Learning Lane Change Behavior to Enable Situation Awareness for Automated Driving on Highways

Von der Fakultät für Ingenieurwissenschaften, Abteilung Maschinenbau und Verfahrenstechnik der Universität Duisburg-Essen zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften
Dr.-Ing.

genehmigte Dissertation

von
Tobias Rehder
aus
Duisburg

Gutachter: Prof. Dr.-Ing. Dr. h.c. Dieter Schramm
Prof. Dr.-Ing. Darius Burschka

Tag der mündlichen Prüfung: 22. Januar 2020

DOI: 10.17185/duepublico/71596
URN: urn:nbn:de:hbz:464-20200514-113958-8

Danksagung

Die vorliegende Arbeit ist im Rahmen meiner Tätigkeit als Doktorand bei der BMW AG in München entstanden.

In besonderer Weise danken möchte ich Herrn Prof. Dieter Schramm vom Lehrstuhl für Mechatronik der Universität Duisburg-Essen. Durch die Übernahme der Betreuung hat er für diese Arbeit nicht nur den Rahmen geschaffen, sondern durch sein entgegengebrachtes Vertrauen, durch die Freiheit zur Ausgestaltung der Forschungsschwerpunkte, sowie durch den wissenschaftlichen Austausch bei regelmäßigen Treffen in München und bei Doktorandenseminaren maßgeblich zum Erfolg dieser Arbeit beigetragen.

Weiterhin danke ich Herrn Prof. Darius Burschka vom Lehrstuhl für Robotik, Künstliche Intelligenz und Echtzeitsysteme der Technischen Universität München für die Übernahme des Koreferates und für die wertvollen Hinweise zu den behandelten Forschungsfragen.

In ganz besonderer Weise bin ich Lawrence Louis zum Dank verpflichtet, der diese Arbeit seitens der BMW AG initiiert hat. Die vermittelten Erfahrungswerte, konstruktiven Diskussionen und vielen gemeinsamen Stunden im Versuchsträger haben diese Arbeit geprägt und werden mir in bester Erinnerung bleiben. Darüber hinaus ist mir seine offene Art und positive Denkweise stets ein Vorbild gewesen, die mich auch persönlich hat reifen lassen.

Hervorheben möchte ich zudem die Unterstützung meiner Kollegen Wolfgang Münst und Alexander König, die an entscheidenen Stellen durch fachlichen Austausch zu wichtigen Impulsen geführt haben, und denen ich freundschaftlich verbunden bleibe. Die Arbeit wurde zudem unterstützt durch die Masterarbeiten von Zdravko Georgiev und Michel Göhl, bei denen ich mich für die gute Zusammenarbeit bedanken möchte.

Schließlich möchte ich meiner Familie - allen voran meinen Eltern - dafür danken, dass sie mich auf meinem bisherigen Weg stets unterstützt haben. Ihr Rückhalt und ihre Zuversicht haben zur notwendigen Kraft beigetragen diesen Weg zu beschreiten.

Abstract

In the automotive industry a robust trend towards assisted and automated driving has evolved during the last years. Although the technology to accomplish this ambition has made great progress recently, there are still a lot of challenges left to make assisted and automated driving a safe and comfortable experience.
One of the main algorithmical challenges is the aptitude of the technical system to acquire situation awareness. Especially the comprehension of the current traffic situation and the anticipation of all traffic participants' future behavior is an indispensable foundation for successful decision-making.
In this thesis a prediction framework is developed that infers a driver's maneuver intention via a hybrid Bayesian network and predicts the future poses of any traffic participant by solving an optimal control problem. The parameters of the methods entailed in the framework are learned from data generated in simulation as well as data acquired from a prototype test vehicle. Different data labeling methods are being investigated to enable the use of supervised machine learning: While the parameters of the Bayesian network are learned using the junction tree algorithm with extensions for hybrid networks, which again is based on the expectation maximization algorithm, the parameters of the optimal control formulation are found via inverse reinforcement learning.

Kurzfassung

In der Automobilindustrie hat sich seit einigen Jahren ein robuster Trend hin zu assistiertem und automatisiertem Fahren entwickelt. Zwar hat die Technologie zur Erreichung dieser Anstrengung deutliche Fortschritte gemacht, jedoch gilt es noch viele Herausforderungen zu meistern, um assistiertes und automatisiertes Fahren zu einem sicheren und komfortablen Erlebnis zu machen.
Eine der größten algorithmischen Herausforderungen ist die Befähigung des technischen Systems Situationsbewusstsein zu entwickeln. Besonders das Verständnis der aktuellen Verkehrssituation und die Antizipation des zukünftigen Verhaltens aller Verkehrsteilnehmer ist eine unverzichtbare Voraussetzung für ein zielvolles Handeln.
In dieser Arbeit wird eine Prädiktionsstruktur entwickelt, die die Manöverintention eines Fahrers mittels eines hybriden Bayesschen Netzes inferiert und die zukünftigen Posen eines Umgebungsfahrzeugs durch die Lösung eines Optimalsteuerungsproblems prädiziert. Die Parameter der in dem Prädiktionsgerüst verwendeten Methoden werden von Daten gelernt, welche sowohl über Simulationen synthetisch generiert, als auch aus Realfahrten in einem prototypischen Versuchsfahrzeug aufgezeichnet wurden. Verschiedene Labelingmethoden werden untersucht, um den Einsatz von Verfahren des überwachten maschinellen Lernens zu ermöglichen: Während das Lernen der Parameter des Bayesschen Netzes mittels des Junction Tree Algorithmus mit enthaltenen Erweiterungen für hybride Netze erfolgt, welcher wiederum auf dem Expectation Maximization Algorithmus basiert, werden die Parameter der Optimalsteuerungsformulierung über inverses bestärkendes Lernen gefunden.

Contents

Abbreviations and Symbols

Abbreviations

Abbreviation	Meaning
ABS	Antilock Braking System
ACC	Adaptive Cruise Control
AD	Automated Driving
ADAS	Advanced Driver Assistance System
AEB	Automatic Emergency Braking
ANN	Artificial Neural Network
BAS	Brake Assist System
BF	Bayesian Filter
BLIS	Blind Spot Information System
BN	Bayesian Network
BSA	Blind Spot Assist
CA	Constant Acceleration
CF	Car Following
CBR	Case-based Reasoning
CCD CPD	Conditional Continuous-Discrete (Conditional Softmax) CPD
CD CPD	Continuous-Discrete (Softmax) CPD
CG CPD	Conditional Gaussian CPD
CLG CPD	Conditional Linear Gaussian CPD
CPD	Conditional Probability Distribution
CPT	Conditional Probability Table
CT	Constant Turn-rate
CV	Constant Velocity
DAS	Driver Assistance System
DBN	Dynamic Bayesian Network
DNN	Deep Neural Network
DW	Distance Warning
EKF	Extended Kalman Filter

EM	Expectation Maximization
ESC	Electronic Stability Control
ESS	Expected Sufficient Statistics
FAD	Fully Automated Driving
FCVM	Forward Vehicle Collision Mitigation
FCVW	Forward Vehicle Collision Warning
FFNN	Feed Forward Neural Network
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
GMM	Gaussian Mixture Model
HAD	Highly Automated Driving
HDA	Highway Driving Assistant
HMM	Hidden Markov Model
IRL	Inverse Reinforcement Learning
KL	Keep Lane (Maneuver)
LC	Lane Change
LCA	Lane Change Assistant
LCB	Lane Change Behavior
LCD	Lane Change Detection
LCI	Lane Change Intention
LCL	Lane Change Left
LCR	Lane Change Right
LCDAS	Lane Change Decision Aid System
LDP	Lane Departure Prevention
LDW	Lane Departure Warning
LG CPD	Linear Gaussian CPD
LIDAR	Light Detection And Ranging
LKAS	Lane Keeping Assistance System
LKS	Lane Keeping Support
LM	Lane Marking
LTI	Linear Time Invariant
NN	Neural Network
MPC	Model Predictive Control
OOBN	Object Oriented Bayesian Network
RADAR	Radio Detection And Ranging
RIP	Running Intersection Property
RL	Reinforcement Learning
RM	Road Model
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic

RVM	Read Vehicle Monitoring | Relevance Vector Machine
SA	Situation Awareness | Side Assistant
SS	Sufficient Statistics
SVM	Support Vector Machine
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
TTB	Time To Brake
TTC	Time To Collision
TTLC	Time To Line Crossing
VDC	Vehicle Dynamics Control
VE	Variable Elimination
VEM	Vehicle Environment Model

Symbols and Notations

Notation	Meaning
a, b, x, y	Scalar
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{y}$	Vector (column format)
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{X}, \boldsymbol{Y}$	Matrix
$\mathbf{x}$	Batch vector (vector of vectors)
$\boldsymbol{\mathcal{X}}$	Batch matrix (matrix of matrices)
t	Point in time
T	Time span
$\dot{x}, \ddot{x}, ...$	Time derivatives of n'th order
$\mathrm{diag}\,(a_1, ..., a_n)$	Diagonal matrix with diagonal values $a_1, ..., a_n$
$\lvert \cdot \rvert$	Cardinality of the argument
$\mathbf{1}\{\cdot\}$	Indicator function (1 if argument is true, 0 otherwise)
	Probability Theory:
$A, B, C, ...$	Discrete random variable
$..., X, Y, Z$	Continuous random variable
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{X}, \boldsymbol{Y}$	Set of random variables
a, b, x, y	Outcome/instantiation of a random variable
$\boldsymbol{\Gamma}$	Set of all continuous random variables
$\boldsymbol{\Delta}$	Set of all discrete random variables
$\mathrm{P}\,(X)$	Probability of random variable X
$\mathrm{P}\,(X \mid Y)$	Conditional probability of random variable X given Y
$\mathrm{L}\,(X)$	Likelihood function
$\ell\,(X)$	Log-Likelihood function
$\mathrm{val}\,(X)$	Set of possible values/outcomes of random variable X
$\sum_X$	Marginalization of random variable X
$\phi(\boldsymbol{X})$	Factor associated to the set of random variables $\boldsymbol{X}$
Φ_C	Potential (factor associated to a clique)
Φ_S	Sepset (factor associated to a separator)
	Graph Theory:
$\mathrm{par}\,(X)$	Parent nodes of node X
$\mathrm{fam}\,(X)$	Family of node X (node itself and its parents or neighbors in the undirected case, respectively)
$\mathrm{dn}\,(X)$	Discrete neighbors of node X
$\mathrm{ccc}\,(X)$	Continuous connected component of node X

Frequently Used Indices

Index	Meaning
$(\cdot)_{\mathrm{ego}}$	Referred to the own vehicle
$(\cdot)_{\mathrm{subj}}$	Referred to the subject vehicle
$(\cdot)_{\mathrm{obj}}$	Referred to the object vehicle
$(\cdot)_{\mathrm{p}_m}$	... m-th preceding object vehicle
$(\cdot)_{\mathrm{s}_m}$	... m-th succeeding object vehicle
$(\cdot)_{\mathrm{p}_m\mathrm{l}_n}$	... m-th preceding object vehicle in n-th lane to the left
$(\cdot)_{\mathrm{p}_m\mathrm{r}_n}$	... m-th preceding object vehicle in n-th lane to the right
$(\cdot)_{\mathrm{s}_m\mathrm{l}_n}$	... m-th succeeding object vehicle in n-th lane to the left
$(\cdot)_{\mathrm{s}_m\mathrm{r}_n}$	... m-th succeeding object vehicle in n-th lane to the right
T_{tc}	Time To Collision
T_{hw}	Time gap/time head way
T_{tb}	Time to brake

CHAPTER 1

Introduction

1.1 Motivation

Automated driving is not science fiction anymore. Many car manufacturers as well as technology companies and suppliers have already announced their plans to boost the existing systems for assisted driving to the next level right up to the holy grail of driving automation: the fully automated or even driverless car.

The development of a fully automated vehicle is not just motivated by the fact that recent progress in hard- and software enable this technology in first place. Rather the vision of individual mobility without any fatalities or serious injuries, which is also called *vision zero* (Johansson, 2009), the increase of comfort during transportation, the maximization of efficiency and many other advantages drive the endeavor to build vehicles that not necessarily need human drivers to transport people or goods.

When looking at the statistics of incidents with personal injuries on German highways from 1970 to 2014 (as shown in Figure 1.1), it can be seen that the number of incidents per 10^9 km has already decreased from 447 in 1970 to 82 in 2014. Several factors have contributed to this pleasant profile, namely the introduction of stricter traffic laws (e.g. the obligation to fasten the seatbelt and a legal alcohol limit for drivers), the improvement of passive vehicle safety systems (optimizing the structural deformation of the vehicle body in case of crashes, the use of airbags etc.) and the start of the era of active safety systems with the introduction of vehicle dynamics control (VDC) systems like the antilock braking system (ABS) or the electronic stability control (ESC).

With the aim to decrease the number of injuries even more, most recently also active exteroceptive systems have been introduced to assist the driver informative or even automate specific tasks of driving. But to distinguish existing assistance systems from future systems fully automating the driving task, six levels of driving automation have been defined by (SAE J3016, 2016) and (BASt, 2012). The automation levels range from *no automa-*

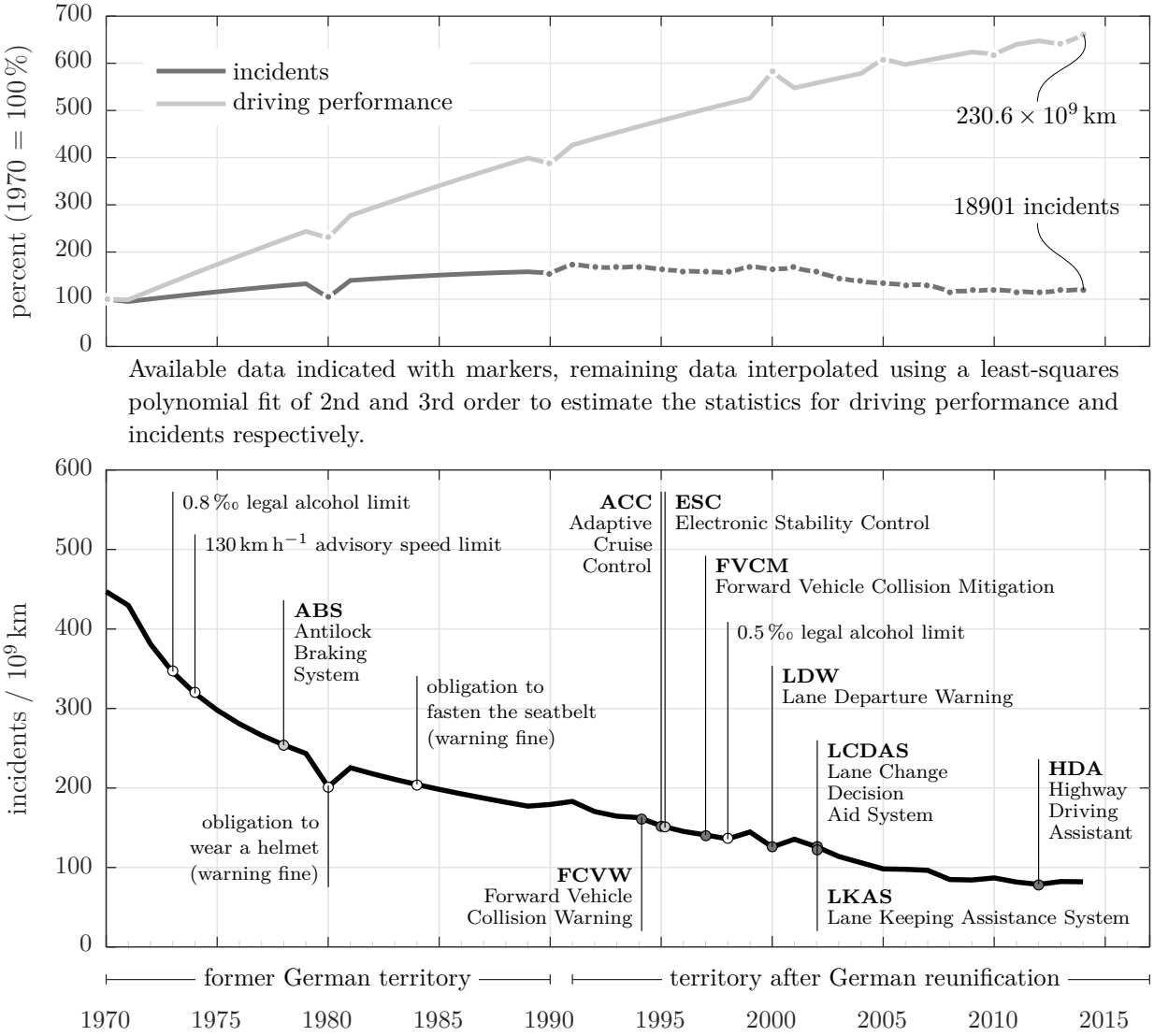
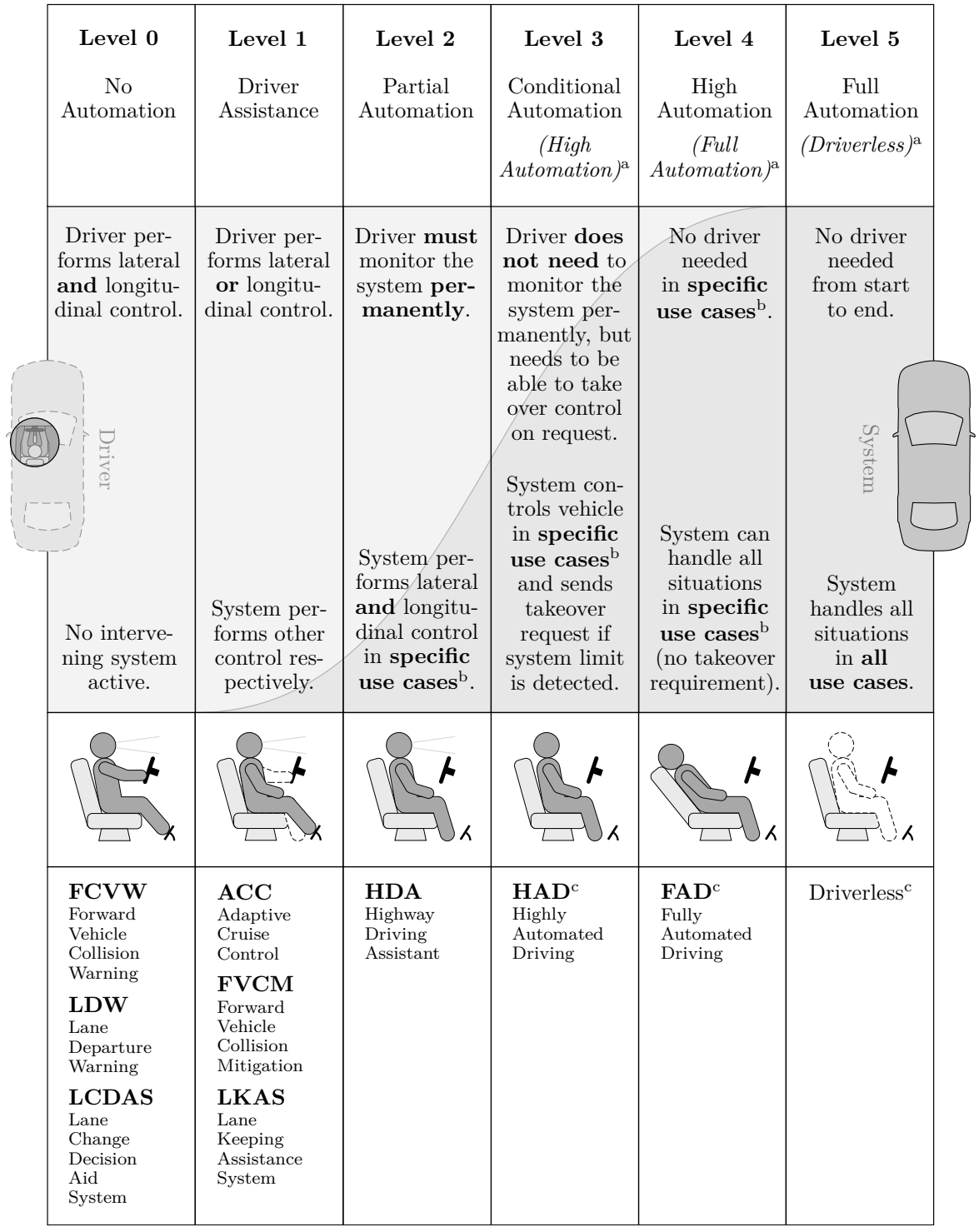


Figure 1.1: Statistics of incidents with personal injuries and driving performance of all motorized vehicles on German highways from 1970 to 2014 (BASt, 2015; Destatis, 2015). Top: Change of incidents and driving performance relative to the year 1970. Bottom: Number of incidents in relation to the driving performance together with introduction of new laws and market launches of vehicle safety and driver assistance systems.

tion in level 0, where a human driver performs all driving tasks manually and is assisted at most informative, to *full automation* which may imply a completely driverless car in level 5 (see Figure 1.2)[1]. Often pure informative assistance systems are also referred to as driver assistance systems (DAS), while systems intervening the vehicle control or partly automating the driving task are called advanced driver assistance systems (ADAS). When the driving task is fully automated (at least in specific use cases) without the driver needed to monitor the system permanently, these systems are here also referred to as automated

[1]The automation levels 3 and 4 are named differently in (BASt, 2012), in particular level 3 corresponds to *high automation* and level 4 is called *full automation*. Level 5 is not defined at all. For a comparison it is also referred to (VDA (en), 2015) versus (VDA (ger), 2015).



[a]Diverging names in (BASt, 2012)[1].
[b]Use cases refer to e.g. road types, speed limits, and environment conditions.
[c]Announced/expected system (provided legal regulations exist).

Figure 1.2: Levels of driving automation according to (SAE J3016, 2016) and (BASt, 2012)[1] together with a timeline of available and expected systems most significant for the development towards automated driving (excluding parking and pure proprioceptive systems).

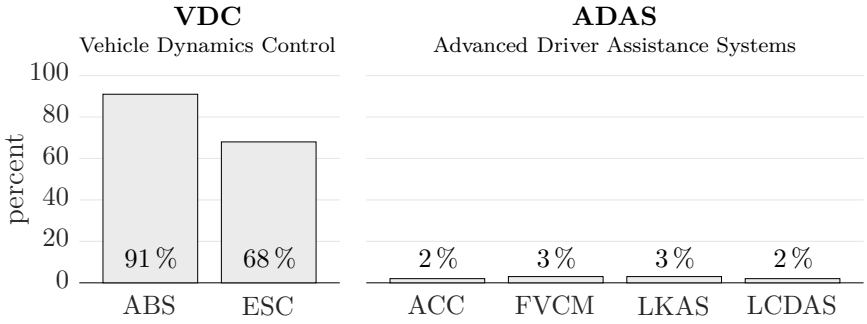


Figure 1.3: Equipment rates of cars with vehicle dynamics control systems and advanced driver assistance systems in 2014 (DAT, 2015).

driving systems (ADS).

Several studies have already investigated the positive impact of ADAS on traffic safety. According to (BASt, 2006), the adaptive cruise control (ACC) has led to 17 % less severe incidents with personal injuries. Forward vehicle collision mitigation (FVCM) systems have caused 28 % less rear-end collisions with injured traffic participants according to (GDV, 2009), and lane keeping assistance systems (LKAS) prevented nearly half of all truck incidents caused by unintended lane departure (AZT, 2006).

But although driving automation systems up to level 2 are available on the market since 1994, it can be observed that the decrease of the incident rate on German highways has slowed down and even stagnated during the last years (see Figure 1.1). One reason for this trend is the market penetration of VDC systems and ADAS as shown in Figure 1.3: In 2014 over 90 % of all cars were equipped with ABS and at least nearly every 7th out of 10 cars utilizes ESC, but the equipment rates for ADAS like the FVCM and LKAS were still only at a maximum of about 3 % (DAT, 2015). This means that the accident prevention potential of VDC systems is nearly depleted, while ADAS may be able to reduce the number of incidents further when the utilization of these systems increases in the future.

On the other hand it has been found out by (Maurer, Stiller, 2005, pp. 161 sqq.) that while warning assistance systems of level 0 generally increase the driver's attention (to avoid the unpleasant warnings), systems that assist or only partly automate the driving task can lead to an overall reduction of attention. When using level 1 driving automation systems the driver retreats from the assisted driving task and the attention is not compensated towards the task that is still being executed manually. Besides this, drivers generally tend to avoid disabling a system performing the vehicle control once it has been activated. Even in critical situations or when the driver notices a driving behavior of the system diverging from the behavior that would have been implemented when driving manually, the driver often bides the system's reactions and only takes over control in the very last moment. So to sum up, ADAS can also have a negative impact on traffic safety as long as the driving task is only partially automated and the systems are not fully reliable, because human drivers tend to over-trust the technology (Waymo, 2017, p. 13). As a consequence, self driving cars fully automating the driving task (at least in specific use cases) are expected to offer a great potential to improve the overall traffic safety.

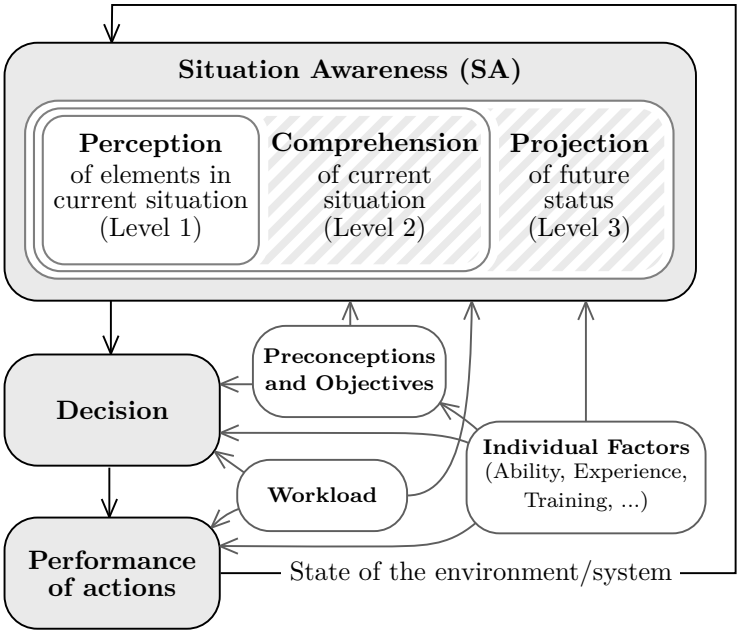


Figure 1.4: Model of situation awareness (SA) and its relationship to decision making and action (Endsley, 1988). The focus of this thesis are methods for the comprehension and projection of traffic situations on highways.

1.2 Problem Statement

To accomplish the mission of a self driving car, many substantial challenges in terms of required vehicle hard- and software have to be overcome. For instance, besides fail-safe steering and braking systems as well as redundant and highly available sensor concepts for sensing the vehicle's environment also new algorithms have to be introduced to realize an adequate decision making of the automated vehicle.

A central and critical task for the driving automation system is to obtain situation awareness (SA), which according to (Endsley, 1995) names the condition of knowing and understanding the state of the environment to a sufficient degree which is critical to the subsequent processes of decision making and performance of action. The process of obtaining situation awareness has the three consecutive levels of *perception*, *comprehension* and *projection*, which are illustrated in Figure 1.4.

The first level of SA is the perception, which for a human is the process of information processing based on stimuli from the environment. But it is assumed that the amount of stimuli that can be processed simultaneously is restricted by a maximum workload. This is why information has to be filtered based on a given context. In general, the human perception process is very complex and it therefore cannot be modeled in full detail.

In a technical system like an automated driving vehicle the perception is defined as the detection of elements in the environment (objects, obstacles, lanes etc.) by a multitude of different environment sensors like ultrasonic sensors (USS), radio and light detection and ranging sensors (RADAR and LIDAR, respectively) and cameras. Thereby the perception also includes the fusion of information coming from these different sensors. Furthermore

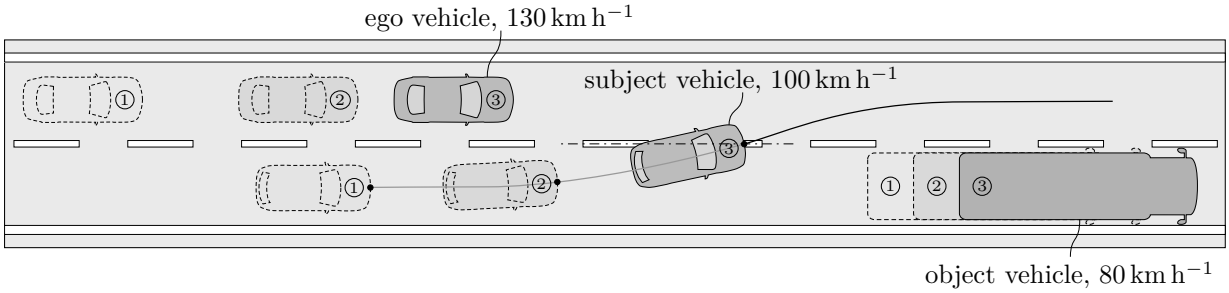


Figure 1.5: Traffic scenario showing a cut-in maneuver of a subject vehicle in front of the ego vehicle because of a slower preceding object vehicle.

elements are filtered, because like in human perception the resources to process information are limited and cannot take all elements and features into account at the same time. Obviously, when using the named environment sensors the detectable features of elements are limited to extrinsic features like the pose of objects or the positions of lane boundaries. Neglecting vehicle-to-vehicle communication, which has not any noteworthy market penetration yet, it is not possible to measure intrinsic features like the driver's head pose et cetera from outside of the vehicle.

In level 2 of SA a holistic picture of the environment is built to comprehend the prevalent situation. This includes interpreting the existence and state of elements in the situation, the recognition of patterns and understanding the impact of the situation on the individual's goals and objectives.

Finally, before any decision can be made and any action can be performed - be it the issue of a warning in an ADAS or the active steering of the vehicle in an ADS - the future statuses of the current situation's elements have to be projected in level 3. Only when the future evolution of a traffic scene is anticipated the driving automation function can make adequate decisions.

A traffic scenario where situation awareness is indispensable and it is therefore necessary to assess the current and predict the future traffic situation is a cut-in maneuver as illustrated in Figure 1.5. To avoid a collision the automated driving ego vehicle has to react to the lane change of a preceding vehicle on the adjacent lane (also referred to as subject vehicle). Depending on the distance and relative velocity between the ego and the subject vehicle the lane change has to be detected several seconds in advance to assure a safe and also comfortable reaction to the cutting-in vehicle. Moreover, the recognition of the lane change intention ahead of the actual lane change execution is needed to empower the ego vehicle to act cooperatively and to enable the lane change of a subject vehicle in first place when the distance to the ego vehicle is too small for the subject vehicle to cut in.

Although the prediction of a traffic scene's evolution is always afflicted with uncertainty (e.g. arising from noisy sensor detections or from the indeterminism of human behavior), still information about the most probable future actions of the traffic participants can improve the vehicle's decision making clearly.

But not only do driving automation systems profit from situation awareness, also driver assistance systems can be enabled to warn the driver in case of critical situations or in-

tervene the vehicle control to avoid collisions. In case of the driver performing at least the longitudinal or lateral vehicle control manually it is beneficial to also anticipate the behavior of the ego vehicle's driver to verify the safety of the maneuvers planned by the ego driver.
To summarize, the advantages of recognizing intentions and predicting maneuvers for assisted and automated driving are:

- **Safety:** Critical situations (e.g. evoked by lane changes) can be evaded and it is possible to maintain a safer distance to other traffic participants even in transient traffic situations.
- **Comfort:** A higher ride comfort can be achieved in terms of a lower acceleration and jerk, for example by starting to brake earlier if a cut-in maneuver is probable.
- **Efficiency:** Better anticipatory driving is enabled, which saves energy and reduces the risk of traffic jams.
- **Cooperativeness:** Altruistic driving is empowered by recognizing maneuver intentions, because traffic participants can be enabled to execute maneuvers that otherwise could not have been performed.

1.3 Document Outline

In chapter 2 an overview of the state of the art of driver assistance systems and automated driving is given for systems that are already available in series production as well as systems that are still part of active research topics. Moreover, existing approaches to a driving behavior prediction are presented and the scientific contribution of this thesis is worked out. A new approach to recognize a driver's maneuver intention on highways based on a hybrid Bayesian network is presented in chapter 3. In chapter 4 this approach is embedded in a behavior prediction framework after the requirements on the prediction for automated driving are discussed. The methodology to learn the parameters of the intention recognition and behavior prediction approach from data is described in chapter 5. Thereby also the process of acquiring driving data is addressed and different strategies to label the data are investigated. The prediction framework is evaluated in chapter 6, and finally the thesis concludes with a discussion and an outlook in chapter 7.

Prior to this thesis several investigations have been conducted that build the foundation of the approach to a driver's maneuver prediction presented here. In particular, in (Rehder, Georgiev, et al., 2015) an entropy-based information filter is used to reduce the feature space for the task of reactively detecting lane change maneuvers with the help of lane-based features. Besides the features' information correlation a potentially achievable prediction time based on the class information entropy is employed as decisive measure for a feature's importance.

In (Rehder, Maas, et al., 2015) a simulator study is being conducted to gather data with lane change motivations manually labeled by the study participants. The same entropy-based feature selection approach is being used to find the most relevant features for classifying a driver's lane change intention.
Investigations towards the performance and practicability of different labeling techniques are carried out in (Rehder, Münst, et al., 2016a). For example, a pilot study in a prototype test vehicle is carried out on a German highway, whereby the study participants are put into responsibility to initiate lane changes from the co-driver's seat. The triggered lane changes (that the driver has to execute after making sure they are safe) serve as labels in the recorded data. The driving data together with the lane change intentions labeled by the co-drivers are used to learn the demonstrated driving behavior in a Bayesian network in (Rehder, Münst, et al., 2016b).
Finally, in (Rehder, König, et al., 2019) the basic idea of the proposed prediction framework is presented, which will be described in this thesis in full detail.

CHAPTER 2

State of the Art

2.1 Automated Driving

Research in the field of automated driving (AD) has already started about three decades ago. In 1987 the PRO-ART project within the scope of the PROMETHEUS (programme for a european traffic of highest efficiency and unprecedented safety) project was one of the first to investigate the automation of the driving task. A van called VAMORS (German *Versuchsfahrzeug für autonome Mobilität und Rechnersehen*, a research vehicle for autonomous mobility and computer vision) was used to test and demonstrate the first automation concepts of lateral and longitudinal guidance on highways (DICKMANNS, GRAEFE, 1988).
The defense advanced research projects agency (DARPA) started challenges for automated driving. For the grand challenges in 2004 and 2005 participating teams had to build and empower a driverless test vehicle to complete an off-road course within a limited time. The DARPA urban challenge carried on the investigations towards more realistic and more complex driving scenarios in 2007, where the participating teams had to design a vehicle and algorithms for navigating through a mock urban environment (KAMMEL et al., 2008).
During these and many following investigations a lot of potential benefits of AD could be pointed out:

- **Safety:** The majority of traffic incidents today can be put down to human error. An inadequate perception, deficient experience, erroneous judgment or missing capacity to react lead to a fatal misbehavior of the human driver. As technical systems are not getting tired or distracted and are able to always act compliant to rules while maximizing the expected utility of their outcome, automated driving offers great potential to increase the road safety - although the validity of any particular safety benefit forecast is based on assumptions and is therefore limited, see (WINKLE, 2016).

- **Economy of time:** No driver is constantly bound to the driving task anymore when using an automation function from level 3 up, so more time can be utilized alternatively (e.g. for work).
- **Energy efficiency:** A foresighted driving style of the driving automation function can be enforced algorithmically, which leads to a reduced energy consumption of automated driving vehicles. Moreover, the transitioning towards new mobility concepts like ride sharing accelerated by automated driving has the potential to reduce the emissions of greenhouse gases (Iglinski, Babiak, 2017).
- **Reduced wear:** A foresighted driving style also leads to a reduction of wear, e.g. because the overall amount of acceleration and braking is reduced.
- **Enabling of new mobility concepts:** Vehicles equipped with a level 5 driving automation system can be used as taxis in a car or ride sharing fleet, picking up the passengers at any desired location and without the need to find a free parking space at the destination. This not only increases the flexibility of individual transportation, but also enables people with impaired mobility (e.g. elderly or disabled people as well as children) to participate in car and ride sharing (Lenz, Fraedrich, 2016).
- **Space efficiency:** An increased acceptance and usage of car and ride sharing leads to a reduction in the total amount of vehicles privately owned (VDV, 2015), leading again to a reduced amount of parking space (which is beneficial especially in cities where space is precious).

But a full automation of the driving task requires many different challenges to be met first, including the controllability of the vehicle through highly available actuators, the sensing of the environment even in bad weather situations, as well as algorithms to comprehend and project the traffic situation and derive adequate decisions for the automated vehicle to transport the passengers to the desired destination safe and comfortably.
Many driver assistance systems up to automation level 2 have been developed and are already available in series production that show the increments in solving partial aspects of the named challenges on the way to a holistic automation of the driving task. The most important systems will therefore be presented in the following.

2.1.1 Driver Assistance Systems

Driver assistance systems (DAS) support the driver when controlling the vehicle - either informatively and alerting or by intervening the vehicle control directly. Strictly speaking even systems like a speedometer and an electric engine starter or vehicle dynamics control systems like the antilock braking system (ABS) and the electronic stability control (ESC) match this definition of DAS. Nevertheless, subsequently only the most significant assistance systems using exteroceptive sensors to observe the vehicle's environment are addressed, because these systems are giving the direction to fully automated driving. In the literature these systems are also referred to as advanced driver assistance systems (ADAS), which are according to (Knapp et al., 2009) specifically characterized by

- supporting the driver in the primary driving task,
- providing active support for the lateral and/or longitudinal control of the vehicle with or without warnings,
- detecting and evaluating the vehicle's environment,
- using complex signal processing and
- interacting directly with the driver.

As of today there are many ADAS available on the market. Figure 1.1 shows a timeline of the market launches of the most significant systems (ignoring parking and pure proprioceptive systems) while Figure 1.2 shows the categorization of the existing and also expected future systems in the six levels of driving automation. In the following a brief description of these systems is given.

Forward Vehicle Collision Warning (FVCW)
One of the first systems that led the way to automating the driving task was the forward vehicle collision warning (FVCW, first introduced as distance warning, DW) which launched the market not later than 1994 (ISO 15623). The FVCW was the first to introduce exteroceptive sensors, in most cases a front-facing radio detection and ranging (RADAR) sensor. This exteroceptive sensor allows for the detection of surrounding vehicles and to measure the distance as well as relative velocity to a preceding vehicle. A warning is raised when the values of these measures (or sophisticated metrics that can be derived from them) exceed specific thresholds. The appropriate selection of the thresholds to initiate a warning is very important, because the driver needs to be warned in time to be able to take an adequate action. But at the same time it needs to be assured that the driver does not get irritated by too many false alarms, which again would lead to a denial of the system by the driver. Because these two requirements are contrary in nature, this fundamental problem in human-machine interaction has been named the *warning dilemma* (Winner, Hakuli, Lotz, et al., 2016, p. 867). Today the ISO 15623 specifies performance requirements and test procedures for FVCW systems.

Lane Departure Warning (LDW)
Computer vision systems were introduced into vehicles with the series production of the lane departure warning (LDW) in the year 2000 (Winner, Hakuli, Lotz, et al., 2016, p. 1225). Cameras paired with hardware running an image processing software are able to detect lane markings in front of the vehicle. This enables a monitoring of the vehicle's lateral position relative to the lane boundaries, whereby the LDW warns the driver when the vehicle is about to cross a lane marking unintendedly (e.g. when no indicator has been activated by the driver). Depending on the system design, the warning is realized either as an acoustic signal, an optical stimulus or a vibration of the steering wheel. For LDW systems the ISO 17361 defines test and operation standards.

Lane Change Decision Aid System (LCDAS)
The lane change decision aid system (LCDAS, sometimes also referred to as lane change assistant, LCA) was introduced in 2002 (Winner, Hakuli, Wolf, 2012, p. 566) extending the environment sensors' field of view to the side of the vehicle to support the driver while

initiating lane changes. Typically ultrasonic sensors are used to sense objects occupying the blind spot, allowing the system to warn the driver about a potential collision when activating the indicator. Therefore these systems are also called blind spot information system (BLIS) or blind spot assistant (BSA). Modern LCDAS often additionally utilize RADAR sensors being capable of detecting objects approaching from behind at greater distance with high relative velocity. These systems are often referred to as rear vehicle monitoring (RVM) or side assist (SA). In ISO 17387 performance requirements and test procedures are standardized individually for a blind spot warning, a closing vehicle warning and a lane change warning (which is a combination of the first two).

Adaptive Cruise Control (ACC)

The first ADAS that not only supports the driver informatively but also actively intervenes the vehicle control is the adaptive cruise control (ACC), available in a series production vehicle since 1995 (Winner, Hakuli, Lotz, et al., 2016, p. 1097). As the first level 1 driving automation system the ACC implements the longitudinal control of the vehicle and maintains a desired speed set by the driver, but as soon as a slower driving vehicle in front is approached the speed is reduced to follow the preceding vehicle (also called target vehicle) at a safe distance. When the target vehicle leaves the predicted driving path of the own vehicle, the desired speed set by the driver is adjusted again. While the first ACC systems were only able to use the throttle and downshifting to control the vehicle's speed, the introduction of brake-by-wire systems enabled all modern systems to also apply the brakes if required. Moreover, the most recent system developments select the target following vehicle not only based on the predicted path of the ego vehicle but also take predictions of the movement of new potential lead vehicles into account (SAE 13863, 2015). While ISO 15622 standardizes the main functionality of an ACC system (i.e., no system operation below a velocity of $5\,\mathrm{m\,s^{-1}}$), ISO 22179 extends the standards to full-speed-range operability.

Forward Vehicle Collision Mitigation (FVCM)

As the quality of the environment detection by the sensors improved it became possible for ADAS to implement even critical interventions to the vehicle control. For example, the forward vehicle collision mitigation (also known as brake assist system, BAS), which was introduced in 1997 (Winner, Hakuli, Lotz, et al., 2016, p. 1152), intervenes the longitudinal control of the vehicle to avoid or at least mitigate the impacts of a collision. A FVCM at first warns the driver and prestresses the brake system for a quicker reaction if a potential front-end vehicle collision is detected (according to the functionality of the FVCW). If the driver reacts and starts to brake, the FVCM applies an additional braking pressure to support the driver's braking maneuver. In case of no reaction by the driver at all, modern systems (which are also known as automatic emergency braking, or autonomous or advanced emergency braking, AEB) apply the braking maneuver fully automated and also react to pedestrians and crossing traffic. As falsely initiated full braking maneuvers clearly pose a high threat to following vehicles, a high detection quality of the environment sensors is crucial for these systems. ISO 22839 specifies operation, performance, and verification requirements for a FVCM system.

Lane Keeping Assistance System (LKAS)
As a consequence of the lane departure warning the lane keeping assistance system (also lane keeping support, LKS, or lane departure prevention, LDP) has been introduced in 2002 (Winner, Hakuli, Lotz, et al., 2016, p. 1225) as a level 1 driving automation system. With the introduction of power steering and brake-by-wire systems the LKAS is able to actively guide the vehicle away from the lane marking back to the center of the lane by applying a steering torque or by wheel-individual braking. ISO 11270 has been introduced to standardize the basic control strategy, minimum functionality requirements, basic driver interface elements, as well as minimum requirements for diagnostics and reaction to failure, and performance test procedures for LKAS.

Highway Driving Assistant (HDA)
The next evolution of ADAS is the highway driving assistant. It is a level 2 driving automation system, which means the system performs the lateral and longitudinal control of the vehicle while the driver must monitor the system permanently. It originates from the combination of the ACC and the LKAS system and is therefore one of the first systems that makes use of different environment sensors at the same time. The HDA at first was designed to work only at low speeds in traffic jams or stop-and-go traffic, which is why it launched the market in 2013 (Winner, Hakuli, Lotz, et al., 2016) as traffic jam assistant. Since then the operation range of the available systems has increased to allow them being activated at full speed range (0 to $>200\,\mathrm{km\,h^{-1}}$) and lane changes to adjacent lanes can be realized automatically when the driver activates the indicator (provided the system detects sufficient free space on the target lane). The system allows the driver to take the hands off of the steering wheel for a short period of time like when using the LKA, but per definition of a level 0 to 2 driving automation system the driver must observe the system permanently and has to be able to react to a system's malfunction instantaneously. A hands-off warning is often implemented to ensure the attentiveness of the driver (which is generally recommended by the ISO 12100 for the manual control of machines whose safe operation depends on permanent, direct control by the operator).

2.1.2 Highly Automated Driving

The transition from a level 2 driving automation system like the highway driving assistant to a level 3 system seems to be a small increment at first glance, because the longitudinal as well as lateral control of the vehicle is already carried out by the technical system. This suggests that the technical aspects of the driving automation are already solved. But in fact the transition of the responsibility for the driving task from the human driver to the technical system increases the demands on the system's operation range and reliability vastly.
In contrast to a level 2 system, where the driver can be assumed to act as a backup in all situations in that the system's operation range is exceeded or when a malfunction of the system occurs, a level 3 system has always to be able to transition to a safe state. Indeed a takeover request can be triggered by the system if an exceedance of the system's

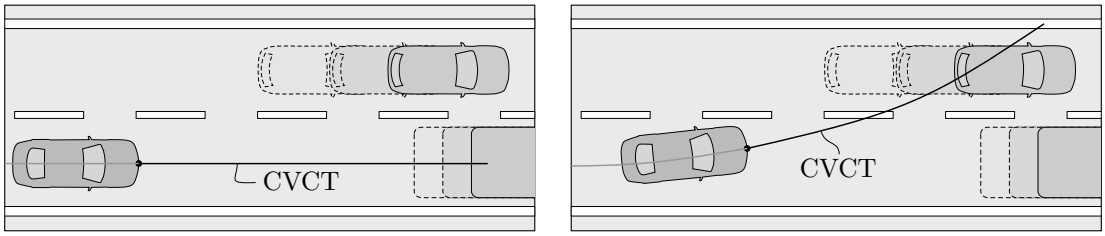
operational limits is predicted, but if the driver is not able to take over the control of the vehicle within a specified takeover time (e.g. because of a medical emergency) the system is required to come to a safe stop. On the one hand this implies that all system components being critical to the system's ability to transition to a safe state have to be fail-safe. On the other hand it has to be assured that the system can reliably handle all use cases for that it has been specified, which greatly increases the effort that has to be spent on tests and coverage.
Up to now no level 3 driving automation system is available in series production, but several car manufacturers and other technology companies have announced to bring a highly automated driving (HAD) systems to market by the end of this decade (provided legal regulations exist). The main use case for which the systems are currently being developed is driving on highways, because the structural separation of lanes with opposing driving direction, the simple road topology (i.e., no intersections) as well as the absence of traffic participants other than cars, trucks and motorcycles constitute a traffic environment with a lower complexity compared to inner-city driving.

2.1.3 Fully Automated Driving

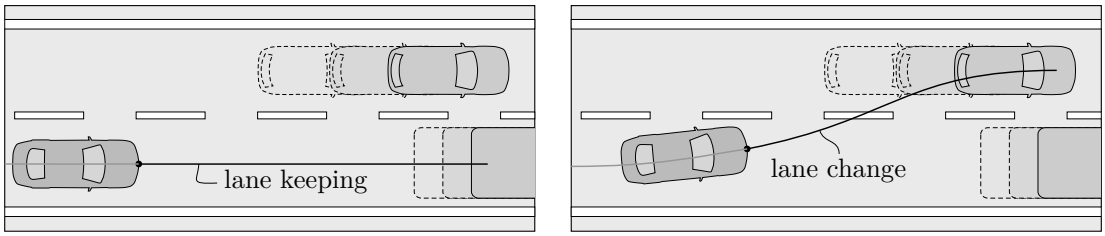
In a fully automated driving (FAD) system the intended use cases of a level 4 system are being enhanced, meaning that more areas will be covered in that the system's operation is facilitated. While HAD systems are mainly focused on highway scenarios, FAD systems are expected to be able to also operate in inner-city areas, requiring the automation function to cope with much more complex traffic scenarios including the presence of pedestrians, cyclists, traffic lights, intersections, roundabouts and suchlike.
As inner-city traffic situations often change much more short-termed (for example cyclists being occluded by other elements in the environment and suddenly crossing the road, forcing the automated driving vehicle to evade or brake immediately), a level 4 driving automation system is not triggering takeover requests. Instead the system has to stay operational without any interaction with the driver.
Finally, a level 5 driving automation system is defined as a fully automated and driverless vehicle.

2.2 Driving Behavior Prediction

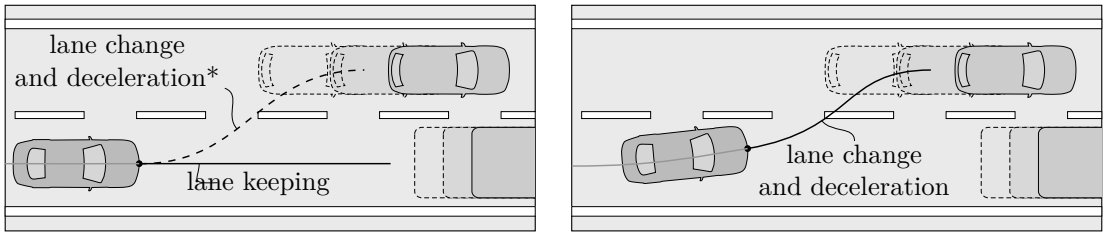
As stated in section 1.2 an essential element required for generating driving behavior and for planning vehicle trajectories is situation awareness, which includes the assessment of the current and especially the projection of the future traffic situation. This is why the term driver behavior prediction as well as the terms that are related and sometimes also used equivalently like intention recognition, behavior identification or state propagation can nowadays often be found in the literature. Many different approaches exist that aim for anticipating human driving behavior for different reasons. Thereby the underlying methodologies are as diverse as the specific prediction objectives and the actual performance of the approaches.



(a) Physics-based



(b) Maneuver-based



(c) Interaction-aware (interaction-based*)

Figure 2.1: Categorization of prediction hypotheses according to (Lefevre et al., 2014) shown for two consecutive time steps with the past trajectory visualized in gray and the predicted future trajectory in black. Left column: Vehicle traversing along right lane at t_n. Right column: Vehicle already started to merge into left lane at t_{n+1}. *Interaction-based category added to refer to prediction solely based on interactions (see dashed line in (c)).

In general, the approaches to driver behavior prediction found in the literature can be sorted according to different categories, which will be presented in the following.

Prediction Hypothesis and Horizon

Different approaches to human driving behavior prediction aim at different prediction hypotheses. The reason for this is that different use cases for the prediction exist that justify varying model assumptions the prediction relies on. Thereby assumptions can either be mild or strong, resulting in varying prediction accuracies and different prediction horizons for the different use cases. In (Lefevre et al., 2014) three common prediction hypothesis models are identified, in particular the *physics-based*, *maneuver-based* and *interaction-aware* motion models.

The **physics-based** models often make the naïve assumption of a vehicle moving with constant velocity (CV) or constant acceleration (CA), sometimes together with for instance also assuming constant turn rate (CT). Obviously these assumptions are wrong when

predicting a vehicle's motion over a longer time horizon, so physics-based models are valid only for a short term prediction of up to one second, which is also illustrated in Figure 2.1a. It can be seen that as soon as a maneuver is executed the assumption of constant velocity and constant turn rate looses its validity and the prediction accuracy decreases.
In (Lin et al., 2000) a single track model with the assumption of constant longitudinal velocity and constant steering angle is employed to project the future path of the ego vehicle to realize a lane departure warning system.
A physics-based prediction for a vehicle collision detection is also accomplished in (Kämpchen et al., 2009) by computing all reachable future vehicle positions limited by the maximum realizable acceleration in terms of Kamm's circle (i.e., taking into account that the sum of the lateral and longitudinal acceleration maximally transmittable by the tires is constant).
In (Barth, Franke, 2008) a constant acceleration and constant turn-rate (CACT) model is used inside an extended Kalman filter to track the motion of oncoming vehicles in a camera image directly.
A survey on kinematic models for the task of vehicle tracking, which also comprises the prediction of future vehicle positions, can be found in (Schubert et al., 2008).
A **maneuver-based** prediction assumes rational driving behavior. It is believed that a vehicle follows the course of lanes and executes maneuvers like turns, lane changes et cetera, but for simplicity no interaction between different vehicles is taken into account. Once a maneuver has been started it can be detected as such and it is assumed that the future movement of the vehicle will match the pre-knowledge about that maneuver. This extends the prediction horizon to the length of the respective maneuver, which for the example of lane changes usually is a maximum of about three seconds.
Figure 2.1b illustrates a maneuver-based prediction and it can be seen that lane keeping behavior is predicted even if most likely a lane change is desired by a driver because of a slower vehicle in front. Not until the vehicle started the lane change (which may be detected on the basis of the vehicle's lateral movement towards an adjacent lane), maneuver-based prediction algorithms detect the maneuver characteristics and adjust the prediction output accordingly. In most approaches this also implies that the predicted maneuver is not necessarily free of collisions. But it has to be emphasized that this is not compellingly a downside of the approach, because the prediction of potential future vehicle collisions would be eliminated by the assumption of maneuvers only being executed if they are safe.
An example of a maneuver-based prediction is presented in (Bahram et al., 2014), where a generative maneuver classification algorithm based on a vehicle's lateral position and lateral velocity with respect to the lane the vehicle is currently driving on is used.
Lane changes on highways as well as turn maneuvers at intersections are classified in (Berndt, Emmert, et al., 2008) using hidden Markov models (HMM). As prediction features signals like gas and brake pedal input, steering wheel angle and map data are used. The algorithm has been trained and tested based on data collected in a prototype vehicle by proficient male drivers.
Turn maneuvers at intersections are investigated by (Eichhorn et al., 2013) modeling the

human driver as the optimizer of an optimal control problem with an unknown terminal state. Cost-to-go gradients are introduced to allow for the anticipation of whether a vehicle is about to cross an intersection or to make a turn.
Unfortunately, maneuver-based prediction approaches don't make use of the fact that maneuvers like lane changes are only executed when a motivation exists to do so. A motivation for a lane change arises for example from the current traffic situation with respect to surrounding traffic participants. Consequently, the use of features describing the relative dynamics to other vehicles are expected to improve the quality of the prediction. To overcome this, approaches have been proposed that extend the feature set of a maneuver-based prediction by features characterizing the relative distance and dynamics to surrounding vehicles. These approaches are referred to as **interaction-aware** prediction models in (Lefevre et al., 2014).
For example, a hand parameterized Bayesian network (BN) is used in (Dagli, Breuel, et al., 2004) (with modifications towards an object-oriented Bayesian network (OOBN) in (Kasper et al., 2011)) to fuse different evidences regarding the trajectory a vehicle might be following, the position relative to the lane the vehicle is currently driving on and the free space with respect to surrounding vehicles on adjacent lanes. Based on these evidences the algorithm infers the probability of a vehicle executing a lane change on highways.
(Weidl, Madsen, 2016) carries on the approaches by extending the BN to a dynamic Bayesian network (DBN) and taking the relative dynamics to the object in front of the subject vehicle into account to classify cut-in maneuvers.
In (Schreier et al., 2014) a BN is modeled and manually parameterized to assess the collision probability of vehicles. Thereby first interaction-aware predictions are carried out for each vehicle individually using causal as well as agnostic inference. Subsequently the collision risk for every pair of vehicles is computed.
However, all named approaches do not distinguish between detecting a maneuver that has already been started and recognizing the motivation of a driver in advance to executing a maneuver solely based on the current traffic situation. This derogates the applicability of these approaches for implementing a cooperative driving behavior.
As a consequence, here it proposed to use an additional category of **interaction-based** prediction models to refer to approaches that are decisively based on the assumption that drivers have a mutual influence and avoid collisions when possible, see Figure 2.1c. Unfortunately, in (Dagli, Reichardt, 2002) it is shown that taking all possible future interaction-based maneuvers into account in a possible worlds structure is not computationally feasible. Therefore the notion of *plausible worlds* is introduced to reduce the computational resources by only considering valid and plausible plans for the prediction of relevant traffic participants. Following this basic idea, in (Lawitzky et al., 2013) and (Deo et al., 2018) first feasible physics- or maneuver-based trajectory hypotheses are generated for each traffic participant individually. Afterwards it is solved for the optimal set of maneuvers in terms of minimizing the overall collision risk. But by doing so still the computational complexity grows exponentially with the number of traffic participants and their feasible maneuvers, which hinders a real time application.
One solution to this problem is to use only asymmetric motion dependencies. In other

words, it is assumed that surrounding traffic affects the future movement of the vehicle of interest, but not vice versa. Thereby often the human decision making process is modeled explicitly, being composed of three sequential steps which are illustrated in Figure 2.2. The first step is the emergence of a *maneuver intention*, which is the motivation to change the lane without knowing yet if the maneuver is realizable at all. According to (Q. Yang, Koutsopoulos, 1996) a lane change intention can have either a discretionary or a mandatory cause. While the intention for a discretionary lane change arises from a higher contentedness on the new lane (e.g. because of a speed advantage), mandatory lane changes are executed to stay on the route to reach a planned destination. The second lane change decision making step is the check whether the lane change is realizable. If the target lane is blocked, the maneuver intention may persist until either the driver finds or enforces a gap on the target lane or until a change of the traffic situation eliminates the intention to leave the current lane (for example because a slower preceding vehicle that motivated the lane change exited the highway). Finally, if a lane change is both desired and realizable, the third step starts, which is the execution of the lane change.

This modelling of the lane change decision making process shows that the very first indication of a potential future lane change maneuver is the maneuver intention. But at this point it has to be noted that several publications misleadingly use the term intention recognition to refer to the detection of the third decision making step only, namely the final execution of the maneuver, as for example in (Kuge et al., 1998; Schlechtriemen, Wedel, Hillenbrand, et al., 2014) (just to name a few). Admittedly it is correct that a driver still has a maneuver intention while moving to the target lane (as depicted in Figure 2.2), but because the maneuver intention emerges even before the maneuver execution and also a maneuver intention may be existing without a maneuver ever being executed (as described before), the usage of the term intention recognition is often incorrect in the literature.

An early approach to model the hierarchy of the human lane change decision making process explicitly is presented in (Gipps, 1986). While the model has been designed for the use in a driving simulation software for simulating logical driving behavior, and not specifically for the task of a behavior prediction (which would require modeling realistic behavior), yet the basic concept and model assumptions are transferable. In particular, the following factors have been found to be most important for modelling the lane change decision process: 1) Whether it is physically possible and safe to change the lane. 2) The location of permanent obstructions. 3) The presence of transit lanes. 4) The driver's intended turning movement. 5) The presence of heavy vehicles. 6) Finally, the speed advantage after a possible lane change.

In (Q. Yang, Koutsopoulos, 1996) modelling the discretionary lane change intention of a driver has also been accomplished by checking admissible adjacent lanes for speed advantages compared to the current lane (if the vehicle's current speed is lower than the driver's desired velocity). Parameters such as an impatient factor or a speed indifference factor are used to estimate the grade of the lane change intent.

The model in (Ehmanns, Hochstädter, 2000) follows a similar approach by estimating the contentedness of a driver on all admissible lanes (composed of the speed relations,

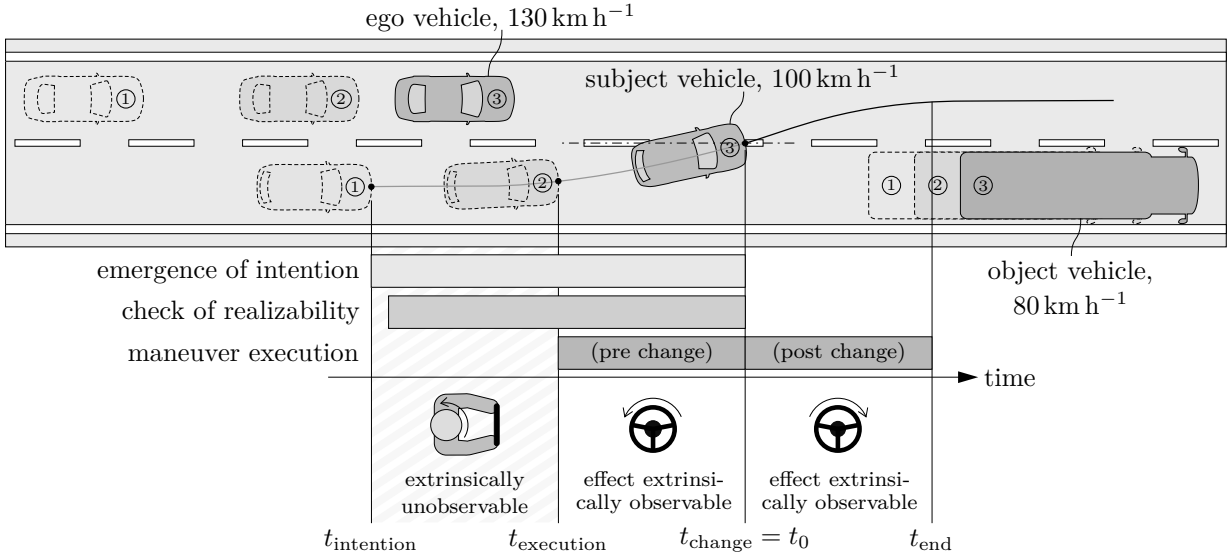


Figure 2.2: Traffic scenario showing a cut-in maneuver of the subject vehicle in front of the ego vehicle because of a slower preceding object vehicle together with the idealized chronological sequence of the lane change decision making process.

vehicle interactions like the pushing of a faster vehicle from behind etc.) and raising a lane change intention if the contentedness on an adjacent lane exceeds the one on the current lane by a specified threshold.

Unfortunately, in all these approaches the parameters have been chosen based on (subjective and individual) expert knowledge and the models have not been validated for the task of predicting human driving behavior in real world traffic.

In contrast, (Graf et al., 2013) uses a case-based reasoning (CBR) approach to classify overtaking maneuvers and learns the cases from data. But as similarity measure for the cases the time series of only the relative distance and velocity to the preceding vehicle are taken into account, which is only a very limited characterization of the traffic situation.

The approach presented in (Bonnin et al., 2014) relies on the time to collision (TTC) and relative velocity to the preceding vehicle, the size of the gap on the target lane, and the time and acceleration that is needed to enter the gap to predict lane changes to the left only. But the model neglects factors like the succeeding traffic or the speed advantage gained through the lane change (if any).

Prediction Scope

Not all approaches are appropriate to predict driving behavior in different driving situations equally well. While some approaches are targeted at predicting lane changes on **highways**, others specifically aim at anticipating intersection turn maneuvers in **urban** environments. Only a few methods are designed to generalize to all kinds of driving situations.

Besides the already mentioned approach of (Eichhorn et al., 2013) another contribution focusing on predicting driving maneuvers at intersection scenarios is for example the approach of (Liebner et al., 2013), where a naïve Bayes classifier with the indicator, velocity and gaze direction as features is used to create hypotheses about possible future paths.

The approach is tested to predict right-turns at intersections. A study was conducted to collect data from 6 subjects on 5 intersections, which summarizes to 200 right turn maneuvers that were used to parameterize the probability distributions of the classifier. Data from a different study containing 15 hours of driving data collected by 12 subjects was used to evaluate the model.

(Petrich et al., 2013) uses an extended Kalman filter (EKF) with a pseudo-measurement in the update step to select a representative set of reasonable trajectories for predicting the future motion of a vehicle in complex scenarios like intersections. The plausibility of each of the trajectories in the set is then assessed by an adaptive multivariate cumulative sum (MCUSUM) algorithm.

An approach specifically targeted at detecting lane changes on highways (as are the already introduced ones e.g. by (Bahram et al., 2014; Berndt, Emmert, et al., 2008; Weidl, Madsen, 2016)) is the one by (Schlechtriemen, Wedel, Breuel, et al., 2014) using Gaussian mixture models (GMM) to represent probable future vehicle positions in longitudinal direction. The model parameters have been trained and validated using data samples from tracked objects on test drives summing up to roughly 87 hours of driving.

(Morris et al., 2011) predicts highway lane change maneuvers of the ego driver by incorporating features like steering wheel angle, indicator state, lane-relative position, the driver's head orientation and relative speed to a preceding vehicle with all values of a short time window of the past in a relevance vector machine (RVM).

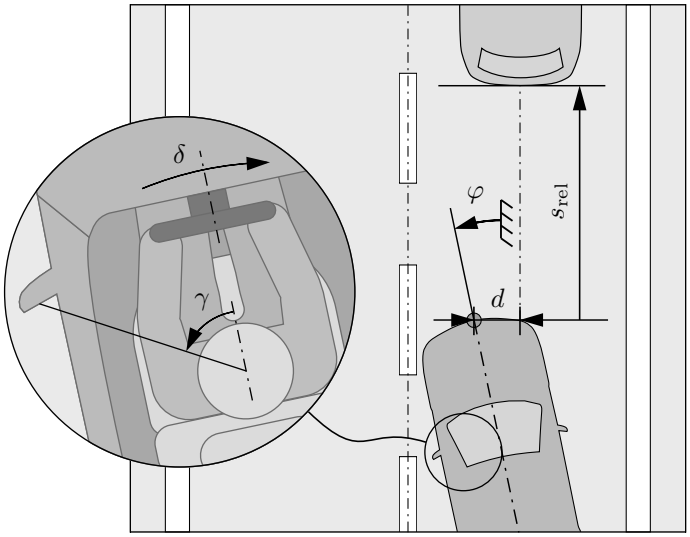
In (Gindele et al., 2015) a hierarchical dynamic Bayesian model describing the physical relationships as well as the driver's behaviors and plans is presented to predict driving behavior in varying traffic situations, but in (Bonnin et al., 2013; Bonnin et al., 2014) it is claimed that methods generic in scope predict behaviors only with low quality, while methods designed for a specific and narrow scope can predict behavior for even complex scenarios. Therefore in (Bonnin et al., 2013; Bonnin et al., 2014) it is strived for a combination of classifiers to cover a wide scope with high prediction quality. The use of a location-based hierarchy of prediction models is proposed, i.e., a generic maneuver classifier at the top of the hierarchy predicts the future traffic scene all the time, but it can be overruled by specific classifiers that are activated depending on the traffic scene or location of the vehicle.

Reviews and comparisons particularly of lane change models (which also served as information source here) can also be found in (Rahman et al., 2013; Zheng, 2013).

Prediction Subject

It can be found that there are mainly two different prediction subjects of interest, namely the **ego vehicle** (which is the own vehicle that is supposed to be equipped with the prediction system in question) and the vehicles that exist in the surrounding of the ego vehicle and that potentially have an influence on the ego vehicle's driving behavior, hereafter called **surrounding** or **traffic vehicles**. For the later of course the first step to making a prediction is the assessment of the relevance of each vehicle in the surrounding.

The ego vehicle is subject to the prediction of the approach e.g. in (McCall et al., 2007), (Kuge et al., 1998) and (D. D. Salvucci et al., 2007). But while the prediction of the



γ	Gaze direction
δ	Steering wheel angle
⋮	
s_{rel}	Distance to surrounding vehicle
d	Distance to center line
φ	Vehicle orientation w.r.t. lane
⋮	

Figure 2.3: Examples of intrinsic and extrinsic prediction features without explicit depiction of features that can be derived from them, like (angular) velocities, accelerations et cetera.

ego vehicle behavior in manual driving mode may be useful as status information to send to other vehicles or to road infrastructure elements via vehicle-to-vehicle communication, or used to enable a monitoring system that detects abnormal or harmful human driving behavior (as passive automated driving function), in general for the task of automated driving it is especially important to anticipate the driving behavior of surrounding vehicles. For example, in (Dagli, Breuel, et al., 2004; Kasper et al., 2011; Bonnin et al., 2013) subject to the prediction are the surrounding vehicles.
In general it shouldn't matter if an approach to driving behavior prediction is used to anticipate the future movements of the ego vehicle or that of surrounding vehicles, but in practice the approaches differ in the information used as input to the prediction algorithm. This limits for example the applicability of algorithms targeted at ego vehicle lane change prediction to predicting cut-in maneuvers of surrounding vehicles, because some features used for the classification may either be of too low quality when measured via exteroceptive sensors (like the lane relative lateral acceleration) or they may be not available at all (e.g. the head pose of the driver).

Prediction Input

Which measurements and features serve as input to the different prediction approaches of course greatly depends on the prediction subject and the prediction objective. For example, the already mentioned approach of (McCall et al., 2007), that classifies lane changes of the ego vehicle, uses **intrinsic features** like gas and brake pedal position, steering wheel angle and the head orientation of the driver (among other features), which can be measured with proprioceptive sensors, and it is shown that the prediction performance decreases when the head movement of the driver is omitted.
(Kuge et al., 1998) relies on the steering wheel angle and steering velocity as input features to a hidden Markov model. The model has been evaluated based on a simulator study

with ten participants.
In (P. Kumar et al., 2013) the lane relative lateral position and steering angle (including their derivatives) are used to predict lane changes of the ego vehicle's driver in highway scenarios in France. Using a stereo camera for the detection of lane markings, 139 lane changes were executed by two different drivers for training and testing their classification approach.
In contrast, **extrinsic features** like lane relative features (lateral offset and velocity), features that describe the relative distance, velocity and acceleration to a gap on the adjacent lane a vehicle could potentially merge to, as well as trajectory features (i.e., recordings over time) like time-to-line-crossing (TTLC), maximum lateral acceleration and the track angle are used in (Dagli, Breuel, et al., 2004) to detect possible cut-in maneuvers. All data is discretized to be used in a probabilistic network that infers the lane change probability of a possible cut-in vehicle. Based on approximately 800 test kilometers driven on German motorways it is pointed out that with a single front-facing light detection and ranging (LIDAR) sensor (36° beam angle) a proper lane change prediction is achieved only within a range of 18 - 60 m.
Trends and levels of distances and relative velocities to the preceding vehicles on the own and the adjacent lane are being used in (Graf et al., 2013) to associate the current traffic situation to known cases that have been stored in memory. Because the future driving maneuver of a vehicle has been recorded for the cases available in memory, this way the purpose of a vehicle to cut-in into the ego vehicle's lane can be inferred.

Prediction Output

Different prediction approaches existing in the literature represent future driving behavior differently. A broad categorization of the prediction output can be achieved by differentiating between the output of **categorical actions** and **future positions** of traffic participants.
A categorical representation of future driving behavior is typically the output of a maneuver classification algorithm. Out of a set of n possible maneuvers, e.g. $M = \{m_1, m_2, ..., m_n\} = \{\textit{lane change left}, \textit{lane change right}, ..., \textit{no maneuver}\}$, the prediction at a discrete point in time t can be given by the realization of a single outcome like $M_t = m_1$. Often a categorical representation is combined with a probabilistic depiction to represent uncertainty about the future actions. The output is then a binomial distribution over the set of actions M denoted as $\mathrm{P}(M_t)$, where e.g. the probability of the realization of the single outcome m_1 would be depicted by $\mathrm{P}(M_t = m_1)$.
In (Dagli, Breuel, et al., 2004) the binary probability of a cut-in maneuver of a particular vehicle is emitted by a probabilistic model. (Kasper et al., 2011) extends the approach to the output of a probability distribution over the discrete maneuvers *lane follow*, *leave lane towards the left* and *leave lane towards the right*, which is computed via an object-oriented Bayesian network (OOBN).
For the representation of driving behavior as future vehicle positions, (Kuchar, L. C. Yang, 2000) gives an overview of state propagation representations for the prediction of future aircraft positions, which have here been adapted to the domain of vehicles. Fig-

ure 2.4 shows the different position representations graphically.
The nominal state propagation (illustrated in Figure 2.4a) describes the future positions of a vehicle over time by a single trajectory. The trajectory represents the true future vehicle positions based on the vehicle's current state as long as the state estimation and the prediction hypothesis by which the trajectory has been generated is correct. Due to sensor noise that influences the state estimation and due to the fact that the vehicle may not behave as expected, the nominal prediction representation can be inaccurate. The representation of the nominal trajectory can e.g. be a single polynomial of reasonable order (Kasper et al., 2011), a spline, a Bézier curve (Gindele et al., 2010) or positions at discrete points in time.
To overcome the inaccuracies of the nominal future state representation, probabilistic state propagation methods take uncertainties in the current state estimation as well as variations in the driving behavior itself into account when estimating future vehicle positions. The distribution of possible future vehicle positions is sometimes also referred to as stochastic reachable set. The greatest challenge in using probabilistic state propagation methods is the quantification of uncertainty. A nominal trajectory can always be derived from a probabilistic representation by following the most likely trajectory, and also a set of possible future positions can be extracted by following any trajectory with equal likelihood. In general different approaches exist to represent uncertainty in state propagation.
First, there exist sampling- or particle-based methods as shown in Figure 2.4b and for instance used in the approaches by (Gindele et al., 2015).
Second, parametric density-based methods can be used, e.g. employing Gaussian densities as illustrated by the σ, 2σ and 3σ confidence intervals in Figure 2.4c and used in (Petrich et al., 2013).
Another way of representing the probability of presence in future time steps is to discretize the state space and to assign a probability to each of the discrete states (e.g. using Cartesian grids for a 2-dimensional vehicle position representation with **occupancy probabilities** assigned to each grid cell as shown in Figure 2.4d). State space discretization is used in (Althoff et al., 2009) to detect probable future vehicle collisions.
As it is quite hard to reuse a probabilistic future state representation e.g. for a trajectory planning algorithm, sometimes a vehicle's future position is represented area-based. An example is a worst case representation which estimates the area of all positions a vehicle is able to reach within a given period of time when performing any of all possible maneuvers (sometimes also referred to as reachable set). This is mostly achieved by applying force field methods, where it is for example assumed that the vehicle's motion is only restricted by the forces transmittable by the tires (prediction areas resulting from a worst case prediction with the so-called circles of forces are exemplarily shown in Figure 2.4e).
One example for an area-based prediction output is the already mentioned approach of (Kämpchen et al., 2009), using Kamm's circle to estimate all future positions reachable by a particular vehicle.
But there also exist approaches with interaction-aware prediction hypotheses that represent the prediction results area-based. For example, in (Schlechtriemen, Wabersich, et al., 2016) the deterministic areas occupied by a vehicle in the future are computed lane

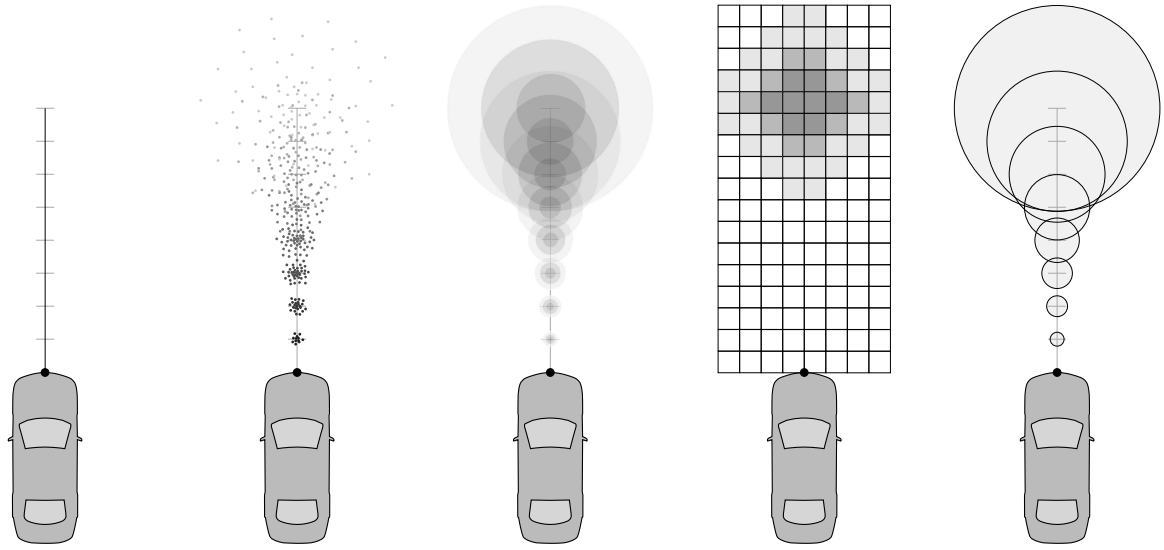


Figure 2.4: Common representations of predicted future vehicle positions (following (Kuchar, L. C. Yang, 2000), but extended by different probabilistic representations). (a) Nominal trajectory prediction. (b) Probabilistic particle or sampling based future position representation. (c) Parametric density-based prediction. (d) Future vehicle position represented by occupancies in a discrete state space (here: probabilistic occupancy grid for a single point in time). (e) Area-based prediction representation.

discrete in a curvilinear coordinate system via confidence bounds on a longitudinal and lateral probability density function which has been learned from data to describe the future probability of presence of surrounding traffic participants. With this area-based representation it is easier to detect potential future vehicle collisions (or compute the safe driving space by inverting the occupancies) and to plan a safe trajectory for the automated driving ego vehicle.

Methodology

Which methodology and which algorithms may be used to predict driving behavior greatly depends on the requirements on the prediction. The algorithms of the approaches referred to so far are already very versatile.

First, as a matter of principal, it can be found that the task of behavior prediction is closely related to the task of behavior planning as anticipating driving behavior means to mimic a subject vehicle's decision making and action. But in practice the premises are often different: On the one hand it is assumed that all intrinsic data of the ego vehicle together with high quality environment perception data is available as model input, but for the task of the surrounding vehicles' behavior prediction only a subset of the features are available. On the other hand (Rolnick, Lubow, 1991; Festner et al., 2016) found out that with a lack of controllability most passengers are more prone to motion sickness and feel uncomfortable when a system which is not controlled by themselves mimics the driving characteristics of a human driver too closely. As a consequence they expect that an

automated vehicle drives more defensive and comfortable compared to the human driver. This is why behavior planning algorithms will not be investigated for the task of behavior prediction here.

Many models to replicate (not necessarily to explicitly predict) human driving behavior exist that are being completely defined by **expert knowledge**. A pundit models the relationship between the environment and human decision making manually and subjectively. Examples of expert models are the deterministic rule-based models by (Gipps, 1980; Gipps, 1986), with the former modeling a vehicle's longitudinal car following (CF) behavior and the later focusing on lane changing (LC) behavior. A lane change is executed by the Gipps model when it is desired, necessary and realizable. (Q. Yang, Koutsopoulos, 1996) extends Gipps' lane change model by probabilistic maneuver decisions and to the application on highways. Moreover a distinction between discretionary and mandatory lane changes is introduced. In contrast, (Hidas, 2005) repeals Gipps' assumption that lane changes are only executed when they are safe, which is why three different lane change categories are introduced: free, cooperative and forced lane changes.

In (Kesting et al., 2007) the lane change model of minimizing overall braking induced by lane changes (MOBIL) is introduced, which is based on the intelligent driver model (IDM) for the longitudinal behavior introduced in (Treiber et al., 2000). A lane change decision is made by anticipating and then comparing all involved vehicles' accelerations for the case of changing the lane versus staying on the current lane. With a politeness factor introduced to weight between egoistic and altruistic behavior a lane change is executed when the overall acceleration induced by the lane change is greater than when staying on the lane.

Utility theory based algorithms are those of (Ahmed, 1999; Toledo, Koutsopoulos, et al., 2007), where lane change decisions are made based on a utility measure that incorporates individual driving characteristics and past lane change decisions.

In (D. D. Salvucci et al., 2007) a technique called model tracing is used, which simulates all possible future maneuvers based on a behavioral model and classifies the actual action of a vehicle afterwards using a similarity measure. The behavioral model in this case is an expert model, which simulates lane keeping and lane changing maneuvers of the ego vehicle.

Another group of lane change models are cellular automata approaches. The models of (Rickert et al., 1995; Wagner et al., 1997; Nagel et al., 1998; Maerivoet, Moor, 2005) are consecutive improvements of (Nagatani, 1994) by incorporating stochastic influences on the lane change decision process. They describe the lane usage inversion phenomenon on highways (i.e., overuse of fast lanes when the traffic density increases) and solve planning conflicts when multiple vehicles select the same cell to drive in.

Fuzzy logic is used in (Wu et al., 2000) and (Moridpour et al., 2009). In these approaches fuzzy sets are defined on the input features like inter-vehicle space gaps, relative speeds and average velocities and then a logic rule deduces the truth value of the current situation being a lane change or no lane change situation.

The downside of the approaches based on expert knowledge described so far is that every expert subjectively defines the models to represent human driving behavior as well as

the characteristics the models depend on. Moreover, tuning the models' parameters is often a very complex and time consuming task, because many dependencies between the parameters exist that are not apparent. In contrast, **machine learning** approaches fit the models' parameters to observed driving data using objective cost functions while taking into account dependencies between the parameters.
An approach to driver behavior prediction based on machine learning is for example presented in (P. Kumar et al., 2013), where a support vector machine (SVM) with a generalized Bradley-Terry model is used to obtain a probabilistic maneuver classification. The output of the SVM is smoothed in a Bayesian filter (BF) afterwards to reduce the rate of false alarms and missed detections.
(Dogan et al., 2011) compares two types of artificial neural networks (ANN, or NN in short), namely a recurrent neural network (RNN) and a feed-forward neural network (FFNN) to the classification performance of a support vector machine.
The downside of SVM and NN is that no probabilistic inference is used to account for uncertainties in the representation of a state of knowledge. This uncertainty arises in human perception as well as in measurements through technical sensors. How a human acts in everyday driving situations depends on the state of knowledge regarding the environment, which is fraught with uncertainty.
Models dealing with uncertainty are (among others) the ones based on Markov processes like presented in (Worrall et al., 1970; Pentland, Liu, 1999; Singh, Li, 2012). (Toledo, Katz, 2009) describes the human decision making process by integrating a hidden Markov model (HMM) into a utility-based approach. A double-layer hidden Markov model is used in (He et al., 2012).
(Boyraz et al., 2009) utilizes hybrid dynamic systems combining stochastic modeling (like a HMM) with control theoretic models. This is justified by constituting that the nature of driving is stochastic and therefore it should be searched for trends in the data instead of exact matches in numerical sense.
In (Schlechtriemen, Wedel, Hillenbrand, et al., 2014) a naïve Bayes classifier is compared to Gaussian filtering and to a hidden Markov model. It is shown that for the use case of an interaction-aware lane change prediction the naïve Bayes classifier performs best.
Bayesian networks (BN) are used for instance in the already mentioned approaches of (Gindele et al., 2015; Dagli, Breuel, et al., 2004; Kasper et al., 2011; Weidl, Madsen, 2016), although only in (Gindele et al., 2015) the parameters of the probabilistic network are indeed learned from data. A hand-parameterized BN is also presented in (Schreier et al., 2014), where the approach is primarily focused on criticality assessment. It is used to predict unlikely vehicle collisions and therefore does not take the complete traffic situation into account. In other words, the implicit premise for predicting driving maneuvers is that drivers do not notice each other, which is why the features taken into account are limited to vehicle-lane relations and the relative dynamics to the vehicle in front only.
In general, the downside of Bayesian networks is that inference in domains with discrete as well as continuous variables is hard, which is why often discretization is used (leading to information loss).

2.3 Summary

While many approaches for a maneuver-based driver behavior prediction exist, only a few approaches can be found that recognize a maneuver intention before the maneuver actually takes place. In a few existing behavior models like the ones by (Gipps, 1986; Q. Yang, Koutsopoulos, 1996; Ehmanns, Hochstädter, 2000) indeed the lane change intention is computed explicitly, but the parameters of all presented models are not chosen empirically and the models are not validated with real world driving data.
Without raising claim to completeness, Table 2.1 summarizes the already mentioned existing approaches in the field of driver behavior prediction. Contributions not emphasizing the task of prediction are omitted. Moreover, if many contributions from the same authors exist across which the presented approach is only changed slightly, only the latest known publication is listed here.
Another classification of existing methods for risk assessment and driver intent inference that focuses on the prediction hypotheses and the underlying prediction methodologies can be found in (Winner, Hakuli, Lotz, et al., 2016, pp. 896 sqq.).

	Hyp.				Scope		Subj.		In		Out			Meth.		Data	
	physics-based	maneuver-based	interaction-aware	interaction-based	highway traffic[a]	inner-city traffic	ego vehicle	surrounding vehicles	intrinsic features	extrinsic features	categorical prediction	nominal trajectory	probabilistic occupancy	expert model	machine learning model	simulator data	real-world driving
(Althoff et al., 2009)	•	•	•			•		•		•			•	•		•	
(Bahram et al., 2014)		•			•			•		•	•				•	•[b]	•
(Berndt et al., 2008)		•			•	•	•		•	•	•				•		•
(Bonnin et al., 2014)				•	•	•		•		•	•			•	•		•
(Deo et al., 2018)	•			•	•			•		•	•	•		•	•		•
(Dogan et al., 2011)		•	•		•		•		•	•	•				•	•[b]	
(Eichhorn et al., 2013)		•				•	•		•	•		•		•			•
(Firl et al., 2011)			•		•			•		•	•				•	•	•
(Gerdes, 2006)			•		•	•	•		•	•	•			•			•
(Gindele et al., 2015)			•			•		•		•			•	•	•	•	
(Graf et al., 2013)				•	•	•		•		•	•				•		•
(He et al., 2012)		•			•		•		•		•				•	•[b]	
(Jordan et al., 2015)			•		•			•		•			•	•	•		•
(Kuge et al., 1998)		•			•		•		•		•				•	•[b]	
(Kumar et al., 2013)		•			•		•		•	•	•				•		•
(Lawitzky et al., 2013)		•		•	•			•		•	•			•		•	•
(Liebner et al., 2013)		•				•	•		•	•	•			•			•
(Lin et al., 2000)	•	•			•		•		•	•			•	•		•	
(Mandalia et al., 2005)		•	•		•		•		•	•	•				•		•
(Meyer-D. et al., 2009)				•	•			•		•	•		•		•	•	•
(Morris et al., 2011)			•		•		•		•	•	•				•		•
(Oliver et al., 2000)			•		•	•	•		•	•	•				•		•
(Petrich et al., 2013)	•	•				•	•	•		•			•	•			•
(Platho et al., 2013)			•			•		•		•		•			•	•	•
(Salvucci et al., 2007)			•		•		•		•	•	•			•		•[b]	•
(Schlechtr. et al., 2014)			•		•			•		•	•				•		•
(Schlechtr. et al., 2015)			•		•			•		•	•		•		•		•
(Schreier et. al., 2014)			•		•	•	•	•	•		•		•	•		•	
(Tomar et al., 2012)			•			•		•		•		•			•		•[c]
(Torkolla et al., 2005)			•		•	•	•		•	•	•				•	•[b]	
(Weidl et al., 2016)			•		•			•		•	•	•		•			•
(Wiest et al., 2012)		•				•	•			•			•		•		•

[a]Including main and rural roads with structural separation of opposing traffic streams.
[b]Data has been obtained from a conducted study with human drivers.
[c]Data has been recorded from a spectator's view (world-fixed sensor installation).

Table 2.1: Categorization of existing prediction approaches.

2.4 Scientific Contribution

Existing models to driver behavior anticipation either not recognize the intention of a driver to execute a specific maneuver explicitly, are too complex to be parameterized by a human expert, or are not validated using real world driving data. On the other hand, many models directly learned from data like e.g. neural networks can only be used as black-box models without any expert being able to interpret the model parameters after the model training phase.
Therefore the need for a new interaction-aware lane change model arises which represents the causal relations of driving behavior and is able to infer a driver's maneuver intention solely based on the predominant traffic situation. Also, as the human behavior is non-deterministic by nature, the model has to be able to account for uncertainties by using probabilistic representations of the dependencies. For this purpose in this thesis a three-layered hybrid Bayesian network is proposed which uses the concept of lane contentedness estimations to model the drivers' lane change decision making process. Thereby the nature of all variables' cardinalities is taken into account without loosing information by discretizing the continuous variables' states.
An extension of the junction tree algorithm is employed for exact inference in the probabilistic graphical model containing both continuous and discrete variables. On top of the inference process a learning method for discrete variables with hidden continuous parent nodes is presented, because when applying the model to predict human behavior the internals of the driver's decision making are unobservable through the vehicle's sensors. Still, to be able to learn the model's parameters from real world driving data while at the same time providing for human interpretable parameters in the hidden layers, two basic approaches are being pursued: First, different data labeling strategies are being investigated to gather information about a driver's intentions. For this purpose test studies in a driving simulator as well as in a prototype vehicle are being carried out. Second, a pre-training step for learning the model's parameters from simulated data is established to guide the optimization process when finally training the parameters based on German highway data gathered from a prototype test vehicle.
In order to not only predict a driver's intentions but also identify the actual future driving behavior in terms of the vehicle's trajectory, a maneuver-based prediction method based on an optimal control problem formulation is proposed. Thereby inverse reinforcement learning is used to find the parameters of this approach for mimicking human driving behavior most realistically.
Finally, a prediction framework is presented that uses a generalized approach to identify observed driving behavior based on any set of reasonable trajectory hypotheses. It also combines the information of a driver's maneuver intention with the outcome of the behavior identification to anticipate a vehicle's most likely future trajectory with a higher accuracy.

CHAPTER 3

Intention Recognition

According to (Knapp et al., 2009) an intention is the aim of the driver to perform an action, which does not imply that the action is actually performed. In case of a lane change intention this means that a driver has a motivation to change the lane, but maybe the target lane is blocked and the lane change cannot be executed at this point in time.
Knowledge about the intentions of a surrounding vehicle's driver is important for automated driving, because it enables a cooperative driving behavior in first place. For example, only if the motivation of a driver to merge in front of the ego vehicle is known, the ego vehicle can slow down to open an adequate gap to enable the lane change in first place. Moreover, the recognition of a driver's lane change intention increases the comfort and safety of automated driving, because potentially a greater maneuver prediction time can be achieved, which enables the ego vehicle to optimize the deceleration behavior and keep a greater distance to a lane changing vehicle. Finally, as it is assumed that a lane change is never executed without a motivation to do so, information about a lane change intention is valuable for the detection of an ongoing lane change maneuver of the same vehicle in terms of improving the maneuver detection accuracy.
The approach to a maneuver intention recognition here is based on the lane change decision model used in the microscopic traffic simulation software Pelops (program for the development of longitudinal microscopic traffic processes in systemrelevant environment), presented in (Hochstaedter et al., 1999). Like in many other lane change models, e.g. the ones presented in (Gipps, 1986; Q. Yang, Koutsopoulos, 1996; Hidas, Behbahanizadeh, 1999), in Pelops the premise for a lane change decision is the desirability or necessity to leave the current lane. It is generated from a comparison of the expected contentedness when driving on each of the available lanes, which is estimated by projecting the vehicle onto each of the available lanes and assessing the traffic situation as if the vehicle is driving on that lane (see Figure 3.1). The overall contentedness on an individual lane is again composed of different contentedness factors, which use a variety of features that characterize the current traffic situation. These features are described in section 3.1.

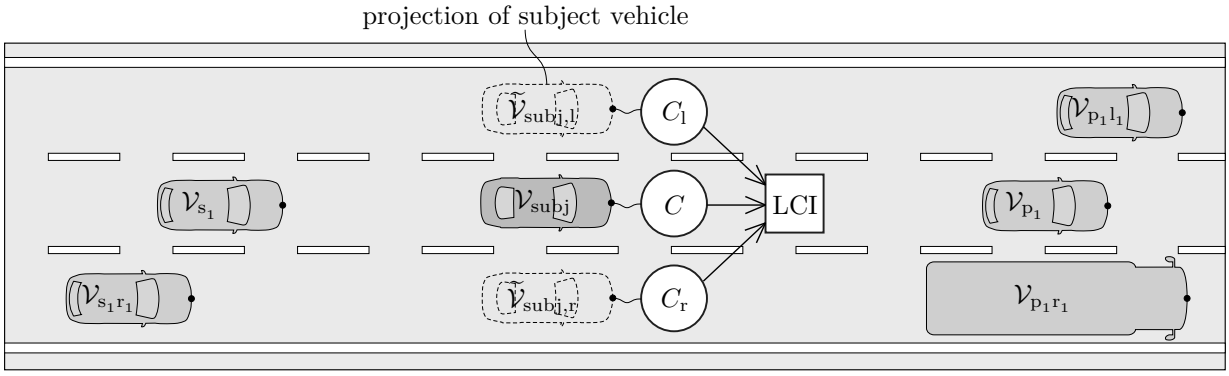


Figure 3.1: Illustration of intention recognition approach based on lane contentedness estimation.

While in PELOPS the computation of the lane contentedness is done deterministically based on parameters tuned by experts, the approach chosen here is a different one: A Bayesian network (BN) is used which integrates the vehicle's contentment on the lanes as random variables and models the dependencies between the features and the contentedness estimations probabilistically. This way the model is able to account for uncertainties arising from sensor noise, limitations in the model's ability to represent the human driving behavior, or even indeterminism of the human driving behavior itself.

In the next section, at first features to characterize a traffic situation are presented, before section 3.2 and 3.3 describe the methodological background of Bayesian networks. Finally, chapter 3 presents the Bayesian network structure to recognize a driver's maneuver intention.

3.1 Features to Characterize a Traffic Situation

As the task of the prediction framework is to anticipate the human behavior, the system first needs to obtain situation awareness before the human decision making and action execution process can be predicted. For a driver to gain proper situation awareness, it is crucial to not only perceive but also to comprehend all relevant elements in the traffic environment, meaning that the relevance and potential impact of objects on the own vehicle must be known. Only if the traffic participants and their relations to the own vehicle are well understood, the future status of the environment can be anticipated and a decision for a future action can be made (as already described in section 1.2 and illustrated in Figure 1.4).
For the technical system to obtain situation awareness, at first the identification of possible influences on the human driving behavior is required, which are expected to be

- the behavior of surrounding traffic participants,
- traffic rules,
- the road course and route,

- the driver's individual driving experience and performance,
- the own vehicle type and equipment,
- weather, sight and road conditions, and
- influences from potential co-drivers,

but of course this list is incomplete and there are many more possible influences that vary the process of decision making over individuals and over time.
Next, features need to be defined which characterize and quantify the most important influences and the current state of the environment. Here, these are features describing attributes of the environment's elements, features characterizing the pose and relative dynamics of a vehicle with respect to the lane the vehicle is driving on, as well as features quantifying inter-vehicle relationships, e.g. the distance and relative dynamics to surrounding vehicles.
For example, attributes for lanes are the lane type (e.g. traffic lane, bus lane or highway on-ramp), lane marking type (e.g. solid or dashed line), the intended driving direction, as well as the speed limit and other traffic rules such as an overtaking ban. The lane geometry is described by the absolute value of the curvature $|\kappa|$ at the vehicle's current position. Additionally a concept from PELOPS is being adopted to compute the maximum relevant curvature ahead (denoted by $\kappa_{\max}$) within a foresight distance that the driver is able to overlook. The basic idea is to find and use the curvature at exactly that position that maximizes the quotient of the curvature at a particular distance divided by that distance. This is to not neglect any upcoming curvature when a higher curvature is recognized further ahead. Consequently it is

$$\kappa_{\max} = \max(|\kappa(s=0)|, |\kappa(s_{\kappa_{\max}})|), \quad \text{with} \tag{3.1}$$

$$s_{\kappa_{\max}} = \underset{s \geq 1}{\operatorname{argmax}}\left(\frac{|\kappa(s)|}{s}\right), \tag{3.2}$$

where s is the longitudinal curvilinear distance ahead (within a maximum distance of 100 m) and $\kappa(s)$ is the curvature at that distance.
For vehicles relevant attributes are the vehicle type classification with the possible types *car*, *truck* or *bike*, the object dimensions with the vehicle's width w and length l (the z-axis is neglected in most applications) and the object's pose.
Another important attribute for vehicles is the indicator state, but (BERNDT, DIETMAYER, 2009) argues that for the task of a maneuver prediction the indicator state should not be used, because the indicator is not utilized reliably by some drivers and the lane change probability would then be underestimated for lane changes without an indicator activation. As no indicator activation does not imply that a vehicle is not changing the lane, but on the other hand an activated indicator should always be treated as information of a driver's lane change intention, the indicator can always be taken into account via a rule-based logical `OR` condition on top of any sophisticated intention recognition algorithm.
A highway scenario together with an illustration of the vehicle-lane relations and the lane-discrete vehicle-vehicle relations is shown in Figure 3.2. A vehicle for which the lane change intention shall be inferred of is called *subject vehicle* and is denoted by $\mathcal{V}_{\text{subj}}$, its

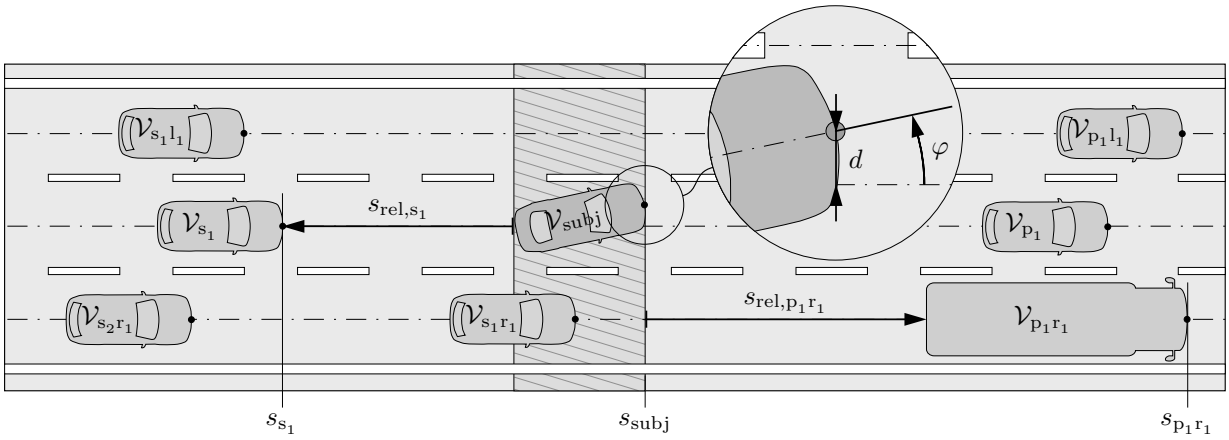


Figure 3.2: Definition of vehicle-to-vehicle and vehicle-to-lane relations. Vehicle-to-vehicle relations are features characterizing the relative distance (s_{rel}) and dynamics of object vehicles $\mathcal{V}_{\text{obj}}$ to a subject vehicle $\mathcal{V}_{\text{subj}}$ for which the future driving behavior shall be inferred. Vehicle-to-lane relations are features characterizing the subject vehicle's pose relative to the center of a lane (like deflection from centerline d or relative angle φ).

surrounding vehicles are called *object vehicles* $\mathcal{V}_{\text{obj}}$, which are the vehicles that $\mathcal{V}_{\text{subj}}$ can possibly interact with.
To depict a particular object vehicle, the subscript p or s together with a one-based index is introduced to refer to a preceding or succeeding vehicle, respectively. To refer to vehicles on neighboring lanes, an l or r together with an additional one-based index is appended to depict specific lanes. For example, $\mathcal{V}_{\text{p}_1}$ depicts the preceding vehicle on the subject vehicle's lane, whereas $\mathcal{V}_{\text{s}_2\text{r}_1}$ denotes the second succeeding object vehicle on the right lane. It is emphasized that in general any observed vehicle can be treated as subject vehicle; the ego vehicle as well as any other traffic vehicle on the road. The notations of the surrounding object vehicles always changes with respect to the currently selected subject vehicle.
For the estimation of both the lateral and longitudinal position of an object vehicle relative to $\mathcal{V}_{\text{subj}}$, the center of each vehicle's front is used as reference point. That is, in a curvilinear coordinate system of the road, where s is the longitudinal axis and d is the lateral offset from s, a vehicle $\mathcal{V}_{\text{s}_1\text{r}_1}$ overtaking $\mathcal{V}_{\text{subj}}$ becomes $\mathcal{V}_{\text{p}_1\text{r}_1}$ when its reference point is ahead of the subject vehicle's reference point and it holds $s_{\text{s}_1\text{r}_1} > s_{\text{subj}}$. However, the distance s_{rel} between two vehicles is specified as clearance, so the free space between the vehicles along the curvature of the lane is measured. This leads to the definition of

$$s_{\text{rel}} = \begin{cases} s_{\text{obj}} - s_{\text{subj}} - l_{\text{obj}} & \text{if } s_{\text{obj}} > s_{\text{subj}} \text{ and } s_{\text{obj}} - s_{\text{subj}} \geq l_{\text{obj}} \\ 0 & \text{if } s_{\text{obj}} > s_{\text{subj}} \text{ and } s_{\text{obj}} - s_{\text{subj}} < l_{\text{obj}} \\ s_{\text{obj}} - s_{\text{subj}} - l_{\text{subj}} & \text{if } s_{\text{obj}} \leq s_{\text{subj}} \text{ and } s_{\text{obj}} - s_{\text{subj}} \leq l_{\text{subj}} \\ 0 & \text{if } s_{\text{obj}} \leq s_{\text{subj}} \text{ and } s_{\text{obj}} - s_{\text{subj}} > l_{\text{subj}}. \end{cases} \tag{3.3}$$

As a consequence in the scenario shown in Figure 3.2 the distance $s_{\text{rel},\text{s}_1\text{r}_1}$ from $\mathcal{V}_{\text{subj}}$ to $\mathcal{V}_{\text{s}_1\text{r}_1}$ is zero as long as the object vehicle is located alongside the subject vehicle, which is indicated by the shaded area.

For convenience, in the following (and throughout the rest of this thesis) the time derivatives of s are denoted by

$$v = \dot{s} \quad \text{with} \quad v_0 = \dot{s}(t = t_0), \tag{3.4}$$

$$a = \ddot{s} \quad \text{with} \quad a_0 = \ddot{s}(t = t_0). \tag{3.5}$$

Based on the surrounding vehicle's distances as well as the relative velocity and acceleration with

$$v_{\text{rel}} = \dot{s}_{\text{obj}} - \dot{s}_{\text{subj}} \quad \text{and} \tag{3.6}$$

$$a_{\text{rel}} = \ddot{s}_{\text{obj}} - \ddot{s}_{\text{subj}}, \tag{3.7}$$

respectively, also more advanced metrics have been introduced in the literature to mimic the human discrimination of motion along the line of sight. For example, according to (Winner, Hakuli, Lotz, et al., 2016, p. 16) it is hard for a human to determine absolute distances. Instead it has been found out that the decisive measure for a driver's situation assessment is the so-called time to collision T_{tc} (first named in (Hayward, 1972) and sometimes denoted as TTC) with

$$T_{\text{tc}} = -\frac{s_{\text{rel}}}{v_{\text{rel}}}. \tag{3.8}$$

It represents the time it takes for two vehicles following the same path to collide, assuming the vehicles' velocities are constant. Also in (Evans, Rothery, 1974) the inverse of the time to collision

$$T_{\text{tc}}^{-1} = -\frac{v_{\text{rel}}}{s_{\text{rel}}} \tag{3.9}$$

has been identified as the most relevant metric for a human to discriminate longitudinal motion as needed in car-following scenarios, although it was not known under this name back then. Compared to the time to collision, the inverse time to collision has the advantage of being continuous at $v_{\text{rel}} = 0$. On the other hand it is $T_{\text{tc}}^{-1} \to \pm\infty$ for $s_{\text{rel}} \to 0$ instead, but in this case the inverse time to collision can more reasonably be set to a maximum (or minimum) value. Moreover the inverse time to collision is positively (instead of negatively) correlated to the risk of a collision.

Besides the time to collision, often a velocity dependent distance measure, the so-called time headway (sometimes denoted by THW, or time gap, TGAP) with

$$T_{\text{hw}} = -\frac{s_{\text{rel}}}{v_{\text{subj}}} \tag{3.10}$$

is used to describe the inter-vehicle relations. For example, ISO 15622 suggests to use the time headway as control parameter in the vehicle following mode of adaptive cruise control systems (see subsection 2.1.1 for a short description of the system's functionality).

For practical reasons it is also useful to to define the inverse of the time headway as

$$T_{\text{hw}}^{-1} = -\frac{v_{\text{subj}}}{s_{\text{rel}}}, \tag{3.11}$$

because like the inverse time to collision the reciprocal of the time headway is positively (instead of negatively) correlated to the collision risk.

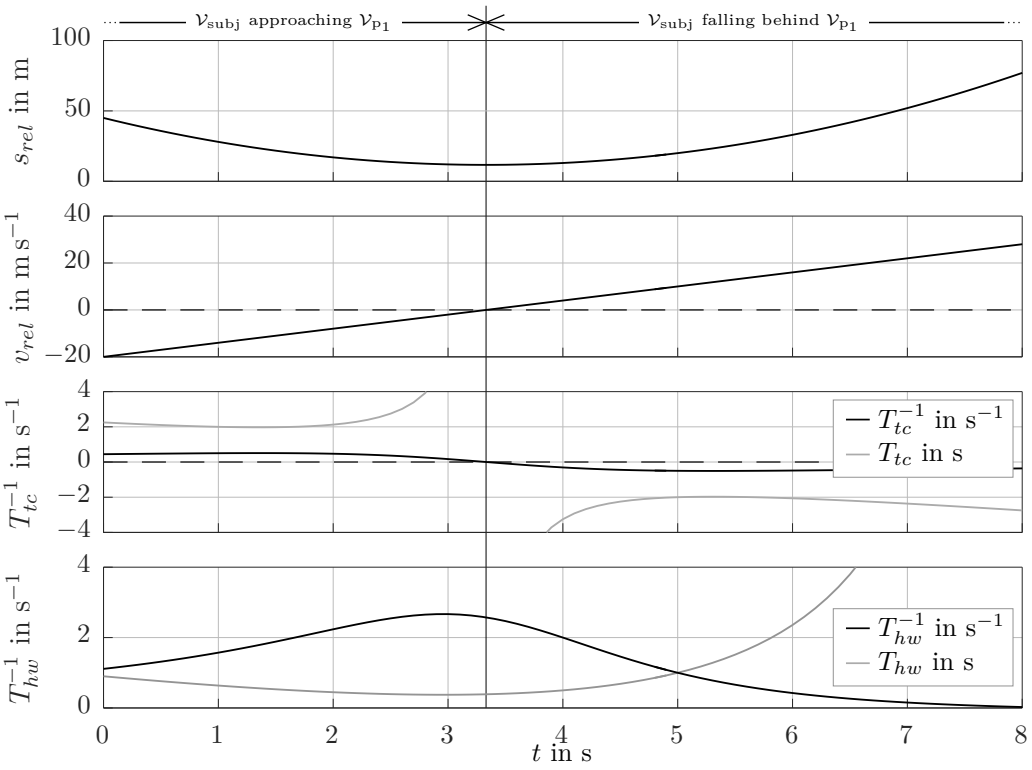


Figure 3.3: Exemplary course of the inverse time to collision and inverse time headway compared to their original definitions for a scenario of a subject vehicle $\mathcal{V}_{\text{subj}}$ approaching a slower object vehicle $\mathcal{V}_{\text{p}_1}$ with a constant acceleration of $-6\,\text{m}\,\text{s}^{-2}$ ($s_{\text{rel,p}_1,0} = 45\,\text{m}$, $v_{\text{rel,p}_1,0} = -20\,\text{m}\,\text{s}^{-1}$).

An exemplary course of the time to collision and time headway compared to their inverse definitions for a scenario of a subject vehicle $\mathcal{V}_{\text{subj}}$ approaching a slower object vehicle $\mathcal{V}_{\text{p}_1}$ with constant acceleration is shown in Figure 3.3.

3.2 Bayesian Networks

A Bayesian network (sometimes called Bayes net in short) is a graphical model that represents the probabilistic dependencies of random variables. It is a popular and efficient model, because it very compactly represents the joint probability distribution over the random variables it contains by exploiting (the assumption of) independence properties of the variables (Pearl, 1988). Variables in a network are modeled as nodes, directed edges between the nodes show their probabilistic dependencies and form the directed, acyclic graph structure $\mathcal{G}$. For each node the probabilistic dependency is quantified by a conditional probability distribution (CPD). A more detailed description of Bayesian networks can be found for example in (Koller, Friedman, 2009; Murphy, 2012).

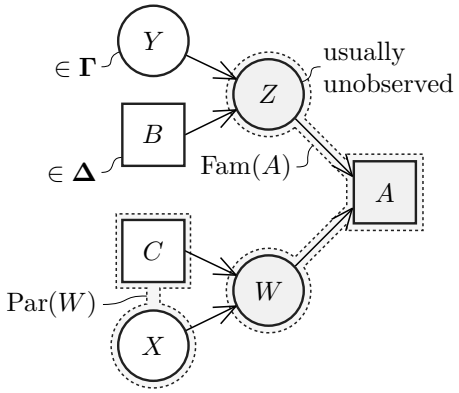
At first some notations have to be defined. In general, random variables in the network are denoted by upper case letters. Often letters from the beginning of the alphabet ($A, B, C, ...$) are used for discrete variables, whereas letters from the end of the alphabet ($..., X, Y, Z$)

represent continuous variables. Lower case letters are used to denote the realization of a random variable, e.g. $X = x$, bold faced letters depict a vector or a set of nodes (e.g. $\boldsymbol{X} = \{X_1, ..., X_n\}$) or their values $\boldsymbol{x}$. The set $\boldsymbol{\Gamma}$ is defined to entail all continuous nodes, $\boldsymbol{\Delta}$ is the set of all nodes in the discrete domain. Moreover, the notation of $\mathrm{Par}(X)$ is used to refer to the parents (the direct predecessors) of a node X, $\mathrm{Fam}(X)$ depicts the node's family, which is the node itself and its parents. Again, lower case letters, e.g. $\mathrm{par}(X)$ and $\mathrm{fam}(X)$, depict the realization of these variables. Furthermore, $\mathrm{val}(A)$ refers to all possible outcomes (or state combinations) of A, e.g. for a binary valued node it is $\mathrm{val}(A) = \{a_1, a_2\}$, for a set of binary valued nodes $\boldsymbol{C} = \{A, B\}$ it is $\mathrm{val}(\boldsymbol{C}) = \{\{a_1, b_1\}, \{a_1, b_2\}, \{a_2, b_1\}, \{a_2, b_2\}\}$. The notation $\mathrm{P}(X)$ denotes the probability distribution for node X, which is a mass function in the discrete case and a density function if X is continuous. $\mathrm{P}(X \mid Y)$ refers to the conditional distribution of X given Y. The notation $\mathrm{P}(X = x)$ or $\mathrm{P}(x)$ represents the probability (mass or density, respectively) that X takes the value x, $\mathrm{P}(X \mid y)$ refers to the conditional distribution of X given $Y = y$ analogously.
Finally, the joint probability distribution of a BN given the graph structure $\mathcal{G}$ and the conditional probability distributions is defined via the chain rule. For n variables $X_1, ..., X_n$ in the network it holds

$$\mathrm{P}(X_1, ..., X_n) = \prod_{i=1}^{n} \mathrm{P}(X_i \mid \mathrm{Par}(X_i)). \tag{3.12}$$

3.2.1 Network Structure

Before the proposed BN structure used for recognizing maneuver intentions is presented in section 3.4, here only a simple exemplary network structure is introduced to give a demonstrative description of the theory behind Bayesian networks in general.
The exemplary network presented in Figure 3.15 aims at inferring a driver's lane change intention (LCI) based on estimates of the driver's contentedness when driving on each of the available lanes on a highway. Thereby the lane contentedness estimation denoted by C is a continuous value scaling between $+1$ (being maximally content on that lane) and -1 (being maximally discontent). For simplicity it is assumed that only two lanes are available, namely the own and an adjacent lane. Furthermore, the network structure is based on the assumption that a driver's lane contentedness depends on the inverse time to collision $T^{-1}_{\mathrm{tc,p_1}}$ to the first preceding vehicle on the respective lane only. The variables $e_{\mathrm{p_1}}$ are used to indicate these object vehicles' existences, which generalizes the network structure for different constellations of surrounding vehicles.
In a BN's graph (generally denoted by $\mathcal{G}$ and for the particular case shown in Figure 3.15) continuous variables are represented as round nodes, discrete variables are plotted as squares. A Bayesian network containing both continuous and discrete variables is called a *hybrid* Bayesian network (Koller, Friedman, 2009, p. 186). Moreover, clear nodes depict variables that are always observed, gray nodes refer to variables that are usually hidden.
For an easier notation the original variable names have been substituted according to the table shown on the right hand side of Figure 3.4.



Subs.	Var.	Description
A	LCI	lane change intention
		own lane:
B	e_{p_1}	existence of $\mathcal{V}_{\mathrm{p}_1}$
Y	$T^{-1}_{\mathrm{tc},\mathrm{p}_1}$	inverse time to collision to $\mathcal{V}_{\mathrm{p}_1}$
Z	C_{own}	lane contentedness
		adjacent lane:
C	$e_{\mathrm{p}_1,\mathrm{adj}}$	existence of $\mathcal{V}_{\mathrm{p}_1,\mathrm{adj}}$
X	$T^{-1}_{\mathrm{tc},\mathrm{p}_1,\mathrm{adj}}$	inverse time to collision to $\mathcal{V}_{\mathrm{p}_1,\mathrm{adj}}$
W	C_{adj}	lane contentedness

Figure 3.4: Simplified Bayesian network to recognize a driver's lane change intention (LCI). Continuous variables are represented as round nodes, discrete variables are plotted as squares. Clear nodes depict variables that are always observed, gray nodes refer to variables that are usually hidden.

3.2.2 Conditional Probability Distributions

While the BN graph structure $\mathcal{G}$ describes the structure of the probabilistic relationship between the network's variables, the conditional probability distributions (CPD) quantify the variables' dependencies.

In general, many different types of CPD exist that can be employed in Bayesian networks, but whether a particular type of CPD is suited to describe the probabilistic relationship between a specific node and its parents depends on the cardinality of the involved nodes, which is the node the CPD is defined for itself and its parents. In the following the CPD types used in the BN of this thesis are described, which are also the ones most commonly used for BN in the literature. An overview of the employed CPD is given in Figure 3.5.

A tabular conditional probability distribution (also called conditional probability table, CPT) is used to model the relationship between a discrete node and its pure discrete parents (Koller, Friedman, 2009, pp. 157 sq.). The table has as many parameters θ as there are state combinations of the discrete node in junction with its parents, that is $|\operatorname{val}(\operatorname{Par}(C))| \cdot |\operatorname{val}(C)|$, e.g. 2^n when there are n all binary valued nodes involved. In practice the number of independent parameters needed to be stored in the table is lower because of the requirement that all conditional probabilities must sum to 1 (see the grayed out parameters in the example shown in Figure 3.5), but this saving is not significant.

In case of a continuous node that has only discrete parents, for each state combination of the discrete parents the first two statistical moments $\theta = \{\mu, \sigma\}$ (i.e., $\mu = \mathbb{E}[X]$ and $\sigma^2 = \mathbb{E}[X^2] - \mathbb{E}[X]^2$) of the Gaussian normal distribution with

$$\mathcal{N}(Z; \mu, \sigma^2) = \underbrace{\frac{1}{\sqrt{2\pi\sigma^2}}}_{p} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right) \tag{3.13}$$

are stored in a table, whereby the term indicated by p is a normalizing constant to assure

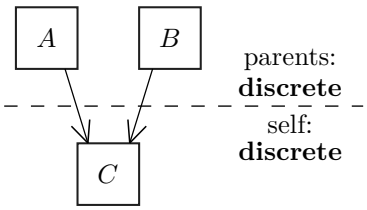
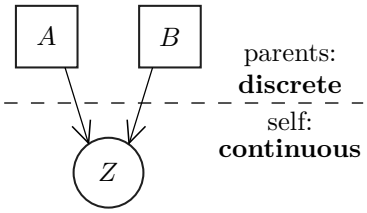
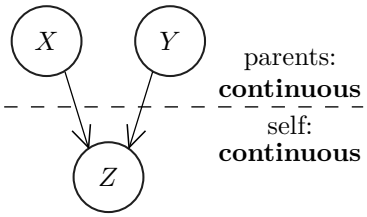
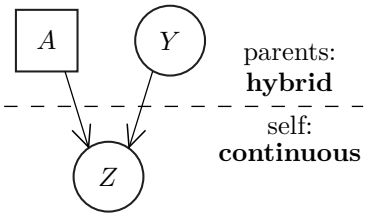
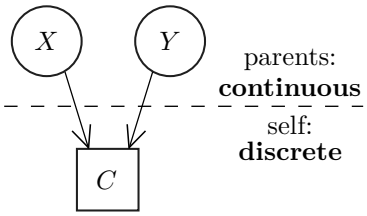
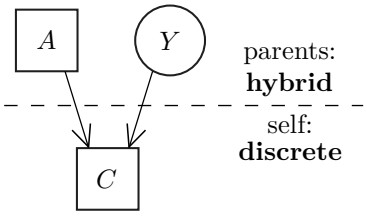
	Nodes' Cardinality	Conditional Probability Distribution (CPD)
Tabular (CPT)		$\mathrm{P}(c_k \mid a_i, b_j) = \theta_{i,j,k}$ A B $C=c_1$ $C=c_2$ a_1 b_1 $\theta_{1,1,1}$ $1-\theta_{1,1,1}$ a_1 b_2 $\theta_{1,2,1}$ $1-\theta_{1,2,1}$ a_2 b_1 $\theta_{2,1,1}$ $1-\theta_{2,1,1}$ a_2 b_2 $\theta_{2,2,1}$ $1-\theta_{2,2,1}$
Conditional Gaussian (CG CPD)		$\mathrm{P}(Z \mid a_i, b_j) = \mathcal{N}(Z; \mu_{i,j}, \sigma^2_{i,j})$ with $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$
Linear Gaussian (LG CPD)		$\mathrm{P}(Z \mid x, y) = \mathcal{N}(Z; \underbrace{\mu_0 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{s}}_{\mu}, \sigma^2)$ with $\boldsymbol{S} = \mathrm{Par}(Z) \cap \boldsymbol{\Gamma}$, $\boldsymbol{\theta} = \{\mu_0, \boldsymbol{w}, \sigma\}$
Conditional Linear Gaussian (CLG CPD)		$\mathrm{P}(Z \mid a_i, y) = \mathcal{N}(Z; \underbrace{\mu_{0,i} + \boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{s}}_{\mu_i}, \sigma_i^2)$ with $\boldsymbol{S} = \mathrm{Par}(Z) \cap \boldsymbol{\Gamma}$, $\boldsymbol{\theta} = \{\boldsymbol{\mu_0}, \boldsymbol{w}, \boldsymbol{\sigma}\}$
Softmax (CD CPD)		$\mathrm{P}(c_j \mid x, y) = \mathcal{S}(c_j; \boldsymbol{b}, \boldsymbol{w})$ with $\boldsymbol{\theta} = \{\boldsymbol{b}, \boldsymbol{w}\}$
Conditional Softmax (CCD CPD)		$\mathrm{P}(c_j \mid a_i, y) = \mathcal{S}(c_j; \boldsymbol{b}_i, \boldsymbol{w}_i)$ with $\boldsymbol{\theta} = \{\boldsymbol{b}, \boldsymbol{w}\}$

Figure 3.5: CPD types used depending on the cardinality of nodes.

$\int_z \mathcal{N}(Z; \mu, \sigma^2) = 1$. Consequently it is

$$\mathrm{P}(Z \mid \mathrm{Par}(Z) = \boldsymbol{a}) = \mathcal{N}(Z; \mu_{\boldsymbol{a}}, \sigma^2_{\boldsymbol{a}}). \tag{3.14}$$

According to (Lerner, 2002) the Gaussian distribution is a convenient choice as a CPD, because Gaussian distributions arise naturally in many real-world domains and the family of Gaussians is mathematically closed under operations such as summation, multiplication and conditioning. Two exemplary parameterizations of a Gaussian normal distribution are shown in Figure 3.6a. In (Murphy, 1999) a table of Gaussians is called a conditional Gaussian (CG) model.

For a continuous node with all continuous parents a so-called linear Gaussian (LG) model is employed (Koller, Friedman, 2009, pp. 186 sqq.). In a LG model the distribution of a continuous variable is a linear function of its continuous parents with Gaussian noise, which is

$$\mathrm{P}(Z \mid \mathrm{Par}(Z) = \boldsymbol{s}) = \mathcal{N}(Z; \underbrace{\mu_0 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{s}}_{\mu}, \sigma^2) \tag{3.15}$$

with $\boldsymbol{w}$ being the coefficient vector of the linear combination of the continuous parents' values $\boldsymbol{s}$. The variance σ does not depend on $\boldsymbol{s}$.

If a continuous node has both continuous and discrete parents, a table with a LG model for each state combination of the discrete parents can be specified. In (Koller, Friedman, 2009, p. 190) this is called a conditional linear Gaussian (CLG) model. If all discrete variables of a CLG are given, then the CPD of the continuous variable is reduced to a LG distribution.

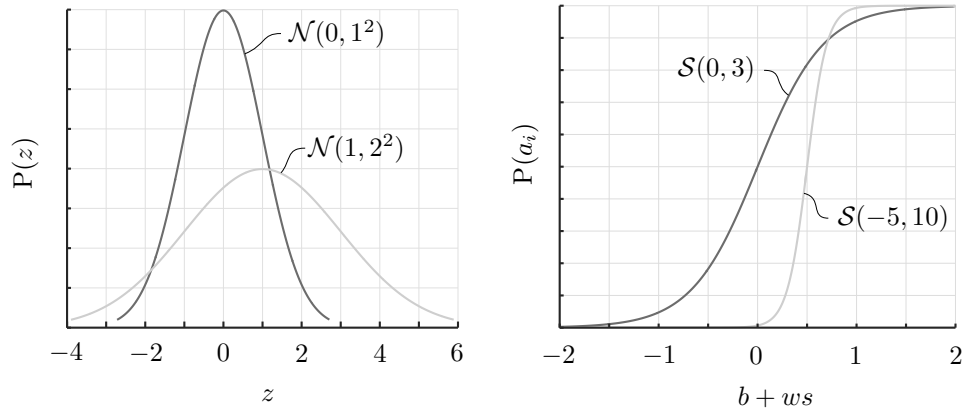
Note that for the case of no continuous parents (i.e., $\boldsymbol{S} = \emptyset$) the CLG CPD simplifies to a CG CPD, while for the case of no discrete parents one can think of the CLG CPD as a 1×1 table, which corresponds to a LG CPD. Therefore the CLG CPD is a generalization of both the LG and CG CPD.

When a discrete node depends on continuous parents, a logistic or softmax distribution (sometimes also called multinomial logistic regression) can be used which "soft thresholds" the node's state given the continuous parents. In case of a linear decision boundary hypothesis the probability of a discrete node C being in a certain state c_j given all values $\boldsymbol{s}$ of its continuous parents $\boldsymbol{S} = \mathrm{Par}(C) \cap \boldsymbol{\Gamma}$ is given by

$$\mathrm{P}(C = c_j \mid \mathrm{Par}(C) = \boldsymbol{s}) = \mathcal{S}(c_j; \boldsymbol{b}, \boldsymbol{w}) = \frac{\exp(b_j + \boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{s})}{\sum_{k=1}^{K} \exp(b_k + \boldsymbol{w}_k^{T}\boldsymbol{s})}, \tag{3.16}$$

where $\boldsymbol{w}_j$ is the normal vector to the j'th decision boundary, b_j is its offset and $K = |\,\mathrm{val}(C)|$. As it has to be $\sum_j \mathrm{P}(c_j \mid \mathrm{par}(C)) = 1$ there are $K - 1$ independent parameters $\boldsymbol{\theta} = \{\boldsymbol{b}, \boldsymbol{w}\}$, which is why often the K'th parameters are fixed ($b_K = 0$, $\boldsymbol{w}_K = \boldsymbol{0}$). Two exemplary parameterizations of a softmax distribution are shown in Figure 3.6b. In (Lerner et al., 2001) it is referred to these models as CD (continuous-discrete) CPD.

Again, if the discrete node also has discrete parents, a table with a softmax function for each state combination of the discrete parents can be specified. This will be called a conditional softmax or CCD CPD. A CCD CPD also generalizes a CD CPD, as for the case of no discrete parents the CCD CPD is simply a 1×1 table with a single CD CPD in it.



(a) Gauss normal distribution. **(b)** Softmax distribution.

Figure 3.6: Exemplary parameterizations of a Gauss normal and a softmax distribution.

All described CPD types are already implemented in the Bayes net toolbox (BNT) for MATLAB (MURPHY, 2001). For an efficient implementation additionally a so-called root CPD exists, which has no parameters and therefore can be used for nodes that are always observed and have no parents.

3.3 Inference in Bayesian Networks

Many different algorithms exist to run inference in a Bayesian network, which means entering evidence for nodes that can be observed and querying the probability of nodes' values that are of interest. This often means computing $\mathrm{P}(\boldsymbol{Q} \mid \boldsymbol{e})$ with a set of query variables $\boldsymbol{Q}$ and the observations or evidence $\boldsymbol{E} = \boldsymbol{e}$.
On a high level, existing inference algorithms can be divided into exact and approximate algorithms. While exact inference algorithms give an exact answer to a probabilistic query, approximate algorithms are used when exact inference is intractable. They are often based on stochastic sampling approaches, practically leading to a long time needed to converge while only giving an approximate answer to a probabilistic query. Therefore approximate inference methods are not suited for most real-time applications.
The majority of approaches to exact inference in BN found in the literature consider the case of inference in pure discrete Bayesian networks, for example the popular clique tree algorithm by (LAURITZEN, SPIEGELHALTER, 1988) which is based on variable elimination as described in (BERTELE, BRIOSCHI, 1973). However, many important real-world applications are both in the continuous *and* discrete domain, which also holds for the task of inferring the probability of a maneuver intention in the BN presented in subsection 3.2.1. While the LCI node as well as variables indicating the existence of vehicles or lanes are discrete in nature, most features characterizing the current traffic situation, e.g. the inverse time to collision or time headway, are continuous valued.
Indeed the continuous variables of the network could be converted to the discrete domain by discretization, but this compulsorily leads to a loss of information. Moreover, when

partitioning the variables' continuous domain into a large number of subsets to reduce the information loss during discretization, this unfortunately leads to an increasing complexity of exact inference, because the size of the CPT in the BN grows exponentially with the number of states of each variable.

Another criteria when choosing an inference algorithm is the temporal aspect of the BN. While the network presented in subsection 3.2.1 is a static BN that deals with static distributions, a model can also be defined as temporal BN to represent stochastic processes, which is called a dynamic Bayesian network (DBN). While in general every DBN can be converted to a static BN by unrolling it in time (making all inference algorithms applicable in theory), in practice the application of exact inference algorithms to DBN is often intractable because of the size of the unrolled network.

In this thesis the inference algorithm presented in (Lerner et al., 2001; Lerner, 2002) is used, which falls into the category of exact methods (although approximate marginalization is employed, which will be described later on). It is based on the clique tree algorithm, but (Lerner, 2002) proposes extensions which make it applicable to run inference in a static hybrid Bayesian network. The clique tree algorithm again is based on variable elimination, which will be described in the following before the clique tree algorithm for hybrid Bayesian networks is described in detail.

3.3.1 Variable Elimination

The variable elimination (VE) algorithm is an algorithm for exact inference in a Bayesian network (Bertele, Brioschi, 1973; Zhang, Poole, 1994). Its basic concept is to make use of the network's structure, which implicates conditional independence assertions and therefore allows for a factorization of the joint distribution over all variables in the network. This again facilitates to evaluate sub-expressions of the joint distribution depending only on a smaller number of variables first and to save the sub-expressions' results for later use. These intermediate results are also called *factors*, whereby a factor $\phi(\boldsymbol{X})$ over the scope $\boldsymbol{X}$ in general is defined to be a function $\phi : \text{val}(\boldsymbol{X}) \mapsto \mathbb{R}$ (Koller, Friedman, 2009, p. 104, p. 296).

The procedure of variable elimination corresponds to performing *dynamic programming* (Koller, Friedman, 2009, p. 296). To describe the basic concept of it subsequently without going too much into detail, it is assumed that all required operations on factors (which are their initialization, the extension/reduction of their scope, multiplication/division, and marginalization) can be performed even in hybrid BN, although originally they have only been defined for pure discrete BN. The representation of factors and a detailed description of the required operations will be given in subsection 3.3.3.

A probabilistic query on the network can be carried out by marginalization, which means integrating (or in case of a discrete domain summing) over all variables different from the query in the joint distribution. Here Σ is used as the marginalization operator. For the example of the network shown in Figure 3.4, if the marginal probability distribution of A

shall be computed, then this means computing

$$\mathrm{P}(A) = \sum_{b,c,w,x,y,z} \mathrm{P}(A, b, c, w, x, y, z). \tag{3.17}$$

According to the conditional independence given by the structure of the Bayesian network the chain rule (Equation 3.12) can be applied, which leads to

$$\mathrm{P}(A) = \sum_{b,c,w,x,y,z} \mathrm{P}(b)\,\mathrm{P}(c)\,\mathrm{P}(x)\,\mathrm{P}(y)\,\mathrm{P}(z \mid b, y)\,\mathrm{P}(w \mid c, x)\,\mathrm{P}(A \mid w, z). \tag{3.18}$$

In the following each of the terms in Equation 3.18 is named a factor ϕ, and it can be seen that inference in the given BN means multiplying seven factors together, resulting in a large factor over all seven variables A, B, C, W, X, Y, Z. In general, for any given BN this means that a factor over all variables in the BN is generated whose size is exponential in the number of variables. This makes this form of inference intractable in large BN (Koller, Friedman, 2009). But fortunately, by re-arranging the terms in Equation 3.18 the marginalizations can be pushed inside, which leads to

$$\mathrm{P}(A) = \sum_{w,z} \Big(\mathrm{P}(A \mid w, z) \sum_{c,x} \Big(\mathrm{P}(c)\,\mathrm{P}(x)\,\mathrm{P}(w \mid c, x)\Big) \sum_{b,y} \Big(\mathrm{P}(b)\,\mathrm{P}(y)\,\mathrm{P}(z \mid b, y)\Big)\Big). \tag{3.19}$$

Starting from the inside, at first $\mathrm{P}(b)$, $\mathrm{P}(y)$ and $\mathrm{P}(z \mid b, y)$ are represented as factors $\phi_b(b)$, $\phi_y(y)$ and $\phi_z(b, y, z)$. Multiplying these factors gives a new factor $\phi_1(b, y, z)$, which leads to

$$\mathrm{P}(A) = \sum_{w,z} \Big(\mathrm{P}(A \mid w, z) \sum_{c,x} \Big(\mathrm{P}(c)\,\mathrm{P}(x)\,\mathrm{P}(w \mid c, x)\Big) \sum_{b,y} \Big(\phi_1(b, y, z)\Big)\Big). \tag{3.20}$$

Performing the marginalization of B and Y generates a factor ϕ_2 which is independent of B and Y, so

$$\mathrm{P}(A) = \sum_{w,z} \Big(\mathrm{P}(A \mid w, z) \sum_{c,x} \Big(\mathrm{P}(c)\,\mathrm{P}(x)\,\mathrm{P}(w \mid c, x)\Big)\phi_2(z)\Big). \tag{3.21}$$

The eliminated variables B and Y will not appear in the following computations anymore. This early elimination of variables is the key to avoid the generation of exponentially large factors.

The same elimination can be done for C and X, so $\mathrm{P}(c)$, $\mathrm{P}(x)$ and $\mathrm{P}(w \mid c, x)$ are first represented as factors $\phi_c(c)$, $\phi_x(x)$ and $\phi_w(c, w, x)$ and next multiplied together, resulting in a factor $\phi_3(c, w, x)$. With ϕ_3 the marginal probability of A can be written as

$$\mathrm{P}(A) = \sum_{w,z} \Big(\mathrm{P}(A \mid w, z) \sum_{c,x} \Big(\phi_3(c, w, x)\Big)\phi_2(z)\Big). \tag{3.22}$$

The marginalization of ϕ_3 eliminates C and X, thus leading to a factor $\phi_4(w)$, so

$$\mathrm{P}(A) = \sum_{w,z} \Big(\mathrm{P}(A \mid w, z)\phi_4(w)\phi_2(z)\Big). \tag{3.23}$$

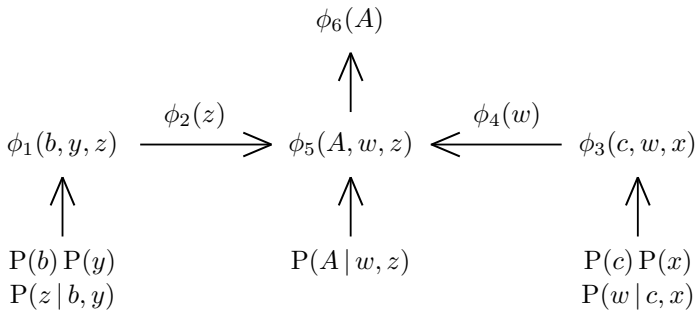


Figure 3.7: Illustration of the variable elimination algorithm for the BN shown in Figure 3.4.

Continuing the process yields $\phi_5(a, w, z) = \mathrm{P}(A \mid w, z)\phi_4(w)\phi_2(z)$, which after marginalizing W and Z finally leads to the queried marginal probability $\mathrm{P}(A)$. To summarize, the following computations have been carried out:

$$\phi_1(b, y, z) = \mathrm{P}(b)\,\mathrm{P}(y)\,\mathrm{P}(z \mid b, y) \tag{3.24}$$
$$\phi_2(z) = \sum_{b,y} \phi_1(b, y, z) \tag{3.25}$$
$$\phi_3(c, w, x) = \mathrm{P}(c)\,\mathrm{P}(x)\,\mathrm{P}(w \mid c, x) \tag{3.26}$$
$$\phi_4(w) = \sum_{c,x} \phi_3(c, w, x) \tag{3.27}$$
$$\phi_5(A, w, z) = \mathrm{P}(A \mid w, z)\phi_4(w)\phi_2(z) \tag{3.28}$$
$$\phi_6(A) = \sum_{w,z} \phi_5(A, w, z) \tag{3.29}$$

This elimination process (which is shown graphically in Figure 3.7) allowed to do inference over the joint distribution without ever generating it explicitly. Compared to the factor over all seven variables in Equation 3.18, the early elimination of variables led to factors of maximally three variables, which reduces the effort of the marginalization process and therefore minimizes the complexity of the variable elimination algorithm. But in general the size of the factors depends on the order in which the variables are eliminated. Although some simple greedy heuristics exist to choose an effective elimination order, finding the optimal order is NP-hard (Kjaerulff, 1990; Koller, Friedman, 2009). Moreover, depending on the network structure even the optimal elimination order can lead to factors which are exponential in the size of the BN, in which case exact inference can become intractable. But fortunately, in many real-world applications an effective elimination order can be found.

Another downside of the variable elimination algorithm is that all presented computations that are needed to find the marginal probability of A need to be carried out again for the inference of any other queried variable. To overcome this the clique tree algorithm (which is based on variable elimination) has been invented. It will be described in the next section.

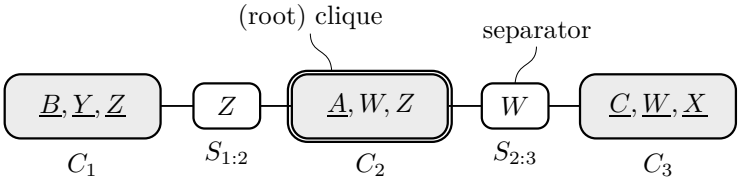


Figure 3.8: Clique tree induced by the variable elimination illustrated in Figure 3.7.

3.3.2 Clique Tree Algorithm

The algorithm most frequently used for exact inference in the literature is the clique tree algorithm (also known as junction tree, cluster tree or join tree algorithm) (Lauritzen, Spiegelhalter, 1988). It basically is a variant of the variable elimination algorithm, but it is more efficient when computing more than just one marginal probability of the network's variables. More precisely, using the clique tree algorithm multiple marginals can be computed while performing at most twice as many computations as are needed for the marginalization of one variable using the variable elimination algorithm (Koller, Friedman, 2009, p. 358). This does not only reduce the computational complexity when querying variables during inference, but also during training when using the expectation maximization algorithm, which will be discussed in section 5.3.
The clique tree algorithm is based on a graph representation. This graph can be seen as being induced by the variable elimination algorithm. Each node in the graph represents a set of variables involved in a factor multiplication. This set of variables is called a *cluster* or *clique*, denoted by C, and the resulting factor that gets associated to the clique is called a *potential*, denoted by Φ_C. An edge in the graph corresponds to the set of variables remaining after marginalizing out variables from a factor, which are the variables that the two cliques that get connected by the edge have in common. An edge in the clique tree is also called *separator* and is denoted by S, the factor of the edge is denoted by Φ_S and called a *sepset*. The weight of a clique is defined to be the number of values that it can take on, which is the product of the cardinalities of all nodes being entailed. Thereby the cardinality of a continuous variable is defined to be 1.
The graph is called a *cluster graph*, and it has some important properties:

- **Tree structure:** The graph has the structure of a tree, meaning that it has no cycles and every path from a clique C_i to C_j is unique. It is therefore also called a *clique tree*.
- **Family preservation property:** The family preservation ensures that each family of a node X in the BN is a subset of the scope of some clique C_i in the clique tree, i.e., $\text{Fam}(X) \subseteq C_i$.
- **Running intersection property:** The running intersection property (RIP) guarantees that when a variable X is entailed in the cliques C_i as well as C_j, then it is also entailed in all cliques in the (unique) path from C_i to C_j. On the other hand, if it is entailed in C_i but not in C_j, then it will also not be entailed in any clique subsequent to C_j.

An example of a clique tree induced from the variable elimination procedure of Figure 3.7 is shown in Figure 3.8. But the clique tree algorithm also includes a procedure to derive the graph from the structure of a BN directly. This will be described in subsection 3.3.4. Again, since the representation of factors (the clique potentials and sepsets) is not yet introduced, for the subsequent description of the general inference process in a clique tree it is assumed that all required operations on factors (initialization, multiplication/division, scope extension/reduction, and marginalization) are well defined.
After constructing the clique tree the clique potentials get initialized with the product of the clique nodes' CPD, but each CPD must only be incorporated into the tree once. This is why every variable of the BN gets associated to exactly one clique that also entails the respective node's parents (in case of multiple qualified cliques the one with the lowest effect in terms of absolute size or increased computational complexity is chosen). The variable's assignments are indicated in the clique tree by underlining the variables associated to the respective clique. For example, in Figure 3.8 the variable Z is entailed in C_1 as well as C_2, but it gets associated to C_1 because it is the only clique that also entails the parents of Z. Moreover, (Murphy, 1999) suggests to only create the initial clique potentials after incorporating any evidence into the respective CPD, which is more efficient and lets observed nodes only contribute a constant factor (namely the variable's observed outcome) to the value of the potential. The clique potentials therefore only have to be defined on the hidden nodes.
The process of marginalizing potentials to obtain the sepsets and multiplying the sepsets onto neighboring potentials in the clique tree is called *message passing*. First, messages (i.e., factors) get passed from the leaves up to the root, which is also called the *collect evidence* phase or *upward pass*. Thereby the *root* is (at least for now) the clique that entails the queried variable in the variable elimination algorithm, which was A when describing the VE process for the exemplary Bayesian network in Figure 3.7. Consequently, here C_2 is chosen to be the root of the clique tree and the upward pass corresponds to sending a message from C_1 to C_2 as well as from C_3 to C_2. After the upward pass, which is analog to the computations in the variable elimination algorithm, the clique C_2 has the correct joint distribution over the variables of its scope and A can be marginalized from its potential.
However, the main advantage of the clique tree algorithm is, that after executing the collect evidence phase a second message passing run can be executed, this time from the root to the leaves (here: from C_2 to C_1 and from C_2 to C_3). This is also called the *distribute evidence* phase or *downward pass*. After each leaf has received a message from the root the tree is *calibrated*, meaning that all cliques agree on the same marginal distribution of any variable they share (Koller, Friedman, 2009, pp. 355 sqq.). Running message passing again would not change the clique distributions anymore. This reveals the effectiveness of the clique tree algorithm: After carrying out at most twice as many computations as in a single run of variable elimination the marginal probability of any variable can be computed from any clique with the variable in its scope.
Finally an important question has to be answered: How are the factors (i.e., the clique potentials and sepsets) represented in a hybrid Bayesian network so that all the operations can actually be performed?

3.3.3 Representation of Factors

For the ability to run message passing in the clique tree, all potentials and sepsets need to be represented using the same type of probability distribution. Therefore the type of distribution that can be used for the representation of factors in a hybrid BN is the least common ancestor in the type hierarchy of all nodes whose CPD get incorporated into the tree during initialization. These are all hidden nodes of the network if the evidence is taken into account before initialization, as suggested by (Murphy, 1999). For example, if all hidden nodes are in the discrete domain, the type of distribution for the potentials is a discrete probability table, also called a discrete factor. If all hidden nodes are represented by a Gaussian distribution, then the clique potentials can be represented as Gaussian factors. However, if there are both discrete and continuous hidden nodes, then the potential representing the joint distribution over the involved nodes needs to be a table of Gaussians with one entry for each instantiation of the discrete variables, which is called a conditional Gaussian factor.

Discrete Factors

A discrete factor over the all discrete variables $\boldsymbol{D}$ can be represented as a probability table with as many entries as there exist state combinations (also called instantiations) of all discrete variables, which is denoted by $|\operatorname{val}(\boldsymbol{D})|$. In the following the required operations on discrete factors are described according to (Murphy, 1999; Koller, Friedman, 2009).

Initialization

A discrete factor can be initialized as "empty" by setting all its entries to 1. Multiplying or dividing by such a factor has no effect.

Initializing a discrete factor from a node's CPT is achieved simply by inheriting the parameters from the CPT.

Multiplication and Division

The product (or quotient) of two discrete factors is computed by multiplying (dividing) all entries with matching instantiations. Thereby it is defined $0 \,/\, 0 = 0$. The result of the operation is a factor whose scope is extended implicitly. For example, the multiplication of two factors $\phi_1(A, B)$ and $\phi_2(B, C)$ results in a factor $\phi_3(A, B, C)$, see Figure 3.9.

A	B	$\mathrm{P}(A,B)$
a_1	b_1	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,1,1}$
a_1	b_2	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,1,2}$
a_2	b_1	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,2,1}$
a_2	b_2	$\underbrace{\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,2,2}}_{\boldsymbol{\theta}_{\boldsymbol{\phi_1}}}$

B	C	$\mathrm{P}(B,C)$
b_1	c_1	$\boldsymbol{\theta}_{\boldsymbol{\phi_2}\,1,1}$
b_1	c_2	$\boldsymbol{\theta}_{\boldsymbol{\phi_2}\,1,2}$
b_2	c_1	$\boldsymbol{\theta}_{\boldsymbol{\phi_2}\,2,1}$
b_2	c_2	$\underbrace{\boldsymbol{\theta}_{\boldsymbol{\phi_2}\,2,2}}_{\boldsymbol{\theta}_{\boldsymbol{\phi_2}}}$

$=$

A	B	C	$\mathrm{P}(A,B,C)$
a_1	b_1	c_1	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,1,1} \cdot \boldsymbol{\theta}_{\boldsymbol{\phi_2}\,1,1}$
a_1	b_1	c_2	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,1,1} \cdot \boldsymbol{\theta}_{\boldsymbol{\phi_2}\,1,2}$
a_1	b_2	c_1	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,1,2} \cdot \boldsymbol{\theta}_{\boldsymbol{\phi_2}\,2,1}$
a_1	b_2	c_2	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,1,2} \cdot \boldsymbol{\theta}_{\boldsymbol{\phi_2}\,2,2}$
a_2	b_1	c_1	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,2,1} \cdot \boldsymbol{\theta}_{\boldsymbol{\phi_2}\,1,1}$
a_2	b_1	c_2	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,2,1} \cdot \boldsymbol{\theta}_{\boldsymbol{\phi_2}\,1,2}$
a_2	b_2	c_1	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,2,2} \cdot \boldsymbol{\theta}_{\boldsymbol{\phi_2}\,2,1}$
a_2	b_2	c_2	$\boldsymbol{\theta}_{\boldsymbol{\phi_1}\,2,2} \cdot \boldsymbol{\theta}_{\boldsymbol{\phi_2}\,2,2}$

Figure 3.9: Discrete factor multiplication.

Scope Extension

Because the scope of a factor is extended implicitly when multiplying two factors over an intersecting set of variables, there is in general no need to extend the scope of a discrete factor explicitly. Nevertheless, extending the scope would be possible by multiplication with an "empty" factor whose scope is the union of the original scope and the scope by which the factor shall be extended. This corresponds to duplicating the table entries for the instantiations of the added variables.

Incorporating Evidence

When entering evidence $E = e$ into an existing discrete factor, all entries of the table being inconsistent with the evidence get eliminated and the remaining probabilities are renormalized to fulfill the requirement of the joint probability summing to 1. For example, suppose the factor $\phi(A, B)$ represents a probability distribution over the binary variables A and B. When entering the evidence $B = b_2$, all table entries with $B \neq b_2$ are eliminated and the remaining entries in the resulting factor $\phi'(A)$ are divided by the sum of the remaining probabilities. This process is illustrated in Figure 3.10.

It can be seen that the size of the discrete factor is reduced after incorporating the evidence. To avoid that large factors are created only to be reduced again as soon as evidence is incorporated, (Murphy, 1999) suggests to already account for any evidence while initializing the factor (as already mentioned before). This way the discrete factor only takes the value of the observed outcome.

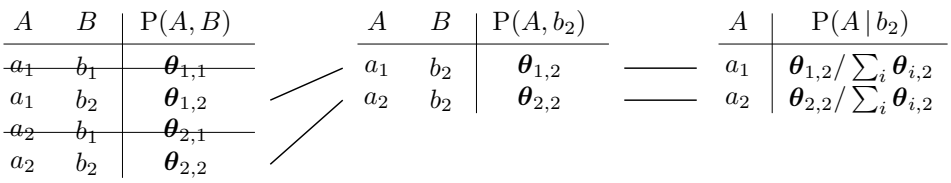


Figure 3.10: Discrete factor reduction due to incorporating evidence.

Marginalization

Marginalizing a variable A from a factor over the discrete variables A, B is defined as

$$\phi(B) = \sum_{a} \phi(A, B), \tag{3.30}$$

which sometimes is also called *summing out* a variable from a factor, because it corresponds to the summation of all entries with a matching instantiation of the variables that shall remain in the factor. In other words, it is summed over all entries of the table for that only the variable to be marginalized has a different value. For the example of marginalizing A from a factor $\phi(A, B)$ of the binary variables A and B this means summing all entries with the same value for B, namely b_1 and b_2. This is illustrated in Figure 3.11.

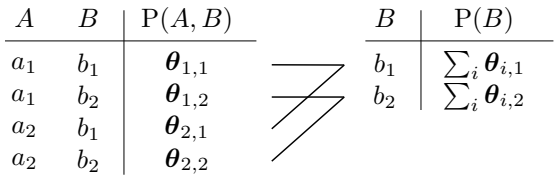


Figure 3.11: Discrete factor marginalization.

Gaussian Factors

If all hidden nodes are represented by a Gaussian distribution, then the clique potentials can also be represented as Gaussians. In this case a multivariate Gaussian normal distribution as a generalization of the univariate Gaussian normal distribution (see Equation 3.13) is defined as

$$\mathrm{P}(\boldsymbol{X}) = \mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right), \tag{3.31}$$

whereby n is the dimension of $\boldsymbol{X}$, $\boldsymbol{\mu}$ is the mean vector of size n with $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{X}]$ and $\boldsymbol{\Sigma} = \mathbb{E}[\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}] - \mathbb{E}[\boldsymbol{X}]\,\mathbb{E}[\boldsymbol{X}]^{\mathrm{T}}$ is the symmetric positive-definite covariance matrix of size $n \times n$.

Following (Koller, Friedman, 2009, p. 609), reformulating the moment form of the Gaussian in Equation 3.31 to

$$\mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \exp\Bigg(\underbrace{-\tfrac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \log\left((2\pi)^{n/2}\det(\boldsymbol{\Sigma})^{1/2}\right)}_{g} + \underbrace{\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}}_{\boldsymbol{h}^{\mathrm{T}}}\boldsymbol{x} - \tfrac{1}{2}\boldsymbol{x}^{\mathrm{T}}\underbrace{\boldsymbol{\Sigma}^{-1}}_{\boldsymbol{K}}\boldsymbol{x}\Bigg) \tag{3.32}$$

leads to the representation of the normal distribution in the canonical form (also called quadratic or natural form) with

$$\mathcal{C}(\boldsymbol{X}; \boldsymbol{K}, \boldsymbol{h}, g) = \exp\left(-\tfrac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{x} + \boldsymbol{h}^{\mathrm{T}}\boldsymbol{x} + g\right). \tag{3.33}$$

The canonical form can be converted to moment form (and vice versa) iff $\boldsymbol{K}$ is of full rank. This implies that the canonical form is more general than Gaussians: The canonical form is well defined even if $\boldsymbol{K}$ is not invertible, but it is not the inverse of a legal covariance matrix (Koller, Friedman, 2009, p. 609). In the following the operations on canonical forms as presented in (Koller, Friedman, 2009, pp. 610 sq.) are summarized.

Initialization

A canonical form can be initialized as "empty" factor by choosing $\boldsymbol{K} = \boldsymbol{0}$, $\boldsymbol{h} = \boldsymbol{0}$, $g = 0$. Multiplying or dividing by such a canonical factor has no effect.

To initialize a canonical factor from a node's LG CPD (see Equation 3.15) the following

conversion can be applied (see (MURPHY, 1999)):

$$\boldsymbol{K} = \frac{1}{\sigma}\begin{bmatrix}\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}} & -\boldsymbol{w}\\ -\boldsymbol{w}^{\mathrm{T}} & 1\end{bmatrix} \tag{3.34a}$$

$$\boldsymbol{h} = \frac{\mu}{\sigma^2}\begin{bmatrix}-\boldsymbol{w}\\ 1\end{bmatrix} \tag{3.34b}$$

$$g = \frac{-\mu^2}{2\sigma^2} - \tfrac{1}{2}\log(2\pi\sigma^2). \tag{3.34c}$$

As $\boldsymbol{K}$ has not full rank it follows that it may not be possible to represent this initial canonical factor as a Gaussian in moment form. But when multiplying the factor onto a clique potential and running message passing in the clique tree the potential represents a joint probability distribution, which according to (MURPHY, 1999) can always be converted to moment form.

Multiplication and Division

Two canonical forms with the same scope can be multiplied (divided) easily by summing (subtracting) its parameters individually:

$$\mathcal{C}(\boldsymbol{K}_1, \boldsymbol{h}_1, g_1) \cdot \mathcal{C}(\boldsymbol{K}_2, \boldsymbol{h}_2, g_2) = \mathcal{C}(\boldsymbol{K}_1 + \boldsymbol{K}_2, \boldsymbol{h}_1 + \boldsymbol{h}_2, g_1 + g_2) \tag{3.35}$$

$$\mathcal{C}(\boldsymbol{K}_1, \boldsymbol{h}_1, g_1) \,/\, \mathcal{C}(\boldsymbol{K}_2, \boldsymbol{h}_2, g_2) = \mathcal{C}(\boldsymbol{K}_1 - \boldsymbol{K}_2, \boldsymbol{h}_1 - \boldsymbol{h}_2, g_1 - g_2) \tag{3.36}$$

Scope Extension

A canonical form can be extended to a superset of the variables it has been defined for originally by simply increasing the dimensions of $\boldsymbol{K}$ and $\boldsymbol{h}$ and initializing the extra entries with zeros. For example, if two factors $\phi_1(X,Y) = \mathcal{C}_1(X,Y;\boldsymbol{K_1},\boldsymbol{h_1},g_1)$ and $\phi_2(Y,Z) = \mathcal{C}_2(Y,Z;\boldsymbol{K_2},\boldsymbol{h_2},g_2)$ shall be multiplied, then their scope first has to be extended to the same variables. For the canonical forms

$$\mathcal{C}_1(X,Y;\begin{bmatrix}K_{1,XX} & K_{1,XY}\\ K_{1,YX} & K_{1,YY}\end{bmatrix},\begin{bmatrix}h_{1,X}\\ h_{1,Y}\end{bmatrix},g_1) \quad \text{and}$$

$$\mathcal{C}_2(Y,Z;\begin{bmatrix}K_{2,YY} & K_{2,YZ}\\ K_{2,ZY} & K_{2,ZZ}\end{bmatrix},\begin{bmatrix}h_{2,Y}\\ h_{2,Z}\end{bmatrix},g_2)$$

this means that they both have to be defined over X, Y and Z, which leads to

$$\mathcal{C}_1(X,Y,Z;\begin{bmatrix}K_{1,XX} & K_{1,XY} & 0\\ K_{1,YX} & K_{1,YY} & 0\\ 0 & 0 & 0\end{bmatrix},\begin{bmatrix}h_{1,X}\\ h_{1,Y}\\ 0\end{bmatrix},g_1) \quad \text{and}$$

$$\mathcal{C}_2(X,Y,Z;\begin{bmatrix}0 & 0 & 0\\ 0 & K_{2,YY} & K_{2,YZ}\\ 0 & K_{2,ZY} & K_{2,ZZ}\end{bmatrix},\begin{bmatrix}0\\ h_{2,Y}\\ h_{2,Z}\end{bmatrix},g_2).$$

The result of the multiplication is then a factor $\phi_3(X,Y,Z)$ with

$$\mathcal{C}_3(X,Y,Z;\begin{bmatrix}K_{1,XX} & K_{1,XY} & 0\\ K_{1,YX} & K_{1,YY}+K_{2,YY} & K_{2,YZ}\\ 0 & K_{2,ZY} & K_{2,ZZ}\end{bmatrix},\begin{bmatrix}h_{1,X}\\ h_{1,Y}+h_{2,Y}\\ h_{2,Z}\end{bmatrix},g_1+g_2).$$

Incorporating Evidence

Entering evidence $\boldsymbol{Y} = \boldsymbol{y}$ in an existing Gaussian factor $\mathcal{C}(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{K}, \boldsymbol{h}, g)$ over the scope $\{\boldsymbol{X}, \boldsymbol{Y}\}$, where

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{K_{XX}} & \boldsymbol{K_{XY}} \\ \boldsymbol{K_{YX}} & \boldsymbol{K_{YY}} \end{bmatrix} \quad \text{and} \quad \boldsymbol{h} = \begin{bmatrix} \boldsymbol{h_X} \\ \boldsymbol{h_Y} \end{bmatrix}, \tag{3.37}$$

results in $\mathcal{C}(\boldsymbol{X}; \boldsymbol{K}', \boldsymbol{h}', g')$ with

$$\boldsymbol{K}' = \boldsymbol{K_{XX}} \tag{3.38a}$$

$$\boldsymbol{h}' = \boldsymbol{h_X} - \boldsymbol{K_{XY}}\boldsymbol{y} \tag{3.38b}$$

$$g' = g + \boldsymbol{h_Y}^{\mathrm{T}}\boldsymbol{y} - \tfrac{1}{2}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K_{YY}}\boldsymbol{y}. \tag{3.38c}$$

But as mentioned before, following (Murphy, 1999) it is most efficient to initialize a factor with evidence instead of incorporating evidence after the factor has been created or even been multiplied onto a clique potential already. When evidence $E = e$ is available for the node whose CPD is converted to a canonical factor, then the canonical form simply takes the values $\boldsymbol{K} = \boldsymbol{0}$, $\boldsymbol{h} = \boldsymbol{0}$, $g = \exp(e)$.

Marginalization

In general, marginalizing a variable X from a factor over the continuous variables $X, \boldsymbol{Y}$ is defined as

$$\phi(\boldsymbol{Y}) = \int_x \phi(X, \boldsymbol{Y}). \tag{3.39}$$

But unfortunately the marginalization operation is not always defined on canonical forms. Let $\mathcal{C}(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{K}, \boldsymbol{h}, g)$ be a canonical form over $\{\boldsymbol{X}, \boldsymbol{Y}\}$ according to Equation 3.37, then the marginalization $\int_{\boldsymbol{Y}} \mathcal{C}(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{K}, \boldsymbol{h}, g)$ is only valid if $\boldsymbol{K_{YY}}$ is positive definite. If it is, then the result of the marginalization is a canonical form $\mathcal{C}(\boldsymbol{X}; \boldsymbol{K}', \boldsymbol{h}', g')$ with

$$\boldsymbol{K}' = \boldsymbol{K_{XX}} - \boldsymbol{K_{XY}}\boldsymbol{K_{YY}}^{-1}\boldsymbol{K_{YX}} \tag{3.40a}$$

$$\boldsymbol{h}' = \boldsymbol{h_X} - \boldsymbol{K_{XY}}\boldsymbol{K_{YY}}^{-1}\boldsymbol{h_Y} \tag{3.40b}$$

$$g' = g + \tfrac{1}{2}\left(\log|2\pi\boldsymbol{K_{YY}}^{-1}| + \boldsymbol{h_Y}^{\mathrm{T}}\boldsymbol{K_{YY}}^{-1}\boldsymbol{h_Y}\right). \tag{3.40c}$$

But (Lerner, 2002, p. 57) shows that during message passing in a clique tree the required marginalization operations can always be executed: The need to marginalize comes up when sending a message from C_i (involving the variables $\boldsymbol{X}_i$) to clique C_j (containing the variables $\boldsymbol{X}_j$), so to obtain the sepset the clique potential $\Phi_{C_i}(\boldsymbol{X}_i)$ needs to be marginalized over the variables $\boldsymbol{Y} = \boldsymbol{X}_i - \boldsymbol{X}_j$. Due to the sequential message passing along the edges of the clique tree, when sending a message to C_j the clique C_i has already received messages from all its neighbors (except for C_j), thus according to the running intersection property of the clique tree all the factors which contain $\boldsymbol{Y}$ in their domain were already multiplied onto C_i. Let $\mathcal{C}(\boldsymbol{X_i}; \boldsymbol{K}, \boldsymbol{h}, g)$ denote the canonical form of the clique potential C_i just before sending the message to C_j, and let $\mathcal{C}(\boldsymbol{X_i} \setminus \boldsymbol{Y}; \boldsymbol{K}, \boldsymbol{h}, g)$ denote the correct marginal distribution over C_i, then it holds $\boldsymbol{K_{YY}} = \boldsymbol{K'_{YY}}$, which means the submatrices of $\boldsymbol{K}$ and $\boldsymbol{K}'$ for $\boldsymbol{Y}$ are the same. Since the marginal distribution of C_i is a Gaussian, $\boldsymbol{K}'$ is positive definite, which implies that also $\boldsymbol{K'_{YY}}$ and consequently $\boldsymbol{K_{YY}}$ are positive definite, so the marginalization operation is possible.

Conditional Gaussian Factors

For the task of inference in a hybrid Bayesian network with both hidden discrete and hidden continuous variables, the rich representation of factors as table of canonical forms is employed. This so-called conditional Gaussian (CG) representation can be seen as the parent of tabular and Gaussian factors in the type hierarchy of factor representations (Murphy, 1999). A CG factor can be employed even without the existence of discrete variables (i.e., $\boldsymbol{D} \subseteq \boldsymbol{\Delta} = \emptyset$); it then represents only a single canonical form over the continuous variables. On the other hand, if the network is purely discrete ($\boldsymbol{X} \subseteq \boldsymbol{\Gamma} = \emptyset$), then the parameters $\boldsymbol{K}_i$ and $\boldsymbol{h}_i$ of every discrete variables' instantiation are equal to zero and the factor takes the form of $\exp(g_i)$ for every $\phi(\boldsymbol{D} = \boldsymbol{d}_i)$.

Following (Murphy, 1999; Lerner, 2002; Koller, Friedman, 2009), the subsequent definitions of all required operations on factors show, that CG factors are a natural generalization of the operations defined on both tabular and Gaussian factors, but there are two exceptions: the marginalization of discrete variables and the incorporation of (C)CD CPD.

Initialization

Initializing an "empty" conditional Gaussian factor is achieved by initializing an "empty" Gaussian factor in canonical form ($\boldsymbol{K} = \boldsymbol{0}$, $\boldsymbol{h} = \boldsymbol{0}$ and $g = 0$, like described before) for every instantiation of the discrete variables in the factor's scope.

The initialization of a conditional Gaussian factor from a CPD is defined as initializing a canonical Gaussian factor for every state combination of the discrete nodes in the scope of the factor with respect to the CPD it is initialized from. In particular, a CLG CPD is converted to a conditional Gaussian factor by initializing each table entry of the factor from the corresponding LG model of the CPD analogously to the initialization of a Gaussian factor from a CPD. But there is one important exception: A conditional Gaussian factor can not directly be initialized from a CCD CPD when no evidence is given for the respective node itself or for its parents' values.

When a clique contains a discrete, unobserved node A with n continuous parents $\boldsymbol{X}$ and no evidence for $\mathrm{Par}(A)$ is available, then initializing the clique's potential entails the multiplication of a CD CPD (i.e., a softmax function) with a Gaussian, which is not defined in closed form. But following (Lerner, 2002, pp. 145 sq.) the product of a softmax and a Gaussian can be approximated as a Gaussian by

$$\begin{aligned}
\boldsymbol{\mu}_a = \mathbb{E}[\boldsymbol{X} \mid A = a] &= \int_{-\infty}^{\infty} \boldsymbol{x}\, \mathrm{P}(\boldsymbol{x} \mid A = a)\, \mathrm{d}\boldsymbol{x} \\
&= \frac{1}{\mathrm{P}(A = a)} \int_{-\infty}^{\infty} \boldsymbol{x}\, \mathrm{P}(A = a \mid \boldsymbol{x})\, \mathrm{P}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \qquad (3.41a) \\
\boldsymbol{\Sigma}_a = \mathbb{E}[\boldsymbol{X}^2 \mid A = a] &= \int_{-\infty}^{\infty} \boldsymbol{x}^2\, \mathrm{P}(\boldsymbol{x} \mid A = a)\, \mathrm{d}\boldsymbol{x} \\
&= \frac{1}{\mathrm{P}(A = a)} \int_{-\infty}^{\infty} \boldsymbol{x}^2\, \mathrm{P}(A = a \mid \boldsymbol{x})\, \mathrm{P}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \qquad (3.41b)
\end{aligned}$$

with the marginal distribution of A as

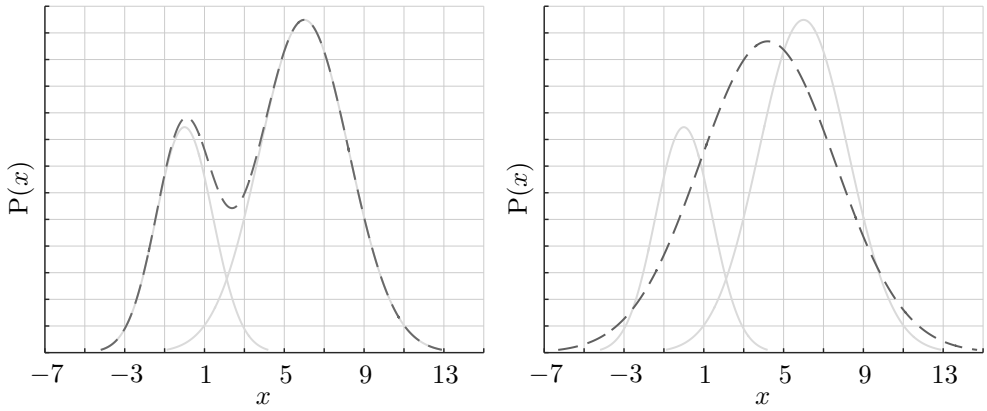
$$\mathrm{P}(A = a) = \int_{-\infty}^{\infty} \mathrm{P}(A = a \mid \boldsymbol{x}) \, \mathrm{P}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}. \tag{3.42}$$

This approximation is exact in terms of the first two statistical moments, but there are two difficulties when evaluating Equation 3.41a and 3.41b: First, during the initialization of the clique potentials the factors do not represent integrable distributions, i.e., the marginal distribution of X can not be obtained yet. The marginal can only be computed when the clique is calibrated.
To overcome this, (Lerner, 2002) proposes to run a pre-calibration of the clique tree without the incorporation of any (C)CD CPD first. In particular, it is suggested to multiply all evidences for observed variables and all CPD of hidden variables onto the cliques they were assigned to (except for the softmax node) and to run the up- and down-pass of the calibration procedure.
After all cliques containing discrete nodes with continuous or hybrid parents have been initially calibrated the (C)CD CPD are multiplied onto their corresponding clique potentials using Equation 3.41. Finally, a complete up- and down-pass of the message passing procedure has to be run to conclude the clique tree calibration.
Unfortunately, although after the pre-calibration phase the cliques have integrable distributions, (Lerner, 2002) emphasizes that in a hybrid BN the marginal distributions that can be obtained from the cliques may be inaccurate. This is due to the fact that conditional Gaussian factors are not closed under the operation of marginalizing discrete variables, which may nevertheless be a required operation during message passing in the pre-calibration phase. The details and a solution to avoid these inaccuracies are described in the section regarding the marginalization operation on conditional Gaussian factors later on.
The second difficulty when approximating the product of a softmax and a Gaussian is that the n-dimensional integrals in Equation 3.41a and 3.41b can not be solved in closed form. Therefore (Lerner, 2002, pp. 118 sqq.) proposes to use a numerical integration procedure like Gaussian quadrature. The Gaussian-Hermite quadrature method can be used to approximate integrals of the form $\int_a^b W(x) f(x) \, \mathrm{d}x$ where $W(x)$ is a nonnegative function (here the Gaussian). The approximated solution of the integral is computed as a weighted sum of n evaluations of the function, so

$$\int_a^b W(x) f(x) \, \mathrm{d}x \approx \sum_{j=1}^{n} w_j f(x_j). \tag{3.43}$$

Multiplication and Division

Conditional Gaussian factors can be multiplied or divided by multiplying or dividing each corresponding instantiation of the discrete variables. For example, the multiplication (or division) of the two conditional Gaussian factors $\phi_1(A, B, X, Y)$ and $\phi_2(B, C, Y, Z)$ leads to a factor $\phi_3(A, B, C, X, Y, Z)$, whereby ϕ_3 has a canonical form for each instantiation of the discrete variables A, B and C. The multiplication/division operation is performed between the corresponding table entries, e.g. for a particular instantiation a, b, c the canonical form is derived as a product of $\phi_1(a, b)$ and $\phi_2(b, c)$ (after extending their scopes respectively).



(a) Mixture of Gaussians. **(b)** Collapsed Gaussian.

Figure 3.12: Comparison of a Gaussian mixture distribution and a collapsed Gaussian distribution resulting from marginalization of a discrete binary variable in a conditional Gaussian potential with $\langle 0.3, \mathcal{N}(0, 2)\rangle$ and $\langle 0.7, \mathcal{N}(6, 5)\rangle$.

Scope Extension

The domain of a conditional Gaussian factor can be extended by adding discrete variables just like in tabular factors and continuous variables in every instantiation of the discrete variables just like in Gaussian factors.

Incorporating Evidence

When evidence $\boldsymbol{d}$ on discrete observations arrives, a canonical table can be reduced by instantiating the observations $\boldsymbol{d}$, which means setting all entries in the table that are not consistent with $\boldsymbol{d}$ to zero. Evidence on continuous values $\boldsymbol{x}$ is incorporated according to Equation 3.38. But again, it is preferred to already take evidence e into account when initializing the factor, in which case it takes the form $\boldsymbol{K} = \boldsymbol{0}$, $\boldsymbol{h} = \boldsymbol{0}$, $g = \exp(e)$.

Marginalization

For the marginalization of variables in a table of canonical forms it has to be distinguished between marginalizing discrete and continuous variables. For marginalizing continuous variables the operations stated in Equation 3.40 can be applied straight forward for every instantiation of the discrete variables, i.e., for each entry in the table of canonical forms. However, the family of canonical tables is not closed under discrete marginalization. One way to marginalize discrete variables while maintaining the correct distribution over the remaining variables is to enrich the conditional Gaussian forms to be mixtures of Gaussians. Such a mixture of Gaussians over a set of discrete and continuous variables $\{\boldsymbol{D}, \boldsymbol{X}\}$ can be defined as

$$\mathrm{P}(\boldsymbol{D}, \boldsymbol{X}) = \langle \boldsymbol{w}, \mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\rangle = \sum_{i=1}^{K} \boldsymbol{w}_i \mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{3.44}$$

with $K = |\operatorname{val}(\boldsymbol{D})|$ and $\boldsymbol{w}_i \propto \mathrm{P}(\boldsymbol{D} = \boldsymbol{d}_i)$ being the weight with whom the i'th normal distribution contributes to the mixture. An example of the resulting distribution over the continuous variable X when marginalizing a single discrete binary variable is shown in Figure 3.12a.

Unfortunately, it can be seen that the marginalized variables in fact are not eliminated from the factor but rather they induce more components into the mixture of canonical forms. This means that exponentially large factors are generated, again making the inference process intractable. To overcome this, the resulting distribution can be approximated using a single Gaussian by collapsing the Gaussian mixtures with

$$\boldsymbol{\mu} = \mathbb{E}[X] \ = \sum_{i=1}^{K} \boldsymbol{w}_i \boldsymbol{\mu}_i, \tag{3.45a}$$

$$\boldsymbol{\Sigma} = \mathbb{E}[X^2] = \sum_{i=1}^{K} \boldsymbol{w}_i \boldsymbol{\Sigma}_i + \sum_{i=1}^{K} \boldsymbol{w}_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^{\mathrm{T}}. \tag{3.45b}$$

This approximation, which is also called *weak marginalization*, is defined iff the canonical forms can be represented as Gaussians. Compared to the exemplary mixture of Gaussians in Figure 3.12a the corresponding collapsed Gaussian distribution is plotted in Figure 3.12b.

Although it can be shown that the approximation in Equation 3.45 preserves the first two statistical moments (means and covariances) of the original distribution (see (Lerner, 2002, pp. 63 sq.)), the quality of the approximation clearly depends on the nearness of the mixture densities to a single Gaussian distribution, so when the distance between the means of the original Gaussians is high the approximation can be quite bad. However, the need for weak marginalization when running the clique tree algorithm can be reduced by adding all discrete variables to the root clique. Due to the running intersection property the clique tree is then characterized by having a so-called *strong root*, because weak marginalization (i.e., the collapsing of modes) can be completely avoided during the collect evidence phase of the message passing.

But as (Lerner et al., 2001) states, the risk of weak marginalization also imposes another requirement on the clique compositions regarding the initialization of (C)CD CPD: As described before, in the clique tree the CPD are assigned uniquely to a clique entailing the respective node's family. This means, that a (C)CD CPD has to be inserted into a clique containing the node itself and its parents using the approximation stated in Equation 3.41, but because the original junction tree algorithm can not preclude that modes of the continuous parents have been collapsed beforehand during pre-calibration (again causing inaccuracies), not only the node's parents but also all discrete nodes making for a multi-modal state of the node's continuous parents have to be entailed in the clique that the CD CPD is assigned to. Therefore any (C)CD CPD is always inserted into the (strong) root clique, which already contains all discrete nodes and only the node's parents have to be added (if they are not already a part of the root).

Of course, requiring a strong root results in larger clique sizes and therefore the algorithmic complexity is increased, but fortunately, when incorporating evidence before initializing the clique potentials as suggested by (Murphy, 1999), the effective size of a factor is determined only by the number of *hidden* nodes it contains. This means that observed discrete nodes can be added to cliques and sepsets without extra costs, still making this procedure applicable to many real world problems.

3.3.4 Clique Tree Construction

While the exemplary clique tree shown in Figure 3.8 has been induced by the variable elimination algorithm described in subsection 3.3.1, the original clique tree algorithm presented in (Lauritzen, Spiegelhalter, 1988) also introduces a graph manipulation method to construct a clique tree from a Bayesian network directly. However, the original clique tree algorithm can not handle inference in hybrid Bayesian networks due to weak marginalization that may occur depending on the graph structure and the undefined operation of multiplying a softmax with a Gaussian distribution. To overcome this, extensions to the original clique tree algorithm have been proposed by (Murphy, 1999; Lerner et al., 2001; Lerner, 2002), which have already been described in the previous section when introducing the representation of factors.
In the following the graph manipulations to construct a clique tree from the structure of a Bayesian network are outlined, already taking into account the requirements on the clique compositions to deal with hybrid BN:

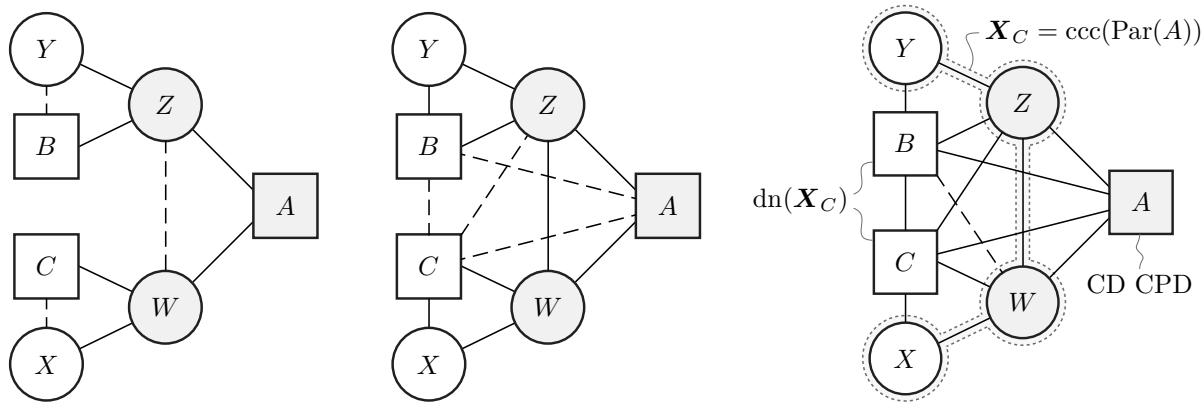
1. The original graph structure $\mathcal{G}$ of the BN gets *moralized*, which means all parents become connected and the directionality of the arcs is dropped. The result is a *moral graph* $\mathcal{G}_M$. Figure 3.13a shows the moralized graph for the exemplary Bayesian network shown in Figure 3.4.

2. The moralized graph $\mathcal{G}_M$ gets *triangulated*, which means that a shortcut for any cycle of length greater than 3 is inserted. This can be achieved by iterating through the graph's nodes in the order of some *elimination order* π and virtually eliminating each node from the graph. During each elimination step, all neighbors of the node to be eliminated become connected. This results in a *triangulated* or *chordal graph* $\mathcal{G}_T$ which has the property of being decomposable into partly independent components, namely the cliques (Lauritzen, Spiegelhalter, 1988).

   In (Kjaerulff, 1990; Koller, Friedman, 2009) heuristics to find an efficient elimination order are presented, because finding the optimal order π^* (i.e., the order which leads to a clique tree with minimal clique sizes) is in general NP-hard. Moreover, in order to avoid weak marginalization during the evidence collection phase of the belief propagation, (Murphy, 1999) proposes to eliminate all continuous nodes before any discrete ones. This leads to a so-called *strong triangulation* and guarantees that the clique tree to be constructed has a strong root. Requiring the clique tree to have a strong root increases the accuracy of the inference process at the cost of a higher algorithmic complexity (because of increased clique sizes).

   It can be seen that for the exemplary BN the moralized graph $\mathcal{G}_M$ is already triangulated, but it does not provide a strong root. Instead, strong triangulation can be achieved with the elimination order $\pi = \{X, Y, W, Z, A, C, B\}$, which as a side condition minimizes the number of inserted edges (called *minimum fill* heuristic, see (Kjaerulff, 1990)). This leads to the graph $\mathcal{G}_T$ shown in Figure 3.13b.

3. Optionally: To assure that in the clique tree to be constructed the potentials with some (C)CD CPD associated to it represent the true, uncollapsed distribution over



(a) Moralization of $\mathcal{G}$ resulting in $\mathcal{G}_M$.

(b) Triangulation of $\mathcal{G}_M$ resulting in $\mathcal{G}_T$.

(c) Connection of $\text{dn}(\boldsymbol{X}_C)$ with $\text{Fam}(A)$ in $\mathcal{G}_T$.

Figure 3.13: Graph manipulations needed to build a clique tree from the exemplary hybrid BN shown in Figure 3.4.

the continuous variables of its scope after pre-calibration, all discrete neighbors of the clique's so-called continuous connected component need to be in the same clique. Thereby a *continuous connected component* is the set of continuous variables $\boldsymbol{X}_C = \text{ccc}(X_i)$ such that there exists a connection of every two variables $X_i, X_j \in \boldsymbol{X}_C$ via only continuous variables in the moralized graph (see Figure 3.13c). The *discrete neighbors* of $\boldsymbol{X}_C$ (denoted as $\text{dn}(\boldsymbol{X}_C)$) constitute the set of discrete variables that is adjacent to some variable in $\boldsymbol{X}_C$. By connecting all $\text{dn}(\boldsymbol{X}_C)$ with the family of the (C)CD CPD nodes, all modes of the probability density will be represented in the clique's probability distribution correctly.

4. In the graph $\mathcal{G}_T$ the maximal cliques C are defined to be the subsets of variables that are completely connected and that are not a subset of another clique (i.e., no pair of cliques C_i and C_j exists such that $C_i \subset C_j$).

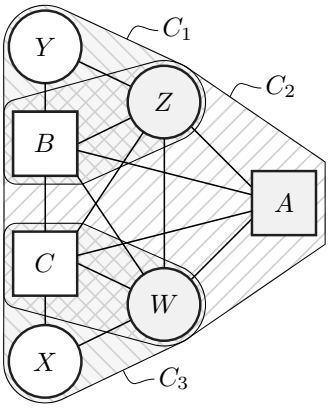
5. The clique tree $\mathcal{T}_C$ is constructed by connecting every pair of cliques C_i and C_j by an edge with a sepset $S_{i:j}$ containing $C_i \cap C_j$.

6. Finally, the (strong) root in $\mathcal{T}_C$ is set to be the node containing all discrete nodes. Thereby each (C)CD CPD gets uniquely associated to the root clique. Any other node in $\mathcal{G}$ gets associated to the clique with the minimum weight, which is the clique with the minimum product of discrete states of all discrete variables it contains.

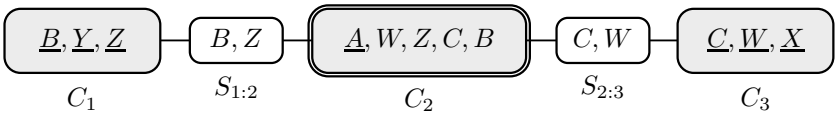
The named modifications to the construction of the junction tree by (Lerner et al., 2001; Lerner, 2002) have been implemented in the Bayes Net Toolbox (BNT) for Matlab (Murphy, 2001). Pseudocode for this approach is shown in section A.1.

3.3.5 Inference

When it comes to the actual inference procedure and evidence $\boldsymbol{e}$ over a set of nodes $\boldsymbol{E}$ is available, according to (Murphy, 1999; Lerner et al., 2001; Lerner, 2002) the modified



(a) Cliques in $\mathcal{G}_T$.



(b) Clique tree derived from $\mathcal{G}_T$.

Figure 3.14: Clique tree for inference in hybrid BN obtained from the graph manipulations described in (Lauritzen, Spiegelhalter, 1988; Murphy, 1999; Lerner et al., 2001; Lerner, 2002).

clique tree algorithm proceeds as follows:

1. To start the inference process, all potentials Φ_C and sepsets Φ_S of the clique tree are initialized as "empty" conditional Gaussian factors, i.e., they are initialized to their identity element.
2. Any discrete node with at least one continuous parent is temporarily removed from the clique tree $\mathcal{T}_C$ due to the fact that a (C)CD CPD can only be multiplied onto a clique potential Φ_C when the tree is (pre-)calibrated. The result is a reduced clique tree $\mathcal{T}_{C\backslash\mathcal{S}}$.
3. All CPD defined in the original BN (except for the (C)CD CPD) are converted to conditional Gaussian factors and all available evidence is instantiated in these factors.
4. Every factor resulting from the conversion of a CPD gets multiplied onto the clique potential it is associated to.
5. To pre-calibrate the tree, messages first get passed from the leaves towards the root during the collect evidence phase. To do so, when sending a message from clique C_i to C_j at first all variables in C_i that are not in C_j get marginalized from the potential Φ_{C_i} to obtain the sepset $\Phi_{S_{i,j}}$. This sepset is then multiplied onto the potential Φ_{C_j}. Afterwards a distribute evidence phase is executed, whereby it is important to take into account that C_j has already received information from C_i. In order to not double-count the information, which would lead to being overconfident about it, when sending a message back from C_j to C_i the potential of C_j is first divided by the sepset generated during the upward pass before the procedure is carried on usually.
6. After the clique tree has been pre-calibrated, the (C)CD CPD are inserted back into the reduced clique tree $\mathcal{T}_{C\backslash\mathcal{S}}$ to restore the original clique compositions of $\mathcal{T}_C$. As suggested in (Lerner et al., 2001), this is achieved by approximating the product of

a Gaussian and a softmax with a Gaussian as stated in Equation 3.41. Because no closed form solution to solve the integrals exist, numerical integration (e.g. Gauss-Hermite quadrature as used in (König, Rehder, Hohmann, 2017)) has to be employed.

7. Now that the (C)CD CPD have been multiplied onto the cliques they are associated to, the evidence in the tree is collected and distributed a second time to re-calibrate the clique potentials. Afterwards each potential in the clique tree holds the (possibly unnormalized) marginal of the joint probability distribution for the entire set of variables it contains.

8. Finally, to obtain the marginal probability of a specific node, any calibrated potential containing the queried variable can be marginalized.

The named modifications to the inference process of the junction tree algorithm by (Lerner et al., 2001; Lerner, 2002) have been implemented in the Bayes Net Toolbox (BNT) for Matlab (Murphy, 2001) in order to infer a driver's maneuver intentions via a hybrid BN as presented in the next section. The pseudocode for the implementation can be found in section A.2.

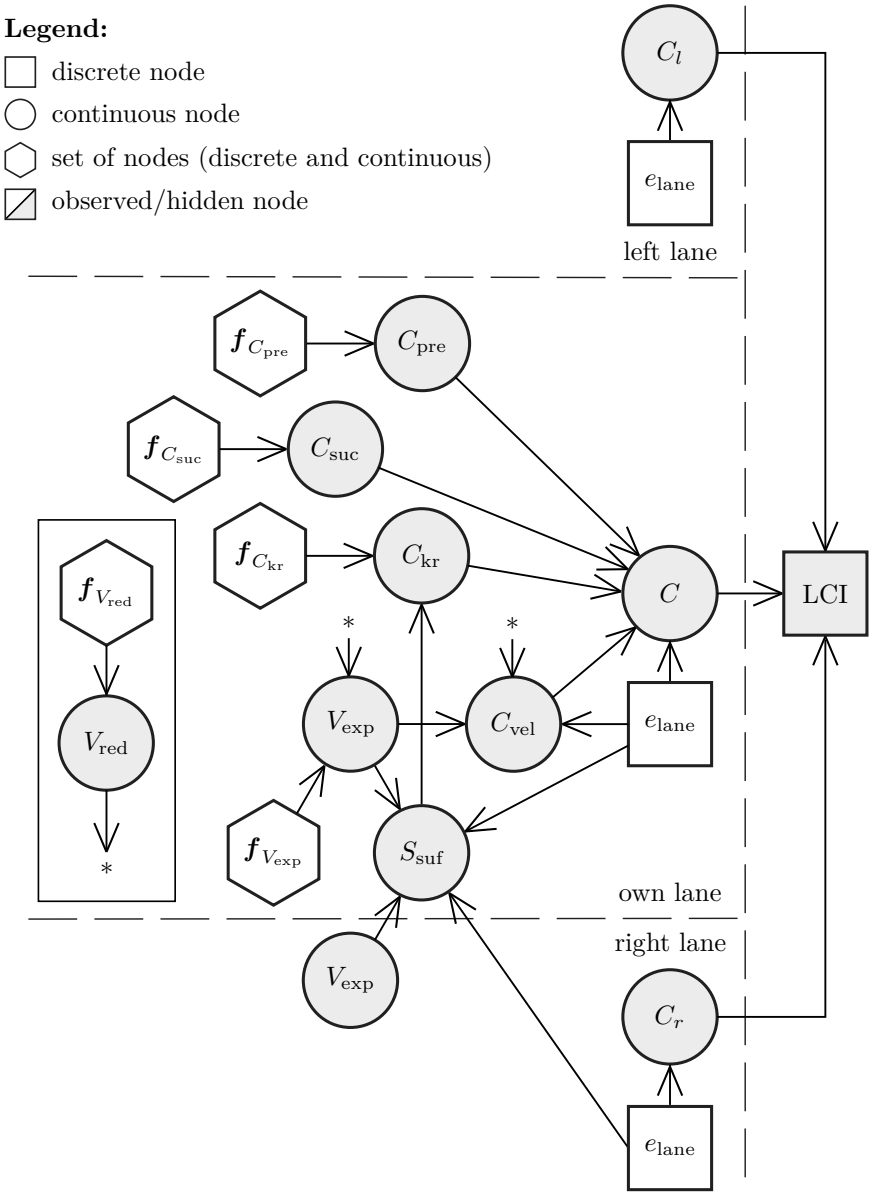
3.4 Bayesian Network for Recognizing Maneuver Intentions

In Figure 3.15 the Bayesian network (BN) for the proposed lane change intention (LCI) recognition approach is shown. In addition to the representation described in subsection 3.2.1, in this graph hexagon shaped nodes refer to sets of variables that may be composed of both continuous and discrete variables (for the sake of a lucid graphic representation).
The graph structure shows the discrete variable indicating a lane change intention on the right. The LCI node has the three states *lane change left*, *keep lane* and *lane change right* (denoted by LCL, KL and LCR in short). As already described, the lane change intention in PELOPS is generated from a comparison of the expected contentedness C on the own lane to the expected lane contentedness values C_l and C_r on the neighboring lanes respectively. The overall contentedness on an individual lane is again composed of different contentedness factors. Eventually the different contentedness factors (which are only shown for the own lane in Figure 3.15 as they repeat for the two adjacent lanes) again depend on features characterizing the traffic situation with respect to surrounding traffic participants like presented in section 3.1.
The contentedness factors that are being taken into account here are described subsequently. As it is not expedient to parameterize the CPD of the BN manually by a human expert, machine learning techniques are used to learn the parameters quantifying the probabilistic influence in the BN directly from data. This will be described in section 5.3.

Contentedness $C_{\mathbf{pre}}$
The contentedness factor C_{pre} regards the nearness of at most two preceding vehicles $\mathcal{V}_{\text{p}_1}$ and $\mathcal{V}_{\text{p}_2}$ on the particular lane, see Figure 3.16a. Vehicles driving slow and close in front reduce a driver's contentment on that lane.
C_{pre} depends on the inverse time headway to $\mathcal{V}_{\text{p}_1}$ to express the nearness and the inverse time to collision to $\mathcal{V}_{\text{p}_1}$ and $\mathcal{V}_{\text{p}_2}$ to characterize the relative dynamics to the preceding traffic. Together with variables e indicating the lane's and object vehicles' existences all features that C_{pre} depends on are

$$\boldsymbol{f}_{C_{\text{pre}}} = \{T^{-1}_{\text{hw,p1}},\ T^{-1}_{\text{tc,p1}},\ T^{-1}_{\text{tc,p2}},\ e_{\text{lane}},\ e_{\text{p}_1},\ e_{\text{p}_2}\}. \tag{3.46}$$

Contentedness $C_{\mathbf{suc}}$
The contentedness C_{suc} considers the nearness of a succeeding vehicle $\mathcal{V}_{\text{s}_1}$ (if any), see Figure 3.16b. Vehicles approaching fast and tailgating the subject vehicle reduce the contentment on that lane.
C_{suc} is based on the nearness and relative dynamics to $\mathcal{V}_{\text{s}_1}$, but the factor is furthermore also influenced by the nearness to $\mathcal{V}_{\text{p}_1}$ and $\mathcal{V}_{\text{p}_2}$, because succeeding vehicles only lower the contentment on a particular lane if the driver is not blocked by vehicles in front. Furthermore it is assumed that the strength of vehicles from behind "pushing the vehicle to the right" depends on how far the vehicle's current lane is away from the rightmost lane compared to the overall number of available lanes. Therefore a factor w is used to express



Set	Composition
$\boldsymbol{f}_{C_{\text{pre}}}$	$T^{-1}_{\text{hw,p1}}$, $T^{-1}_{\text{tc,p1}}$, $T^{-1}_{\text{tc,p2}}$, e_{lane}, e_{p_1}, e_{p_2}
$\boldsymbol{f}_{C_{\text{suc}}}$	$T^{-1}_{\text{hw,s}_1}$, $T^{-1}_{\text{tc,s}_1}$, $T^{-1}_{\text{hw,p}_1}$, $T^{-1}_{\text{hw,p}_2}$, w, e_{lane}, e_{s_1}, e_{p_1}, e_{p_2}
$\boldsymbol{f}_{C_{\text{kr}}}$	$T^{-1}_{\text{hw,p}_1}$, $T^{-1}_{\text{hw,p}_2}$, $T^{-1}_{\text{hw,s}_1}$, $T^{-1}_{\text{tc,s}_1}$, e_{lane}, $e_{\text{lane},r}$, w, e_{p_1}, e_{p_2}, e_{s_1}
$\boldsymbol{f}_{V_{\text{exp}}}$	$v_{\text{pred,p}_1}$, $v_{\text{pred,p}_2}$, $T^{-1}_{\text{hw,p}_1}$, $T^{-1}_{\text{hw,p}_2}$, v, e_{lane}, e_{p_1}, e_{p_2}
$\boldsymbol{f}_{V_{\text{red}}}$	κ, $\kappa_{\max}$, v_{des}

Figure 3.15: Bayesian network for lane change intention recognition (LCI). The network structure for a single lane contentedness C is repeated for the adjacent lanes' contentedness C_l and C_r.

how many lanes there are to the left that succeeding vehicles could possibly use to overtake without the need for the subject vehicle to change to the right. With m being the number of all lanes and i being the zero-based index of the current lane counted from right to left, w is computed by

$$w = \frac{i}{m-1}. \tag{3.47}$$

To sum up, all features that C_{suc} depends on are

$$\boldsymbol{f}_{C_{\text{suc}}} = \{T^{-1}_{\text{hw,s}_1},\ T^{-1}_{\text{tc,s}_1},\ T^{-1}_{\text{hw,p}_1},\ T^{-1}_{\text{hw,p}_2},\ w,\ e_{\text{lane}},\ e_{\text{s}_1},\ e_{\text{p}_1},\ e_{\text{p}_2}\}. \tag{3.48}$$

Contentedness C_{vel}

The factor C_{vel} represents the contentment of a driver with the desired velocity compared to the viable velocity with respect to other vehicles driving on a particular lane, see Figure 3.16c. Thereby the driver's desired velocity is reduced by limitations from the track like the road curvature and speed limits, which is why C_{vel} depends on the driver's reduced desired velocity V_{red} and the expected velocity V_{exp} on that lane.
The driver's reduced desired velocity V_{red} again depends on the features

$$\boldsymbol{f}_{V_{\text{red}}} = \{\kappa,\ \kappa_{\max},\ v_{\text{des}}\} \tag{3.49}$$

with κ being the curvature on the lane at the current vehicle's position, $\kappa_{\max}$ the maximum curvature ahead (see Equation 3.1) within a foresight distance that the driver is able to overlook (which here is set to 100 m due to the sensor limitations), as well as the driver's desired velocity v_{des}.
The desired velocity of all vehicles' drivers is assumed to be $130\,\text{km}\,\text{h}^{-1}$. This obviously is wrong in many cases, because the true desired velocity is influenced by many different factors, especially a driver's personal preference. However, on German highways this is the advisory speed limit, which makes this assumption sensible.
The expected velocity V_{exp} depends on the predicted velocity v_{pred} of at most two preceding vehicles at a lookahead time of $t_{\text{pred}} = 1\,\text{s}$ under the assumption of constant acceleration, so

$$v_{\text{pred}} = v + a \cdot t_{\text{pred}}. \tag{3.50}$$

Moreover V_{exp} is affected by the nearness to these vehicles expressed via the time headway, which is motivated by the assumption that the further the preceding vehicles are away, the less is their influence on the subject vehicle's velocity. In summary V_{exp} depends on the features

$$\boldsymbol{f}_{V_{\text{exp}}} = \{v_{\text{pred,p}_1},\ v_{\text{pred,p}_2},\ T^{-1}_{\text{hw,p}_1},\ T^{-1}_{\text{hw,p}_2},\ v,\ e_{\text{lane}},\ e_{\text{p}_1},\ e_{\text{p}_2}\}. \tag{3.51}$$

Contentedness C_{kr}

The factor C_{kr} reflects a driver's contentment regarding the compliance with the obligation to drive on the right hand side of the road (holding in many European countries including Germany, see § 2(2) of the German road traffic regulations (StVO, 2013)). It is assumed that a driver is not content driving on the left lanes if the right lanes are clear (see Figure 3.16d), but this obligation is only obeyed when it does not come at the cost of a high speed reduction with respect to the driver's desired velocity.

To compare the expected velocity V_{exp} on the current lane to the expected velocity on the next lane to the right a so-called sufficient speed factor S_{suf} is used. Basically S_{suf} makes the right lane more attractive if the expected velocity is roughly the same on both lanes. Together with characteristics describing the relative dynamics to the vehicles ahead and behind on the current lane, all features that C_{kr} additionally depends on are

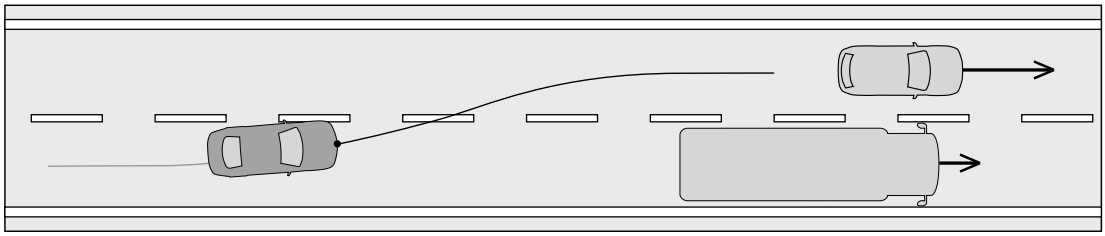
$$\boldsymbol{f}_{C_{\text{kr}}} = \{T^{-1}_{\text{hw,p}_1},\ T^{-1}_{\text{hw,p}_2},\ T^{-1}_{\text{hw,s}_1},\ T^{-1}_{\text{tc,s}_1},\ e_{\text{lane}},\ e_{\text{lane},r},\ w,\ e_{\text{p}_1},\ e_{\text{p}_2},\ e_{\text{s}_1}\ \}. \tag{3.52}$$

Besides the named contentedness factors the driver model in PELOPS is using even more factors which are not used in the approach to a driver's intention recognition here. These factors are described in the following together with the reason why they have not been taken into account.

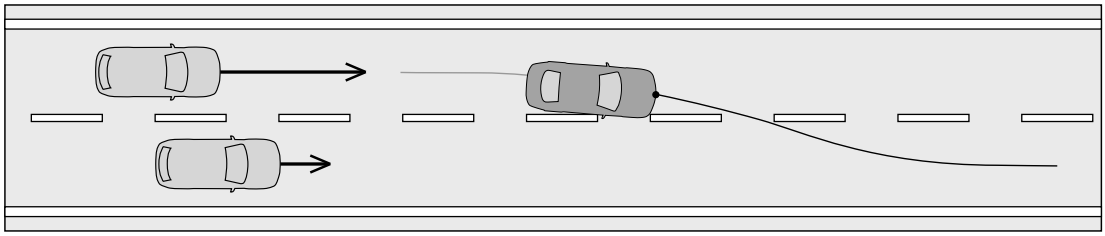
Contentedness C_{rte}
C_{rte} is a contentedness factor regarding the driver's planned route. It reduces the contentment on lanes which do not lead to the driver's desired destination, see Figure 3.17a.
As in the test vehicle no digital map information is available yet, an estimation of the driver's desired route can not be carried out, which is why this factor is not being taken into account.

Contentedness C_{co}
The factor C_{co} reflects the contentment arising from the cooperation with surrounding vehicles that are about to change lanes, e.g. on highway on-ramps or when detecting an activated indicator, see Figure 3.17b.
This contentedness factor cannot be taken into account yet, because a traffic vehicle's activated indicator cannot be detected by the environment sensors available in the test vehicle and again no digital map information is available to gather information about the end of a lane.

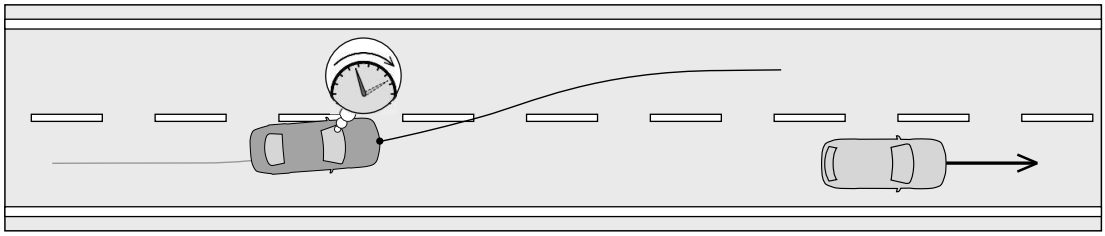
Contentedness C_{fav}
The factor C_{fav} favours a specific lane. It is used in PELOPS to enforce a custom driving behavior, e.g. ignoring the obligation to drive on the right lane.



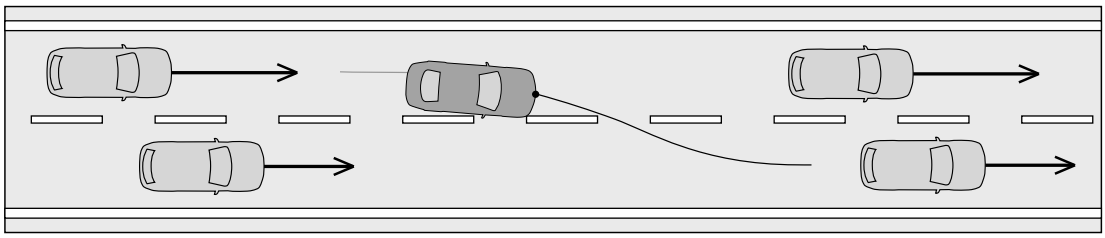
(a) Exemplary scenario of C_{pre} motivating a lane change because of a lane obstruction.



(b) Exemplary scenario of C_{suc} motivating a lane change because of a vehicle approaching from behind fast.

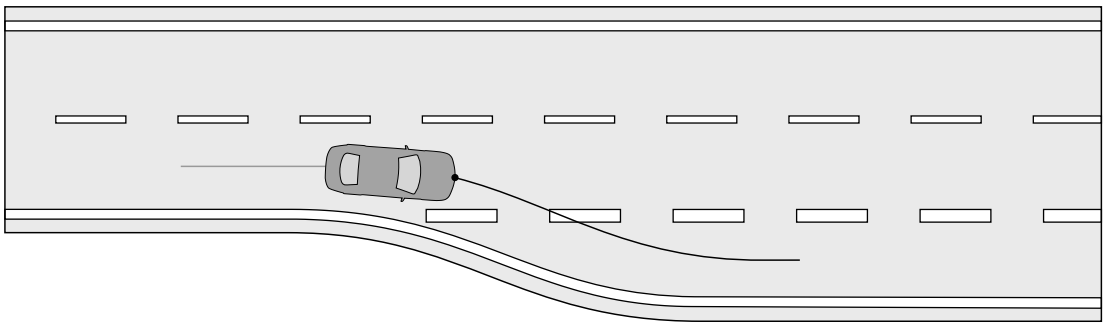


(c) Exemplary scenario of C_{vel} motivating a lane change due to unsatisfied speed desire.

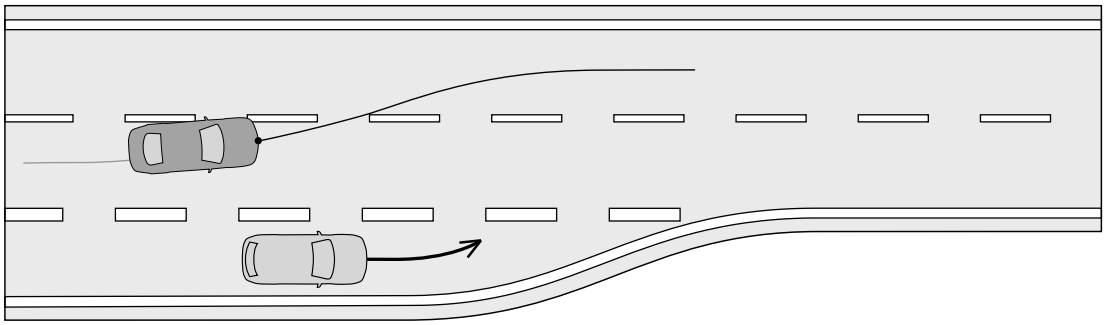


(d) Exemplary scenario of C_{kr} motivating a lane change to be compliant to the obligation to drive on the right lane.

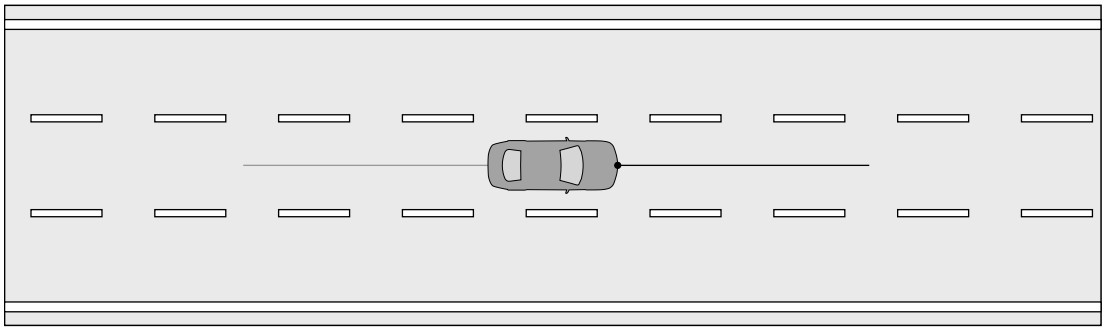
Figure 3.16: Exemplary scenarios of lane contentedness factors in PELOPS which are being taken into account in the Bayesian network to recognize a driver's lane change intention.



(a) Exemplary scenario of C_{rte} motivating a lane change due to the need to follow a desired route.



(b) Exemplary scenario of C_{co} motivating a lane change due to a cooperative behavior with respect to a traffic vehicle intending to change the lane.



(c) Exemplary scenario of C_{fav} motivating to drive on a specific lane.

Figure 3.17: Exemplary scenarios of lane contentedness factors in PELOPS which are *not* being taken into account in the Bayesian network to recognize a driver's lane change intention.

CHAPTER 4

Driving Behavior Prediction

When dealing with time series data in probability theory, the term *prediction* is normally used to refer to the inference of a yet unknown system state X at time t_{k+n} (shortly denoted by X_{k+n}) given a series of observations $\boldsymbol{o}_{1:k}$ from the past including the present, so a prediction refers to $\mathrm{P}(X_{k+n}|\boldsymbol{o}_{1:k})$, or more generally to $\mathrm{P}(X_{k+1}|\boldsymbol{o}_{j:k})$ conditioned on a subset of the past observations. Moreover, estimating $\mathrm{P}(X_k|\boldsymbol{o}_{j:k})$ is called *filtering* (also *tracking*), computing $\mathrm{P}(X_{k-n}|\boldsymbol{o}_{j:k})$ is named *smoothing* (see Figure 4.1 for the example of $n = 1$).
However, in the literature the term prediction is often used to refer to the inference of a generally unknown system state - may it be in the sense of estimating $\mathrm{P}(X_{k+n}|\boldsymbol{o}_{j:k})$ when the system state is yet unknown but can be observed in the future, or in the sense of $\mathrm{P}(X_k|\boldsymbol{o}_{j:k})$ with a latent system state X_k. Also in machine learning, where the goal is to find the unknown function f to compute $X = f(\boldsymbol{o})$, the inference step is often called prediction regardless of the time domain, no matter if f is designed such that $f(\boldsymbol{o})$ acts as prediction, filtering or smoothing function. In supervised machine learning this behavior is controlled by different data labeling strategies, which will be discussed in section 5.2.
In this thesis the terms maneuver *detection* and intention *recognition* are used when inferring the probability of a maneuver currently taking place or being intended, i.e., detecting an ongoing maneuver by estimating $\mathrm{P}(M_k|\boldsymbol{o}_{k-1})$ or inferring a maneuver intention by com-

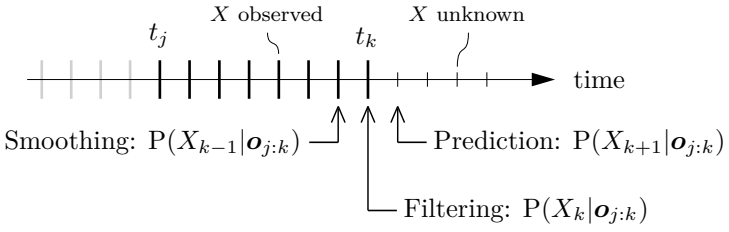


Figure 4.1: Prediction vs. filtering vs. smoothing.

puting $P(I_k|\boldsymbol{o}_{k-1})$, respectively. This terminology is based on the lane change decision process commonly used in many lane change models, for example in (Gipps, 1986; Q. Yang, Koutsopoulos, 1996; Hidas, Behbahanizadeh, 1999), where the lane change process is composed of three sequential steps, which have also been indicated in Figure 2.2:

1. The intention to change the lane is made up from the discretion or necessity to leave the current lane, starting at $t_{\text{intention}}$.
2. The realizability of a lane change is checked and an acceptable gap on the target lane is selected (or, as extended in the model of (Hidas, 2005), the decision to force a gap on an occupied lane is made) in the phase from $t_{\text{intention}}$ to $t_{\text{execution}}$.
3. Finally, the lane change is executed starting from $t_{\text{execution}}$ and in this definition ending at the time of the vehicle's lane assignment change at $t_{\text{change}} = t_0$.

Finally, when a maneuver intention has been recognized or the start of a maneuver execution has been detected, then the term *prediction* refers to the future vehicle states $X_{k+1:h}$ in the prediction horizon of t_{k+1} to t_{k+h}, which can be estimated based on the maneuver information.
The approach to driving behavior prediction in this thesis is inspired by the three layer hierarchy after (Donges, 1982), which subdivides the driving task into three layers. In the first layer the navigation task is carried out in which the driver decides which route to take to reach the target destination. According to (Werling, 2015) the navigation can be seen as an optimization of the trip duration (or route length, fuel consumption etc.) given the street network and it has an optimization horizon of up to several hours. The result of the navigation is a route, which serves as input to the lane guidance task. Primarily with respect to safety and comfort aspects the driver refines the route in the second layer. Taking the road course and the vehicle dynamics into account, the lane guidance is optimized as fast as possible to be able to react to changing traffic situations quickly. With an optimization horizon of several seconds the result is a desired target lane the driver wants to drive on, including the maneuver to reach or stay on the target lane. The third layer is the stabilization layer in which the driver reacts to disturbances like wind or the coarseness of the road surface. During the stabilization the vehicle's actuators (like pedals and steering wheel) are operated continuously by the driver.
To apply the hierarchy to the task of automated driving and to bring it in line with the process of obtaining situation awareness proposed by (Endsley, 1988) (see Figure 1.4 in section 1.2), the three layer hierarchy here has been extended by a driving behavior prediction framework to project the future statuses of the current situation's vehicles, see Figure 4.2. The general idea behind this framework is to take up the three layer hierarchy again and to generate a set of plausible hypotheses as result of the driving task in each layer first. Based on a variety of plausible future vehicle trajectories, which are the output of the hypotheses generation, the prediction of the actual driving behavior is achieved afterwards by comparing the vehicle movement to the predicted trajectories in the subsequent time steps. In other words, the correctness of the maneuver hypotheses is classified to identify the current and future driving behavior.

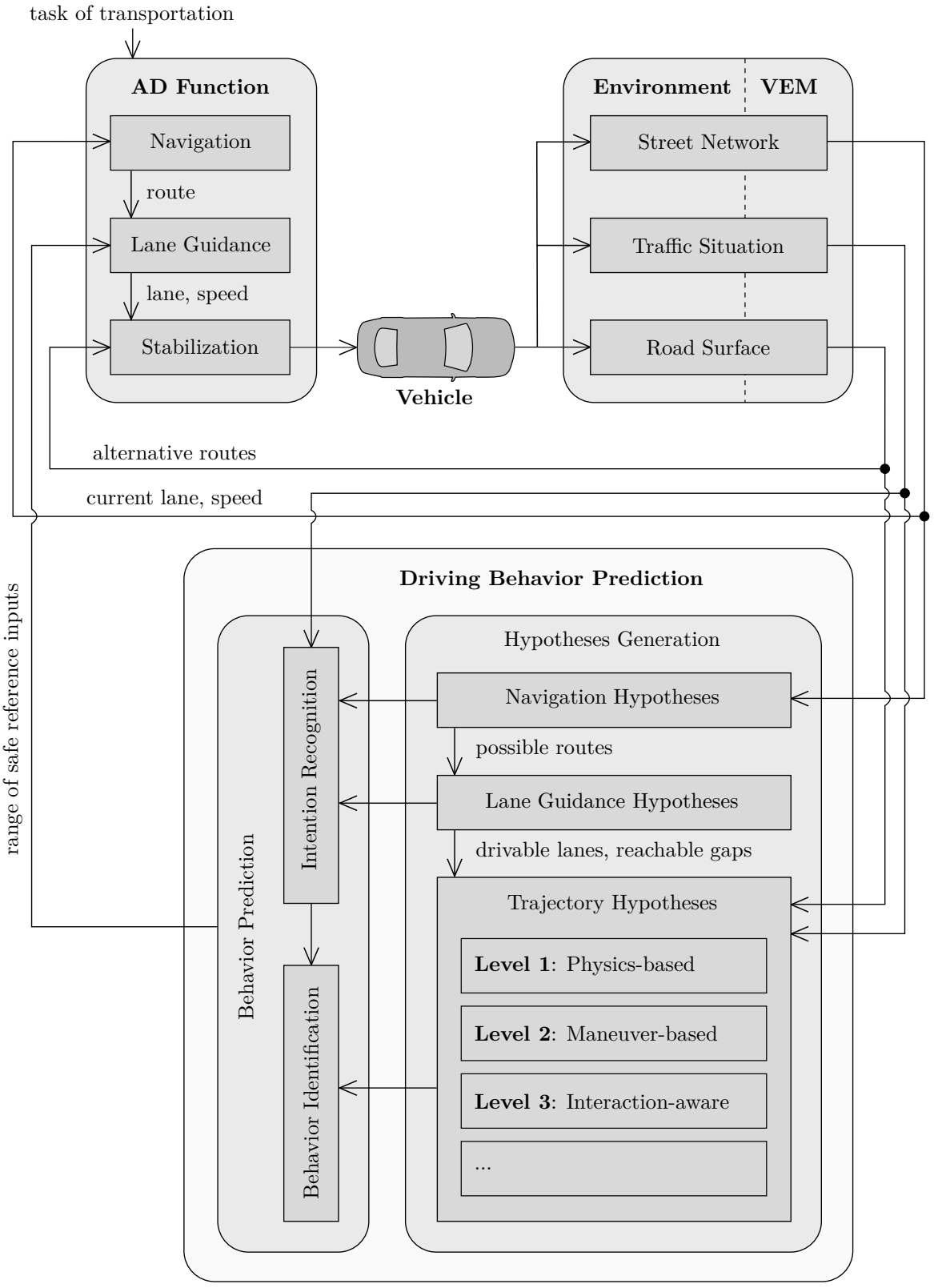


Figure 4.2: Representation of the three layer hierarchy after (Donges, 1982), extended by the prediction framework as part of the process to obtain situation awareness.

In detail, the first layer of the hypotheses generation finds plausible navigation hypotheses based on the street network, which is provided by the vehicle environment model (VEM). The results are possible routes an observed subject vehicle can take. For every possible route in the next layer lane guidance hypotheses are generated to obtain the lanes a vehicle can possibly maneuver to, together with the reachable gaps between surrounding vehicles already driving on that lane. Finally, trajectory hypotheses are generated by the stabilization layer to estimate the positions at which a vehicle is likely to be in the future. The advantage of this framework is its modularity. For example, different approaches to generate the trajectory hypotheses on the stabilization layer can be employed either to account for different strengths of model assumptions with varying prediction horizons, for different target destinations the driver of a vehicle might want to reach or even for limited computational resources when generating the hypotheses.
Based on the current traffic situation, as well as the possible routes and drivable lanes, also the intention of a driver to follow any drivable lane is inferred. This is accomplished by estimating how content a driver would be when driving on any of the available lanes on a particular route (as explained in chapter 3). The lane contentedness estimation can then on the one hand be used as informed prior for estimating the probability of a vehicle following one of the generated trajectory hypothesis along the respective lane. On the other hand the estimated lane contentedness can directly be used to infer the motivation of a driver to change the lane, which is an indispensable information for the automated driving (AD) function to act cooperatively and to allow the observed vehicle to realize the lane change, e.g. by slowing down to create an adequate gap.
The lane change intention probabilities and likelihoods of the trajectory hypotheses are a required input to the driving strategy, trajectory planning and collision checking modules of the AD function on the lane guidance level.

In the next sections, at first the requirements on the prediction in terms of the prediction horizon are worked out. Then, in section 4.2 the approach to generate trajectory hypotheses is presented in detail. Finally, section 4.3 describes the process of identifying the actual vehicle behavior, which leads to the final driving behavior prediction.

4.1 Prediction Requirements

Assuming that a slower subject vehicle on a highway merges into the ego vehicle's lane without recognizing the faster ego vehicle approaching from behind, how much time in advance to the actual cut-in of the subject vehicle needs the automated driving ego vehicle to predict the lane change situation in order to avoid a collision? To estimate this requirement on the prediction of any surrounding vehicle's driving behavior, the time a braking maneuver has to be started in advance to a cut-in situation such that a collision can be only just avoided is computed. According to (Kopf, 1994) this time span is also known as the time to brake T_{tb}. The latest point in time a cut-in maneuver has to be detected is when T_{tb} approaches zero.

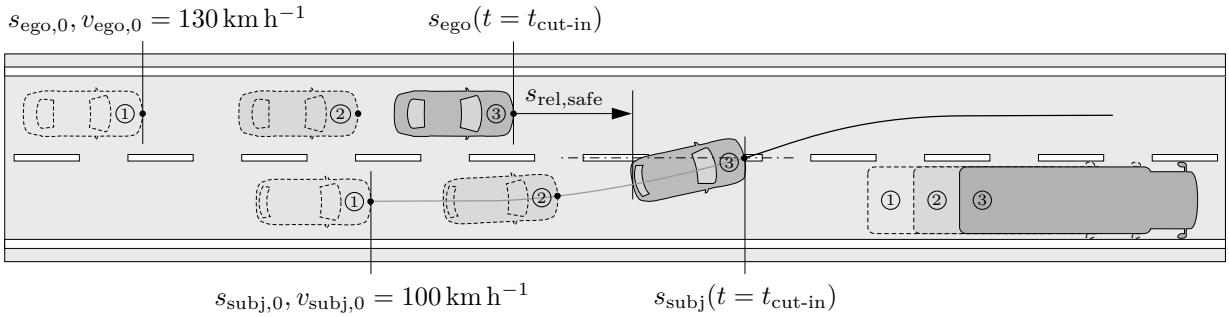


Figure 4.3: Vehicle positions over time during a lane change.

In Figure 4.3 a typical traffic scenario on a highway is shown. The ego vehicle driving at $v_{\text{ego},0} = 130\,\text{km}\,\text{h}^{-1}$ is facing a cut-in maneuver of the subject vehicle, which drives at $v_{\text{subj},0} = 100\,\text{km}\,\text{h}^{-1}$ and changes the lane due to a slower object vehicle in front. The lane change of the subject vehicle takes place at $t = t_{\text{cut-in}}$, which is defined to be the point in time when the center of the vehicle's front crosses the lane marking. Because of the relative velocity of $v_{\text{rel},0} = -30\,\text{km}\,\text{h}^{-1}$ obviously the ego vehicle has to decelerate in order to avoid a rear-end collision with the lane changing subject vehicle.
In general, when the acceleration a of a vehicle is time variant and when $t_0 = 0$, the velocity v and position s of a vehicle at a specific time t_s calculate to

$$v(t = t_s) = v_0 + \int_{t_0}^{t_s} a(t)\,\mathrm{d} \quad \text{and} \tag{4.1}$$

$$s(t = t_s) = s_0 + v_0\,t_s + \iint_{t_0}^{t_s} a(t)\,(\mathrm{d}t)^2. \tag{4.2}$$

For the particular scenario shown in Figure 4.3, to compute the remaining time until the ego vehicle has to start decelerating in order to adapt to the speed of a cut-in vehicle while maintaining a safety distance of $s_{\text{rel,safe}}$, it is postulated that

$$v_{\text{rel}}(t = t_{\text{cut-in}}) \stackrel{!}{=} 0 \quad \text{and} \tag{4.3}$$

$$s_{\text{rel}}(t = t_{\text{cut-in}}) \stackrel{!}{=} s_{\text{rel,safe}}. \tag{4.4}$$

(The relative distance s_{rel} between the object and the subject vehicle is specified as clearance, see Equation 3.3.) Assuming constant velocity of the subject vehicle ($v_{\text{subj}} = \text{const}$) and a piecewise constant acceleration of the ego vehicle with

$$a_{\text{ego}}(t) = \begin{cases} 0 & \text{if } t < t_{\text{brake}} \\ a_{\text{ego,min}} & \text{if } t_{\text{brake}} \leq t \leq t_{\text{cut-in}}, \end{cases} \tag{4.5}$$

whereby t_{brake} denotes the point in time the ego vehicle instantly starts decelerating, it follows from Equation 4.3 and 4.4 together with Equation 4.2 and 4.1, respectively, that

$$t_{\text{brake}} = \frac{v_{\text{rel},0}}{2\,a_{\text{ego,min}}} - \frac{s_{\text{rel},0} + s_{\text{rel,safe}}}{v_{\text{rel},0}}. \tag{4.6}$$

(See Appendix B for a detailed deduction.) Thereby $a_{\text{ego,min}}$ is the minimal realizable (or acceptable) acceleration (i.e., the maximal realizable/acceptable deceleration) of the ego

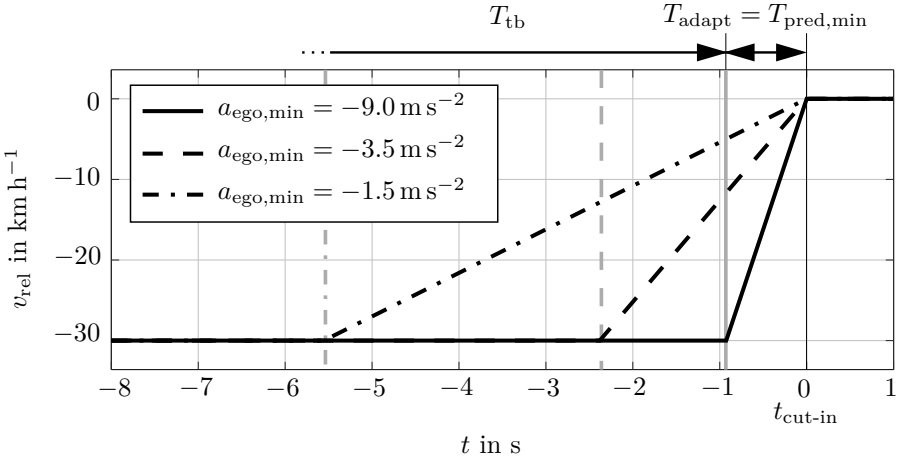


Figure 4.4: Minimal required lane change prediction time $T_{\text{pred,min}}$ equals the time T_{adapt} which is required to eliminate v_{rel}.

vehicle. For the case of $t_0 = 0$ this point in time is equal to the time span T_{tb}, which is the so-called time to brake (Hillenbrand, 2007, pp. 120 sqq.), sometimes also denoted by TTB. It basically is the time to collision T_{tc} (see Equation 3.8) reduced by the time it takes for the ego vehicle to decelerate to the velocity of the preceding subject vehicle, which is denoted by

$$T_{\text{adapt}} = \left|\frac{v_{\text{rel},0}}{a_{\text{ego,min}}}\right|. \tag{4.7}$$

In all situations where T_{tb} is greater than or equal to zero the time span T_{adapt} is the minimal required prediction time in order to only just avoid a collision, so it holds

$$T_{\text{pred,min}} = T_{\text{adapt}}. \tag{4.8}$$

To visualize the minimal required prediction time depending on the piecewise constant acceleration profile of Equation 4.5 with the minimal realizable/acceptable acceleration $a_{\text{ego,min}}$ of the ego vehicle, in Figure 4.4 the trajectory of the relative velocity v_{rel} is plotted for the cut-in scenario of Figure 4.3. It can be seen that with increasing values for $a_{\text{ego,min}}$, which lead to an improved comfort for the passengers of the ego vehicle, a larger required minimal prediction time is required to avoid a rear-end collision. In this particular scenario the minimal required prediction time is 2.4 s when the maximum deceleration is set to a safe value of $-3.5\,\text{m}\,\text{s}^{-2}$. This prediction horizon is used as the target value throughout the rest of this thesis.
Figure 4.5 shows the remaining T_{tb} until a braking maneuver with $a_{\text{ego,min}} = -9.0\,\text{m}\,\text{s}^{-2}$ has to be started in order to only just maintain a safety distance $s_{\text{rel,safe}}$ to the cutting-in vehicle.

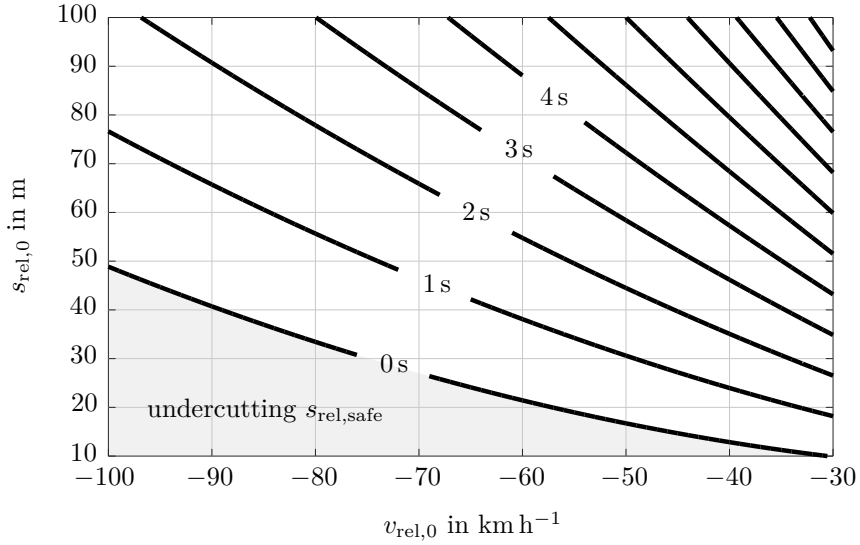


Figure 4.5: Remaining time to brake T_{tb} until a braking maneuver with $a_{\mathrm{ego,min}} = -9.0\,\mathrm{m\,s^{-2}}$ has to be started in order to only just maintain a safety distance $s_{\mathrm{rel,safe}}$.

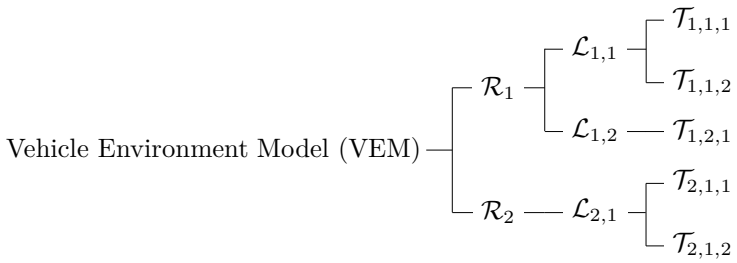
4.2 Hypotheses Generation

To predict a trajectory a vehicle is likely to follow, the key notion is to generate multiple probable trajectory hypotheses first and to infer the posterior probability of the vehicle following a specific trajectory afterwards by observing the vehicle motion relative to each trajectory for a period of time.

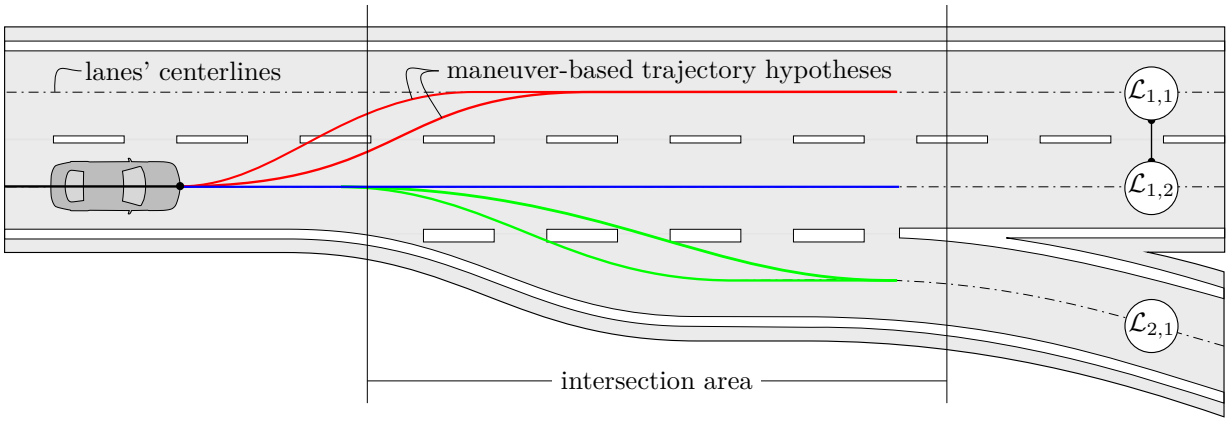
As already described and depicted in Figure 4.2, the hypotheses generation follows a three layer approach. First, all possible routes $\mathcal{R}$ a driver can take are identified. Thereby a route is defined to be the set of all adjacent lanes that connect the current lane to the next reachable intersection. An intersection is a nodal point where lanes that are not adjacent get connected.

Next, for each possible route all adjacent lanes $\mathcal{L}$ a driver is allowed to use are identified and for each lane the centerline of the lane geometry is extracted, because when following a specific lane it is assumed that the human driver steers the vehicle towards a goal state while trying to operate at minimum cost with respect to an individual optimization objective.

As for the task of driving not only a single but rather a whole sequence of states and control inputs is of interest, the use of optimal control to mimic this behavior is motivated. Therefore a linear-quadratic regulator (LQR, also called Riccatti regulator) is employed to simulate the possible future trajectories $\mathcal{T}$ of an observed vehicle, which compared to a standard linear regulator can be efficiently solved and easily extended by state restrictions later on. Because the goal state of any surrounding vehicle is yet unknown, the simulation of the controller is done for every possible goal state (i.e., every centerline of all lanes $\mathcal{L}$ on a single route $\mathcal{R}$). Moreover, as the individual optimization objective of a single driver is yet unknown, multiple trajectory hypotheses $\mathcal{T}$ for a single lane can be generated to account for different driving styles (e.g. trying to reach the goal state as fast as possible might correspond to an aggressive driving style while minimizing the vehicle's overall acceleration



(a) Hierarchy of prediction hypotheses.



(b) Visualization of exemplary prediction hypotheses.

Figure 4.6: Example of route, lane and trajectory hypotheses on a highway.

represents a defensive driving behavior).
An example of different route, lane and trajectory hypotheses on a highway is shown in Figure 4.6. Here the subject vehicle can either keep on following the highway on route $\mathcal{R}_1$ or can take the highway exit to follow the route $\mathcal{R}_2$. On route $\mathcal{R}_1$ the vehicle is allowed to follow the course of lane $\mathcal{L}_{1,1}$ or $\mathcal{L}_{1,2}$, so at least two trajectory hypotheses will be generated; each one using the respective lane's centerline as goal state for solving the optimal control problem. But in the example in Figure 4.6 two trajectory hypotheses for each lane change hypothesis have been generated to demonstrate the consideration of different driving styles (i.e., different optimization objectives for solving the LQR problem).

4.2.1 Linear-Quadratic Regulator as Hypotheses Generator

Following (Lunze, 2014, pp. 285 sqq.) and (Borelli et al., 2015, pp. 167 sqq.) the state space equations of a linear time invariant (LTI) regulator which is discrete in time are given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{4.9a}$$

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k \tag{4.9b}$$

with the system's state, input and output $\boldsymbol{x}_k$, $\boldsymbol{u}_k$, and $\boldsymbol{y}_k$ at time step k, the system's time invariant state, input and output matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ and with $k \in 0, 1, ...N-1$, where N is the finite optimization horizon.

Linear constraints are applied to each system input u_i and output y_i at every time step k with

$$u_{i,\min} \leq u_i(k) \leq u_{i,\max} \quad \text{and} \tag{4.10a}$$

$$y_{i,\min} \leq y_i(k) \leq y_{i,\max}. \tag{4.10b}$$

The goal of optimal control is to find the control input sequence $\mathbf{u} = [\boldsymbol{u}_0, ..., \boldsymbol{u}_{N-1}]^{\mathrm{T}}$ that minimizes a given cost functional $J(\boldsymbol{x}_0, \mathbf{u})$, such that

$$J(\boldsymbol{x}_0, \mathbf{u}^*) = \min_{\mathbf{u}} J(\boldsymbol{x}_0, \mathbf{u}) \tag{4.11}$$

with the initial state $\boldsymbol{x}_0 = \boldsymbol{x}(0)$. The optimal control input sequence $\mathbf{u}^*$ is sometimes also referred to as the decision vector which contains all future inputs. Following (Gutjahr et al., 2016) the cost functional used for the LQR when minimizing the deviation to an arbitrary reference state $\boldsymbol{x}_{\mathrm{ref},k}$ is given by

$$\begin{aligned} J(\boldsymbol{x}_0, \mathbf{u}) = {} & (\boldsymbol{x}_N - \boldsymbol{x}_{\mathrm{ref},k})^{\mathrm{T}} \boldsymbol{P} (\boldsymbol{x}_N - \boldsymbol{x}_{\mathrm{ref},k}) \\ & + \sum_{k=1}^{N-1} (\boldsymbol{x}_k - \boldsymbol{x}_{\mathrm{ref},k})^{\mathrm{T}} \boldsymbol{Q} (\boldsymbol{x}_k - \boldsymbol{x}_{\mathrm{ref},k}) + \sum_{k=0}^{N-1} \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{R} \boldsymbol{u}_k \end{aligned} \tag{4.12}$$

with the positive semi-definite state penalty $\boldsymbol{Q} \geq 0$, the positive semi-definite terminal state penalty $\boldsymbol{P} \geq 0$ and with the positive definite input penalty $\boldsymbol{R} > 0$. The term $(\boldsymbol{x}_0 - \boldsymbol{x}_{\mathrm{ref},0})^{\mathrm{T}} \boldsymbol{Q} (\boldsymbol{x}_0 - \boldsymbol{x}_{\mathrm{ref},0})$ is not entailed in the cost functional, because these costs cannot be affected by the system input $\boldsymbol{u}$; only future states can be manipulated.

The diagonal elements of the cost matrix $\boldsymbol{Q}$ determine how fast the particular system states are driven towards their origin. Because all other non-diagonal elements do not allow for an interpretation with respect to their effect on the system behavior they are usually chosen to be zero. In a similar way the diagonal elements of the cost matrix $\boldsymbol{R}$ define the scale of the control variables and thereby also affect the speed at which the controller acts. Because the overall control behavior is mostly affected only by the ratio of $\boldsymbol{Q}$ and $\boldsymbol{R}$, the later is often chosen to be the identity matrix and $\boldsymbol{Q}$ is tuned until the desired control behavior is achieved.

The final state penalty $\boldsymbol{P}$ is used to minimize the system states at the end of the prediction horizon in case the states could not be driven to their origin within the optimization horizon. This matrix is therefore only used in a finite horizon optimal control problem formulation like the one presented here. In practice $\boldsymbol{P}$ has to be chosen such that $\boldsymbol{P} > \boldsymbol{Q}$, so for example it could be set to a positive multitude of $\boldsymbol{Q}$.

In general there exist two approaches to solve the dynamic optimization problem of Equation 4.11 as stated in (Borelli et al., 2015, pp. 167 sqq.). On the one hand there exists a so-called *batch approach* that formulates the complete sequence of inputs as a function of the initial state at once. On the other hand the solution can be derived recursively with *dynamic programming* which leads to a policy expressing the control action as a function of the state at each time, which can be applied to each state individually.

While the recursive dynamic programming approach can be solved more efficiently in practice and is also applicable to an infinite time horizon problem, the batch approach allows for taking into account (soft) constraints in closed form when optimizing over a finite time horizon. To be able to extend the trajectory prediction by optimization constraints in the future, in the following the batch approach is used to solve for the optimal control input sequence $\mathbf{u}^*$.
To obtain the system trajectory $\mathbf{x}$, the system's states could be computed sequentially for each time step with the computation series

$$\begin{aligned} \boldsymbol{x}_1 &= \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{B}\boldsymbol{u}_0 \\ \boldsymbol{x}_2 &= \boldsymbol{A}^2\boldsymbol{x}_0 + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{B}\boldsymbol{u}_1 \\ &\vdots \end{aligned}$$

But to be able to solve for the whole optimal control input sequence $\mathbf{u}^*$, the equations are formulated using batch matrices, which results in

$$\underbrace{\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix}}_{\boldsymbol{\mathcal{A}}} \boldsymbol{x}_0 + \underbrace{\begin{bmatrix} \boldsymbol{B} & 0 & \dots & 0 \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \dots & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} \end{bmatrix}}_{\boldsymbol{\mathcal{B}}} \underbrace{\begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{bmatrix}}_{\mathbf{u}} \tag{4.13}$$

and

$$\underbrace{\begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} \boldsymbol{C} & 0 & \dots & 0 \\ 0 & \boldsymbol{C} & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \boldsymbol{C} \end{bmatrix}}_{\boldsymbol{\mathcal{C}}} \underbrace{\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix}}_{\mathbf{x}} \tag{4.14}$$

This can be written with the indicated substitutions in short as

$$\mathbf{x} = \boldsymbol{\mathcal{A}}\boldsymbol{x}_0 + \boldsymbol{\mathcal{B}}\mathbf{u} \quad \text{and} \tag{4.15}$$

$$\mathbf{y} = \boldsymbol{\mathcal{C}}\mathbf{x}. \tag{4.16}$$

Analogously the constraints in Equation 4.10b can be converted to batch form by substitution in Equation 4.15 and Equation 4.16 with

$$\mathbf{y} \leq \quad \mathbf{y}_{\max} \Rightarrow \quad \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{B}}\mathbf{u} \leq \quad \mathbf{y}_{\max} - \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{A}}\boldsymbol{x}_0 \quad \text{and} \tag{4.17a}$$

$$-\mathbf{y} \leq -\mathbf{y}_{\min} \Rightarrow -\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{B}}\mathbf{u} \leq -\mathbf{y}_{\min} - \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{A}}\boldsymbol{x}_0. \tag{4.17b}$$

For the cost functional stated in Equation 4.12 it follows the batch form

$$J(\boldsymbol{x}_0, \mathbf{x}, \mathbf{u}, \mathbf{x}_{\text{ref}}) = [\mathbf{x} - \mathbf{x}_{\text{ref}}]^{\mathrm{T}} \boldsymbol{\mathcal{Q}} [\mathbf{x} - \mathbf{x}_{\text{ref}}] + \mathbf{u}^{\mathrm{T}} \boldsymbol{\mathcal{R}} \mathbf{u} \tag{4.18a}$$

$$= \mathbf{x}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \mathbf{x} - \mathbf{x}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \mathbf{x}_{\text{ref}} - \mathbf{x}_{\text{ref}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \mathbf{x} + \mathbf{x}_{\text{ref}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \mathbf{x}_{\text{ref}} + \mathbf{u}^{\mathrm{T}} \boldsymbol{\mathcal{R}} \mathbf{u} \tag{4.18b}$$

$$= \mathbf{x}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \mathbf{x} - 2\mathbf{x}_{\text{ref}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \mathbf{x} + \mathbf{x}_{\text{ref}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}} \mathbf{x}_{\text{ref}} + \mathbf{u}^{\mathrm{T}} \boldsymbol{\mathcal{R}} \mathbf{u} \tag{4.18c}$$

with the reference trajectory

$$\mathbf{x}_{\text{ref}} = [\boldsymbol{x}_{\text{ref},1}, ..., \boldsymbol{x}_{\text{ref},N}]^{\text{T}} \tag{4.19}$$

as well as the penalties

$$\boldsymbol{\mathcal{Q}} = \text{diag}(\underbrace{\boldsymbol{Q}, ..., \boldsymbol{Q}}_{N \text{ times}}, \boldsymbol{P}) \quad \text{and} \quad \boldsymbol{\mathcal{R}} = \text{diag}(\underbrace{\boldsymbol{R}, ..., \boldsymbol{R}}_{N \text{ times}}). \tag{4.20}$$

Substituting Equation 4.15 into Equation 4.18c yields

$$\begin{aligned}
J(\boldsymbol{x}_0, \mathbf{u}, \mathbf{x}_{\text{ref}}) &= [\boldsymbol{\mathcal{A}}\boldsymbol{x}_0 + \boldsymbol{\mathcal{B}}\mathbf{u}]^{\text{T}} \boldsymbol{\mathcal{Q}} [\boldsymbol{\mathcal{A}}\boldsymbol{x}_0 + \boldsymbol{\mathcal{B}}\mathbf{u}] \\
&\quad - 2\mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{\mathcal{Q}} [\boldsymbol{\mathcal{A}}\boldsymbol{x}_0 + \boldsymbol{\mathcal{B}}\mathbf{u}] + \mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{\mathcal{Q}} \mathbf{x}_{\text{ref}} + \mathbf{u}^{\text{T}} \boldsymbol{\mathcal{R}} \mathbf{u} && (4.21a) \\
&= \boldsymbol{x}_0^{\text{T}} \boldsymbol{\mathcal{A}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{A}} \boldsymbol{x}_0 + \boldsymbol{x}_0^{\text{T}} \boldsymbol{\mathcal{A}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}} \mathbf{u} + \mathbf{u}^{\text{T}} \boldsymbol{\mathcal{B}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{A}} \boldsymbol{x}_0 + \mathbf{u}^{\text{T}} \boldsymbol{\mathcal{B}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}} \mathbf{u} \\
&\quad - 2\mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{A}} \boldsymbol{x}_0 - 2\mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}} \mathbf{u} + \mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{\mathcal{Q}} \mathbf{x}_{\text{ref}} + \mathbf{u}^{\text{T}} \boldsymbol{\mathcal{R}} \mathbf{u} && (4.21b) \\
&= \boldsymbol{x}_0^{\text{T}} \boldsymbol{\mathcal{A}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{A}} \boldsymbol{x}_0 + 2\boldsymbol{x}_0^{\text{T}} \boldsymbol{\mathcal{A}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}} \mathbf{u} + \mathbf{u}^{\text{T}} \boldsymbol{\mathcal{B}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}} \mathbf{u} \\
&\quad - 2\mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{A}} \boldsymbol{x}_0 - 2\mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}} \mathbf{u} + \mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{\mathcal{Q}} \mathbf{x}_{\text{ref}} + \mathbf{u}^{\text{T}} \boldsymbol{\mathcal{R}} \mathbf{u} && (4.21c) \\
&= \mathbf{u}^{\text{T}} [\underbrace{\boldsymbol{\mathcal{B}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}}_{\boldsymbol{H}}] \mathbf{u} + 2[\boldsymbol{x}_0^{\text{T}} \underbrace{\boldsymbol{\mathcal{A}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}}}_{\boldsymbol{F}} - \mathbf{x}_{\text{ref}}^{\text{T}} \underbrace{\boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{B}}}_{\boldsymbol{G}}] \mathbf{u} \\
&\quad + \boldsymbol{x}_0^{\text{T}} [\underbrace{\boldsymbol{\mathcal{A}}^{\text{T}} \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{A}}}_{\boldsymbol{Y}_1}] \boldsymbol{x}_0^{\text{T}} - 2\mathbf{x}_{\text{ref}}^{\text{T}} [\underbrace{\boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{A}}}_{\boldsymbol{Y}_2}] \boldsymbol{x}_0 + \mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{\mathcal{Q}} \mathbf{x}_{\text{ref}} && (4.21d) \\
&= \mathbf{u}^{\text{T}} \boldsymbol{H} \mathbf{u} + 2[\boldsymbol{x}_0^{\text{T}} \boldsymbol{F} - \mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{G}] \mathbf{u} + \underbrace{\boldsymbol{x}_0^{\text{T}} \boldsymbol{Y}_1 \boldsymbol{x}_0}_{\text{const}} + \underbrace{2\mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{Y}_2 \boldsymbol{x}_0}_{\text{const}} + \underbrace{\mathbf{x}_{\text{ref}}^{\text{T}} \boldsymbol{\mathcal{Q}} \mathbf{x}_{\text{ref}}}_{\text{const}}. && (4.21e)
\end{aligned}$$

This is a *quadratic program* that can be solved efficiently with static, constrained optimization approaches like interior point, active set or gradient projection methods (Borelli et al., 2015, pp. 49 sqq.). Thereby terms in Equation 4.21e indicated as constant are independent of $\mathbf{u}$, meaning that they have no influence on finding the optimal solution and can therefore be omitted.

4.2.2 Maneuver-based Trajectory Hypotheses

For the maneuver-based prediction of lane changes the focus lies on the lateral movement of a vehicle to be predicted. To this point constant velocity of the subject vehicle for the longitudinal movement is assumed. Therefore the state vector is chosen to be $\boldsymbol{x} = [\, d, \dot{d}, \ddot{d}\,]^{\text{T}}$ with the lane's curvilinear coordinates d as the lateral position, $\dot{d}$ the lateral velocity and $\ddot{d}$ the lateral acceleration of the subject vehicle. The control input $\boldsymbol{u}$ is the lateral jerk $\dddot{d}$. The reference state $\boldsymbol{x}_{\text{ref},k}$ is the course of the center line of a single lane hypothesis $\mathcal{L}$ at time step k with the lane relative lateral velocity and lateral acceleration equal to zero, so $\boldsymbol{x}_{\text{ref},k} = [d_{\text{ref},k}, 0, 0]^{\text{T}}$.
With a time interval of T_s the system matrices of a simple double integrator model are

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_s & \frac{1}{2}T_s^2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \frac{1}{6}T_s^3 \\ \frac{1}{2}T_s^2 \\ T_s \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

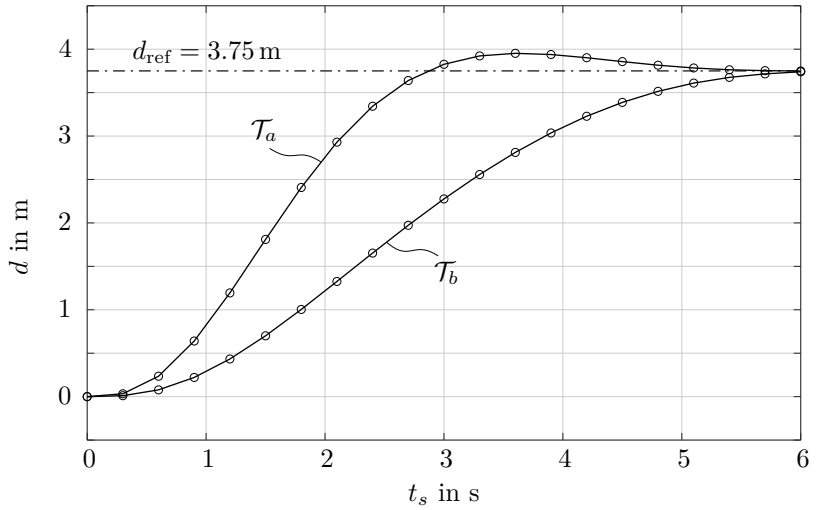


Figure 4.7: Comparison of lane change trajectories resulting from two different LQR cost functional parameterizations a and b.

The optimization horizon of the LQR problem is chosen to be $N = 20$ with a step size of $T_s = 0.3\,\text{s}$, which results in a prediction horizon of $T_{\text{pred,horizon}} = 6.0\,\text{s}$.
In Figure 4.7 two lane change trajectories $\mathcal{T}_a := \mathbf{x}_a^*$ and $\mathcal{T}_b := \mathbf{x}_b^*$ are plotted for two different exemplary parameterizations a and b of the LQR cost functional. An initial state of $x_0 = [0, 0, 0]^{\text{T}}$ and a constant reference trajectory of

$$\mathbf{x}_{\text{ref}} = [\underbrace{\boldsymbol{x}_{\text{ref},0}, ..., \boldsymbol{x}_{\text{ref},0}}_{N \text{ times}}]^{\text{T}} \quad \text{with} \quad \boldsymbol{x}_{\text{ref},0} = [3.75, 0, 0]^{\text{T}} \tag{4.22}$$

is assumed for this demonstration. The two exemplary parameterizations are chosen to be

$$\boldsymbol{Q}_a = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{Q}_b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

with $\boldsymbol{R} = 1$ and $\boldsymbol{P} = 100 \cdot \boldsymbol{Q}$ common to both parameterizations. The resulting trajectories are plotted in solid black with circles marking the discrete time steps every $t_s = k \cdot T_s$ seconds.
It can be seen that parameterization a drives the state much faster towards the reference, because the lateral position d is strongly penalized in relation to the penalties for the lateral velocity $\dot{d}$ and acceleration $\ddot{d}$. In contrast, parameterization b lets the controller strive for driving the state towards the reference while at the same time reducing the lateral acceleration because of its relatively high cost. So while parameterization a could represent an aggressive driving style, parameterization b represents a more defensive lane changing behavior.
But when using a controller for predicting driving behavior it is not expedient to parameterize the weight matrices of the LQR by an expert who optimizes the controller parameters to a specific objective. Instead, the goal is to ascertain the optimization objective of a given, observed system, namely the drivers of the surrounding vehicles of that the driving

behavior is of interest. To do so, inverse reinforcement learning (IRL) with the algorithms proposed in (Ng, Russell, 2000) is used, which finds the weights of the cost matrices such that the demonstrated behavior appears (near)-optimal. This procedure is described in section 5.4.

4.2.3 Receding Horizon Approach

The controller used for the generation of trajectory hypotheses presented in subsection 4.2.1 solves a finite horizon open loop optimal control problem and generates the optimal state trajectory $\mathbf{x}^*$ with respect to the defined state and control input costs.
A trajectory generated by the LQR shall be used to predict the future movement of a vehicle for the time steps t_k with $k \in 1, ..., N$, where $t_k = k \cdot T_s$. But in practice, as the optimal input trajectory $\mathbf{u}^*$ is obviously not implemented in the observed vehicle, the observed state $\tilde{\boldsymbol{x}}_k$ will differ at least slightly from the predicted optimal state $\boldsymbol{x}_k^*$ even at $k = 1$ and this error will propagate through the subsequent states up to $\tilde{\boldsymbol{x}}_N$. To overcome this, the current state is measured in each time step and the N-step optimization problem is solved again with an updated initial condition $\boldsymbol{x}_0$. This procedure is exemplarily visualized in Figure 4.8.

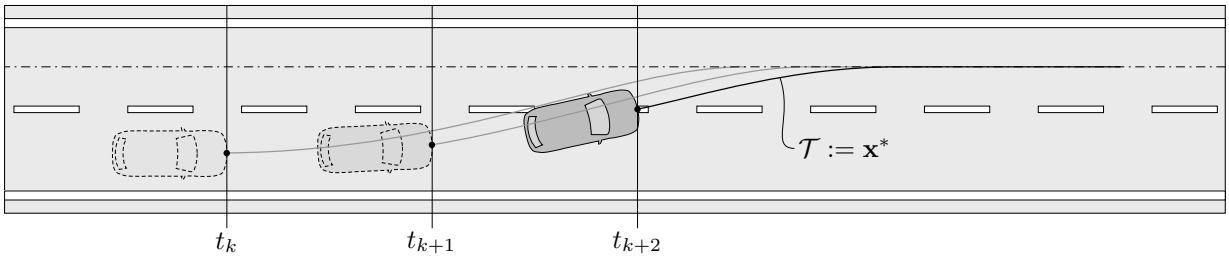
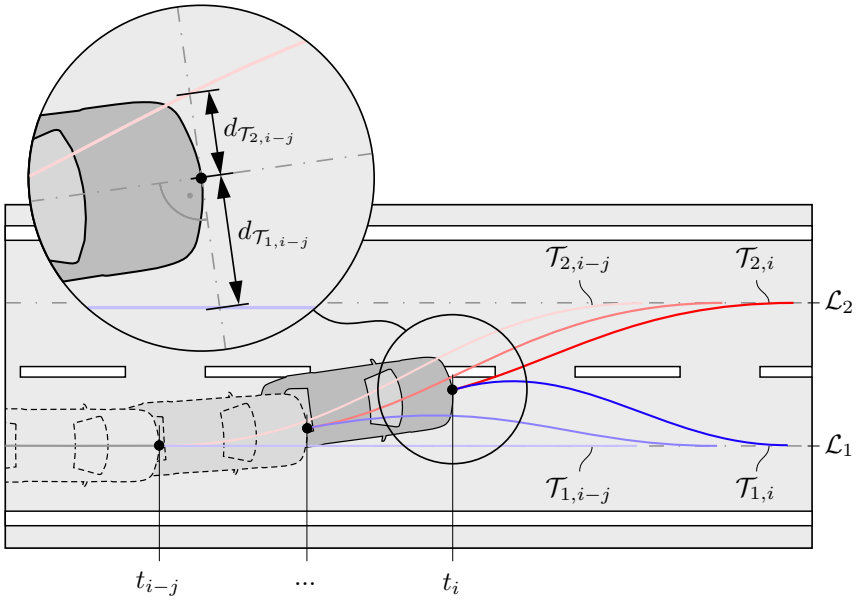


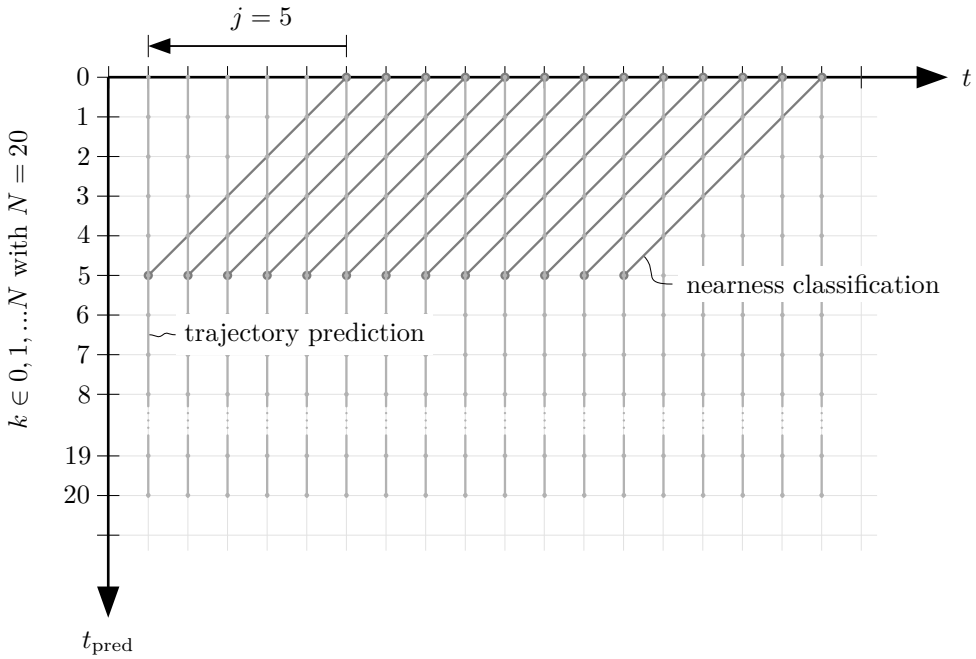
Figure 4.8: Example of receding horizon trajectory hypotheses generation.

4.3 Behavior Identification

To identify which of the trajectory hypotheses generated by the approach presented in subsection 4.2.2 describes the actual vehicle behavior best, a classification of the vehicle's nearness with respect to each of the generated trajectories is carried out. This corresponds to the idea of model tracing. Here the use of a logistic regression is suggested as it is computationally very efficient, but in general also more sophisticated classification methods can be employed.
As each trajectory generated by the LQR is using the current vehicle's state as $\boldsymbol{x}_0$, the behavior identification cannot be carried out together with the trajectory generation at the same time. Instead, the manifestation of the currently ongoing maneuver has to be awaited before the correctness of a trajectory hypothesis can be inspected. This is illustrated in Figure 4.9a, where a lane changing vehicle is depicted. A set of trajectory hypotheses $\boldsymbol{\mathcal{T}}_1$ and $\boldsymbol{\mathcal{T}}_2$ to reach the centerline of each of the lanes $\mathcal{L}_1$ and $\mathcal{L}_2$ have been generated



(a) Classification of ongoing maneuver. In this example the vehicle is following the trajectory hypotheses for $\mathcal{L}_2$ (shown in red) more closely as for $\mathcal{L}_1$ (blue), which would lead to a prediction of a *lane change left* maneuver.



(b) Time matrix of trajectory nearness classification for the case of $j = 5$.

Figure 4.9: Receding horizon trajectory hypotheses classification.

for the time steps from t_{i-j} to t_i. But the trajectories generated at t_i give no indication of the maneuver currently taking place. However, when inspecting the distance of the vehicle to the trajectories generated at t_{i-j} it can be seen that the vehicle is following $\mathcal{T}_{2,i-j}$ more closely than $\mathcal{T}_{1,i-j}$, which indicates a currently ongoing lane change maneuver. The maneuver classification is carried out continuously as new trajectories are predicted via the receding horizon approach (as described in subsection 4.2.3), which is illustrated as a time matrix in Figure 4.9b.
In the linear logistic regression classifier the probability of a vehicle's currently observed trajectory $\widetilde{\mathcal{T}}_i$ being equal to a specific trajectory hypothesis $\mathcal{T}_{i-j}$ is given by

$$\mathrm{P}(\widetilde{\mathcal{T}}_i = \mathcal{T}_{i-j} \,|\, \boldsymbol{f}_{\mathrm{LCD}}) = \frac{1}{1 - \exp(-b - \boldsymbol{w}\, \boldsymbol{f}_{\mathrm{LCD}})} \tag{4.23}$$

with the feature vector $\boldsymbol{f}_{\mathrm{LCD}}$, which here only consists of the lateral distance to the respective trajectory hypothesis denoted by $d_{\mathcal{T}_{i-j}}$ (see Figure 4.9a) and the parameters $\theta = [b,\, \boldsymbol{w}]^T$ defining the logistic regression decision boundary. Due to the fact that the neighborly relations of the lanes relative to the observed vehicle are known (e.g. in this case $\mathcal{L}_1$ is the current lane and $\mathcal{L}_2$ is left to $\mathcal{L}_1$) the probabilities of Equation 4.23 can be mapped to a set of detected lane change maneuvers LCD = {*lane change left*, *keep lane*, *lane change right*}.
As a maneuver is more likely to be observed when the driver has an intention to execute the maneuver (and of course also more unlikely if no motivation to leave the lane is given at all) the lane change detection probabilities (LCD) are scaled with the lane change intention recognition results (LCI) and re-normalized afterwards. Finally, the probabilities to identify the actual lane change behavior (LCB) of the subject vehicle are given by

$$\mathrm{P}(\mathrm{LCB} \,|\, \boldsymbol{f}_{\mathrm{LCD}}, \boldsymbol{f}_{\mathrm{LCI}}) = \alpha\, \mathrm{P}(\mathrm{LCD} \,|\, \boldsymbol{f}_{\mathrm{LCD}})\, \mathrm{P}(\mathrm{LCI} \,|\, \boldsymbol{f}_{\mathrm{LCI}}) \tag{4.24}$$

with the re-normalization constant α.

CHAPTER 5

Learning Model Parameters from Data

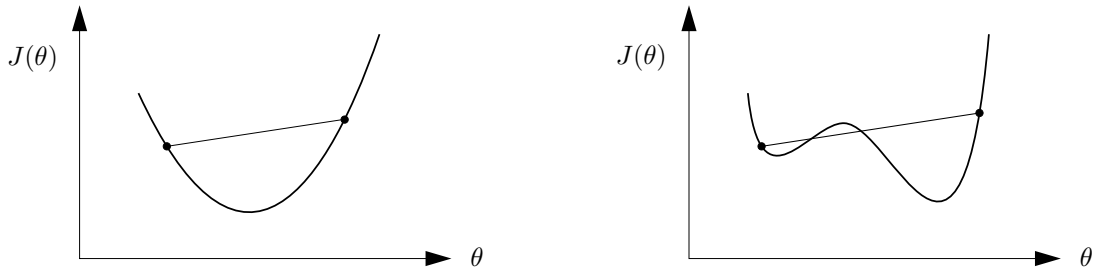
The approaches to a driver's intention recognition and behavior prediction presented so far can be transferred to machine learning problems. In the context of this thesis *learning* means progressively improving the performance on the parameters of the models by means of optimizing an objective function, such that optimal model behavior is achieved based on a set of training examples.
An objective function of a parameter learning problem is in general given as

$$J(\boldsymbol{\theta}, \mathcal{D}) \quad \text{with} \quad \mathcal{D} = \{\boldsymbol{d}[1], ..., \boldsymbol{d}[M]\} \tag{5.1}$$

and with the parameters $\boldsymbol{\theta}$. Thereby a single training data instance is given by $\boldsymbol{d}[m] = [x_1, ..., x_N]^{\mathrm{T}}$ with $x_i[m]$ being the instantiation of the model's variable X_i in the training case m.
The objective function is either going to be minimized if it represents the costs induced by the parameters with respect to some desired system behavior, or it is maximized if the system's state given the parameters is formulated as a reward. For example, in probabilistic models often the log-likelihood (the probability of the parameters explaining the data) is maximized (see section 5.3), whereas in control theory cost functions for the system's states and the controller input can be defined that are to be minimized in order to derive optimal control behavior (see section 5.4). It is noted that maximizing a reward function can always be converted to minimizing a cost function with

$$\max\ J(\boldsymbol{\theta}, \mathcal{D}) = \min\ -J(\boldsymbol{\theta}, \mathcal{D}). \tag{5.2}$$

The advantage of machine learning techniques in general is that parameter tuning by a human expert can be avoided, because on the one hand it is often very complex and time consuming (depending on the model's complexity and the mutual dependency of the parameters' values) and on the other hand the process of manual parameter tuning is prone to errors. Moreover, different experts tend to tune parameters differently as the task is often more subjective rather than objective in nature.



(a) Convex objective function. **(b)** Non-convex objective function.

Figure 5.1: Convex vs. non-convex objective function.

However, a disadvantage of applying machine learning is the often large amount of training examples that has to be handed to the objective function in order to generalize the model's parameters well and to not only fit the parameters for a few data instances exactly (which is called *overfitting*). Furthermore, for many machine learning problems only objective functions exist that are not convex, which practically breaks the guaranty to find the optimal model parameters, because many optimization techniques are only guaranteed to find a local optimum (see Figure 5.1). At least running the optimization multiple times with random initial parameters reduces the risk of getting stuck in a local optimum with relatively high overall parameter costs.
Obviously, as when using machine learning the model's parameters are optimized such that they fit the training data well, it is crucial to have an understanding of how the training data can be acquired, what exactly is represented in the data and which aspects of the domain can not be measured by sensors and thus must be labeled subsequent to the data acquisition process - either automatically or by a human expert. Therefore, subsequently first details regarding the data acquisition and annotation are given (section 5.1 and 5.2) before the approach to learn the parameters of a Bayesian network is presented in section 5.3. In section 5.4 the methodology of inverse reinforcement learning is presented to fit the parameters of the optimal control approach.

5.1 Data Acquisition

In the past several methodologies and concepts have been developed to acquire vehicle and traffic data for different purposes. The most important concepts are categorized in Figure 5.2 and can be distinguished between acquiring data from observation or from simulation. Here the term *observation* is used to refer to measuring the effects of real human driving, while *simulation* always includes synthetic models which serve as an abstraction of the domain to be simulated.
In particular, when acquiring data from observation, driving behavior can be recorded from the perspective of a vehicle moving in traffic, which is often a *test vehicle* equipped with additional environment sensors. Thereby obviously the quality and observability of the data is limited by the technical specifications of the sensors being used. Alternatively a *stationary sensor setup* like cameras or inductive loops built into the road surface can

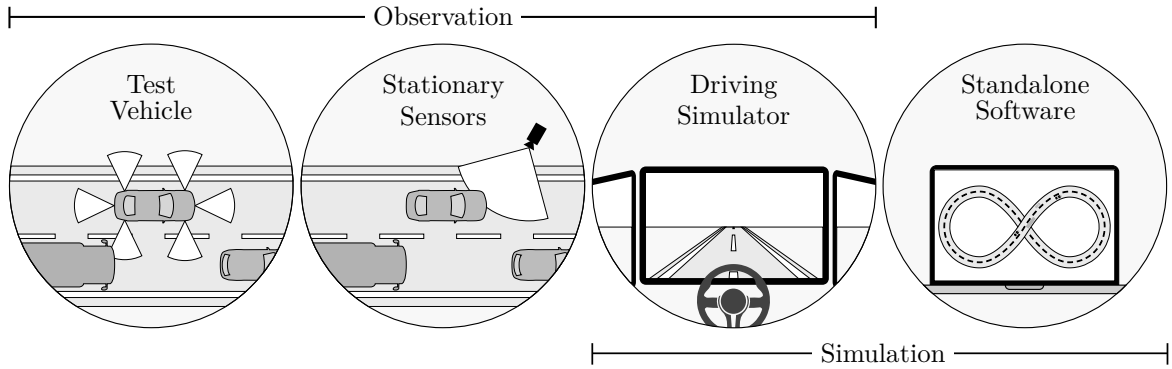


Figure 5.2: Categorization of driving data acquisition methods.

be utilized to obtain the relevant features characterizing a predominant traffic situation. In a *driving simulator* a human driver is acting in a virtual traffic environment via a vehicle mock-up. In this case all features characterizing the traffic situation can be extracted from the simulation and no environment sensors are needed.
Finally, in contrast to a driving simulator, in a *standalone driving simulation software* all elements of the traffic are simulated by synthetic models without any interaction of a human.

In this thesis multiple data sources are used for different purposes:

- **Standalone driving simulation software:** The standalone driving simulation software Pelops is employed for three reasons: First of all, data generated from simulation is free of sensor noise due to the lack of environment sensors. This allows to narrow down possible malfunction sources when testing new algorithms. Second, the simulation model allows to track variables which are usually unobservable through environment sensors. Here the lane change intentions and lane contentedness factors from the driver model are sampled to pre-train the parameters in the Bayesian network for the approach to intention recognition presented in section 3.4. This data can also be used as ground truth when evaluating the algorithms for learning hidden variables in a BN as presented in subsection 5.3.4. Finally, the simulation software does not need to run in real-time. Instead, the speed at which simulated data can be generated is only limited by the available computational resources. This allows to generate a large amount of data in comparatively short time, and this data may also include a lot of instances for very rarely occurring events in real traffic. Pelops is introduced in subsection 5.1.1 in detail.
- **Driving simulator:** The driving simulator combines the advantage of a noise free environment measurement like in the case of a standalone driving simulation with the ability to observe real human driving behavior. Moreover the driver in a simulator can be instructed to carry out secondary tasks without any harm for the driver or traffic participants. Here this advantage has been exploited to investigate the feasibility of manually labeling maneuver intentions through the push of a button while driving, which is discussed in section 5.2. The setup of the driving simulator

is presented in subsection 5.1.2.

- **Test vehicle:** Eventually, the only source of genuine driving behavior is real world traffic. Indeed, all features characterizing the predominant traffic situation are measured by environment sensors, which have a limited range and are always afflicted with noise. Moreover, many variables like a driver's contentedness on a particular lane are intrinsic and unobservable in a test vehicle. But to allow for learning the parameters of a model which represents true driving behavior as realistic as possible, real world traffic data has to be used as training and test data for the parameter learning algorithms presented in section 5.3 and 5.4. Furthermore, it is always advisable to provide the training algorithms with the same data characteristics as those that are prevalent for the model's inference in the target application. In particular, the same sensor setup should be used to collect the training data and to eventually provide observations for variables in the final implementation. Details about the setup of the test vehicle used here are given in subsection 5.1.3.

For the last given argument of advisably not changing the data characteristics between training and inference, data acquisition via stationary sensors is not deemed to be appropriate here, because the sensors being used differ significantly from those in current prototype vehicles for automated driving. For example, while stationary sensor setups like the one used for recording the NGSIM (next generation simulation) data sets provided by (FHWA, 2017) most often consist of vision based systems (i.e., cameras), information from a variety of different sensor types like RADAR, LIDAR and cameras is fused in the test vehicle. This leads to different characteristics like accuracy, sensor noise, measurable features and occlusion in the data, which is inappropriate when using the data for machine learning, because the characteristics of the training data would differ from the characteristics of measurement data being used for the model's inference while driving.

5.1.1 Driving Simulation Software PELOPS

Acquiring driving data from a test vehicle is a time-consuming process. Moreover, the perception of the environment through sensors is always limited and often noisy. In contrast, a standalone driving simulation software like PELOPS (program for the development of longitudinal microscopic traffic processes in systemrelevant environment, see (HOCHSTAEDTER et al., 1999)) is able to generate a large amount of perfectly observable driving data samples in comparatively short time.
Most importantly, as the approach to intention recognition with a Bayesian network (as described in chapter 3) is based on the lane change decision model used by PELOPS, the simulation software itself can be used to generate training data with ground truth for the variables representing the contentment of a driver on a particular lane. This information is obviously not available when using sensor data from a test vehicle. The intrinsic data can be used to pre-train the parameters of the BN and also to evaluate the process of learning hidden variables, which will be presented in subsection 5.3.4.

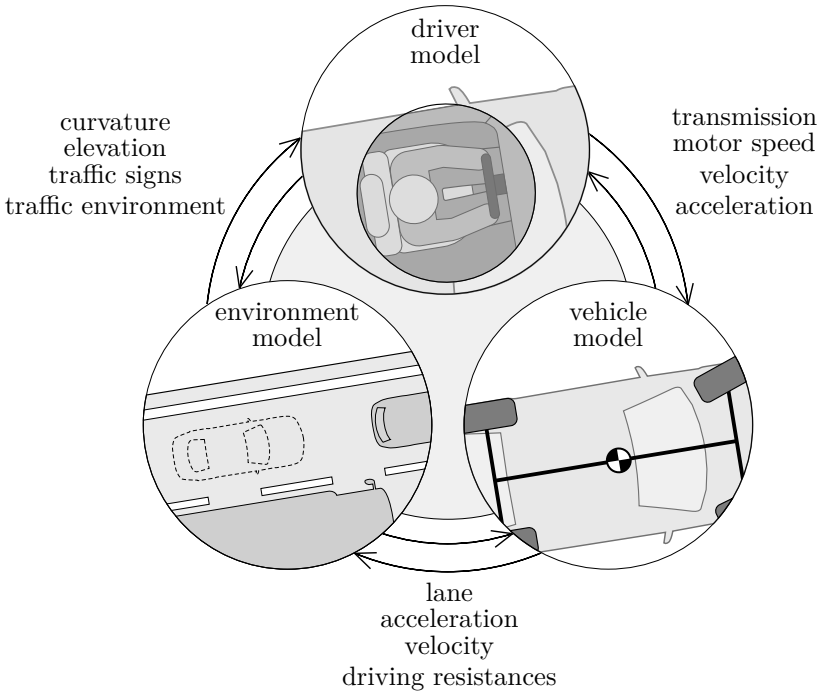


Figure 5.3: Modules of the traffic simulation software PELOPS (EHMANNS, 2002).

PELOPS has been developed at the institute of automotive engineering of the technical university Aachen, Germany, in cooperation with the BMW AG starting from 1988. Since then it has been extended and improved by several contributions, see for example (LUDMANN, 2000; NEUNZIG et al., 2000; EHMANNS, 2002). The aim when developing the simulation software was to provide a tool to investigate the environmental impact, the effect on the traffic flow and a driver's comfort when a vehicle is equipped with driver assistance systems. The software has already been used for a variety of different applications, see (WALLENTOWITZ, NEUNZIG, 1999) for an exemplary overview.

The simulation software is structured into three modules, which are intended to simulate the basic elements of traffic: the driver, the vehicle and the environment (EHMANNS, 2002) (see Figure 5.3). While the environment model entails a detailed description of the road topology including the geometry of the lanes and the signage, the description of the driving behavior is included in the driver model. Thereby the later is subdivided into a behavior generation and an action execution part. In the behavior generation part the maneuver intention is determined based on the perceived traffic situation. The behavior for vehicle following and lane changing is considered separately and is combined only subsequently.

While the longitudinal behavior is generated from the Wiedemann following model described in (WIEDEMANN, 1974), the lane change model follows the approach by (EHMANNS, 2002), which has already been described in chapter 3 and 3.2.1 as it is the inspiring model for the intention recognition approach of the behavior prediction framework.

Finally, a vehicle's viable acceleration is checked with respect to the following constraints:

- Environment conditions that affect the behavior in curves, at traffic signs and at varying lane widths.
- Obstacles and the end of lanes.
- Traffic on adjacent lanes that must not be passed by with a too high relative velocity.
- Vehicles driving on left lanes that must not be overtaken on the right.

The result of the behavior generation model is a desired acceleration and a target lane, which are implemented in the action execution part of the driver model by actuating brake and acceleration pedals, setting a steering wheel angle and activating the indicators of the vehicle model.
In Pelops two types of vehicle models can be employed: On the one hand, there exists a very detailed model called *real* vehicle, which is based on the cause- and effect-method and aims at simulating the vehicle and its dynamics most realistically. In this model the drive force is computed based on the engine's current operating point, the clutch, transmission and differential and finally the tractive and the drive resistance forces are balanced. When the driver adjusts the load (cause) the operating point is changed, which again leads to a change in power and therefore also to a change of speed (effect). Characteristic maps are used to describe the engine's behavior. The lateral vehicle dynamics are represented through a single track model with transient behavior. On the other hand, for the purpose of a faster data generation there also exists a so-called *synthetic* vehicle model which makes some simplifying assumptions about the vehicle. In fact, the synthetic vehicle model only uses a limited set of parameters to model the vehicle. The simulation of the lateral vehicle dynamics is accomplished via a single track model for steady-state skid pad testing.
A variety of parameters can be set to modify the driving characteristics of an individual driver, the vehicle dynamics, as well as environment conditions like the time of day and the weather. Driver-specific parameters are for example a driver's desired speed, need for safety, estimation ability, exhaust of the vehicle acceleration and reaction times. Vehicle dynamics can be controlled e.g. by varying the vehicle's engine performance, the vehicle mass, the maximum deceleration or the drag coefficient. The environment can be influenced e.g. by adjusting the amount of rain, the friction coefficient of the road or the range of vision. The very last is an important parameter when using Pelops for pre-training the BN, because it can be set according to the detection range of the ego vehicle's sensors. That way the BN is not optimized based on environment characteristics that are not observable when using the prediction framework in the test vehicle.

5.1.2 Driving Simulator

Driving simulators have turned out to be a helpful instrument to investigate the human driving behavior. This is not only advantageous to help understanding the human perception and decision process while driving, but also to support the development of new driver assistance systems or even functions for automated driving. While according to (Maas, 2017) in a driving simulator the grade of immersion and the ability to reflect all

(a) Rear view. **(b)** Cockpit.

Figure 5.4: Dynamic driving simulator of the chair of mechatronics at the university of Duisburg-Essen.

cause-effect relationships realistically is always limited and can never represent the reality, the simulators are nevertheless beneficial due to the following reasons:

- **Comparability:** Traffic situations can be created deterministically to assure that every test driver discovers the same environmental conditions.
- **Cost saving:** The need to build a cost-intensive prototype vehicle can be postponed to a point in time when the development of a new technology has advanced already. As developing new vehicular systems in the simulator often means to develop a new software component only, the costs to build expensive prototype vehicles can be reduced.
- **Economy of time:** The saved effort to build expensive prototype vehicles also means a saving of time, because the development of a new, pure software-based component is not only much more affordable, but also faster. A faster development also leads to an early detection of development errors.
- **Risk reduction:** Obviously, system failures or vehicle collisions in a virtual traffic environment do not implicate any damages or even a risk to health. Therefore, in a driving simulator even those new technologies can be tested that may be fraught with risk when testing them in real world driving.

Especially the last point is turned into account here: In a driving simulator test propositi can be asked to label their maneuver intentions while driving without risking any harm arising from distraction. This way data can be acquired together with labels needed for a supervised learning approach at the same time, which will be described in detail in section 5.2.
The driving simulator used is a dynamic driving simulator from the chair of mechatronics at the university of Duisburg-Essen. This human-centered simulator consists of a one-man cockpit that is placed in the center of a circular projection screen. The cabin is

mounted on a moving platform that is able to drive the platform with a frequency of 40 Hz and a maximum acceleration of $2\,g$, which is sufficient to reproduce the majority of the accelerations that a driver experiences in a vehicle in real traffic. The screen covers the driver's field of view entirely by embracing the cabin in an angle of 250°. The simulator is shown in Figure 5.4.

5.1.3 Test Vehicle

To acquire data of authentic human driving behavior a BMW 5 series (F10) test vehicle was driven roughly 9 h (∼1100 km) on German highways and relevant ego vehicle data as well as data from additionally installed proprioceptive sensors has been recorded.
The proprioceptive sensor setup of the test vehicle consists of six LIDAR sensors (covering a 360° field of view around the vehicle) and a monocular vision system mounted behind the front windshield. The vehicle and the sensors are connected to a computer running the robot operating system (ROS) (Quigley et al., 2009) as the vehicle's central framework mainly managing the communication between software components needed for automated driving.
The LIDAR sensors of the vehicle feature a horizontal opening angle of 120° each. The theoretical sensor range is 0.3 m to 200 m with an angular resolution of 1°. An object tracking system embedded on a separate control unit provides high level object data in form of an object list fused from all six LIDAR sensors. The tracking algorithm, which is capable of tracking 65 objects simultaneously at a scan frequency of 25 Hz, uses a Kalman filter with a single track model for the ego vehicle motion compensation. Therefore the ego vehicle's CAN bus data is passed to the control unit via a gateway. The tracking algorithm predicts the objects' states up to 1 s into the future in order to recognize the same vehicle in case of it being covered or hidden for a short period of time.
The monocular vision system captures the scene at a horizontal field of view of 38°. A computer vision software embedded in the sensor tracks objects, lane boundaries and traffic signs in the image. Together with other relevant ego vehicle bus data like the vehicle velocity etc. the high level data delivered by the vision system is forwarded to ROS via a gateway.
Images of the proprioceptive sensor setup are shown in Figure 5.5a to 5.5d, while the sensor range planes are shown in Figure 5.5e and the system setup is depicted in Figure 5.5f.

Vehicle Environment Model

In a standalone driving simulation software or a (dynamic) driving simulator as presented in subsection 5.1.1 and 5.1.2 a synthetic model of the environment is already part of the software. It can be used to answer questions like: What is the curvature of the lane the vehicle is currently driving on? What are possible routes a vehicle can take? How far ahead is the next preceding vehicle?
However, in a test vehicle a so-called *vehicle environment model* (VEM) first has to be generated from the information provided by the environment sensors. The process to

generate an environment model can be broken down to (at least) the following steps:

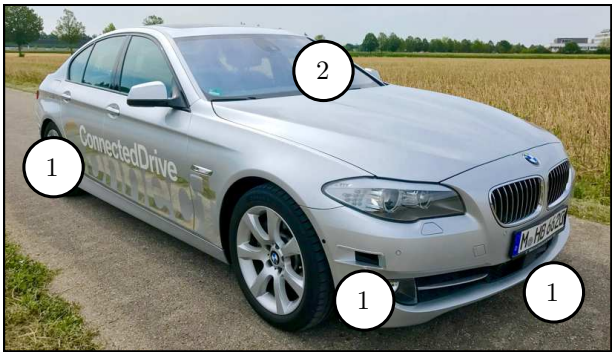
- **Sensor fusion:** Often a multitude of different types of sensors is employed which provides partly redundant and partly complementary information about the environment. For example, objects can in general be detected by RADAR, LIDAR as well as camera sensors, but while only the RADAR sensor is capable of sensing the relative velocity of objects via the Doppler effect directly, the best object classification (e.g. whether an object is a car, truck, bike or pedestrian) is instead provided by the camera. The LIDAR sensor again often yields the most accurate pose estimation. But because of different susceptibilities to failures of the varying sensor types (e.g. in harsh environment conditions), even redundant information is fused to come out with a reliable overall environment detection system.
- **Self localization:** Primarily when a precise digital map of the environment is available in the vehicle, it is obviously a crucial task to accurately estimate the ego vehicle's pose in the global coordinate frame of the map. While systems like the global positioning system (GPS) provide this estimate with an accuracy within the magnitude of a few meters, for the task of automated driving the accuracy needs to be within the magnitude of only a few centimeters to be able to follow the course of a lane precisely. To accomplish this, algorithms are being developed that match the position of memorized landmarks and road boundaries in the digital map with current detections in the vehicle's surrounding via the environment sensors. Moreover, odometry systems are employed to support keeping track of the vehicle's position even in situations where no global positioning signal is available (e.g. in tunnels).
- **Road model estimation:** Like in the case of the sensor fusion, the information about the road course from a digital map needs to be supported by at least one additional source of information. Particularly as the road course may change due to construction sites, current and accurate measurements of the road are crucial for automated driving. Therefore algorithms to estimate the topology and course of the road from lane markings being detected (for example by a camera sensor) are employed.
- **Object lane assignment:** One important step towards situation awareness is the understanding of the relations between individual traffic objects and the road's lanes. This understanding is established by assigning traffic objects to the lanes they are currently driving on, e.g. by checking the geometric overlap of a vehicle and a lane. This object lane assignment is also the basis for determining inter-vehicle relations like determining the preceding or succeeding vehicle.

As many algorithms for the named steps towards an accurate and reliable vehicle environment model are still in active development and have not been available for the implementation of investigations here, the following simplifications and specifics hold for the data acquisition with the described test vehicle:
First of all, for detecting and tracking objects in the vehicle's environment, solely the object list provided by the fusion system of the six LIDAR sensors is used. Algorithms for a sensor

fusion of additional RADAR and camera sensors were not available at a sophisticated level. A quantitative performance evaluation of the particular LIDAR sensors in the test vehicle setup can be found in (Hirsenkorn et al., 2017).
Furthermore, due to a lack of a precise digital map, a self localization of the ego vehicle is obsolete. But all the more a foresighted and accurate road model estimation is necessary to be able to compute all relevant features to characterize the predominant traffic situation as described in section 3.1.
Due to the horizontal field of view of only 38° the camera in the utilized test vehicle is only capable of detecting lane markings of the ego vehicle's current lane. Moreover, the distance at which lane markings can be detected ahead of the vehicle is limited to ∼80 m (provided good weather conditions and an occlusion free view on the lane markings) and the accuracy of the detection degrades with increasing distance. This is why for the investigations in this thesis a non-causal filter algorithm (i.e., a filter which output depends on past, present and also future inputs) has been employed to improve the road model estimation and to mimic the information given by a precise digital map. The use of a non-causal filtering is justified with the fact that a precise digital map is expected to be available in automated driving vehicles in the future, which will make the non-causal filtering obsolete. (In any case, obviously the use of non-causal filtering is only possible when testing in simulation. It cannot be employed in the vehicle while driving.) A detailed description of the process to generate the foresighted road model via a non-causal filter can be found in Appendix C.
Regarding the sensing of traffic vehicles, although the theoretical coverage of the environment sensors is 200 m surrounding the ego vehicle (provided ideal weather conditions and no occlusions), experience with the data provided by the environment model has shown that the quality of object detections farther than 100 m decreases tremendously. Especially the object lane assignment is often erroneous for objects at great distances, because errors of the object pose estimation and lane marking detection add together in the worst case. Therefore the range in which surrounding vehicles are taken into account has been limited to 100 m.

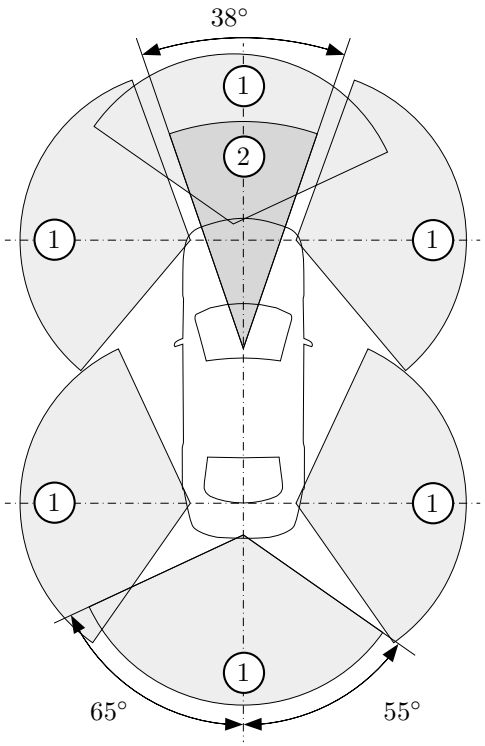
(a) Sensors on the left and rear.

(b) LIDAR sensor.

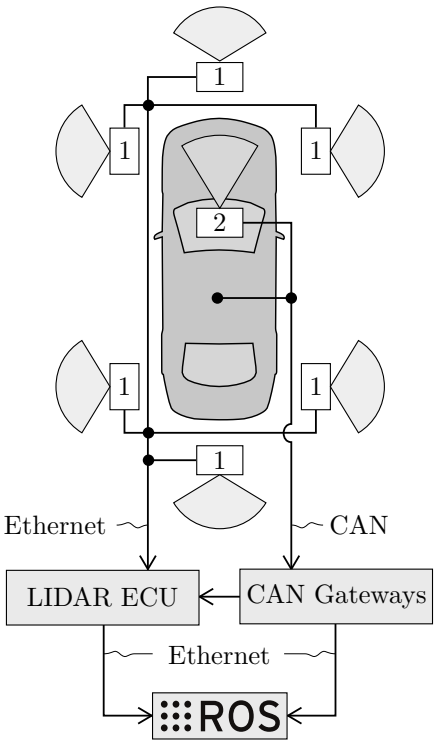


(c) Sensors on the right and front.

(d) Camera sensor.



(e) Sensor range planes.



(f) System setup.

Figure 5.5: Test vehicle system and proprioceptive sensor setup. 1: LIDAR sensor (see (b)). 2: Camera sensor (see (d)).

5.2 Labeling

In general, supervised classification methods need class labels in a data set that the algorithm learns the model parameters from. But the task of labeling is not standardized in any way and many factors have to be taken into account.
First of all, the correct definition of the labels depends on the type of classification model that shall be learned from data. For example, when learning a smoothing model, current observations refer to a label in the past. In a filter model labels indicate the current state of a class, whereas in a prediction model a future class state is labeled (see Figure 5.6).
Moreover, depending on the class that shall be predicted, labels can either be of subjective or of objective type. In case of a maneuver detection approach the lane change label can be defined objectively as there can be set up a clear definition of when a vehicle is matched to a specific lane and when the matched lane changes. Even the start of the lane change at $t_{\text{execution}}$ can be determined in an objective manner by taking into account the lateral movement over the entire duration of the lane change, although it is hard for any algorithm to label this point in time reliably because of sensor noise. For the label of $t_{\text{intention}}$, there exists no distinct definition as the motivation to change the lane depends on a driver's individual awareness of the situation and on the personally accepted minimal time distances to surrounding vehicles. Therefore, the label of $t_{\text{intention}}$ is a subjective label.
In Figure 5.7a and 5.7b the true states (called *ground truth*) of a maneuver intention and execution are shown for two exemplary scenarios. Figure 5.7a depicts the typical lane change scenario motivated by a slower vehicle in front together with the idealized course of the subject vehicle's curvilinear lateral position d, lateral velocity $\dot{d}$ as well as lateral acceleration $\ddot{d}$. Foremost noticeable is the discontinuity of d, which is characteristic for a vehicle's lane change because of the changing lane assignment, which also changes the reference lane for the computation of d. This discontinuity can be used as an indicator for the estimation of t_{change}, which determines the end of both a maneuver intention and execution phase, because it is assumed that no lane change is executed without an intention to do so. After the lane has been changed the intention has obviously been accomplished.
In Figure 5.7b a slightly different scenario is shown, where indeed also a motivation to change the lane due to a slower vehicle in front arises and a lane change maneuver is started, but before the vehicle crosses the lane marking the preceding vehicle leaves the lane. This causes the lane change motivation to vanish and consequently the subject

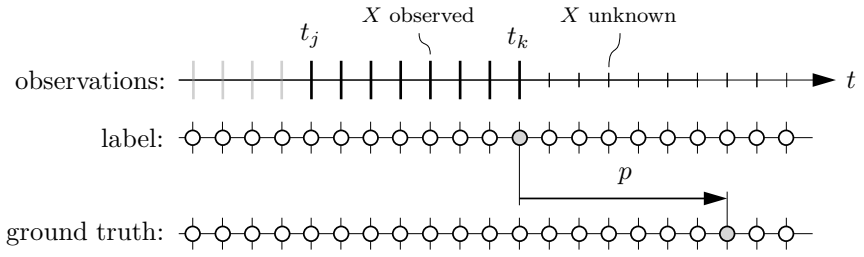
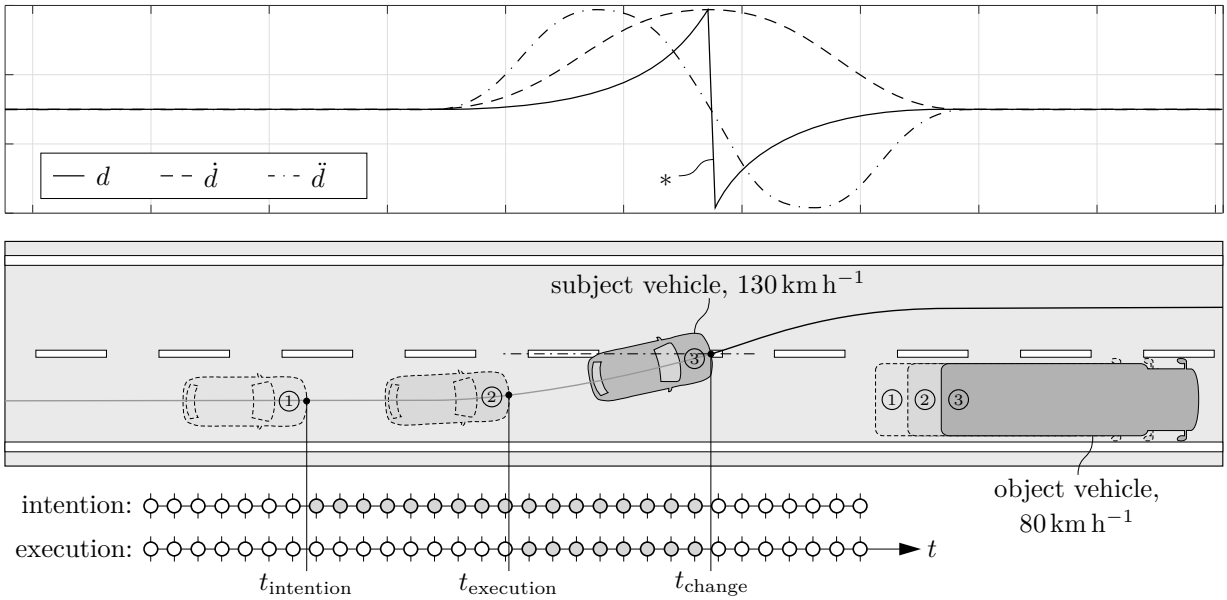
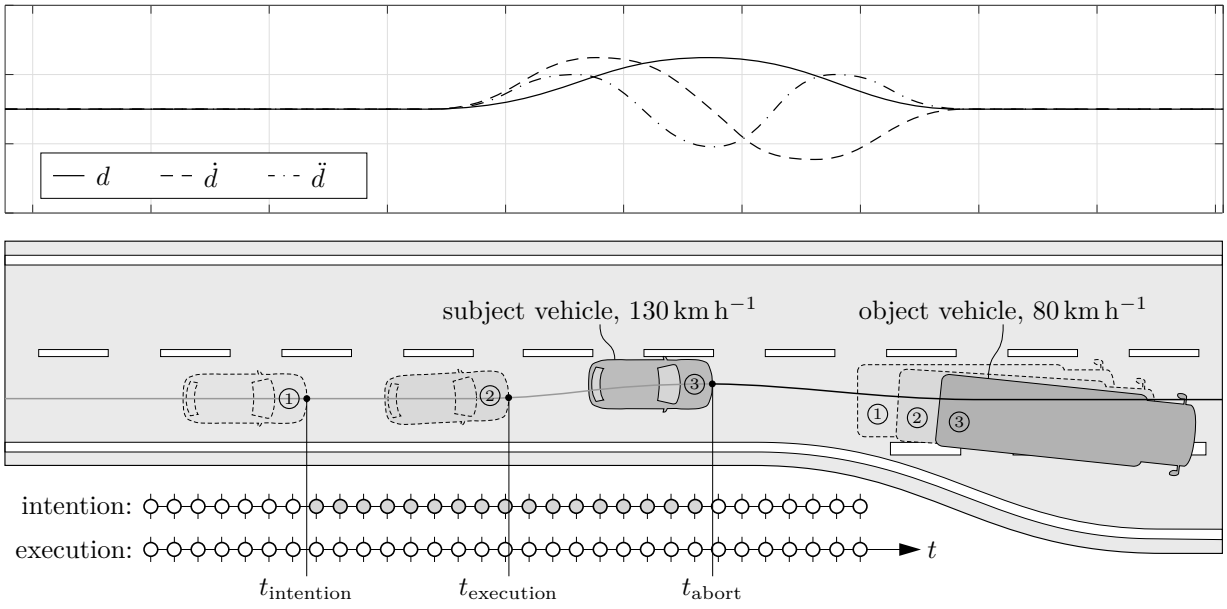


Figure 5.6: Label offset p depending on the model to be learned. For learning a prediction model it is $p > 0$ (as illustrated), for smoothing $p < 0$ and for filtering $p = 0$.



(a) Lane change motivated by a slower vehicle in front. *Characteristic discontinuity of d due to the lane assignment change of the subject vehicle.



(b) Aborted lane change due to a slower vehicle in front leaving the lane.

Figure 5.7: Ground truth of lane change intention and maneuver execution. Grey dots indicate samples labeled as positive (i.e., lane change intention or execution active), white dots indicate negative labels (i.e., intention to keep lane).

vehicle to abort the lane change. This shows that unfortunately using only t_{change} the state of the lane change intention can not be reliably labeled, because the maneuver intention is completely intrinsic. This is even worse in the opposite case: It is always unknown if an intention to keep the lane exists, because at any time a lane change intention could exist without ever seeing its effects in terms of an executed maneuver.
Also, during the investigations of this thesis it turned out that it is hard for any human to label the state of a lane change intention without taking into account the realizability of the maneuver at the same time. It seems that the lines between the individual steps of the human driver's lane change decision making process (emergence of intention, check of realizability and maneuver execution, see also Figure 2.2) are blurry. Therefore, in the following an overview over possible maneuver labeling methods and simplifying assumptions about the presence of a lane change intention is given.

5.2.1 Manual Offline Labeling

Labeling data for supervised learning is often done offline (i.e., outside of the vehicle after the data has been measured) by a human expert who has a deep understanding of the domain the data is taken from. For this purpose relevant driving data is prepared graphically in a way that the expert gains awareness of the predominant traffic situation, which is crucial especially when defining subjective labels like the lane change intention.
Valuable elements of the graphical preparation are images or video streams of the vehicle's environment, a top view of the traffic situation and plots of characteristic features over a sliding time window. Ideally the data can be played back in real-time so the expert is able to see things from the driver's perspective and define the labels as close to the original driver's intentions as possible, because the quality of the labels depends on the grade of immersion into the driving situation. The grade of immersion again depends on the the quality of the recorded data and on the representation of the data.
In this thesis all these elements are provided by a software tool that has been developed especially for the task of labeling. Thereby an expert with domain knowledge can define labels for a maneuver intention and maneuver execution of both the ego vehicle and surrounding traffic participants manually.
The main advantage of manual labeling is that measurement errors (like self-localization errors or false detections of traffic objects) can be recognized by a human expert quickly and can be marked as invalid, so they will not be used for training. Moreover, an expert is able to detect relevant characteristics in the data even if the measurements are noisy.
The disadvantage of the manual labeling process is obviously the time it takes to review all data and define all labels by hand. Thus manual labeling does not scale with the amount of driving data for the training and evaluation of the learning algorithms.

5.2.2 Automatic Offline Labeling

Particular characteristics in the data allow for finding the start and/or the end of a lane change automatically. Once an expert manually sets up a definition of the maneuver

based on these characteristics, more maneuvers matching that definition can be found automatically.
For example, as already described before and depicted in Figure 5.7a, the end of a lane change can be defined as the point in time when the lane assignment of a vehicle changes and the characteristic discontinuity of the vehicle's lateral position with respect to its current lane is observed. To be more robust against noise in the data, additionally the course of the signal before and after a potential lane change is checked, in particular if the vehicle constantly moved towards the lane marking before and away from the lane marking after the change.
Further assumptions about the process of the lane change (e.g. a constant average duration $T_{\text{intention}}$ of the lane change motivation and a constant average duration $T_{\text{execution}}$ of the process to steer towards the target lane) can then be used to compute the start time of the intention or execution with

$$t_{\text{intention}} = t_{\text{change}} - T_{\text{intention}} \quad \text{and} \tag{5.3}$$

$$t_{execution} = t_{\text{change}} - T_{\text{execution}}. \tag{5.4}$$

This automatic labeling via the assumption of a constant lane change duration $T_{\text{execution}}$ has been used in (Schlechtriemen, Wirthmueller, et al., 2015; P. Kumar et al., 2013; Bonnin et al., 2014), where an expert defined the lane change duration. A different approach is to define a maximum entropy for the separation of the two classes $LC = \{lane\ change, keep\ lane\}$ and to compute the mean lane change duration with the help of mutual information like presented in (Rehder, Georgiev, et al., 2015). This concept will be described in the following.
Let D be the set of all values of the lateral position d relative to the lane the vehicle is driving on at time $t_k < t_{\text{change}}$. Presumed every value of N measurement instances of d at time t_k is unique, there exist $N-1$ possible cut-points $s \in S$ to split D in two portions D_1 with $D < s_n$ and D_2 with $D > s_n$. According to (Fayyad, Irani, 1993), every cut-point s_n induces a class information entropy

$$\mathrm{H_I}(D, s) = \frac{|D_1|}{|D|}\,\mathrm{H_C}(D_1) + \frac{|D_2|}{|D|}\,\mathrm{H_C}(D_2), \tag{5.5}$$

where $\mathrm{H_C}(D)$ is the class entropy defined as

$$\mathrm{H_C}(D) = -\sum_i \mathrm{P}(C_i, D)\log_2 \mathrm{P}(C_i, D). \tag{5.6}$$

The cut-point s_n minimizing the class information entropy $\mathrm{H_I}$ is the entropy-optimal cut-point s^* with $\mathrm{H_{I,min}} = \mathrm{H_I}(D, s^*)$. $\mathrm{H_{I,min}}$ can be seen as a quality indicator of the class split, so as long as $\mathrm{H_{I,min}}$ is lower than a defined threshold $\delta_{\mathrm{H,max}}$, D can be split well enough into the two classes of C, which means that the average lane change started even before t_k. Computing $\mathrm{H_{I,min}}$ is iteratively repeated for a decreasing t until $\mathrm{H_I}(D, s^*) >= \delta_{\mathrm{H,max}}$, which marks the average start time $t_{\text{execution}}$ of a lane change relative to $t_0 = t_{\text{change}}$ (see Figure 5.8).

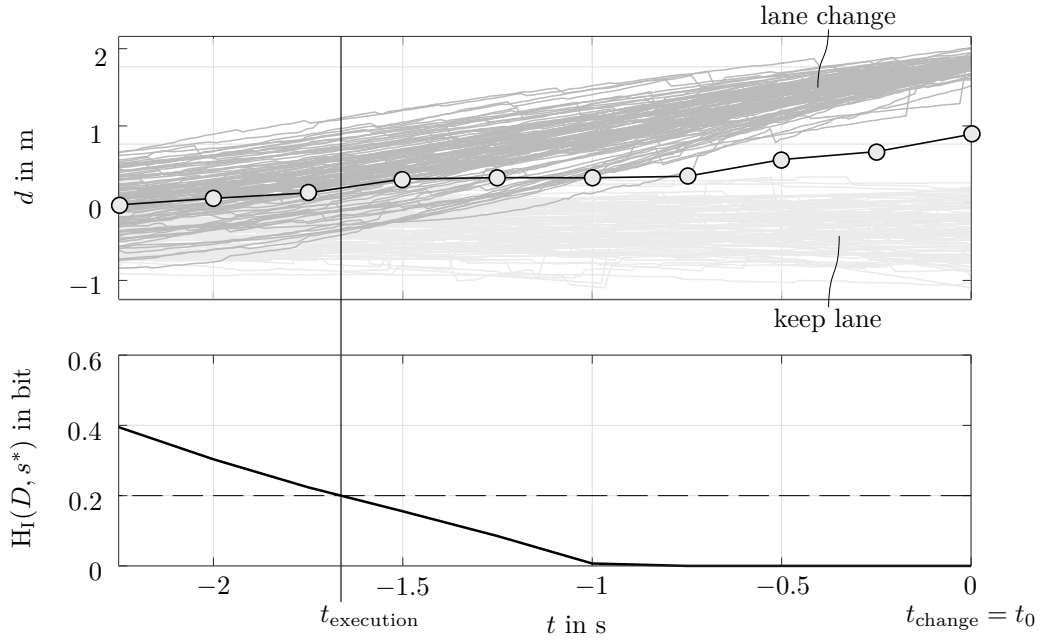


Figure 5.8: Class split entropy of the classes *lane change* and *keep lane*. Top: Lateral position of all recorded sequences. Bottom: Optimal split class information entropy plotted as solid line, entropy threshold $\delta_{\mathrm{H,max}} = 0.2$ as dashed.

The main advantage of the automatic labeling process in general is that it is fast and scales with a growing amount of data that needs to be labeled. Compared to defining the lane change duration for the automatic labeling by hand, the estimation of $T_{\mathrm{execution}}$ with help of class information entropy, the labeling relies more on the true characteristics of the data. The disadvantage of the automatic labeling methods presented here is that all maneuvers are labeled with the same maneuver duration, regardless of the characteristic of the individual maneuver. For example, a slow and a fast lane change are both labeled with the same $T_{\mathrm{execution}}$. Another downside is that there are parameters (the entropy threshold $\delta_{\mathrm{H,max}}$ or the label duration $T_{\mathrm{execution}}$ itself) that need to be tuned manually. Moreover, estimating the class information entropy requires computational costs, that must not be neglected completely (especially for high dimensional feature spaces, i.e., when more features than only d are taken into account).
When using automatic labeling, the labels of maneuver intentions can only be approximated by assuming the presence of an intention in the time of a maneuver execution (plus an additional constant offset to take into account the duration of a lane change realizability check).

5.2.3 Semi-automatic Offline Labeling

The downside of the manual labeling process being very time consuming and the automatic labeling not taking into account the characteristics of a maneuver individually can be improved by combining those two approaches. In particular, the end time t_{change} of a lane change maneuver can be found automatically in the data relatively easy, but the duration

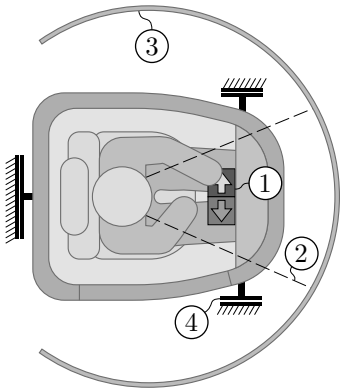
$T_{\text{execution}}$ of an individual lane change is hard to be determined algorithmically.
To overcome this, a human supervisor can be employed to adjust the maneuver start time for each automatically found label instance manually. This way the expert does not need to inspect all data samples to search for instances of lane changes, but instead it can be iterated over the automatically found lane change instances quickly. This can be done for labels regarding the driving behavior of the ego vehicle in the same way as for the behavior of traffic vehicles.

5.2.4 Manual Online Labeling

Sometimes even for an expert it is hard to label data by just seeing recorded videos or analyzing sensor measurements, because driving scenarios can be very complex. To overcome this, it is investigated if labels can be acquired directly while driving. The idea is that the driver can either define labels by recording audio comments or by manipulating buttons. While audio comments need to be converted to signal-based labels retrospectively and are hard to be aligned to the point in time a label is intended to refer to, push-buttons can be evaluated programmatically, which makes the processing a lot easier.
The advantage of labeling while driving is that the expert has a good situational awareness and can define the label authentically. Moreover, no additional time for the labeling process is consumed.
A negative impact is that the driver can only define manual online labels for the own driving behavior. Indeed, in theory at least the co-driver not being loaded with the driving task can define labels even for surrounding traffic vehicles while moving through the traffic in a vehicle taking the measurements, but self conducted tests have shown that it is too hard for the co-driver to coordinate the definition of the label together with the selection of the object vehicle that the label refers to in real-time.
Moreover, the labeling task may distract the driver from driving, which on one hand is safety-critical and on the other hand affects the quality of the labels. Depending on the number of different labels that need to be defined and depending on the complexity of the current traffic situation, the driver can be overexerted with the additional task.
In order to not cause any harm to the driver or other traffic participants, two approaches of labeling the own driving behavior have been investigated: Labeling while driving in a simulator study and labeling while being the co-driver in charge of lane change decisions in a test vehicle study.

Simulator Study

In order to label a driver's lane change intention without any effects from the check of a lane change realizability, it is proposed to conduct a simulator study with a simulated automated driving vehicle. Thereby the simulator automatically follows the course of the lane and keeps a safe distance to preceding vehicles, but does not initiate a lane change by itself. Instead, buttons for the propositus to indicate a lane change intention are used to trigger the automated execution of a lane change, whereby the simulator only changes the



1 | Buttons to indicate a lane change intention.
2 | Propositi's restricted forward facing field of view.
3 | Screen showing the simulated vehicle's environment.
4 | Linear actors moving the vehicle mock up.

Figure 5.9: Setup of the manual online labeling process in the driving simulator.

lane as soon as it is safe. The propositus' field of view on the traffic has to be restricted (i.e., side and rear view mirrors have to be disabled) to make sure that the realizability of a lane change can not be checked by the driver. This setup is illustrated in Figure 5.9.
The advantage of a simulator study for the task of manual online labeling is that due to the driver's restricted field of view perfect lane change intention labels without a mixing of the emergence of an intention and the check of the realizability can be acquired.
In addition, human driving data is acquired while obtaining the labels at the same time, so no extra time is claimed by the labeling process.
A disadvantage is that because of the driver's restricted view not all motivations that cause a lane change intention are perceptible by the driver at the same time. For example, as no vehicles approaching fast from behind and potentially causing an intention to change the lane to the right are seen by the driver, no data is acquired to learn this particular aspect of a driver's lane change behavior. A different study has to be conducted to capture the propositi's lane change intention solely based on observations of the traffic behind. Moreover, the effort of conducting a simulator study is high compared to an automatic labeling process.
Finally, as in a driving simulator the ability to reflect all cause-effect relationships realistically is always limited (as already discussed in subsection 5.1.2) it is questionable if the lane change intention labels acquired in the simulator reflect real-world driving behavior realistically.

Test Vehicle Study

Because of the driving simulator's limited ability to reflect all aspects of real-world driving realistically, it is desirable to do a manual online labeling of lane change intentions in real traffic. But unfortunately, as currently no level 4 (or higher) driving automation system is available to legally conduct a study with the driver triggering an automated execution of lane changes without a human monitoring the safety of the vehicle's actions, it is not possible to use the same setup of the study as in the case of the driving simulator described before. Instead, it is claimed that a co-driver can be instructed to do the labeling instead by ordering the execution of lane changes. The driver imitates an automation function

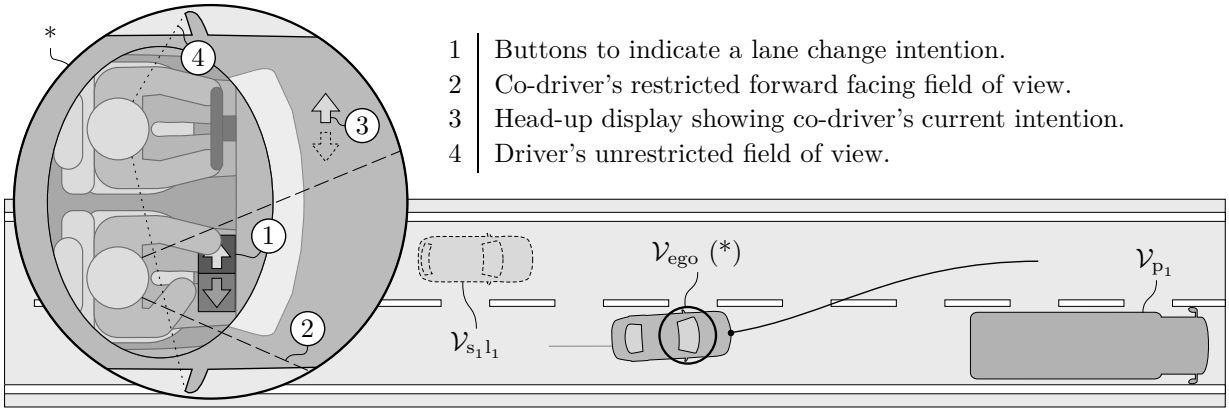


Figure 5.10: Setup of the manual online labeling process in the test vehicle.

and is liable of driving safe (i.e., following the course of the lane, keeping a safe distance to preceding vehicles and checking the realizability of lane changes, but *not* initiating a lane change without the co-driver's order). This way the co-driver can turn full concentration towards the labeling process without the driving safety being affected.

For the implementation of this study the test vehicle described in subsection 5.1.3 is used. The side and rear view mirrors are adjusted to the driver, so the co-driver has a restricted forward facing field of view preventing a check of a lane change realizability (like in the case of the simulator study). During the study the co-driver is instructed to indicate a lane change intention via the press of a button. The current status of a lane change request is displayed to the driver via a head-up display and the driver is instructed to execute the lane change as soon as it is safe. This setup is illustrated in Figure 5.10.

Similar to the simulator study, the advantage of manual online labeling is the simultaneous acquisition of driving data and maneuver intention labels, but in a test vehicle the labels potentially reflect real-world driving behavior more realistically.

On the other hand, the effort of conducting a study is still high compared to an automatic labeling process. Furthermore, it remains an open question whether the traffic situation can be assessed from the co-driver's perspective as good as from the driver's point of view, especially because the co-driver is not in responsibility of the vehicle control.

5.2.5 Summary

The presented labeling strategies all have their strengths and weaknesses with respect to the time that needs to be spent on the labeling process and the quality of the labels. Thereby the latter depends on the validity of the assumptions that the labels are based on, which again clearly affects the prediction performance.

In (Rehder, Münst, et al., 2016b) and (Rehder, Münst, et al., 2016a) the presented labeling strategies have been evaluated for the task of labeling the start of a lane change execution at $t_{\text{execution}}$ and subjectively labeling the start of a maneuver intention at $t_{\text{intention}}$. In particular, in (Rehder, Münst, et al., 2016b) a pilot simulator study with the setup described in section 5.2.4 has been carried out. In 45 min driving time 10 participants (1

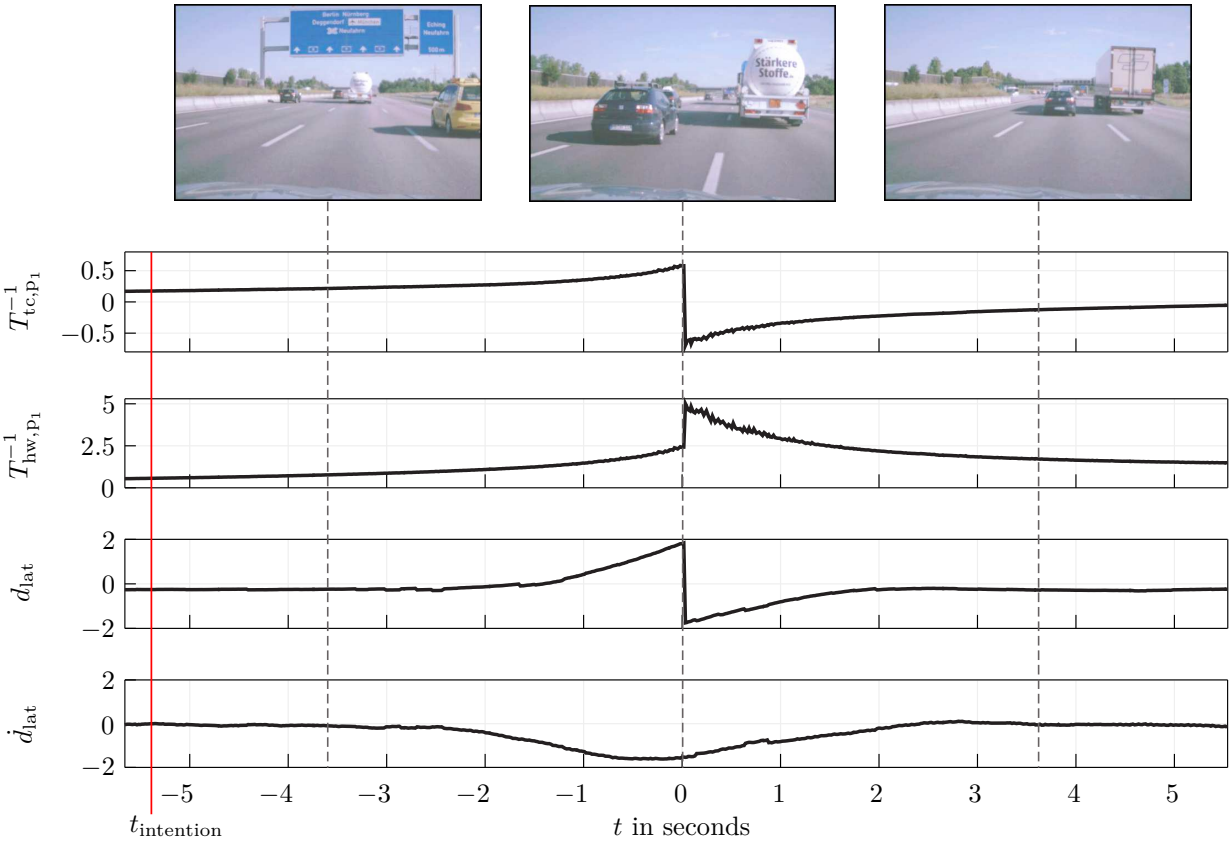


Figure 5.11: Example of a lane change scenario with a manual online label for $t_{\mathrm{intention}}$ (red vertical line). The lane change takes place at $t_{\mathrm{change}} = 0$, the dashed vertical lines mark the time reference for the camera images shown at the top.

women and 9 men) drove 112 km on a simulated highway and labeled their intentions to change the lane. The simulation software executed an intended lane change as soon as its realization was safe.

On the other hand, in (Rehder, Münst, et al., 2016a) a pilot study with 13 participants (3 women and 10 men) as co-drivers for a manual online labeling process in the test vehicle was conducted (according to the setup described in section 5.2.4). Altogether the participants completed 158 min (294 km) of highway driving while labeling their lane change intentions, which the driver of the test vehicle executed as soon as the traffic situation allowed for it. An exemplary traffic scenario with a label for $t_{\mathrm{intention}}$ of the co-driver indicating an intent to change the lane is shown in Figure 5.11. In this case a slower vehicle in front motivated a lane change to the left, which the driver executed 5.3 s later after letting a faster vehicle approaching from behind pass by.

To sum up, the results showed that the effort of conducting user studies for labeling maneuver intentions manually online can not be justified by the corresponding prediction results, at least with the particular setup of the studies in the test vehicle and in the simulator as described before.

Also the semi-automatic offline labeling process as described in subsection 5.2.3 did not sanctify the time that has to be spent on manually defining the start of a maneuver, as

Set	Description	Ego		Traffic	
		Time	Distance	Time	Distance
$\mathcal{D}_{\text{veh}}^{(1)}$	Entire (unlabeled) set of real world driving data collected in prototype test vehicle.	28 h	3200 km	72 h	6000 km
$\mathcal{D}_{\text{veh}}^{(1.1)}$	Subset of set $\mathcal{D}_{\text{veh}}^{(1)}$ with manually defined labels for measurement rejections due to faulty sensor data (see Appendix D).	9 h	1100 km	21 h	2000 km
$\mathcal{D}_{\text{veh}}^{(1.1.1)}$	Subset of set $\mathcal{D}_{\text{veh}}^{(1.1)}$ with manual labels defined for the execution of lane changes of the ego as well as traffic vehicles.	5 h	590 km	12 h	1100 km
$\mathcal{D}_{\text{sim-study}}$	Data set collected in driving simulator during test study.	1 h	110 km	-	-
$\mathcal{D}_{\text{veh-study}}$	Data set collected in prototype test vehicle during test study.	3 h	300 km	-	-
$\mathcal{D}_{\text{PELOPS}}$	Data set gathered from driving simulation software PELOPS	50 h	13 000 km	-	-

Table 5.1: Data sets used for investigations in this thesis.

the prediction results did not outperform the automatic labeling.
Finally, in the test vehicle the best prediction results in terms of a correct classification rate were achieved based on automatic labeling, whereby choosing a maneuver duration based on the class information entropy as described in subsection 5.2.2 can be helpful to find the best assumption of the constant maneuver duration $T_{\text{execution}}$ (or $T_{\text{intention}}$, respectively) for the labeling process.

5.3 Learning Parameters in Bayesian Networks

The overall goal of probabilistic machine learning techniques is to derive and parameterize a probabilistic model for that the probability distribution $\mathrm{P}(\mathcal{X} : \boldsymbol{\theta})$ over the underlying domain's variables $\mathcal{X}$ it represents via the parameters $\boldsymbol{\theta}$ is close to the true probability distribution $\mathrm{P}^*(\mathcal{X})$. In case of machine learning with Bayesian networks this goal decomposes into two parts: Learning the graph structure $\mathcal{G}$ and learning the parameters $\boldsymbol{\theta}$ of the network's CPD. Sometimes the later is also called *system identification*.
Since in section 3.4 expert knowledge has been used to define the structure of a hybrid BN manually, here the parameters of the network's parametric CPD are to be learned from data. An overview of the parameters used in all employed CPD types is given in Figure 3.5 of subsection 3.2.2. The general data set is given by $\mathcal{D} = \{\boldsymbol{d}[1], ..., \boldsymbol{d}[M]\}$ with

M instantiations $\boldsymbol{d}$ of $N = |\mathcal{X}|$ variables measured from observed driving behavior, so $\boldsymbol{d} \in \mathbb{R}^{N \times 1}$. In general, with respect to the observability of each variable represented in $\mathcal{D}$, two cases have to be distinguished:

- **Complete data:** Every variable represented in the model is observed at any instantiation m in the data set $\mathcal{D}$. All variables are called to be *fully observable*.
- **Incomplete data:** In some or all data instances one or more variables may be unobserved. A variable of which only a limited number of instantiations is missing is called *partially observable*, whereas a variable whose value is always missing is called a *latent* or *hidden variable*.

In the following at first the case of learning parameters from complete data is discussed, which is already implemented in the Bayes Net Toolbox (BNT) for MATLAB (MURPHY, 2001). The harder case of learning partially observed and even completely hidden variables in hybrid BN is addressed in subsection 5.3.3 and 5.3.4.

5.3.1 Learning From Complete Data

As the domain's true probability distribution $\mathrm{P}^*(\mathcal{X})$ is unknown, the objective for learning a BN's parameters cannot be formulated as an absolute error between $\mathrm{P}(\mathcal{X} : \boldsymbol{\theta})$ and $\mathrm{P}^*(\mathcal{X})$. Instead, the goal is to find the parameters $\boldsymbol{\theta}$ that explain the samples of recorded driving behavior best. For this case *maximum likelihood* (ML) estimation can be employed, where the learning objective is the *likelihood* L of the parameters given the data. In other words, the parameter values to be found are the ones that most likely generate the samples in the data set $\mathcal{D}$. The higher the score of the likelihood function, the more likely are the parameters to generate the data sampled (i.e., measured) from the true distribution $\mathrm{P}^*(\mathcal{X})$. Often the logarithm of the likelihood is used, because it is more convenient to work with summations rather than with products. This is called the *log-likelihood*, denoted by ℓ.
According to (KOLLER, FRIEDMAN, 2009, pp. 725 sqq.) the conditional log-likelihood of a BN's parameters is given by

$$\log \mathrm{L}(\boldsymbol{\theta} : \mathcal{D}) = \ell(\boldsymbol{\theta} : \mathcal{D}) = \log \prod_{m=1}^{M} \mathrm{P}(\mathcal{X} = \boldsymbol{d}[m] : \boldsymbol{\theta}). \tag{5.7}$$

Using the chain rule this can be written as

$$\begin{aligned} \ell(\boldsymbol{\theta} : \mathcal{D}) &= \log \prod_{m=1}^{M} \prod_{n=1}^{N} \mathrm{P}(y_n[m] \mid \boldsymbol{x}_n[m] : \boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \left[\sum_{m=1}^{M} \log \mathrm{P}(y_n[m] \mid \boldsymbol{x}_n[m] : \boldsymbol{\theta}) \right] \end{aligned} \tag{5.8}$$

with $y_n[m]$ being the m'th instantiation of the network's variable Y_n and $\boldsymbol{x}_n[m] = \mathrm{par}(Y_n)$ being the m'th instantiation of Y_n's parents in the graph.

In Equation 5.8 it can be seen that in the complete data case the likelihood function for a Bayesian network decomposes as a product of local likelihood functions for the individual variables given their parents. This allows to also decompose the parameter estimation problem and to learn the parameters of each involved CPD individually. Thereby learning the parameters means maximizing the log-likelihood function with respect to the CPD's parameters:

$$\widetilde{\boldsymbol{\theta}}_{\mathrm{ML}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta} : \mathcal{D}). \tag{5.9}$$

To reduce the algorithmic complexity (in time and especially in space) of solving Equation 5.9 for $\widetilde{\boldsymbol{\theta}}_{\mathrm{ML}}$, often an intermediate function called *sufficient statistic* is used to compress all information in $\mathcal{D}$ with respect to a particular type of probability distribution. Instead of directly calculating $\widetilde{\boldsymbol{\theta}}_{\mathrm{ML}}$ from operations on the raw data matrix (requiring $\mathcal{D}$ to be kept in memory entirely), the lower dimensional sufficient statistics serve as a compact representation which can be updated on the arrival of a new data instance $\boldsymbol{d}[m+1]$ sequentially. Eventually $\widetilde{\boldsymbol{\theta}}_{\mathrm{ML}}$ is derived from the sufficient statistics. More formally, for all probability distributions in the exponential family the function $s(\mathcal{D})$ is a sufficient statistic if $\mathrm{P}(\boldsymbol{\theta} \mid \mathcal{D}) = \mathrm{P}(\boldsymbol{\theta} \mid s(\mathcal{D}))$, see (Murphy, 2012, p. 74).
For example, following (Murphy, 2012, pp. 73 sq.) for a variable X being Bernoulli distributed (which would correspond to a single, unconnected binary variable in a BN) with $\mathrm{P}(X = 1) = \theta$ and $\mathrm{P}(X = 0) = 1 - \theta$, the likelihood function is

$$\mathrm{L}(\theta : \mathcal{D}) = \theta^{M_{X=1}}(1-\theta)^{M-M_{X=1}}, \tag{5.10}$$

with M being the number of samples in $\mathcal{D}$ (as before) and $M_{X=1}$ being the number of times the outcome is $X = 1$. As described before, to simplify the optimization problem the log-likelihood is used, which is

$$\ell(\theta : \mathcal{D}) = M_{X=1} \log \theta + (M - M_{X=1}) \log(1-\theta). \tag{5.11}$$

Solving Equation 5.11 for the maximum with respect to θ can be achieved in closed form by differentiation and equating the gradient to 0 in order to find the parameters where the log-likelihood function is stationary. This leads to the maximum likelihood estimate

$$\widetilde{\theta}_{\mathrm{ML}} = \frac{M_{X=1}}{M} \tag{5.12}$$

which simply is the frequency of $X = 1$ in the training data. It can be seen that M and $M_{X=1}$ are all that needs to be known about $\mathcal{D}$ to derive θ. On the other hand, θ alone would not be sufficient to update the estimate for a new data instance $\boldsymbol{d}[m+1]$. Therefore M and $M_{X=1}$ are the sufficient statistics of the data. (Another option would be $M_{X=1}$ and $M_{X=0} = M - M_{X=1}$.)
For any discrete node B with purely discrete parents $\boldsymbol{A}$ the CPT's parameter for an individual instantiation $B = b$ and $\boldsymbol{A} = \boldsymbol{a}$ can be computed using the sufficient statistics with

$$\theta_{b|\boldsymbol{a}} = \frac{M_{B,\boldsymbol{a}}}{M_b}, \tag{5.13}$$

whereby the counts are defined as

$$M_z = \sum_{m=1}^{M} \mathbf{1}\left\{Z[m] = z\right\}, \tag{5.14}$$

see (Koller, Friedman, 2009, p. 724). The notation $\mathbf{1}\{\cdot\}$ is used to refer to the indicator function, which is 1 if its argument is true and 0 otherwise.
While for CLG CPD (including the special cases of CG and LG CPD) the sufficient statistics and a closed-form solution to Equation 5.9 are given in (Murphy, 1998), unfortunately the parameters of a (C)CD CPD cannot be compressed in sufficient statistics because the softmax distribution is not in the exponential family. Moreover, no closed form solution for the parameter optimization is available. According to (Murphy, 2012, pp. 252 sq.) the log-likelihood function of the softmax model is given by

$$\ell(\boldsymbol{\theta} : \mathcal{D}) = \log \prod_{m=1}^{M} \prod_{k=1}^{K} \mathrm{P}(a[m] = k \mid \boldsymbol{x}[m] : \boldsymbol{\theta})^{\mathbf{1}\{a[m]=k\}} \tag{5.15a}$$

$$= \sum_{m=1}^{M} \sum_{k=1}^{K} \mathbf{1}\left\{a[m] = k\right\} \log \mathrm{P}(a[m] = k \mid \boldsymbol{x}[m] : \boldsymbol{\theta}) \tag{5.15b}$$

$$= \sum_{m=1}^{M} \sum_{k=1}^{K} \mathbf{1}\left\{a[m] = k\right\} \log \frac{\boldsymbol{b}_k + \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}[m]}{\sum_{k'=1}^{K} \boldsymbol{b}_{k'} + \boldsymbol{w}_{k'}^{\mathrm{T}} \boldsymbol{x}[m]} \tag{5.15c}$$

with $\boldsymbol{x} = \mathrm{Par}(A) \cap \boldsymbol{\Gamma}$ and $K = |\,\mathrm{val}(A)|$. Intuitively, through the indicator function in Equation 5.15 only the probabilities matching the respective class assignment add to the likelihood score. In particular, if the class $a = k$ is observed but not predicted with probability 1.0, a negative score will be added because of the logarithm.
Equation 5.15 is also called the *cross-entropy error* function. Its global maximum can be found using standard optimization algorithms like gradient ascent.

5.3.2 Parameter Tying

The specific structure of the BN for inferring a driver's maneuver intention presented in section 3.4 has an important property: The sub-structure for the estimation of a driver's contentedness on an individual lane repeats for all three lanes that are taken into account (which is the own lane, the left and the right lane). The reason is that the way the lane contentedness is estimated should be independent of any lane in particular. Indeed, when running inference in the network, available evidence is taken into account for each lane individually. But the quantification of the observed variables' influence on the driver's contentedness on each lane should be identical for all lanes.
This property motivates to use the same CPD parameters for every sub-structure, which is called *parameter tying*. Tying parameters effectively reduces the number of parameters that need to be learned from data. At the same time this implies an increased amount of data that can be used to learn the parameters of an individual CPD. In particular, whereas without parameter tying the CPD's parameters in a single sub-structure could solely be estimated based on data measured on the single corresponding lane, with parameter tying

data from all lanes can be used to estimate a single sub-structure's parameters. Eventually for the task of inference the same parameter estimates are used in all repeating sub-structures.
For the BN structure presented in Figure 3.15 the number of free parameters (i.e., parameters that need to be learned from data) is 1641 (see Figure 3.5 for an overview of the number of parameters depending on the type of CPD used). Parameter tying for all CPD in the sub-structures for each lane reduces the number of parameters to 561.

5.3.3 Learning Partially Observed Variables

According to (Koller, Friedman, 2009, pp. 869 sqq.), when learning the parameters of a hybrid BN from incomplete data, actually two tasks have to be solved at once: A distribution over the possible outcomes of each unobserved variable has to be inferred to complete the training data, and based on the data of course parameter estimates for the network's CPD have to be computed.
With the inference procedure for hybrid BN presented in subsection 3.3.5 and the formulas for the parameter optimizations in the case of complete data as described in section 5.3 it seems that each of the tasks can be achieved given the result from the other: Given the parameters the clique tree algorithm can be used to query the value of any unobserved variable in the network (to complete the missing data), whereas given complete data the sufficient statistics and the cross-entropy error can be used to estimate the network's parameters.
The *expectation maximization* (EM) algorithm solves this circular dependency iteratively by first inferring a probability distribution over the values of unobserved variables (which is called the *expectation step* or *e-step*) and then computing a maximum likelihood estimate of the parameters similar to the case of complete data (called *maximization step* or *m-step*). The first e-step is based on a set of random initial parameters and it can be shown that each iteration of EM is guaranteed to improve the log-likelihood until convergence, see (Koller, Friedman, 2009, pp. 870 sq.).
Unfortunately, while the clique tree algorithm as presented in subsection 3.3.5 can be applied in the e-step straight forward, two difficulties arise in the m-step: First, in the e-step a posterior probability distribution rather than a single most likely value of each hidden variable is inferred, so the sufficient statistics can not be computed like in the case of complete data. To overcome this, the standard approach is to compute so-called *expected sufficient statistics* (ESS), which can be seen as using soft estimates for all unobserved variables' values (Koller, Friedman, 2009, pp. 870 sqq.). For example, when computing the maximum likelihood parameters of a CPT according to Equation 5.13 the definition of the counts changes from Equation 5.14 to

$$M_z = \sum_{m=1}^{M} \mathrm{P}(Z = z \mid \boldsymbol{d}[m] : \boldsymbol{\theta}). \tag{5.16}$$

The expected sufficient statistics for CLG CPD are presented in (Murphy, 1998).

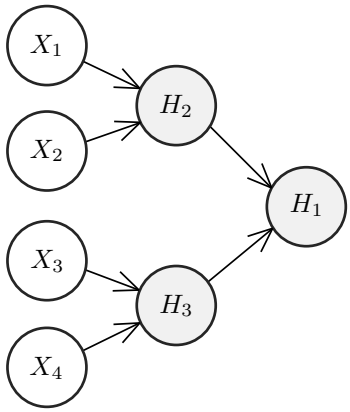
The second difficulty of the m-step in case of partially observed data is that the ML estimates of any CD CPD's parameters can not be computed using Equation 5.15c, because $\boldsymbol{x} = \text{par}(A)$ may be unobserved. The solution proposed in (König, Rehder, 2016) is inspired by the procedure to incorporate (C)CD CPD in the clique tree algorithm: In every iteration of the gradient ascent algorithm for maximizing the log-likelihood of the softmax distribution, the marginal probability of the softmax node is computed by

$$\mathrm{P}(A = a) = \int_{-\infty}^{\infty} \mathrm{P}(A = a \,|\, \boldsymbol{x})\, \mathrm{P}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, \qquad \text{(3.42 revisited)}$$

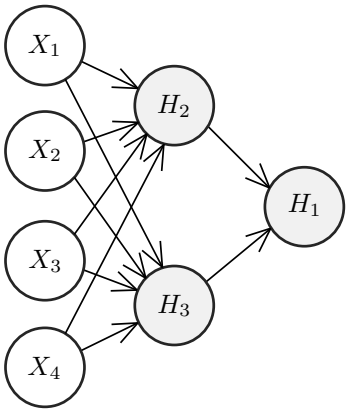
which requires numerical integration to be solved. This marginal can be used in Equation 5.15b to compute the log-likelihood score of the softmax distribution.
While the Bayes Net Toolbox (BNT) for Matlab (Murphy, 2001) already offers an implementation to learn parameters from partially observed data in the case of CPT and CLG CPD (including LG and CG CPD), the proposed methodology to also obtain parameter estimates for (C)CD CPD with unobserved parent values has been implemented additionally. Pseudocode for the EM algorithm in case of hybrid BN is presented in section A.3.

5.3.4 Learning Hidden Variables

Hidden variables (also latent variables) are variables for that no evidence can be acquired when running inference in the model. In case of the BN for recognizing maneuver intentions described in section 3.4, all variables representing lane contentedness values and contentedness factors can never be obtained from any sensor measurement. (Note that here the variable representing the lane change intention itself is not considered to be a hidden variable, because during the training phase labels for the lane change intention are defined in terms of a supervised learning approach.)
In general, even when a variable is never observed in the training data set, the expectation maximization algorithm as presented in subsection 5.3.3 can still be used to find a maximum likelihood estimate of the latent variables' parameters. But as for the task of parameter learning in a BN usually many local optima for the optimization of a single probability distribution exist, it cannot be guaranteed that the hidden variables still have any semantic meaning in the context of the problem domain.
As stated by (Koller, Friedman, 2009, pp. 930 sqq.), methods to initialize a hidden variable so as to guide the algorithm towards semantically reasonable regions of the parameter space are generally heuristic and no guarantees can be given. Here, two measures are taken to learn hidden variables with defined semantic meanings:
First, when a hidden variable is connected to only those nodes that indeed have a semantic influence on the intended meaning of the hidden variable, the parameterization of the hidden variable is significantly biased and therefore the risk of learning parameters that are meaningless and only capture the noise in the training data is reduced (Koller, Friedman, 2009, p. 931). See for example the two BN structures in Figure 5.12. While the structure in Figure 5.12a is constrained such that for example H_2 can only depend on X_1 and X_2, the structure in Figure 5.12 is unconstrained (in this case the layers of the network are fully connected) and H_2 may be influenced by all observed features X_1 to X_4.



(a) Non-overlapping, constrained structure.



(b) Overlapping, unconstrained structure.

Figure 5.12: Comparison of network structures with hidden variables.

The BN for a driver's intention recognition presented in section 3.4 exactly follows the idea of a constrained structure. In particular, the node representing the overall contentedness on a single lane is connected to only those contentedness factors regarding this same lane. The contentedness factors again are connected only to those features having a semantic influence on a single contentedness factor.

The second measure to learn semantically meaningful hidden variables is a pre-training of the model's parameters based on simulated data. That is, an expert model including all variables and dependencies of the domain to be represented (here: the driver model in PELOPS) is used to sample training data, which again is used to learn the parameters $\widetilde{\boldsymbol{\theta}}_{\text{sim}}$ of the BN's CPD. As all variables are modelled explicitly in the simulation, even data for variables being otherwise hidden can be sampled. The parameters from the pre-trained model are then used as initial parameters for the training of the final model, resulting in the parameters $\widetilde{\boldsymbol{\theta}}$.

Of course, the parameters of the simulation model are chosen by an expert, so it is not guaranteed that the model correctly quantifies the variables' dependencies compared to observed human driving behavior. But it is believed that using the parameters from pre-training as initial parameters for the final model ($\boldsymbol{\theta}_{\text{init}} = \widetilde{\boldsymbol{\theta}}_{\text{sim}}$), the chance of the likelihood function converging in a local optimum in the parameter region with a semantic meaning is high. This idea is visualized in Figure 5.13.

Unfortunately, the true likelihood function of the BN cannot be plotted like it is shown in Figure 5.13, because the parameter space is high dimensional. Thus it is hard to examine if the presented (heuristic) measures indeed maintain the semantic meaning of hidden variables. Especially when the true distribution of the problem domain (represented by the measurement samples) differs from the simulation model significantly, the same semantic meaning of the hidden variables may be located in a completely different parameter region with vastly differing conditional probability distributions. As an aid the Mahalanobis distance can be employed to measure the distance of the ground truth samples generated by PELOPS to the learned probability distributions, which will be described in subsection 6.1.3

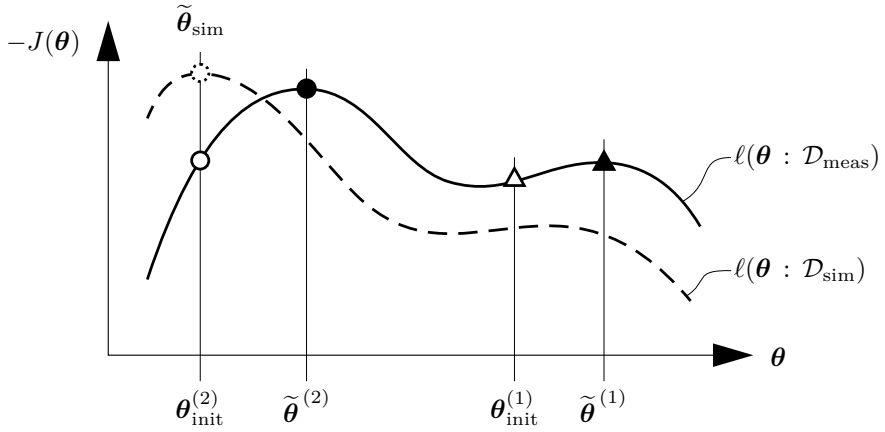


Figure 5.13: Illustration of parameter initialization from pre-training (parameters indicated by superscript $^{(2)}$) compared to random initialization (indicated by superscript $^{(1)}$). While the learned parameters $\widetilde{\boldsymbol{\theta}}^{(2)}$ stay within the region of $\widetilde{\boldsymbol{\theta}}_{\text{sim}}$, the initial parameters $\boldsymbol{\theta}_{\text{init}}^{(1)}$ are chosen randomly, which may lead to $\widetilde{\boldsymbol{\theta}}^{(1)}$ being far from $\widetilde{\boldsymbol{\theta}}_{\text{sim}}$.

in detail. But at the end a human expert is needed to investigate the true sense of the hidden variables.

5.4 Inverse Reinforcement Learning

In section 4.2 a linear-quadratic controller was presented that can be used to generate trajectory hypotheses to predict the future movement of vehicles. The weight matrices $\boldsymbol{Q}$, $\boldsymbol{P}$ and $\boldsymbol{R}$ that define the behavior of the controller via the cost functional

$$\begin{aligned} J(\boldsymbol{x}_0, \mathbf{u}) = (\boldsymbol{x}_N - \boldsymbol{x}_{\text{ref}})^T \boldsymbol{P} (\boldsymbol{x}_N - \boldsymbol{x}_{\text{ref}}) \\ + \sum_{k=1}^{N-1} (\boldsymbol{x}_k - \boldsymbol{x}_{\text{ref}})^T \boldsymbol{Q} (\boldsymbol{x}_k - \boldsymbol{x}_{\text{ref}}) + \sum_{k=0}^{N-1} \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k \end{aligned} \quad \text{(4.12 revisited)}$$

were chosen manually to show the working principle of using the LQR approach for the generation of trajectories.
In general, manually choosing the weight matrices via trial and error is common practice for designing the LQR. Indeed, in the literature effort has been spent to algebraically select the weights (see for example (V. E. Kumar, Jerome, 2014; Ata, Coban, 2017)), but in the end these approaches just transfer the task of directly choosing $\boldsymbol{Q}$ and $\boldsymbol{R}$ to manually choosing other application specific objectives that are linked to these weights.
However, for the task of predicting driving behavior it is *not* expedient to parameterize the weight matrices of the LQR approach by an expert who optimizes the controller parameters to an individual objective. Instead the goal is to ascertain the (unknown) optimization objective of a given, observed system; namely the drivers of the surrounding vehicles of that the driving behavior shall be predicted. This task is also called *learning from demonstration*, *imitation learning* or *apprenticeship learning*. Specifically *inverse reinforcement*

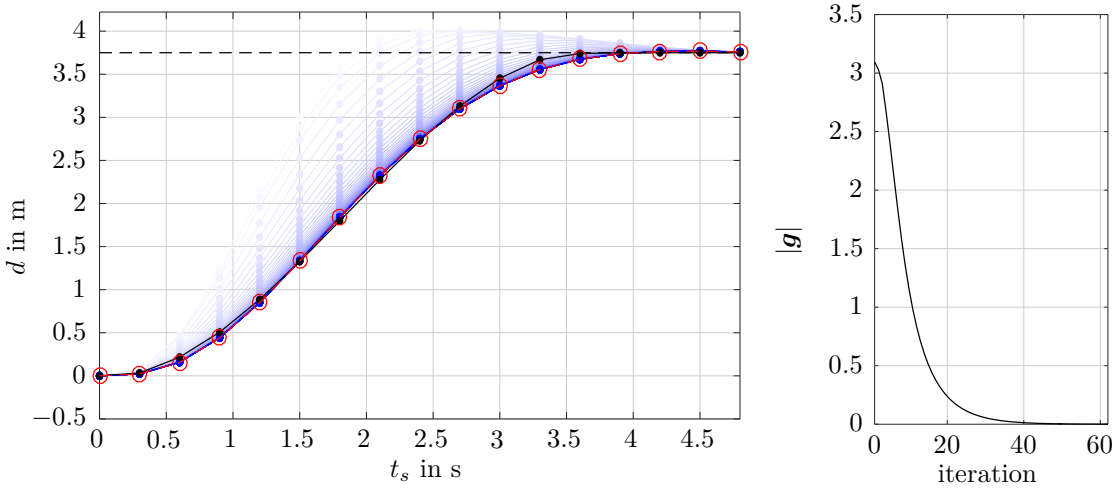
learning (IRL) was found to exactly fulfill the described problem: Recovering the unknown reward function of an agent assumed to be performing optimal behavior (Abbeel, Ng, 2004).
Unfortunately, for any learning algorithm finding the optimal parameters gets harder with an increasing degree of freedom of the underlying system. Consequently only the least possible amount of parameters not excessively limiting the solution space should be exposed to the IRL algorithm. Consequently a variety of rules and presuppositions to reduce the number of free parameters in the LQR formulation is introduced in the following.
First of all, according to (Lunze, 2014, pp. 308 sqq.) in systems with a single input even under weak conditions only the diagonal elements of $\boldsymbol{Q} = \mathrm{diag}(q_1, q_2, q_3)$ have to be selected without restricting the solution space. In addition, to incorporate foreknowledge about the system's stability, the lower bound of the diagonal elements in $\boldsymbol{Q}$ can be set to 0, which ascertains that the learned state penalty is positive definite.
The final state penalty is chosen to be $\boldsymbol{P} = 100 \cdot \boldsymbol{Q}$, which showed good results in practice to assure that the controller equalizes the divergence from the reference trajectory within the optimization horizon. Moreover, as many pairs of $\boldsymbol{Q}$ and $\boldsymbol{R}$ exist that yield the same control law, it is common practice to choose either $\boldsymbol{Q}$ or $\boldsymbol{R}$ to be the identity matrix $\boldsymbol{I}$ and only tune the other weight matrix, respectively, see (Murray, 2006). Therefore here $\boldsymbol{R}$ has been chosen to 1.
Finally, as for the task of a behavior prediction it is sufficient to learn the characteristics of the subject vehicle's future positions (rather than the lateral velocities and accelerations), the state penalty is simplified to $\boldsymbol{Q} = \mathrm{diag}(q, 1, 1)$, so the task of learning the cost parameters breaks down to finding the optimal value of q for which the state trajectory matches the demonstrations.
As data basis n trajectories to learn the parameters from are given by $\widetilde{\mathbf{x}}_1, ..., \widetilde{\mathbf{x}}_n$ and each of the time discrete trajectory demonstrations $\widetilde{\mathbf{x}}_i$ has been subsampled to have the same number of samples N and sample size T_s as stated in the LQR problem formulation.
For each trajectory a feature vector is computed that describes the characteristics of the trajectory at each point in time and serves as a similarity measure to learn a model of the trajectories and generate similar trajectories in new situations. In this approach the sum of squares of each element of the state vector $\boldsymbol{x} = [\, d, \dot{d}, \ddot{d}\,]^T$ is used, so the empirical feature vector to represent the characteristics of a single observed trajectory is

$$\boldsymbol{f}(\widetilde{\mathbf{x}}_i) = \sum_{k=0}^{N} \widetilde{\mathbf{x}}_{i,k}^2. \tag{5.17}$$

For a multitude of demonstrated trajectories the mean of the feature values is taken, which is denoted by

$$\widetilde{\boldsymbol{f}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{f}(\widetilde{\mathbf{x}}_i) \tag{5.18}$$

with $\widetilde{\mathbf{x}}_i$ being the i'th demonstrated instance of all n trajectories.
According to (Kuderer, Gulati, et al., 2015) the goal is to find the parameters in $\boldsymbol{Q}$ that yield a conditional probability distribution over trajectories $\mathrm{P}(\mathbf{x} \,|\, \boldsymbol{Q})$ for that the expected



(a) Lane change trajectories.

(b) Learning curve.

Figure 5.14: Learning iterations of the maximum entropy IRL algorithm. The demonstrated lateral position trajectory is plotted solid black while the trajectories generated by the LQR in every learning iteration are plotted in shades of blue. The reference trajectory is shown as dashed black, the optimal trajectory found at convergence is plotted dashed red.

feature values match the observed feature values, so it is required that

$$\mathbb{E}_{\mathrm{P}(\boldsymbol{x}\,|\,\boldsymbol{Q})}[\boldsymbol{f}] \stackrel{!}{=} \widetilde{\boldsymbol{f}}. \tag{5.19}$$

In general there exists not a unique distribution for that Equation 5.19 is true. In (Ziebart et al., 2008) it is proposed to use the one that maximizes entropy, which is justified with the fact that this distribution indeed matches feature expectations, but it does not favor any other outcome besides this constraint, so while maximizing entropy any other amount of prior information is minimized.
In (Kuderer, Kretzschmar, et al., 2012) it is shown that maximizing the entropy corresponds to maximizing the likelihood of the training data when assuming an exponential family distribution. The resulting optimization problem cannot be solved in closed form, but the gradient $\boldsymbol{g}$ given by the difference between the expected and the empirical feature values

$$\boldsymbol{g} = \mathbb{E}_{\mathrm{P}(\boldsymbol{x}\,|\,\boldsymbol{Q})}[\boldsymbol{f}] - \widetilde{\boldsymbol{f}} \tag{5.20}$$

can be used to employ an iterative optimization technique like gradient descent.
Unfortunately the computation of the expected feature values is very expensive as it is hard to compute the high dimensional integral

$$\mathbb{E}_{\mathrm{P}(\mathbf{x}\,|\,\boldsymbol{Q})}[\boldsymbol{f}] = \int_{\mathbf{x}} \mathrm{P}(\mathbf{x}\,|\,\boldsymbol{Q})\boldsymbol{f}(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{5.21}$$

This is why (Kuderer, Kretzschmar, et al., 2012) suggests to use the approximation of only taking the expected features of the most probable trajectory, which leads to

$$\mathbb{E}_{\mathrm{P}(\mathbf{x}\,|\,\boldsymbol{Q})}[\boldsymbol{f}] \approx \boldsymbol{f}(\operatorname*{argmax}_{\mathbf{x}} \mathrm{P}(\mathbf{x}\,|\,\boldsymbol{Q})). \tag{5.22}$$

To summarize, the approach for learning the optimal state penalty $\boldsymbol{Q}$ from the trajectory demonstrations $\widetilde{\mathbf{x}}_1, ..., \widetilde{\mathbf{x}}_n$ is as follows:

1. The empirical feature vector $\widetilde{\boldsymbol{f}}$ is computed according to Equation 5.18 for all demonstrated trajectories.
2. The state penalty is initialized with $\boldsymbol{Q} = \text{diag}(1, 1, 1)$.
3. The optimal control problem is solved for every demonstrated trajectory $\widetilde{\mathbf{x}}_i$, which leads to an optimal state trajectory $\mathbf{x}^*$ according to the current state penalty $\boldsymbol{Q}$.
4. The expected feature vector $\mathbb{E}_{\text{P}(\mathbf{x}^* \mid \boldsymbol{Q})}[\boldsymbol{f}]$ of the optimal state trajectory is computed using Equation 5.22.
5. The gradient $\boldsymbol{g}$ of the features is computed according to Equation 5.20. If $|\boldsymbol{g}| < \epsilon$ then $q_j = \boldsymbol{Q}(j, j)$ with $j \in 1, ..., \dim(\boldsymbol{Q})$ is updated by taking a gradient descent step with $q_j = q_j - \alpha \cdot \boldsymbol{g}$ (whilst taking into account the lower parameter bound of 0).
6. Steps 3 to 5 are repeated until convergence.

The pseudocode for the outlined algorithm is presented in section A.4. A demonstration of how the IRL algorithm minimizes the distance to a single demonstrated trajectory during the learning iterations is shown in Figure 5.14. The exemplary demonstration is plotted solid black, the predicted trajectory of the LQR in every training iteration is plotted with shades of blue, starting from light blue with the initial penalty matrix $\boldsymbol{Q} = \text{diag}(100, 1, 1)$ and turning to a darker color as the training evolves and $|\boldsymbol{g}|$ converges to 0. The final trajectory is plotted dashed red with circles marking the discrete time steps every $t_s = k \cdot T_s$ seconds ($T_s = 0.3\,\text{s}$, $k = 0, ..., N-1$ with $N = 20$).

CHAPTER 6

Evaluation

According to (Gipps, 1986) it is not possible to validate a model. Indeed, finding a fault means the model is incorrect, but not finding a fault does not prove the model is correct as the fault may just be undetected. For automated driving this poses a great challenge, because the health of the vehicle occupants as well as of other traffic participants is at risk if the automation system shows erratic behavior.
One way to counter this challenge is to test the holistic system as much as possible, covering as many different driving scenarios and traffic situations as possible in order to reduce the risk of *not* encountering system deficiencies - if there are any. But unfortunately, as many building blocks for an automated driving vehicle are still in active development, here the proposed approaches to a maneuver intention recognition and driving behavior prediction presented in chapter 3 and 4 can not be tested in a holistic system in closed-loop driving. Therefore, the approach taken in the following is to evaluate the prediction models based on a large amount of driving data isolated against different ground truth hypotheses.
It has to be noted that in fact human driving behavior can never be predicted without any faults. Classifying a driver's maneuver intention or estimating the future vehicle trajectory is always afflicted with uncertainty, not least because human behavior itself is not deterministic. Nevertheless, inaccurate predictions not necessarily lead to an erratic behavior of the automated vehicle at the end. For example, depending on the particular traffic situation, the driving strategy of the vehicle may account for the uncertainty in the predictions and compute an evasive vehicle trajectory as a backup plan as soon as a traffic participant's observed behavior diverges from its prediction. Such an evasive maneuver may indeed come at the cost of discomfort for the passengers compared to a maneuver based on correct predictions, but it not necessarily affects the safety of the vehicle's driving behavior. As a consequence, an isolated evaluation of the prediction accuracy is not suited to finally give evidence of the approaches' suitability for automated driving. But nevertheless it can be used to quantify the prediction performance, to compare rivaling approaches, and to facilitate causal research in case of prediction errors.

6.1 Evaluation Setting and Metrics

For the training and evaluation of the proposed prediction approaches all data acquired from the sources described in section 5.1 and listed in Table 5.1 has been splitted into a training set (75 %), an evaluation set (12.5 %, needed for the training of a neural network for comparison, see subsection 6.2.3) and a test set (12.5 %).
As probabilistic predictors tend to underestimate the likelihood of classes that are less represented in a data set (which is then also called to be *skewed*), the training data has been randomly undersampled to an equal amount of data instances for all maneuver classes, which is a common practice (Japkowicz, 2000).
The evaluation of the intention recognition and behavior prediction approaches involves the evaluation of different outcomes and aspects of the models. Not only shall the classification and prediction time of the maneuvers predicted by the intention recognition, maneuver detection and behavior prediction approach be evaluated, but also the nearness of the learned probability distributions to the ground truth and the accuracy of the predicted positions in the generated trajectory hypotheses is of interest. The metrics to evaluate these aspects are described in the following sections individually.

6.1.1 Classification Performance Metrics

Metrics for the evaluation of classification models are in general statistics about the amount of correct and false classifications of the model compared to the ground truth. Thereby the ground truth is either the known true condition of the domain, the prediction of a reference classifier or any other reasonable hypothesis defined as labels.
Commonly a *contingency table* (also called *confusion matrix*) is used, which shows the frequency distribution of the classification results. A 2×2 contingency table for the case of a binary classifier together with the *true positive rate* (TPR, also called sensitivity) and *false negative rate* (FNR), as well as *false positive rate* (FPR or fallout) and *true negative rate* (TNR or specificity) that can be computed based on the table is given as follows:

ground truth (g) \ **prediction** (y)	$+$	$-$	
$+$	true positive (TP)	false negative (FN)	$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$, $\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}}$
$-$	false positive (FP)	true negative (TN)	$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$, $\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{FP} + \mathrm{TN}}$

Using the classification rates as performance metric has the advantage of being robust against skewed class distributions, because their computation is solely based on the entity of all positive and all negative ground truth values, respectively.

The *informedness* (also called Youden's index or J statistic) given by

$$J = \mathrm{TPR} + \mathrm{TNR} - 1 = \mathrm{TPR} - \mathrm{FPR} \tag{6.1}$$

summarizes the classification performance in a single metric. It ranges from -1 to 1. Thereby 1 indicates a perfect classification, 0 certifies a random classifier, and -1 reports a classifier with inverted semantics, see (Powers, 2011).

While these statistics at first seem easy to be computed and evaluated, when using a probabilistic classifier the fundamental question is: When is a class outcome actually predicted?

As a probabilistic classifier only outputs a class affiliation $\mathrm{P}(Y = y) \in \mathbb{R}^{[0,1]}$, two standard approaches exist for the evaluation of a binary problem (also called *dichotomous diagnostic task*): Either the outcome with the maximum probability is predicted (i.e., $y^+ = \mathbf{1}\{\mathrm{P}(Y = y) = \mathrm{argmax}(\mathrm{P}(Y))\}$), or a threshold δ is defined that maps the probabilistic outcome to a Boolean value with $y^+ = \mathbf{1}\{\mathrm{P}(Y = y) \geq \delta\}$. (Note that for the dichotomous case predicting the outcome with the maximum probability corresponds to choosing $\delta = 0.5$.)

Using a threshold has the advantage of the human expert being able to adjust the classification behavior. In particular, choosing $\delta < 0.5$ in general means being more conservative about the prediction of a positive class outcome (lower false positive rate at the cost of a lower true positive rate), while choosing $\delta > 0.5$ lets the classifier be more optimistic (higher true positive rate at the cost of a higher false positive rate). A receiver operating characteristic (ROC) curve together with the informedness measure can be used to find an optimal trade-off for the selection of δ. The ROC is a plot of a binary classifier's TPR over its FPR for all possible choices of δ in the interval $[0, 1]$, see (Fawcett, 2005).

However, for the case of a probabilistic n-class problem the evaluation is more cumbersome. One common evaluation method is the so-called *one-vs-all* strategy. The idea is to successively pick one class as positive and evaluate it against all other classes being treated as negative, which transfers the n-class evaluation to a binary class evaluation problem for every choice of the positive class. Unfortunately, in an n-class problem the threshold to decide on the class value can not be chosen individually for all outcomes. For example, for a discrete variable with the three states $\mathrm{LC} = \{\textit{lane change left}, \textit{keep lane}, \textit{lane change right}\} = \{\mathrm{LCL}, \mathrm{KL}, \mathrm{LCR}\}$, which is also called a *trichotomous decision task*, it is not possible to choose $\delta_{\mathrm{LCL}} = 0.7$, $\delta_{\mathrm{KL}} = 0.6$ and $\delta_{\mathrm{LCR}} = 0.8$, because an outcome of e.g. $\mathrm{P}(\mathrm{LC}) = \{0.3, 0.4, 0.3\}$ would lead to no class assignment being predicted at all, which in this case is a semantic error.

On the other hand, predicting the class with the highest probability (which is called a *winner-takes-all* strategy in the case of an n-class problem) is always possible, but it lacks a parameter to trade-off a conservative against an optimistic classification behavior with respect to each class. Indeed, approaches exist to plot ROC curves even for multi-class decision tasks, see for example (Mossman, 1999) for the trichotomous case, but as the ROC space in general has $n(n-1)$ dimensions the evaluation complexity increases with the number of classes and the clarity and interpretability of the evaluation results decreases.

For the case of a lane change prediction therefore a single weight w is introduced to scale the probabilities of the positive class outcomes being defined as $\mathrm{LC}^+ = \{\mathrm{LCL}, \mathrm{LCR}\}$ against the negative outcome with $\mathrm{LC}^- = \mathrm{KL}$. This allows for an evaluation of the classification with a winner-takes-all strategy while maintaining the ability to trade-off the classification behavior with a single parameter. A score s with

$$s(\mathrm{LC}^+) = (1 - w) \cdot \mathrm{P}(\mathrm{LC}^+) \quad \text{and} \quad s(\mathrm{LC}^-) = w \cdot \mathrm{P}(\mathrm{LC}^-) \tag{6.2}$$

is computed for each outcome as a weighted probability, and the predicted class outcome is the one that maximizes the score. The weight w is then varied from 0 to 1 to obtain the ROC curve. (Note that for the case of $w = 0.5$ the prediction evaluation is equal to the classical winner-takes-all strategy.)

6.1.2 Prediction Time

The time a maneuver is predicted in advance to the actual execution is a metric that here is computed based on a positive lane change classification outcome with a true positive prediction in the end; a *lane change left* or *lane change right* that was continuously predicted until the execution of the maneuver.
See for instance the exemplary lane change classification in Figure 6.1, where a lane change takes place at t_0. The course of the probabilistic classification outcome P(LC) together with the prediction threshold δ is shown at the top. For each sampled time step the labeled ground truth and prediction result as well as the evaluation result according to the definitions in subsection 6.1.1 are shown at the bottom. It can be seen that the positive classification of the lane change at t_0 already starts at t_{-6}, although the positive label interval for the lane change only starts at t_{-3}. As a consequence the evaluation of the prediction correctness is false positive from t_{-6} to t_{-4} and true positive only from t_{-3} to t_0. Nevertheless, the time span the maneuver is predicted is $T_{\mathrm{pred}} = |t_{-6} - t_0|$. On the other hand, the positive lane change classification from t_{-11} to t_{-10} is not taken into account, because no lane change is being executed at the end of the positive prediction interval.
The downside of this prediction time computation is that it is only coupled to the maneuver classification outcome and the final maneuver execution, but not to the classification evaluation result. Hence the computation is prone to overestimate the maneuver prediction time. This effect becomes obvious in case of a maneuver classification constantly predicting a lane change. While in general a long prediction time is desirable, in this case the maneuver prediction time would be equal to the entire time span between two consecutive maneuvers and would therefore not represent a reasonable evaluation metric.
However, if the classification performance shows reasonable results in terms of the maximum informedness of the classifier, it can be assumed that also the computation of the prediction time based on the maneuver classification outcome shows reasonable results. Moreover, if the computation of the prediction time would instead be coupled to the classification evaluation (i.e., computing the time span of true positive prediction intervals only) the prediction time would be limited to the duration of the lane change labels. In case of the lane change labels being defined using the assumption of a constant maneuver

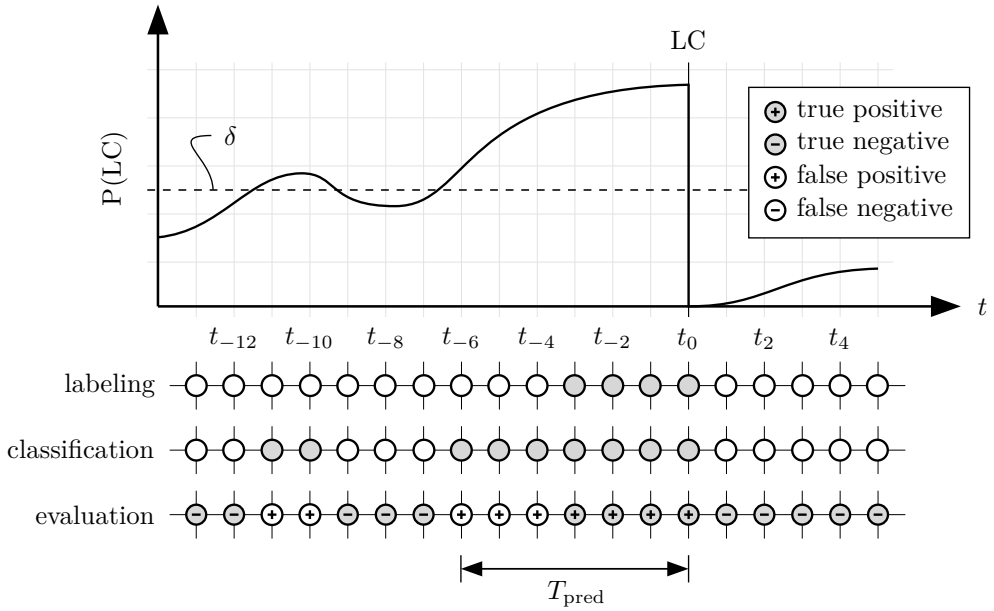


Figure 6.1: Exemplary probabilistic lane change classification outcome together with the definition of the maneuver prediction time T_{pred}. A lane change is predicted as soon as the lane change probability P(LC) exceeds a given threshold δ.

duration the computation would be prone to underestimate the true maneuver prediction time.

6.1.3 Distribution Accuracy Metrics

For the evaluation of a driver's contentedness estimation on a particular lane (including the contentedness factors that the lane contentedness depends on) after the pre-training phase of the approach to a driver's maneuver intention recognition, the Gaussian estimation of the contentedness (denoted by $\widetilde{C} = \mathbb{E}(C)$) can be compared to the scalar ground truth value of the contentedness C, which is acquired from PELOPS.
According to (MURPHY, 2012, p. 98) the distance of a scalar point x to a distribution $Y \propto \mathcal{N}(\mu, \Sigma)$ can be measured with the Mahalanobis distance as

$$D_{\text{M}}(x, Y) = \sqrt{(x-\mu)^{\text{T}}\Sigma^{-1}(x-\mu)}. \tag{6.3}$$

Thereby $D_{\text{M}}(x, Y)$ measures how many standard deviations away x is from μ.

6.1.4 Position Accuracy Metrics

When evaluating the performance of the trajectory hypotheses generation approach as presented in section 4.2 the accuracy is measured as the root mean squared error (RMSE) of the vehicle's position.
With $\hat{d}$ being the lateral predicted position with respect to the original lane, the RMSE is

computed as

$$\mathrm{RMSE}_d = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{d}_n - \tilde{d}_n)^2}. \tag{6.4}$$

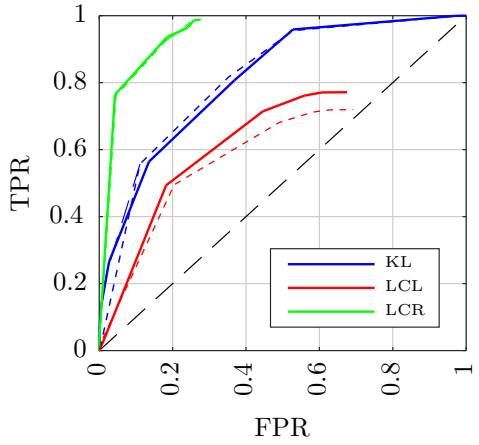
The measured ground truth trajectory given by $\tilde{d}$ is linearly interpolated to match the time of the predicted trajectory at every relative time step $t_k = k \cdot T_s$ with $k = 0, ..., N$ ($T_s = 0.3\,\mathrm{s}$, $N = 20$).

6.2 Intention Recognition Evaluation

In the following the results of training the approach to a driver's maneuver intention recognition as presented in section 3.4 with the particular BN structure depicted in Figure 3.15 are described. Thereby the BN is trained and also evaluated on driving data gathered from varying sources.
To distinguish the different evaluations the notation $\mathrm{BN}\langle\, \mathcal{D}_{\text{test}} \,|\, \mathcal{D}_{\text{training}} \,\rangle$ is used to refer to the evaluation of the BN on the test data source given the parameters of the BN obtained from training on the training data source. For example, $\mathrm{BN}\langle\, \mathcal{D}^{(1.1)}_{\text{veh,ego}} \,|\, \mathcal{D}_{\text{PELOPS}} \,\rangle$ refers to the evaluation of the BN on real world driving data with the ego vehicle as subject, whereby for the training phase data gathered from PELOPS has been used (see Table 5.1 for an overview over the different data sets). Unless stated otherwise, the different data sources for training and testing are always subdivided into a training, an evaluation and a test data set. So for example, when using $\mathcal{D}_{\text{PELOPS}}$ for training, only 75 % of the data is used, while each 12.5 % of the data remains for evaluation and test purposes.

6.2.1 Performance on Simulation Data

For pre-training the BN's parameters approximately 50 h (∼13 000 km) highway driving data have been simulated using PELOPS (see data set $\mathcal{D}_{\text{PELOPS}}$ in Table 5.1). The data has a sample time of $T_s = 0.3\,\mathrm{s}$ and is being undersampled to synthesize equally distributed maneuver classes. Using 75 % of the data for training results in approximately 16 400 samples of simulated driving behavior being available for the parameter optimization.
Labels for samples in the interval of 1.5 s after an executed lane change have been omitted, because in PELOPS a lane change intention is maintained until the vehicle reaches the center of the target lane. Since the employed BN does not take into account any time series information, these labels would lead the learning algorithm to the wrong conclusion of an existing lane change intention when driving on lanes with the highest lane contentedness already.
All variables of the BN (including the lane contentedness variables) are fully observed when using data from PELOPS, so the parameters can be learned using the standard approach described in subsection 5.3.1. However, it is also possible to use the wider applicable EM algorithm as described in subsection 5.3.3, which in case of complete data converges already after the second iteration.



	$D_{\mathrm{M}}(C, \widetilde{C})$	RMSE
C	0.32	0.06
C_{pre}	0.14	0.04
C_{suc}	0.27	0.10
C_{kl}	0.38	0.10
C_{vel}	0.26	0.04

Figure 6.2 & Table 6.1: Left: ROC of lane change intention recognition performance for pre-trained BN based on simulated driving data from PELOPS. Solid lines indicate the test set performance, dashed lines show performance on training set. Right: Mahalanobis distance and root mean squared error of the lane contentedness estimation and the contentedness factors.

For the evaluation of BN$\langle \mathcal{D}_{\mathrm{PELOPS}} \,|\, \mathcal{D}_{\mathrm{PELOPS}} \rangle$ the performance metrics are shown in Figure 6.2 in terms of the ROC curves for the maneuver intention classification and in Table 6.1 in terms of the Mahalanobis distance and root mean squared error of the lane contentedness estimations. It can be seen that the distance of the contentedness estimations to the true contentedness values computed by the driver model in PELOPS is in the low range of 0.14 to 0.38 standard deviations, signifying that the estimation fits the ground truth quite well. On the other hand, for the maneuver intention classification the ROC curves show poor results on the test set. The maximum informedness averaged over all maneuver classes is only at $J_{\max} = 0.49$ ($w = 0.5$).

As explanation for the low classification performance overfitting can be excluded, because the classification performance on the training set does not considerably differ from the performance on the test set (see the dashed lines in Figure 6.2). Moreover, although the particular structure of the BN is only a simplification of the driver model employed in PELOPS the good performance of the lane contentedness estimation does not allow the conclusion of the BN's hypothesis space (in terms of the used features, the network structure and the employed CPD types, see section 3.4) being too restrictive. Instead, the reason for the poor classification performance can be found in the particular parameterization of the lane contentedness computation in PELOPS.

As shown in Figure 6.3, the intention to change the lane is made up from only slight differences in the contentedness values of individual lanes, resulting in wrong classifications even in case of only low deviance of the lane contentedness estimation. However, since the semantic meaning of the BN's lane contentedness nodes is correct after learning the CPD's parameters from simulation data, the parameters can be used as a reasonable initialization when learning the parameters from real world driving data acquired from the test vehicle subsequently.

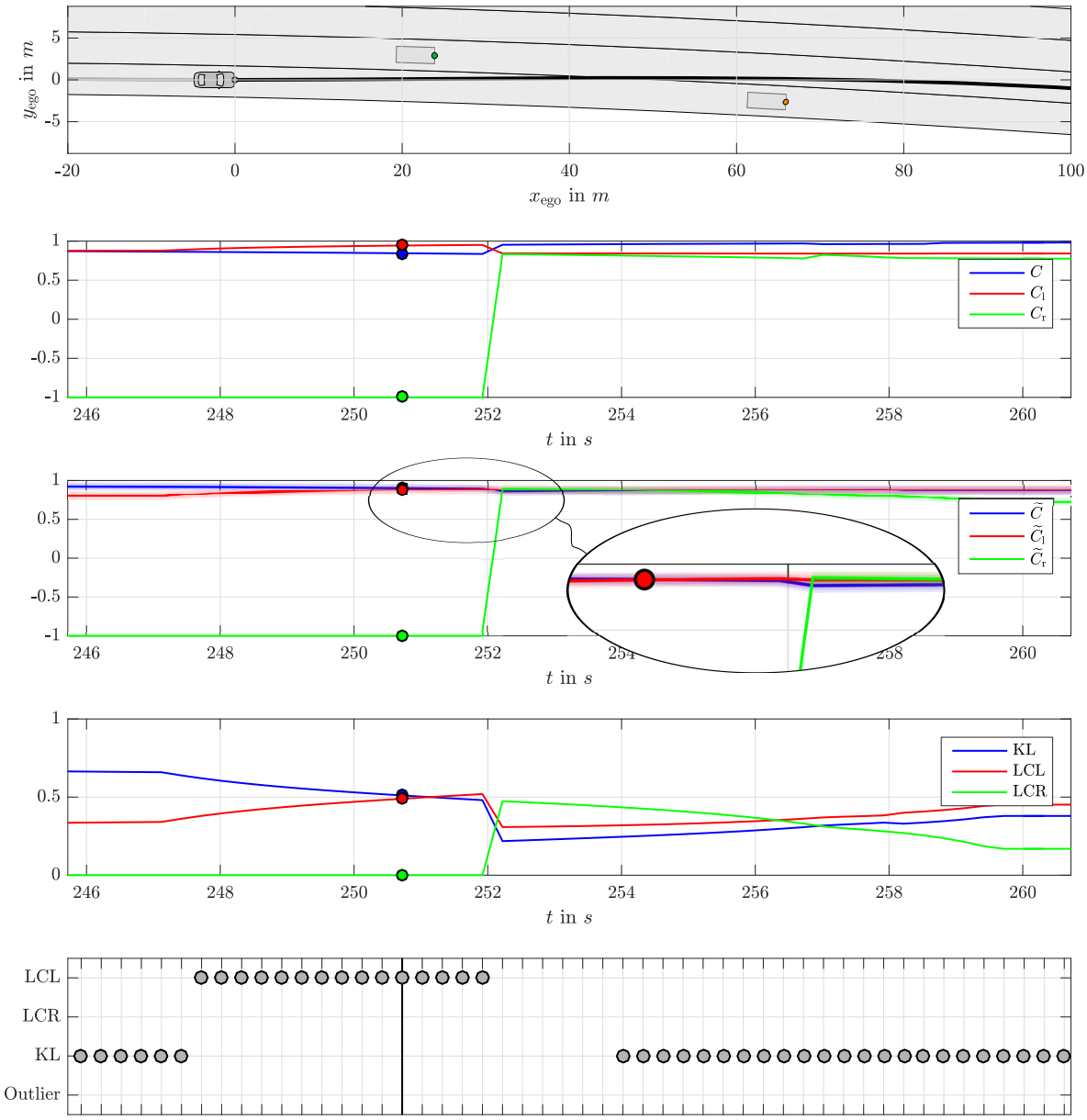


Figure 6.3: Exemplary lane change scenario in PELOPS. From top to bottom: 1: Top view of the traffic situation. 2: Ground truth of lane contentedness values gathered from the driver model in PELOPS. 3: Estimation of lane contentedness from Bayesian network (together with 1σ, 2σ and 3σ standard deviation intervals). 4: Classification of maneuver intention by Bayesian network. 5: Labels used for training and evaluating the maneuver classification.

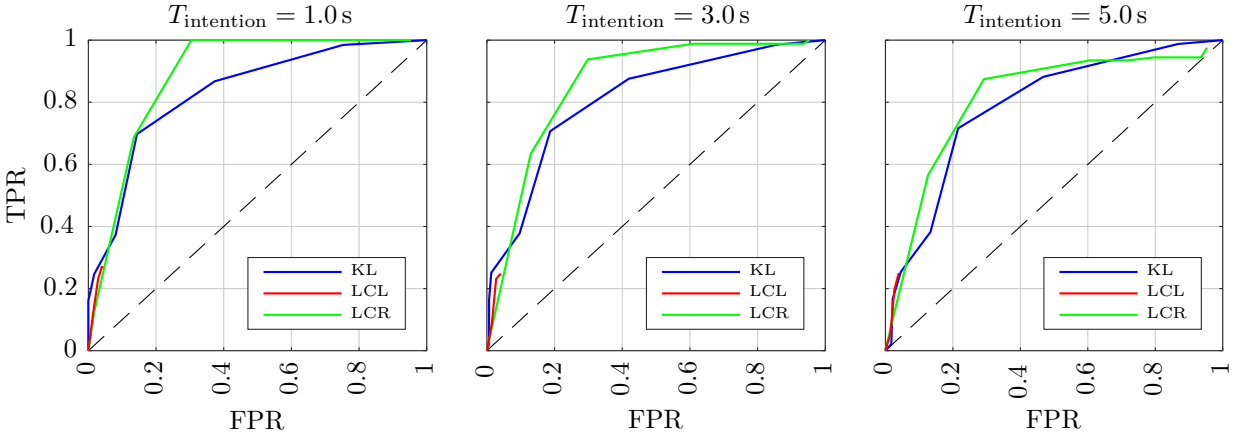


Figure 6.4: ROC of lane change intention recognition performance for pre-trained BN based on test set of vehicle data for varying assumptions of $T_{\text{intention}}$ during evaluation.

6.2.2 Performance on Test Vehicle Data

From the test vehicle approximately 28 h (∼3200 km driven by ego vehicle, see data set $\mathcal{D}_{\text{veh}}^{(1)}$ in Table 5.1) highway driving data have been acquired, resulting in 504 857 data samples for the ego vehicle and 1 316 028 samples for all surrounding traffic vehicles. But unfortunately ∼67 % of the data instances had to be manually rejected because of faulty sensor detections, e.g. due to construction sites, irrecognizable lane markings or heavy rain. Thus, the resulting data set $\mathcal{D}_{\text{veh}}^{(1.1)}$ is reduced to 9 h (165 917 samples for ego vehicle, 379 820 samples for traffic vehicles). Some examples of the data defects being sorted out are shown in Appendix D.
As a result of the discussion on different labeling strategies in subsection 5.2.5 the driver's maneuver intentions have been labeled automatically based on detected lane change executions. Thereby different assumptions of the time a maneuver intention arises ahead of a maneuver execution have been investigated, in particular $T_{\text{intention}} = 1.0\,\text{s}$, 3.0 s and 5.0 s.

Ego Vehicle as Subject

First, the BN trained on simulation data is evaluated on real world driving data without any parameter modifications. With the ego vehicle as subject the evaluation of $\text{BN}\langle\, \mathcal{D}_{\text{veh,ego}}^{(1.1)} \,|\, \mathcal{D}_{\text{PELOPS}} \,\rangle$ is shown in Figure 6.4 in terms of the ROC curves for varying assumptions of the maneuver intention duration $T_{\text{intention}}$ used for labeling.
It can be seen that indeed a Bayesian network as function approximator for the driver model of the traffic simulation PELOPS is able to predict the lane changing behavior of the ego vehicle's driver up to a certain extend. Clearly, with a maximum informedness of $J_{\max} = 0.42$ ($w = 0.6$), 0.39 ($w = 0.6$) and 0.36 ($w = 0.6$) averaged over all maneuver classes (when assuming $T_{\text{intention}} = 1.0\,\text{s}$, 3.0 s and 5.0 s, respectively) the model's practicability is very limited, but this performance proves that it is reasonable to use the simulation as a pre-training step for learning the model's parameters from real-world driving data.
When learning the BN's parameters using data from the test vehicle the EM algorithm is employed, because all contentedness nodes of the BN are now unobserved (see subsec-

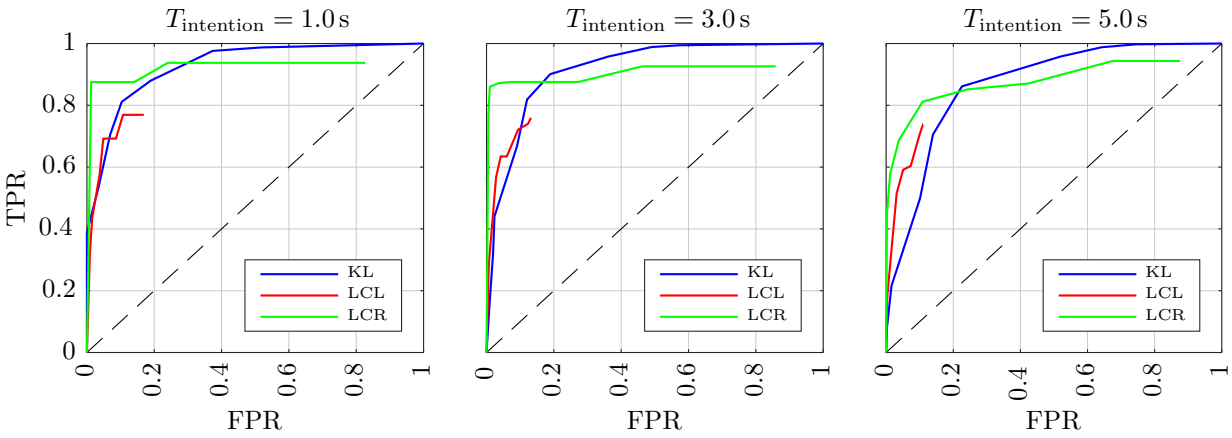


Figure 6.5: ROC of BN's lane change intention recognition performance on test set of vehicle data with ego vehicle as subject for varying assumptions of $T_{\text{intention}}$ during training and evaluation.

tion 5.3.4). To bear up the semantic meaning of the hidden nodes, the pre-trained BN's parameters serve as reasonable initialization.

The different time intervals for the automatic labeling process affect the amount of training samples being available after undersampling the data to synthesize equally distributed maneuver classes. While with the sample time of $T_s = 0.2\,\text{s}$ the assumption of $T_{\text{intention}} = 5.0\,\text{s}$ results in a total of ∼8400 samples available for training, choosing $T_{\text{intention}} = 3.0\,\text{s}$ reduces the samples to roughly 5100. Choosing $T_{\text{intention}} = 1.0\,\text{s}$ leads to about 1900 samples.

The results of the prediction performance for BN$\langle\, \mathcal{D}^{(1.1)}_{\text{veh,ego}} \,|\, \mathcal{D}^{(1.1)}_{\text{veh,ego}} \,\rangle$ are shown in Figure 6.5 in terms of the ROC for each assumption of $T_{\text{intention}}$ individually. In summary, the maximum informedness of the lane change intention recognition is $J_{\max} = 0.70$ ($w = 0.5$) for $T_{\text{intention}} = 1.0\,\text{s}$, $J_{\max} = 0.68$ ($w = 0.6$) for $T_{\text{intention}} = 3.0\,\text{s}$ and $J_{\max} = 0.61$ ($w = 0.6$) for $T_{\text{intention}} = 5.0\,\text{s}$ - a significant improvement of at average 71 % compared to using the pre-trained BN directly.

In order to show the influence of failures in the environment perception the performance of the BN has also been evaluated on the same data set while omitting the labels for measurement rejections (i.e., including data samples with faulty sensor detections, see Appendix D). With a maximum informedness of the lane change intention recognition of $J_{\max} = 0.65$ ($w = 0.6$), $J_{\max} = 0.65$ ($w = 0.6$) and $J_{\max} = 0.52$ ($w = 0.5$) for the same labeling intervals of $T_{\text{intention}} = 1.0\,\text{s}$, $3.0\,\text{s}$ and $5.0\,\text{s}$ the performance decreases at average by 9 %.

An exemplary traffic scenario together with the BN's predictions and the ground truth labels is depicted in Figure 6.6. The camera images and the top view show that the ego vehicle is initially driving on the center lane of three available lanes, changing the lane to the left in order to overtake a slower vehicle in front and changing back to the center lane after the slower vehicle has been passed.

The lane contentedness estimations show that at the beginning the contentedness on the starting lane is the highest compared to the neighboring lanes on the left and on the right,

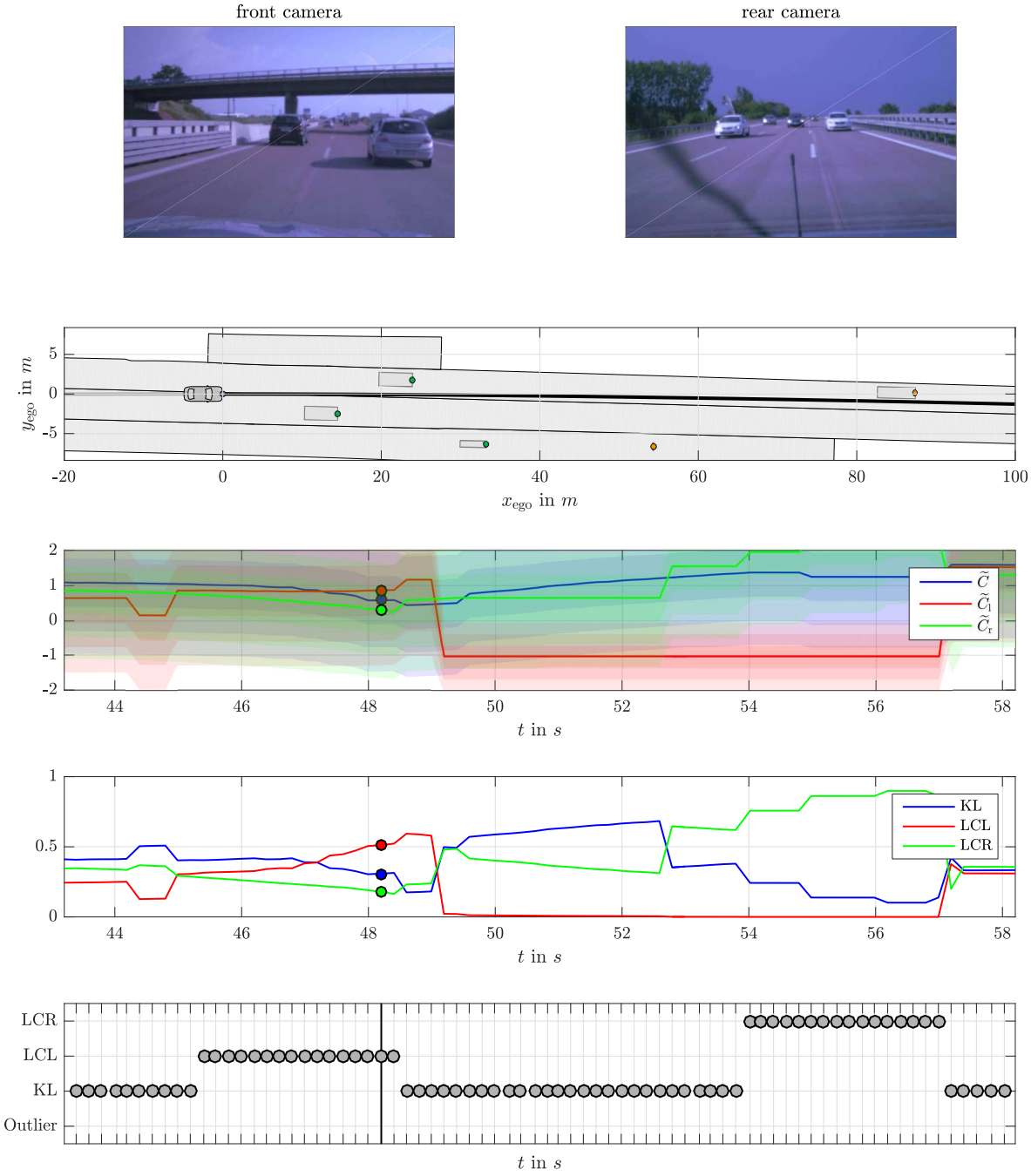


Figure 6.6: Exemplary lane change scenario of ego vehicle from test vehicle data. From top to bottom: 1: Front and rear camera image of the traffic situation at marked time in plots below. 2: Top view of the traffic situation. 3: Estimation of lane contentedness from Bayesian network (together with 1σ, 2σ and 3σ standard deviation intervals). 4: Classification of maneuver intention by Bayesian network. 5: Labels used for training and evaluating the maneuver classification.

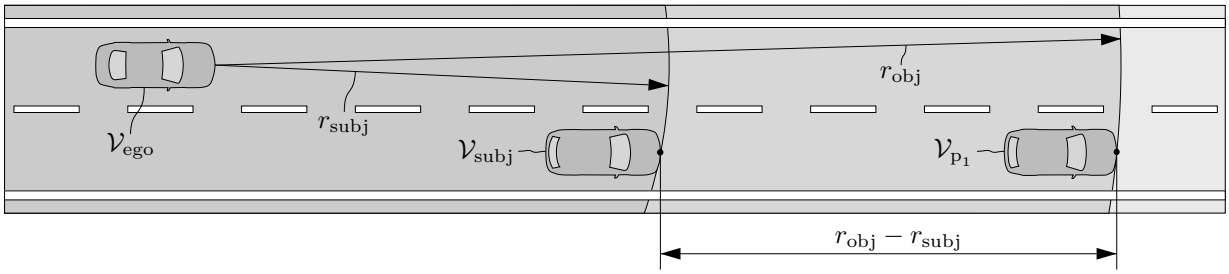


Figure 6.7: Illustration of observation ranges for potential subject and object vehicles.

thus an intention to keep the lane is predicted. The closer the ego vehicle comes to the preceding vehicle, the lower the contentedness on the own lane is estimated. Since the relative dynamics to the vehicle in front on the left lane are at some point advantageous compared to the own lane, the overall lane contentedness $\widetilde{C}_l$ exceeds $\widetilde{C}_r$ at $t = 47.2\,\text{s}$, therefore leading to a predicted intention to change towards the left lane.
It can be seen that in this case the lane change left maneuver is labeled with $T_{\text{intention}} = 3.0\,\text{s}$ prior to the lane change at $t_{\text{change}} = 48.4\,\text{s}$, leading to the interval of 45.4 s to 48.4 s being labeled as LCL while the lane change is predicted in the interval of 47.2 s to 49.0 s. This implicates false negative predictions of the LCL maneuver in the beginning (false positive predictions of KL, respectively) and false positive predictions of LCL at the end.
The false predictions at the beginning are owed to the fact that the true start of the driver's lane change intention is latent and the automatic labeling process defines the start of the intention based on the simple assumption of a constant duration of the intention ahead of each lane change. On the other hand, the spurious lane change left intention right after the executed lane change at $t_{\text{change}} = 48.4\,\text{s}$ is established based on an erroneous road model. Here, the camera wrongly considered the guard railing to the left of the leftmost lane as a lane marking, leading to the belief of an additional lane. Since no vehicles are found to be driving on that lane, the BN predicted a high contentedness when driving on that lane.
One important observation is that while in the driver model of Pelops the lane contentedness values (C, C_l and C_r) have been defined to lie in the closed interval of $[-1, 1]$, after training the BN's parameters based on test vehicle data the estimated lane contentedness values (denoted by $\widetilde{C}$, $\widetilde{C}_l$ and $\widetilde{C}_r$) range from -1.96 to 3.98. This suggests that in the test vehicle driving situations have been encountered that have never been simulated using Pelops.

Traffic Vehicles as Subjects

For the prediction of the traffic vehicles' maneuver intentions all vehicles surrounding the ego vehicle in the range of $r_{\text{subj}} = 50\,\text{m}$ are taken into account as potential subject vehicles. This limit is chosen based on the maximum range of $r_{\text{obj}} = 100\,\text{m}$ in which object detections are generally treated as trustworthy (see subsection 5.1.3). The range r_{subj} should be lower than r_{obj} to be able to observe potential preceding (or succeeding) object vehicles of the subject vehicle, see the illustration in Figure 6.7. In this case the minimum distance at which preceding (or succeeding) object vehicles are being detected ahead (or behind) of

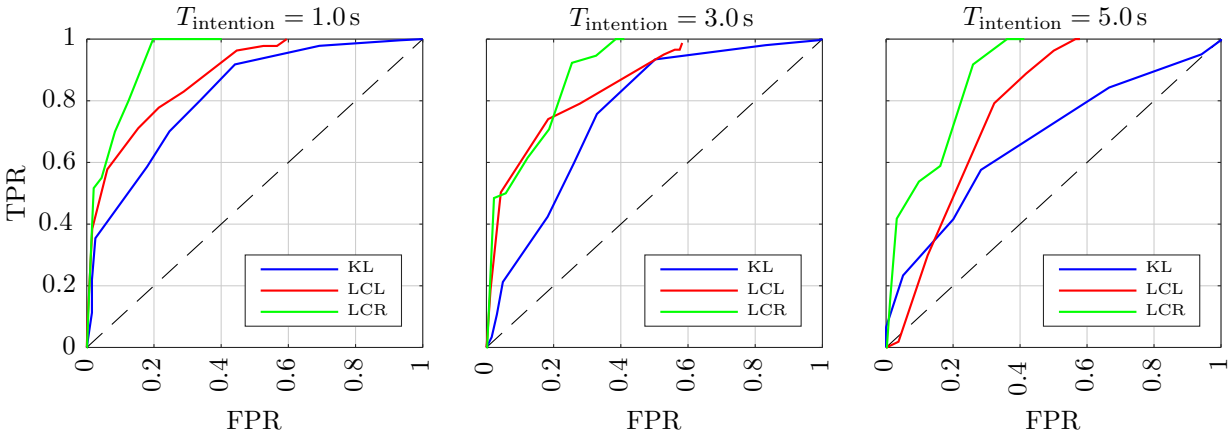


Figure 6.8: ROC of lane change intention recognition performance for BN trained on vehicle data with ego and evaluated with traffic vehicles as subjects for varying assumptions of $T_{\text{intention}}$ during training and evaluation.

the subject vehicle is $r_{\text{obj}} - r_{\text{subj}} = 50\,\text{m}$.
The resulting amount of surrounding vehicle's driving data collected in the prototype test vehicle is 2000 km (respectively 21 h, see $\mathcal{D}_{\text{veh}}^{(1.1)}$ in Table 5.1).
At first the performance when using the BN trained on ego vehicle data is evaluated on traffic vehicle data to find out how well the model generalizes from predicting the ego vehicle's lane change intentions to predicting the surrounding traffic vehicles' maneuver intentions. The ROC curves when evaluating $\text{BN}\langle\, \mathcal{D}_{\text{veh,traffic}}^{(1.1)} \,|\, \mathcal{D}_{\text{veh,ego}}^{(1.1)} \,\rangle$ are shown in Figure 6.8 for the varying assumptions of $T_{\text{intention}}$. In terms of the maximum informedness the BN predicts lane change intentions of surrounding vehicles with $J_{\max} = 0.55$ ($w = 0.3$) when labeling lane change intentions 1.0 s in advance to the actual lane change. Using $T_{\text{intention}} = 3.0\,\text{s}$ leads to $J_{\max} = 0.48$ ($w = 0.6$), and $T_{\text{intention}} = 5.0\,\text{s}$ results in at least $J_{\max} = 0.43$ ($w = 0.4$). This means that compared to predicting the ego vehicle's lane change intention, using the same BN parameterization traffic vehicles are being predicted with a performance decrease of 36 % on average.
However, when re-training the BN based on data with traffic vehicles as subjects (i.e., evaluating $\text{BN}\langle\, \mathcal{D}_{\text{veh,traffic}}^{(1.1)} \,|\, \mathcal{D}_{\text{veh,traffic}}^{(1.1)} \,\rangle$) the prediction performance improves. With $J_{\max} = 0.64$ ($w = 0.4$), 0.55 ($w = 0.5$) and 0.53 ($w = 0.5$) for the same varying assumptions of $T_{\text{intention}} = 1.0\,\text{s}$, 3.0 s and 5.0 s respectively, the performance of predicting the surrounding vehicles' intentions increases on average by 15 %. Compared to the performance of predicting the ego vehicle's lane change intentions this is a decrease of only 16 %, which is 20 percentage points better than before re-training the BN's parameters. The corresponding ROC curves are shown in Figure 6.9.
Again the performance of predicting the traffic vehicles' lane change intention is also evaluated when omitting the labels for measurement rejections. Just like when inferring the lane change intentions for the ego vehicle the prediction performance decreases by 9 % at average when faulty sensor detections are not filtered out.
In Figure 6.10 a lane change scenario of a traffic vehicle is shown for which the lane change intention was correctly classified - albeit with the predicted intention starting even 2.2 s

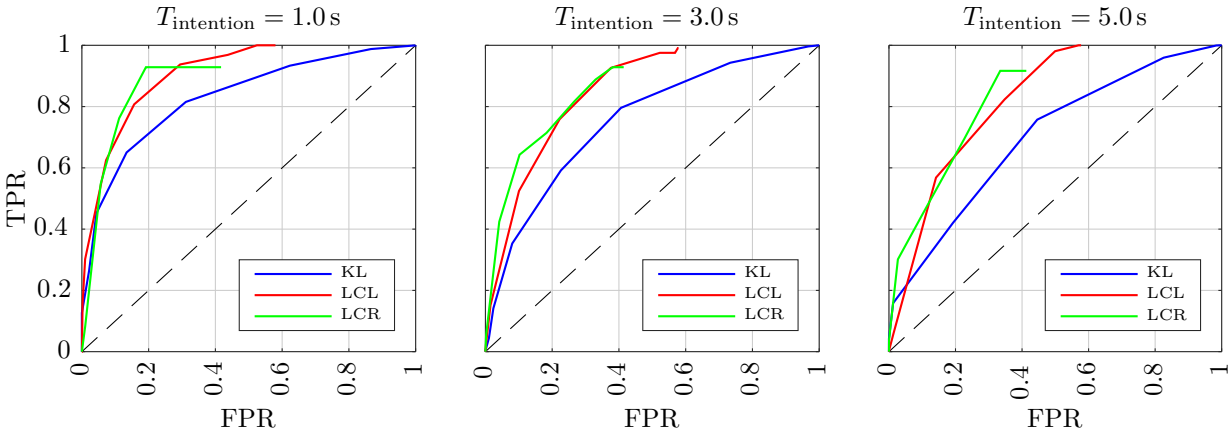


Figure 6.9: ROC of lane change intention recognition performance for BN trained and evaluated on vehicle data with traffic vehicles as subjects for varying assumptions of $T_{\text{intention}}$ during training and evaluation.

ahead of the ground truth labels, which is based on the assumption of a constant intention duration of $T_{\text{intention}} = 3.0\,\text{s}$ ahead of the lane change, thus causing false positive evaluations in the beginning. Note that this time the estimated lane contentedness factors $\widetilde{C}_{\text{pre}}$ are plotted (instead of the overall estimated lane contentedness values $\widetilde{C}$) to show that the vehicle's lane change is motivated from a higher contentedness regarding the preceding traffic as soon as the ego vehicle passed the traffic vehicle on the left lane.
A different lane change scenario with a traffic vehicle cutting in at about 35 m in front of the ego vehicle is shown in Figure 6.11. In the top view of the traffic situation the subject vehicle and its preceding object vehicle (at $x_{\text{subj}} = 64\,\text{m}$) are only visualized as boxes with a minimal assumed length of 0.1 m, because the LIDAR sensors have only seen the vehicles from behind so far and therefore cannot estimate the objects' true lengths.
Moreover, the cut-in maneuver of the subject vehicle is only predicted at the single point in time that is marked in the graph, because the subject's preceding object vehicle (which motivated the lane change) has not been detected by the sensors before. In other words, the most important aspect of the traffic situation, namely the decreasing distance between the subject vehicle and its preceding object vehicle, were unknown to the prediction model, thus resulting in many false negative prediction instances.

6.2.3 Comparison to Neural Networks

Artificial neural networks (ANN, or simply neural networks, NN) are in general computing systems vaguely inspired by the physiology of the human brain (McCulloch, Pitts, 1943). They are based on directed connections between artificial neurons. Each neuron (also called unit) sends signals to all neurons it is connected to by weighting the incoming signals and sending an output based on a specific (possibly non-linear) activation or transfer function. For a detailed introduction to NN see for example (Bishop, 2006, pp. 225 sqq.). According to (Hornik et al., 1989) neural networks are universal approximators and even a network with a single hidden layer is able to approximate any continuous mapping from

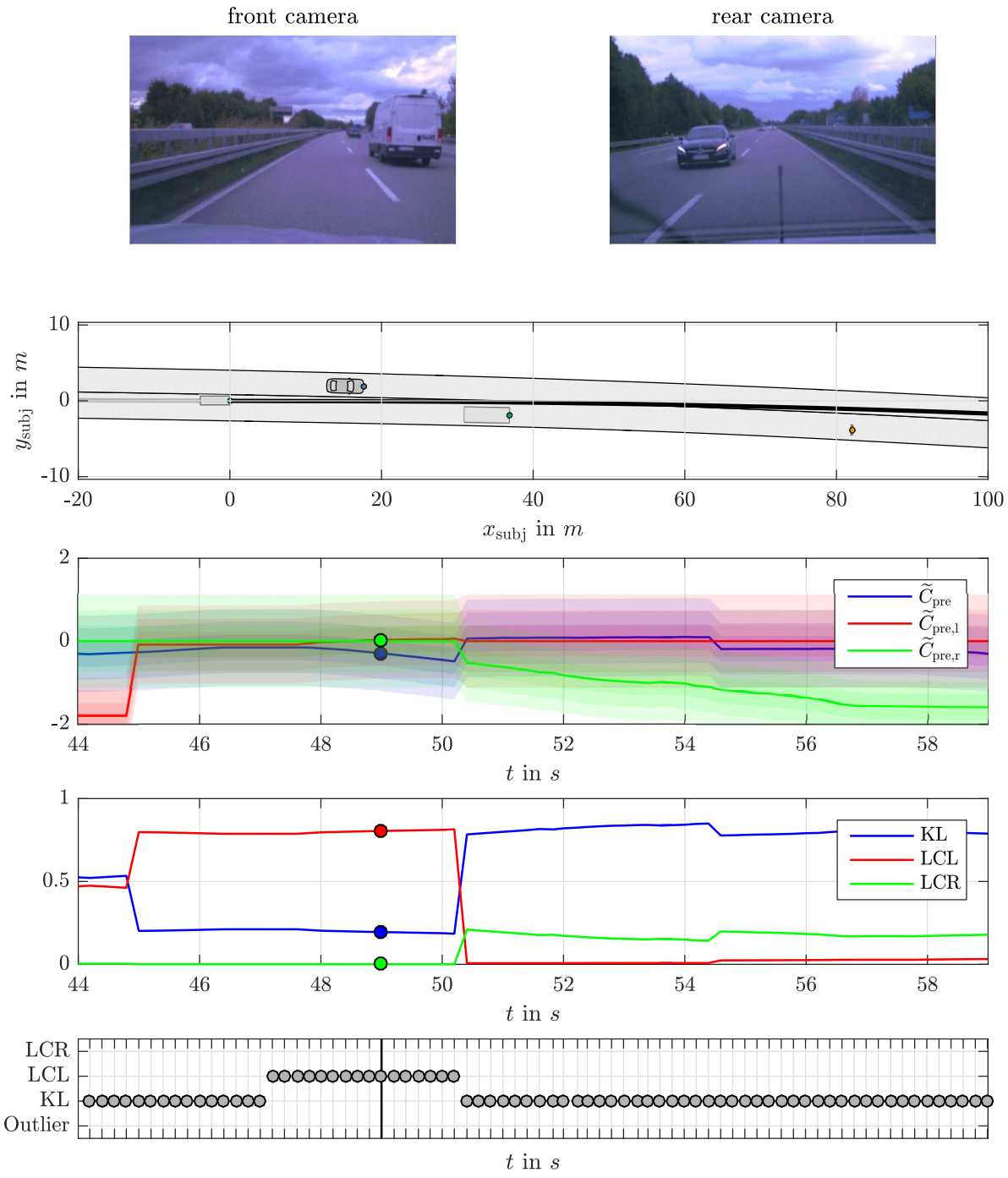


Figure 6.10: Exemplary lane change scenario of traffic vehicle from test vehicle data. From top to bottom: 1: Front and rear camera image of the traffic situation at marked time in plots below (ego vehicle perspective, subject vehicle visible in rear camera image). 2: Top view of the traffic situation relative to subject vehicle. 3: Estimation of lane contentedness factor C_{pre} from Bayesian network (together with 1σ, 2σ and 3σ standard deviation intervals). 4: Classification of maneuver intention by Bayesian network. 5: Labels used for evaluating the maneuver classification.

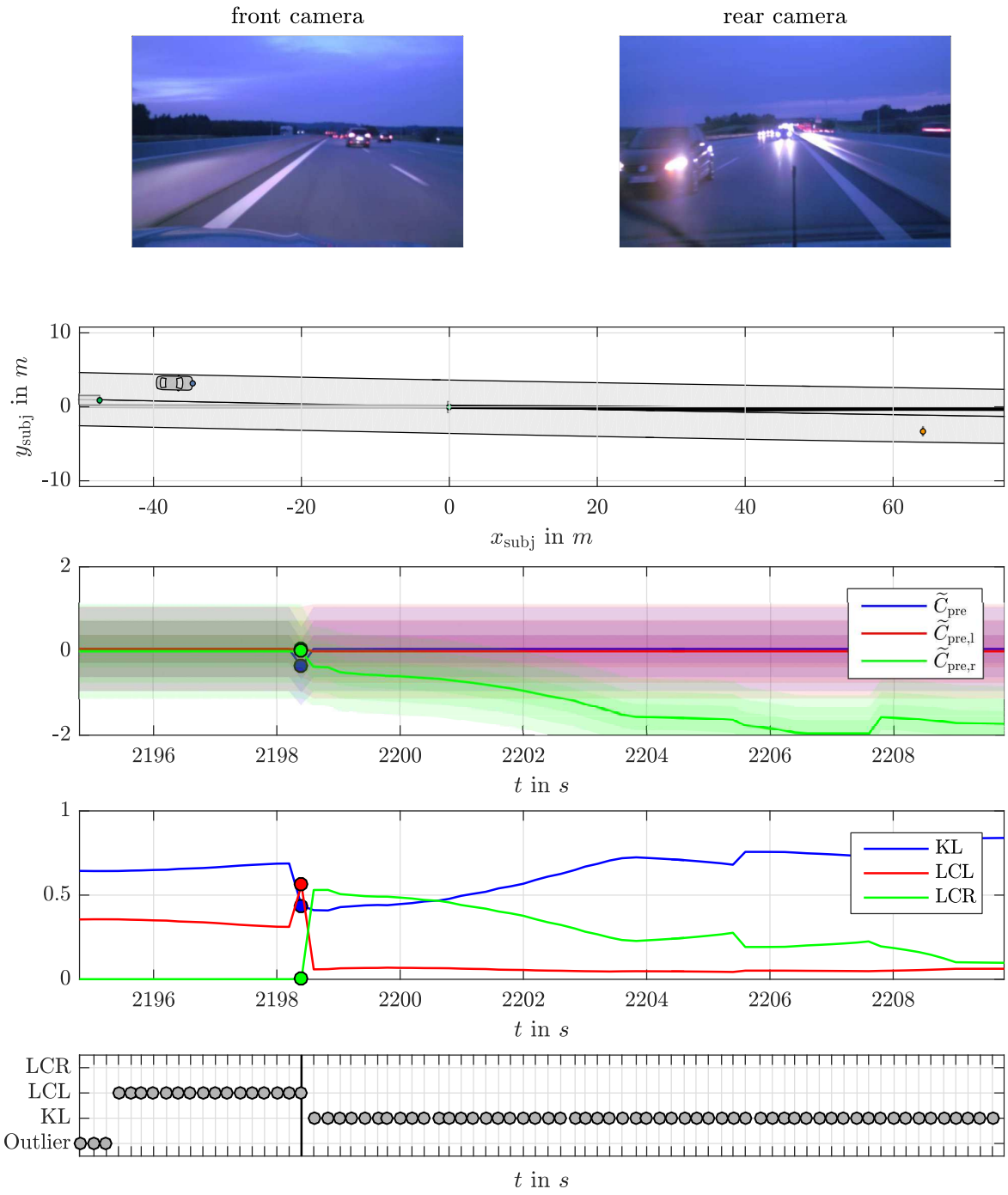


Figure 6.11: Exemplary cut-in scenario of traffic vehicle from test vehicle data. From top to bottom: 1: Front and rear camera image of the traffic situation at marked time in plots below (ego vehicle perspective, subject vehicle visible in front camera image). 2: Top view of the traffic situation relative to subject vehicle. 3: Estimation of lane contentedness factor C_{pre} from Bayesian network (together with 1σ, 2σ and 3σ standard deviation intervals). 4: Classification of maneuver intention by Bayesian network. 5: Labels used for evaluating the maneuver classification.

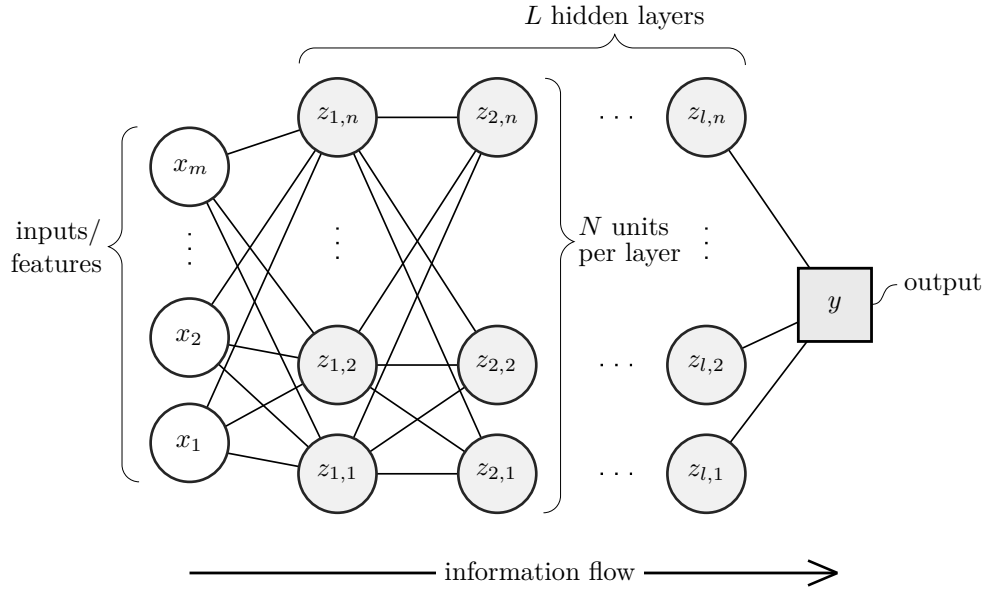


Figure 6.12: General structure of a feed-forward neural network for classification with $M = |\boldsymbol{x}|$ inputs, L hidden layers of N neurons each, and a discrete output variable y. The direction of information flow is from left to right (arcs of connections are omitted for clarity).

finite inputs to outputs (provided a sufficiently large amount of hidden units and a non-polynomial activation function is used). The backpropagation algorithm (together with increasing computational power becoming accessible) has leveraged the use of NN for many complex tasks in the field of artificial intelligence, because it depicts a method to train the parameters even of networks with many hidden layers (also called deep neural networks, DNN) based on a large amount of data (Werbos, 1974). DNN have shown outstanding results in many domains, for example in image classification (Krizhevsky et al., 2012) and speech recognition (Hinton et al., 2012). (An overview over the field of learning deep architectures for artificial intelligence is given in (Bengio, 2009).)
On the other hand, according to (Ghahramani, 2016) deep neural networks are in general

- very data hungry,
- very compute-intensive to train and deploy,
- poor at representing uncertainty,
- finicky to optimize, because the parameter cost functions are non-convex, the choice of the network architecture is not fixed and the parameter learning as well as the initialization procedure has great influence (which requires expert knowledge and experimentation), and they are
- uninterpretable black-boxes (lacking in transparency, difficult to trust).

A vast amount of different kinds of NN exists by now. In the following the most commonly used type of NN, namely a standard fully connected feed-forward neural network with sigmoid activation functions in the hidden layers and a softmax activation function in the output layer is used to compete the Bayesian network for a maneuver intention

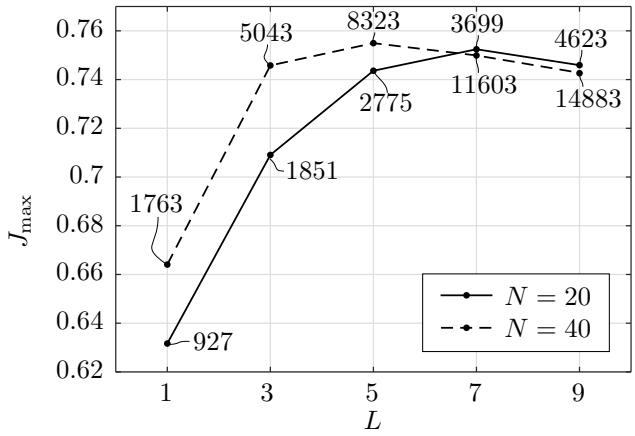


Figure 6.13: Maximum informedness $J_{\max}$ of neural network on simulated test data set depending on the number of layers L and the number of neurons per layer N. Additionally the resulting number of overall network parameters is shown for every evaluated network configuration.

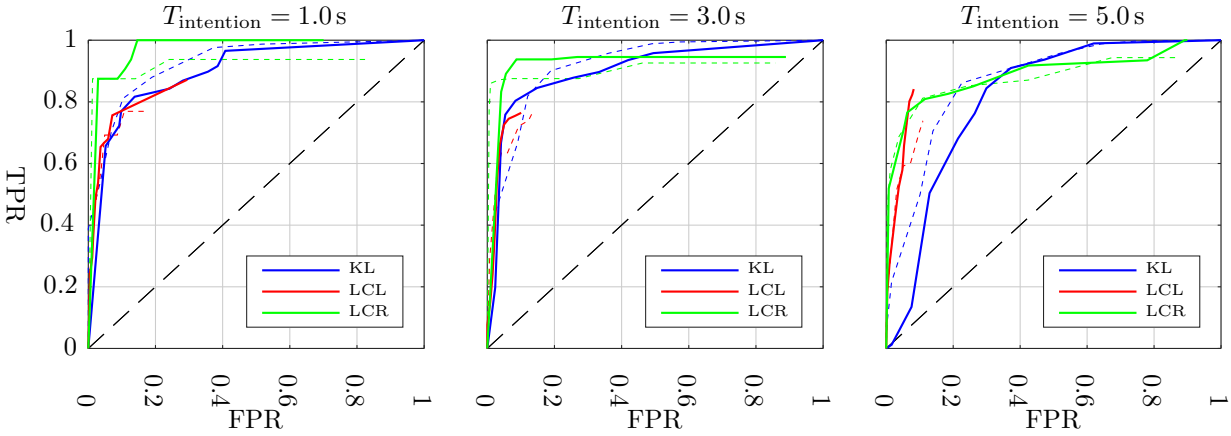


Figure 6.14: ROC of NN's lane change intention recognition performance on test set of vehicle data for varying assumptions of $T_{\text{intention}}$ during training and evaluation. The BN's ROC (originally shown in Figure 6.5) are added as dashed lines for comparison.

recognition. While the general structure of a fully connected feed-forward NN is illustrated in Figure 6.12, the specific design of the network in terms of the number of hidden layers and units per layer has to be chosen such that neither overfitting (too many units) nor underfitting (too few units) occurs and thus falls into the category of hyperparameter optimization.

For the particular comparison here a network structure with a feature vector $\boldsymbol{x}$ identical to all input features of the respective BN (see section 3.4) is chosen, which consists of a total of 40 features. The output is the discrete LCI node representing the lane change intention via the three states LCL, LCR and KL.

For the hidden layers each 20 neurons in 7 layers are used. This choice is based on a comparison of different network architectures shown in Figure 6.13. In particular, two variants of units per layer have been investigated: $N = 40$ based on the number of input

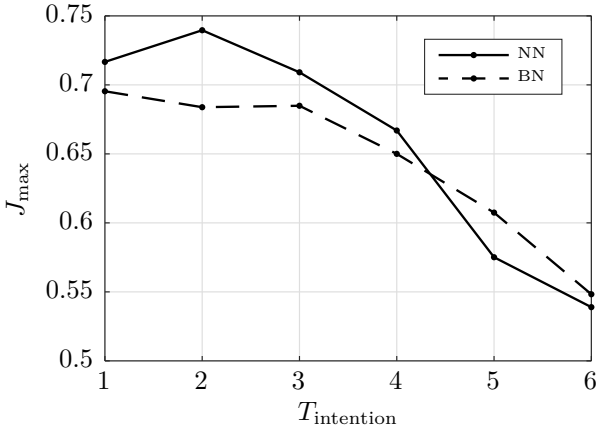


Figure 6.15: Informedness of NN and BN for lane change intention recognition with varying assumptions of $T_{\text{intention}}$ for training and evaluation.

features and $N = 20$, which is the rounded down mean between the number of inputs and outputs. Moreover, the number of hidden layers has been varied from 1 to 9. In favor of a preferably low amount of overall network parameters the configuration of 20 neurons per 7 hidden layers performed best in terms of the maximum informedness $J_{\max}$ evaluated on simulated data from PELOPS ($\mathcal{D}_{\text{PELOPS}}$).

For the parameter optimization the scaled conjugate gradient (SCG) algorithm, introduced in (MOLLER, 1993) and implemented in MATLAB's neural network toolbox, is employed. It uses backpropagation for the computation of the gradient.

Eventually the NN has been trained on test vehicle data with the ego vehicle as subject vehicle ($\mathcal{D}^{(1.1)}_{\text{veh,ego}}$), using automatic labeling of the lane change intention with $T_{\text{intention}} = 1.0\,\text{s}$, $3.0\,\text{s}$ and $5.0\,\text{s}$. The prediction performance is shown in Figure 6.14 in terms of the ROC curves in comparison to the evaluations of the BN in Figure 6.5 (the ROC curves of the BN are copied as dashed lines). Figure 6.15 shows the resulting maximum informedness $J_{\max}$ for the varying assumptions of $T_{\text{intention}}$ during training and evaluation.

It can be seen that the prediction performance of the NN is quite comparable to the BN's performance with the results deviating only 2 % at average (max. 8 %) in terms of the maximum informedness. But especially the uninterpretability of the parameters has motivated to *not* use DNN for the task of a maneuver prediction in this thesis in first place. In (SZEGEDY et al., 2014) it is shown that the uninterpretable solutions a DNN learns can lead to counter-intuitive properties. One such property may be a highly discontinuous mapping from input to output, which causes the network to make prediction errors elicited by even minimal perturbations on the inputs.

In Figure 6.16 an example of such a counter-intuitive characteristic is shown. The neural network predicts an intention to change to the left lane without this lane actually existing. No data instance existed in the training data that demonstrated this behavior, so it remains unclear why this particular behavior is learned. Moreover, as the parameters of the network are uninterpretable, there is no way for a human expert to manually fix this erroneous classification behavior manually.

This characteristic is especially dangerous in safety critical applications like automated

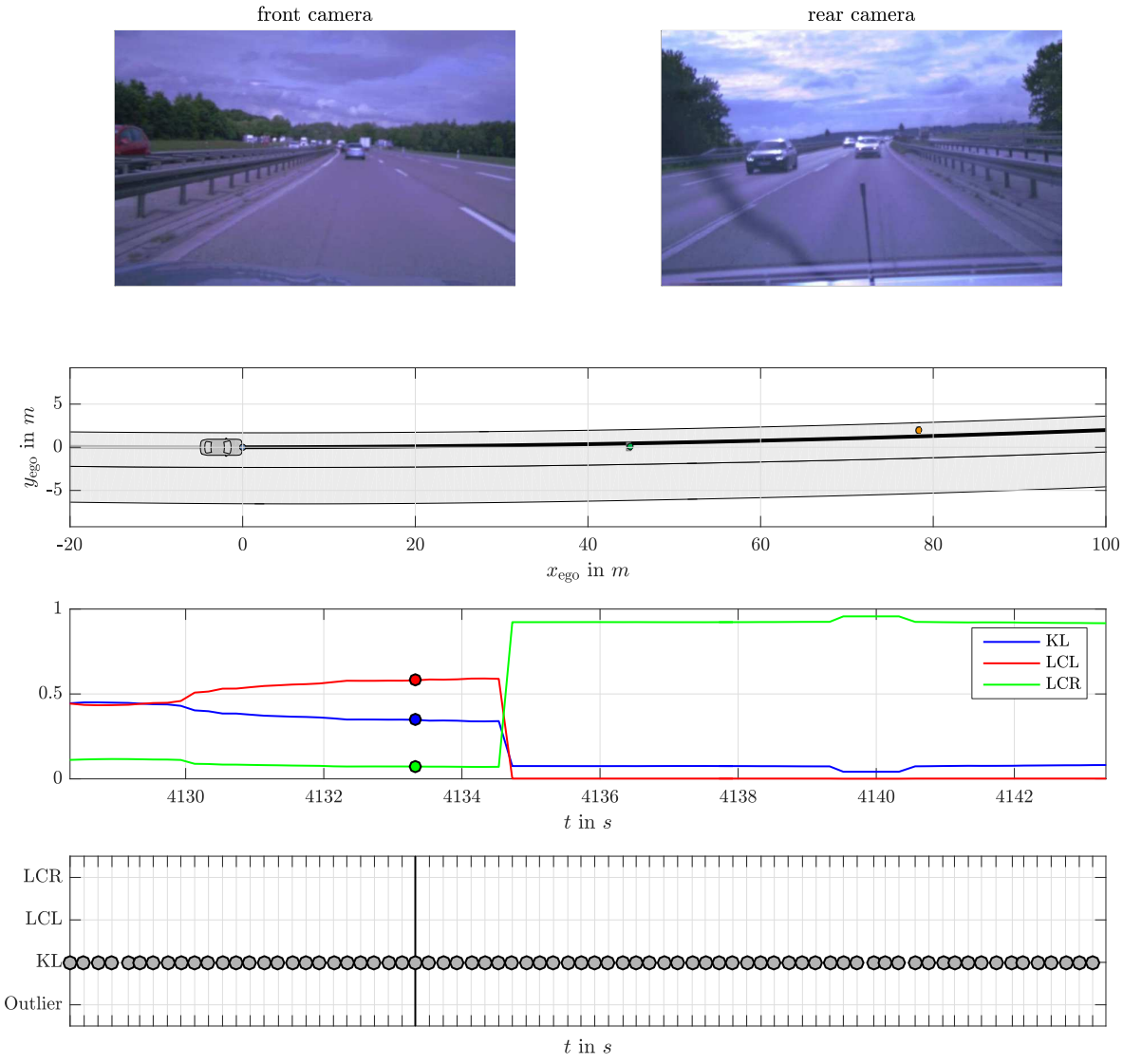


Figure 6.16: Example of counter-intuitive prediction of neural network, predicting a lane change intention without the existence of the corresponding lane. From top to bottom: 1: Front and rear camera image of the traffic situation at marked time in plots below. 2: Top view of the traffic situation. 3: Classification of maneuver intention by neural network. 4: Labels used for training and evaluating the maneuver classification.

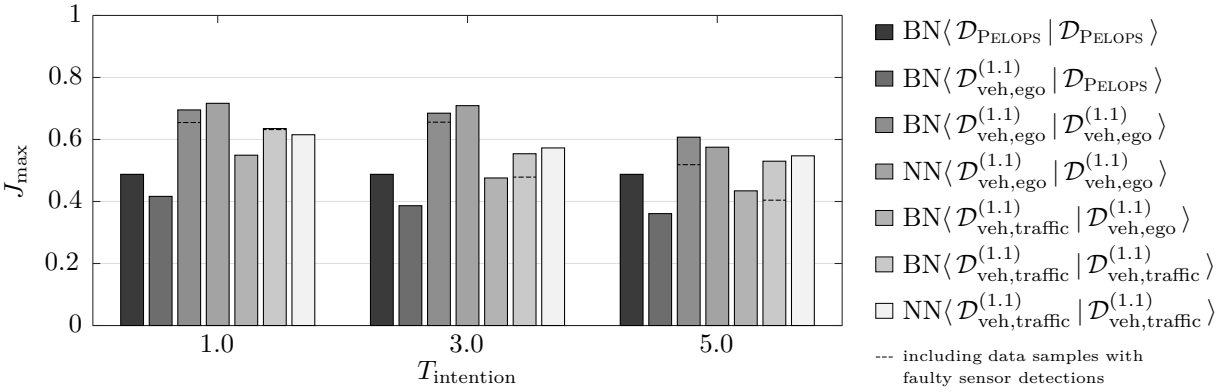


Figure 6.17: Comparison of intention recognition performances in terms of the maximum informedness $J_{\max}$ averaged over all maneuver classes. The dashed lines indicate the performance when including data samples with faulty sensor detections, i.e., when omitting the manually defined measurement rejections.

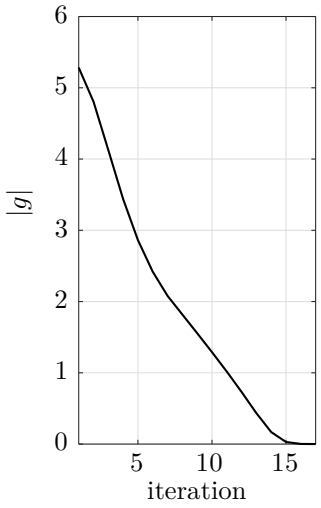
driving, which leads to the conclusion of the particular feed-forward neural network used for comparison here being inferior to the use of a Bayesian network for the task of a lane change intention recognition as proposed in this thesis.

6.3 Trajectory Prediction Accuracy

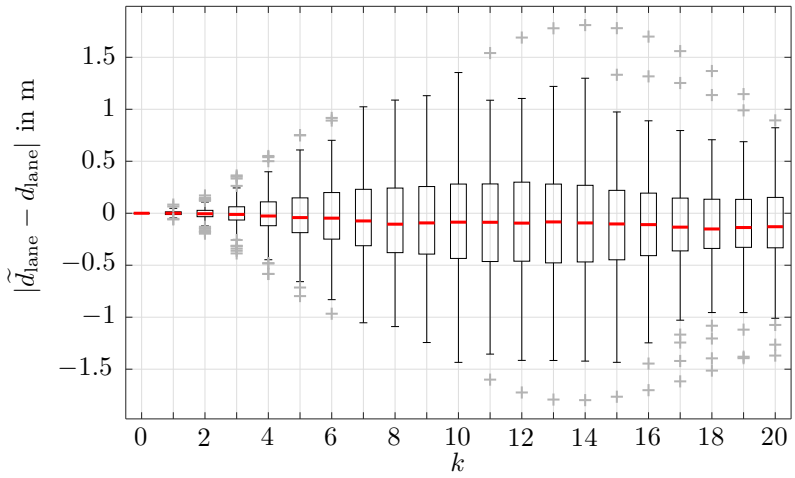
For the task of learning the characteristics of a vehicle's movement during a lane change maneuver with the LQR approach described in section 5.4 it is not sufficient to label the start of the maneuver execution automatically. Instead, the exact start of the vehicle's lateral movement with respect to the lane has to be marked precisely in the data in order to not mix up the characteristics of a vehicle's trajectory for different maneuvers during learning.

Using 103 manually labeled lane change left and 109 lane change right trajectories of the ego vehicle, as well as 122 lane change left and 140 lane change right trajectories of surrounding traffic vehicles (referred to as data set $\mathcal{D}^{(1.1.1)}_{\text{veh}}$, see Table 5.1), the IRL algorithm computed the weight parameters for the LQR that best explained the demonstrated lane changing behavior. In Figure 6.18 the lateral deviations of the trajectory predictions to the corresponding demonstrated trajectories in the training data is shown for every discrete time step in the LQR optimization horizon of $k = 0, ..., N$ with $N = 20$ individually. The prediction error at time step t_0 ($k = 0$) is of course zero, because the first state of each demonstrated trajectory is used as the initial state $\boldsymbol{x}_0$ for solving the optimal control problem given in Equation 4.21e.

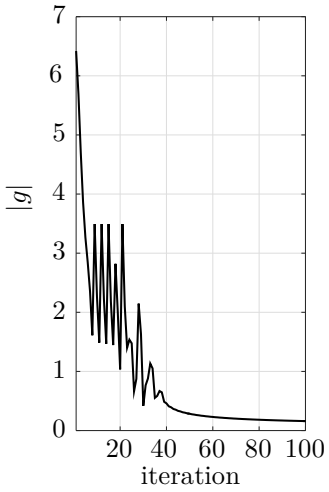
An example of the trajectory predictions for the case of a lane change right maneuver of the ego vehicle on the highway is shown in Figure 6.19. First, Figure 6.19a shows that trajectory hypotheses are generated for every possible maneuver, in this case keeping the lane illustrated with a blue line, and changing to the right lane depicted with a green line. (A lane change to the left is not taken into account, because the vehicle is already driving



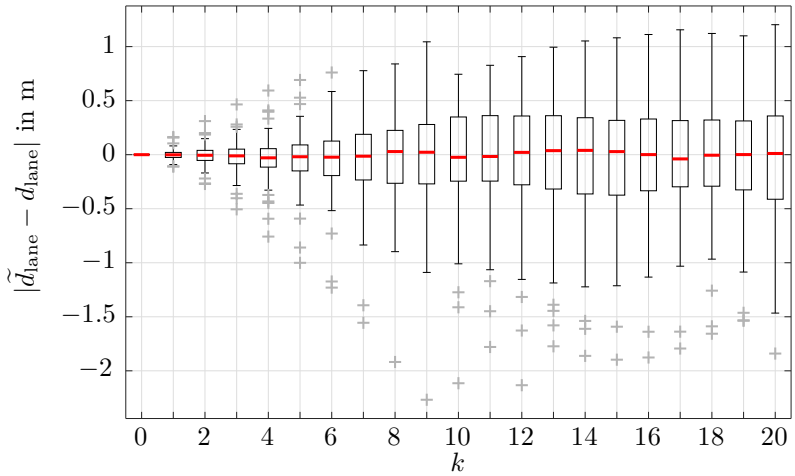
(a) Learning curve ego.



(b) Prediction error ego vehicle.



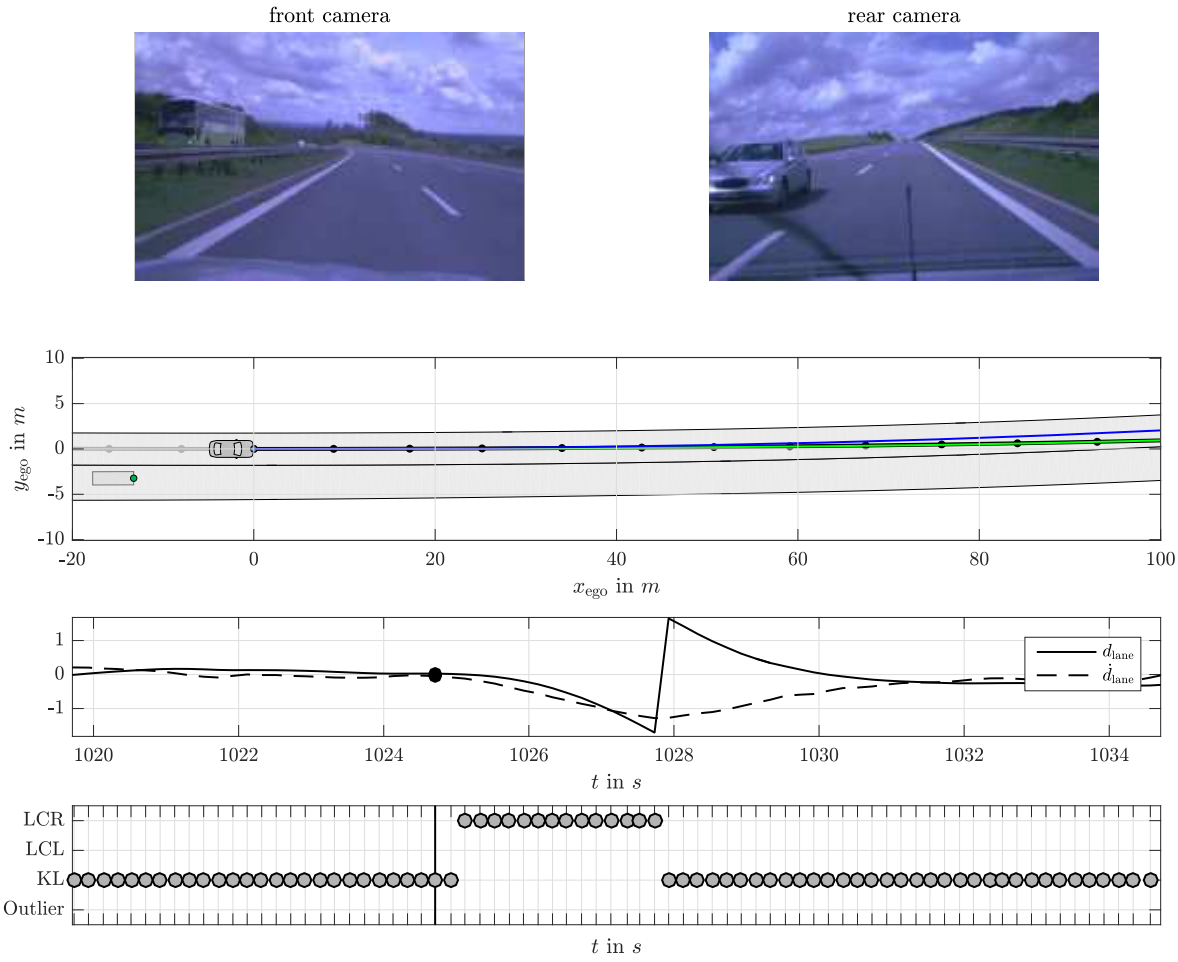
(c) Learning curve traffic.



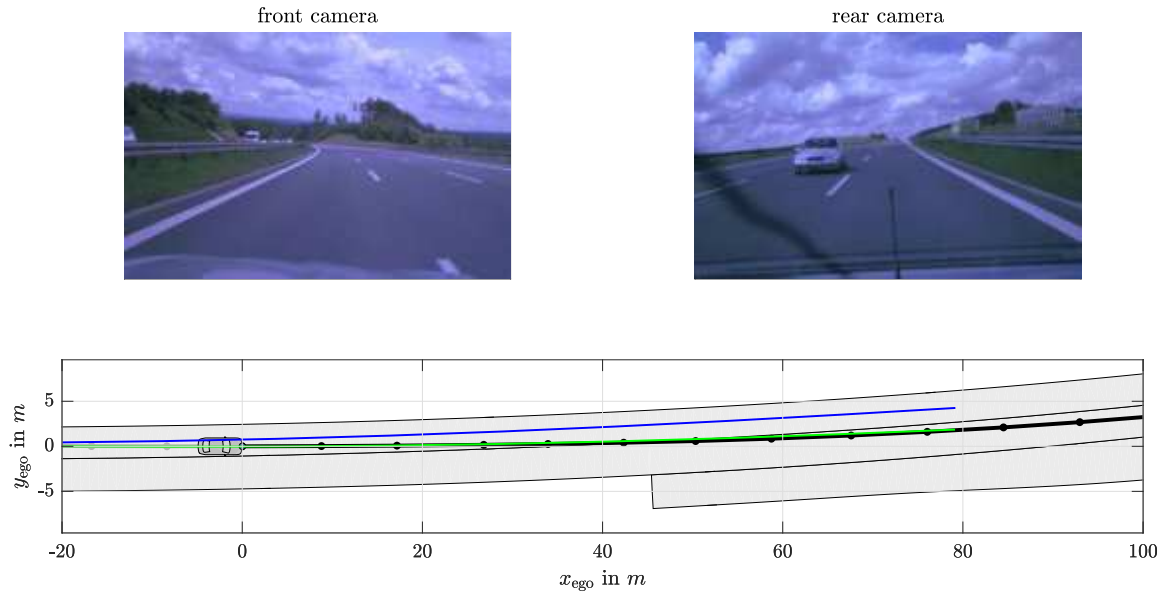
(d) Prediction error traffic vehicles.

Figure 6.18: Learning curve and boxplot* of performance of trajectory prediction during lane changes. While (a) and (c) show the feature difference during the iterations of the IRL algorithm, (b) and (d) visualize the absolute lateral position error of the predicted lane change trajectories over the entire optimization horizon when learning the trajectory characteristics of the ego vehicle and surrounding traffic vehicles, respectively.

*In a boxplot the central mark (red) shows the median, the edges of the box are the 25th and 75th percentiles. The whiskers extend to the most extreme data points not considered outliers, whereby points are drawn as outliers (gray crosses) if they are larger than $q_3 + w(q_3 - q_1)$ or smaller than $q_1 - w(q_3 - q_1)$ with q_1 and q_3 being the 25th and 75th percentiles, respectively.

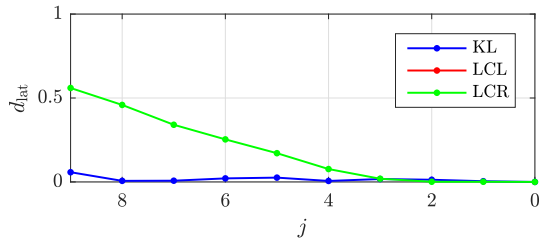


(a) $t = 1024.71\,\text{s}$ (just before start of lane change): Hypotheses for LCL (red) and KL (blue).

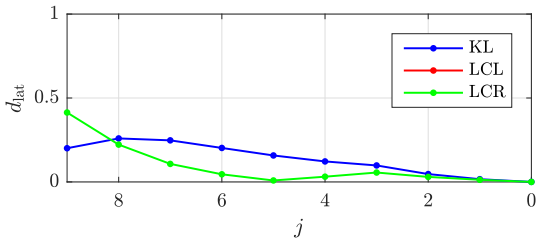


(b) $t = 1026.73\,\text{s}$: Vehicle position compared to former hypotheses (see Figure 6.19a).

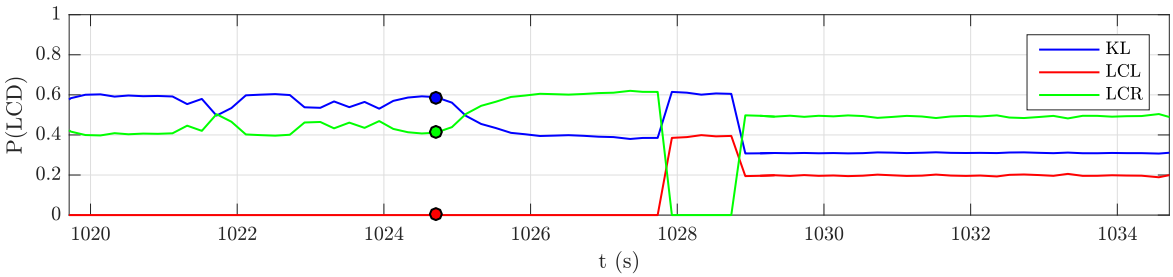
Figure 6.19: Exemplary lane change scenario together with trajectory hypotheses.



(a) Trajectory distances at $t = 1024.71\,\text{s}$.



(b) Trajectory distances at $t = 1026.73\,\text{s}$.



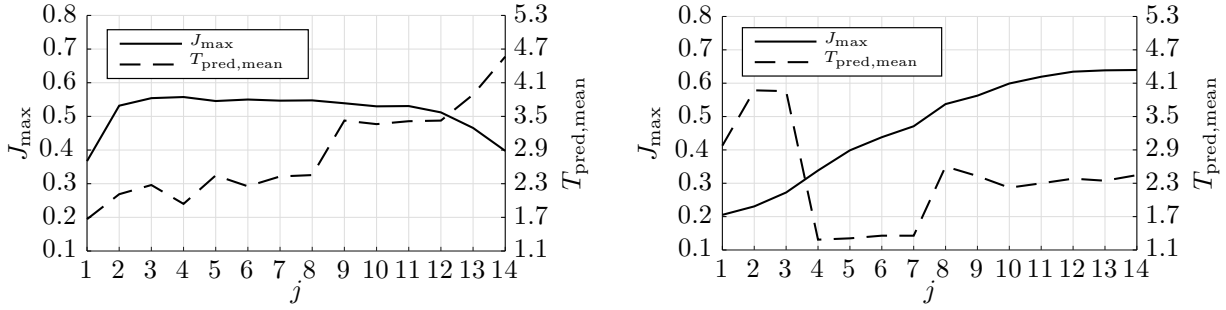
(c) Lane change detection (LCD) probability at $t = 1024.71\,\text{s}$ for the case of $j = 5$.

Figure 6.20: Lane change detection for scenario shown in Figure 6.19. Note the reversed abscissa orientation in (a) and (b) to emphasize that higher values of j correspond to classifying the nearness to trajectories more time steps ago.

on the leftmost lane). In Figure 6.19b the very same trajectories are shown together with the traffic situation after 2 s, demonstrating that the subject vehicle indeed followed the course of the predicted *lane change right* trajectory hypothesis.

6.4 Maneuver Detection Accuracy

The maneuver a driver is currently implementing is being detected by classifying the nearness of a vehicle to the trajectory hypotheses generated in the time steps before (as described in section 4.3). Since the generated trajectory hypotheses are maneuver-based (see subsection 4.2.2) as input to the logistic regression classifier the lateral distance from the subject vehicle to the formerly generated trajectory in the curvilinear coordinate system of the trajectory is used.
Using the data set $\mathcal{D}_{\text{veh}}^{(1.1.1)}$ and the same manually defined labels for the lane change execution that were used for the training and evaluation of the trajectory hypotheses in section 6.3 the maneuver detection has been trained for varying choices of j from 1 to 14. An example of the classification is shown in Figure 6.20 for the traffic situation known from Figure 6.19, where the ego vehicle is the subject. The lateral distance to each of the maneuver-based trajectory hypotheses generated $j = 0, ..., 9$ time steps before is plotted in Figure 6.20a and 6.20b, whereby the time offset to the trajectory hypothesis being classified corresponds to $\Delta t = -j \cdot T_s$ with $T_s = 0.3\,\text{s}$. In the given example the offset j is chosen to be 5, consequently the classification of an ongoing maneuver is carried out with respect



(a) Results for the ego vehicle.

(b) Results for traffic vehicles.

Figure 6.21: Maximum informedness $J_{\max}$ and mean prediction time $T_{\text{pred,mean}}$ of the maneuver detection approach for (a) the ego vehicle and (b) surrounding traffic vehicles.

to the corresponding trajectory that has been predicted 1.5 s before. For this particular case the resulting maneuver detection probability is shown in Figure 6.20c. But it is again emphasized that although for the maneuver classification a formerly predicted trajectory is being used, a new trajectory is generated in every time step for the prediction of the vehicle's future position, see subsection 4.2.3. However, the downside of a large offset j is that when a new surrounding vehicle is started to be observed the time span of $j \cdot T_s$ has to be awaited until a first classification can be performed, because the history of prediction hypotheses has to be generated first.

In Figure 6.21 the maximum informedness $J_{\max}$ and mean prediction time $T_{\text{pred,mean}}$ of the maneuver detection approach are shown for the ego vehicle (Figure 6.21a) and the surrounding traffic vehicles (Figure 6.21b) individually.

As it can be seen the maximum informedness $J_{\max}$ increases with an increasing offset j. This is expected, since when trying to classify the nearness of the vehicle to each hypothesis too soon (i.e., with a too small offset j) the signals rather represent imprecise driving behavior and sensor noise instead of the course of a currently implemented maneuver. Only if a larger time span is awaited a reliable point regarding the implemented driving behavior can be made.

However, while for the ego vehicle a plateau of the maximum informedness is already reached at $j = 3$, the classification performance for detecting traffic vehicles' maneuvers increases until $j = 12$. This shows that for the traffic vehicles the estimation of the distance to the trajectory hypotheses is much more afflicted with noise than for the ego vehicle. This could also be expected, since for the ego vehicle the distance estimation only incorporates the detection of the lane markings relative to the very same vehicle, while for traffic vehicles both the lane markings as well as the vehicle state itself are being detected through (different) environment sensors relative to the ego vehicle and are put into the perspective of the corresponding traffic vehicle only afterwards.

In contrast, the shown results of the mean prediction time $T_{\text{pred,mean}}$ may be surprising, because one might think that the greater the offset j for the trajectory classification, the lower the prediction time, because the classification is carried out to a later point in time relative to the start of a maneuver. In fact, Figure 6.21 shows that this assumption is

not unrestricted true. Since a maneuver's prediction time is computed as the duration of a sequence constantly predicted positive, a low informedness of the classifier is likely to be (although not necessarily) negatively correlated with the mean prediction time (as already explained in subsection 6.1.2). So as the classification performance computed as the maximum informedness $J_{\max}$ increases with an increasing offset j, the mean prediction time may decrease up to a certain point (and vice versa).
Based on the evaluations of Figure 6.21 an adequate trade-off between $T_{\text{pred,mean}}$ and $J_{\max}$ is chosen to be $j = 3$ in case of detecting lane changes of the ego vehicle and $j = 12$ for detecting maneuvers of surrounding traffic vehicles. These configurations will also be used for the evaluation of the overall behavior prediction in section 6.5.
In Figure 6.22 an exemplary cut-in scenario is shown. The vehicle $\mathcal{V}_{\mathrm{p_1 r_1}}$ cuts-in into the gap in front of the ego vehicle at 4990.42 s. At 4988.63 s this maneuver is being detected, which in this particular case corresponds to a prediction time of $T_{\text{pred}} = 1.79\,\text{s}$.
Another exemplary scenario is shown in Figure 6.23. Here it can be seen that the subject vehicle cuts-in into the gap behind the ego vehicle at 2108.27 s. As this maneuver is detected already at 2105.64 s the prediction time computes to $T_{\text{pred}} = 2.63\,\text{s}$.
The detection of a cut-out maneuver of the ego vehicle's preceding vehicle is shown in Figure 6.24. Already 2.00 s before the subject vehicle's lane assignment change the maneuver is being detected.
To further confirm the presented results the maneuver detection shall also be evaluated on the larger data set $\mathcal{D}^{(1)}_{\text{veh}}$. However, for this data set no manual lane change labels have been defined, because the manual labeling procedure is very time consuming and thus expensive. Therefore, first the effect of using automatically defined labels compared to the expert labels has to be quantified. This is done by evaluating the performance on the same data set $\mathcal{D}^{(1)}_{\text{veh}}$ with labels being automatically defined via the assumption of a constant lane change duration with $T_{\text{change}} = 3.0\,\text{s}$.
While for the ego vehicle the maximum informedness of the maneuver detection is $J_{\max} = 0.56$ ($w = 0.5$) on manually labeled data, using the same data set with automatic labeling results in $J_{\max} = 0.61$ ($w = 0.5$); an improvement of 8.9 %. At the same time the average maneuver prediction time slightly decreases from $T_{\text{pred,mean}} = 2.33\,\text{s}$ to 2.07 s. For traffic vehicles a similar tendency can be observed: The classification performance slightly raises from $J_{\max} = 0.63$ ($w = 0.4$) on manually labeled data to $J_{\max} = 0.66$ ($w = 0.4$) for the same data being automatically labeled. However, the prediction time increases from $T_{\text{pred,mean}} = 2.38\,\text{s}$ to 2.43 s.
The reason for the classification improvement is that the automatically defined labels are only an approximation of the true lane change duration. Consequently, as the immediate start of a lane change is hard to detect by any classification algorithm (the vehicle's movement could also just be an oscillation around the lane center line, see the exemplary trajectories shown in Figure 5.8 in subsection 5.2.2) the maneuver detection often results in predicting false negative instances at the start of a lane change, thus leading to a decrease in the prediction performance with manually defined labels.
Finally, when the larger data set $\mathcal{D}^{(1)}_{\text{veh}}$ is used as basis for the evaluation, the maximum informedness of the ego vehicle's maneuver detection on the automatically labeled data is

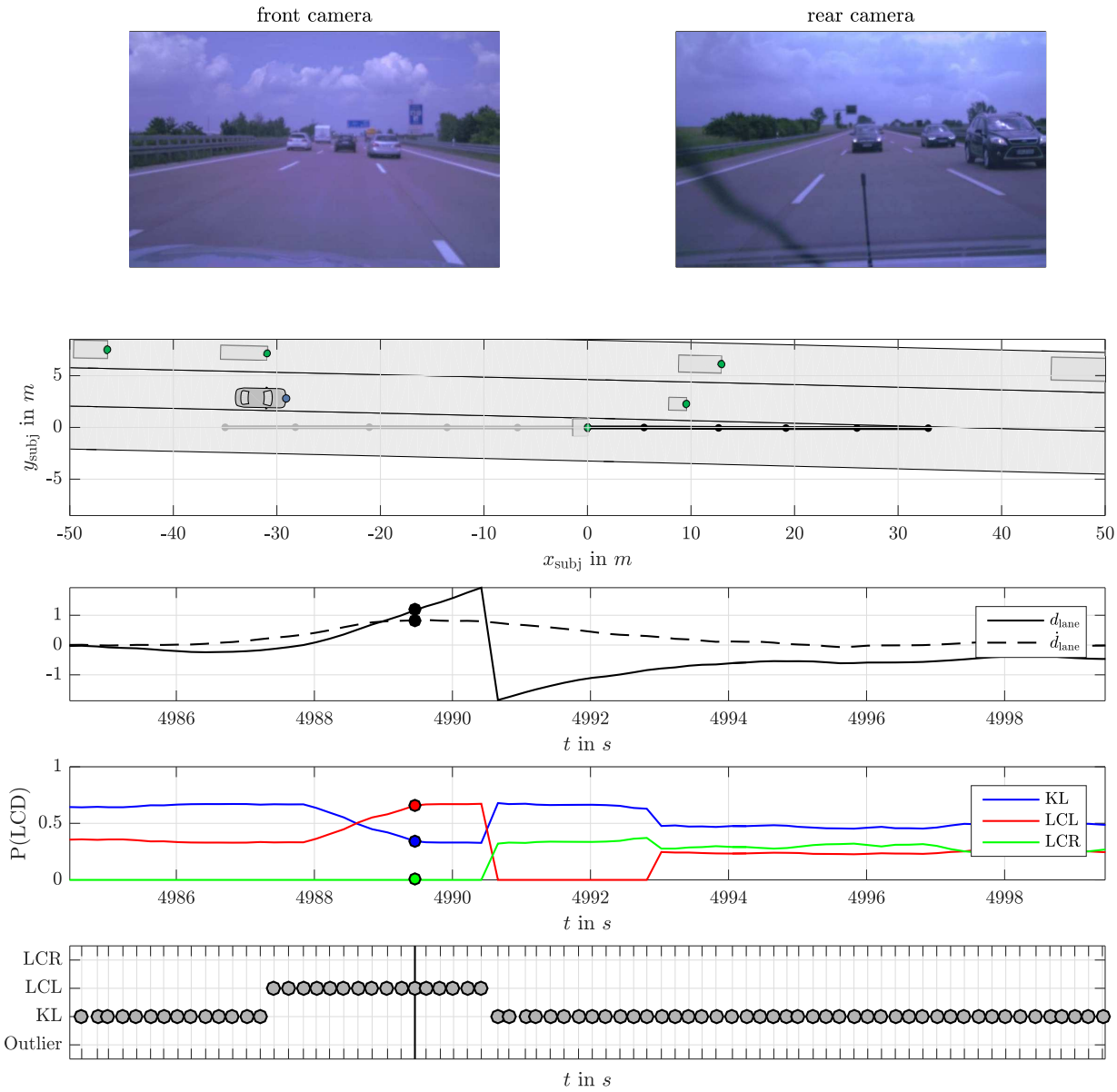


Figure 6.22: Example of a cut-in maneuver detection in front of the ego vehicle. From top to bottom: 1: Front and rear camera image of the traffic situation at marked time in plots below. 2: Top view of the traffic situation. 3: Lateral distance and velocity of the subject vehicle. 4: Course of maneuver detection probability. 5: Manually defined labels used for training and evaluation of the maneuver classification.

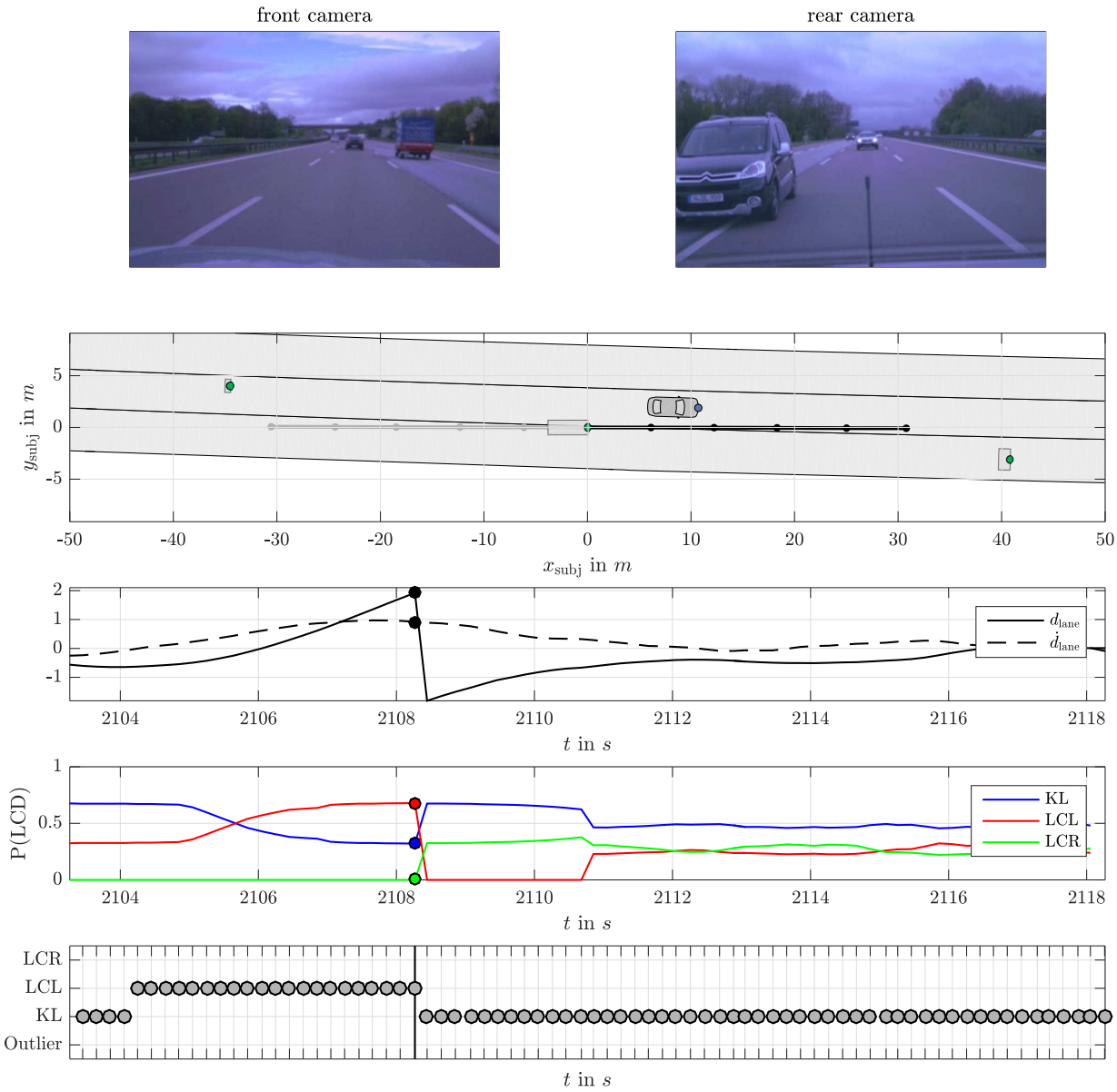


Figure 6.23: Example of a maneuver detection behind the ego vehicle. From top to bottom: 1: Front and rear camera image of the traffic situation at marked time in plots below. 2: Top view of the traffic situation. 3: Lateral distance and velocity of the subject vehicle. 4: Course of maneuver detection probability. 5: Manually defined labels used for training and evaluation of the maneuver classification.

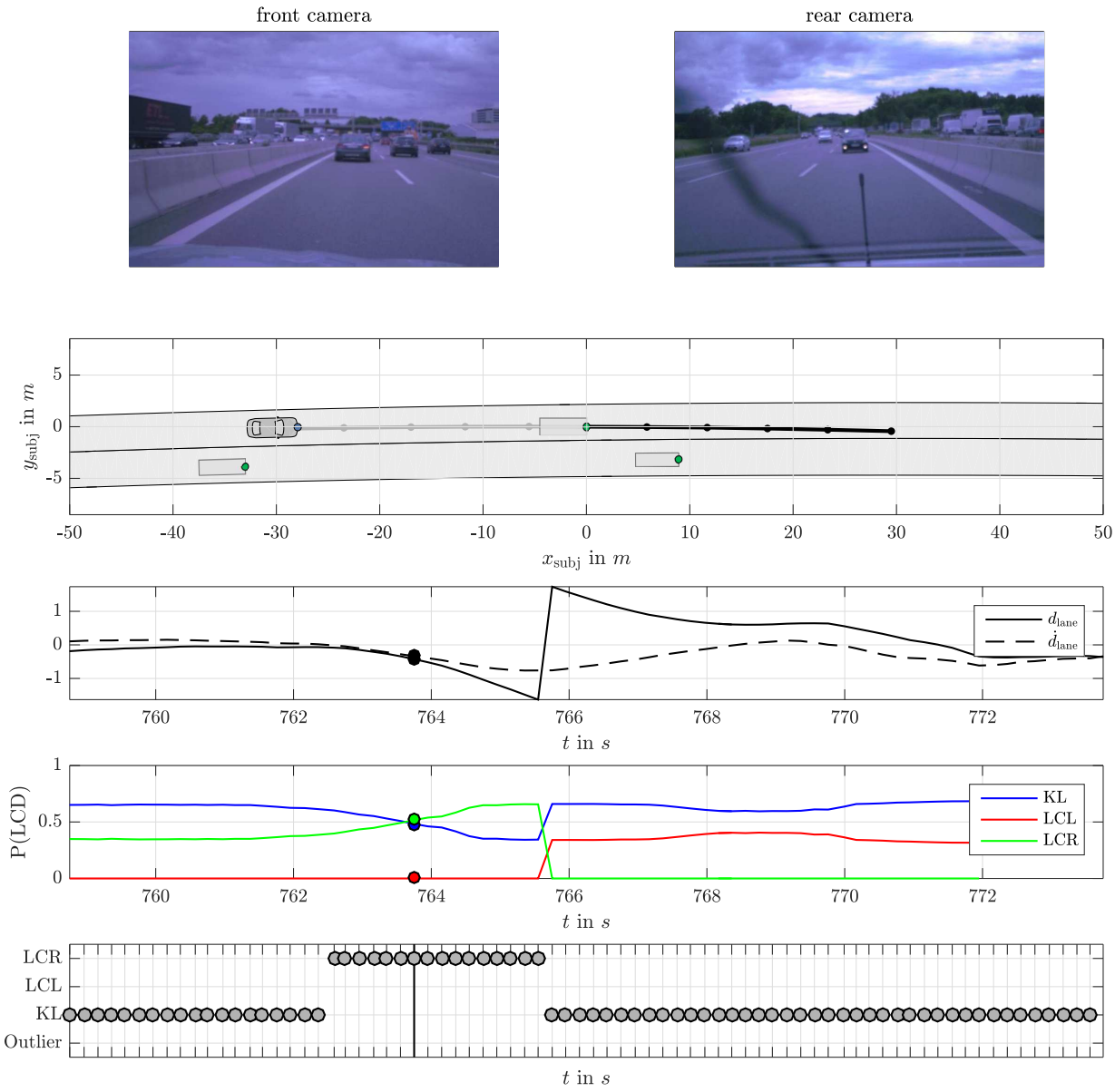


Figure 6.24: Example of a cut-out maneuver detection in front of the ego vehicle. From top to bottom: 1: Front and rear camera image of the traffic situation at marked time in plots below. 2: Top view of the traffic situation. 3: Lateral distance and velocity of the subject vehicle. 4: Course of maneuver detection probability. 5: Manually defined labels used for training and evaluation of the maneuver classification.

$J_{\max} = 0.68$ ($w = 0.5$, $T_{\text{pred,mean}} = 2.13\,\text{s}$), for the traffic vehicles' lane change detection it is $J_{\max} = 0.77$ ($w = 0.4$, $T_{\text{pred,mean}} = 2.74\,\text{s}$). In other words, the maximum informedness of the maneuver detection increased by 10.3 % for the ego vehicle and 14.3 % for the traffic vehicles using the larger data set. These results preclude the occurrence of overfitting while training the maneuver detection algorithm.

6.5 Behavior Identification Accuracy

Finally the identification of the vehicles' behavior is evaluated. As presented in section 4.3 the behavior identification basically is the combination of the intention recognition and the detection of the currently ongoing maneuver.

Like when evaluating the maneuver detection approach, for the evaluation of the behavior identification first the data set $\mathcal{D}_{\text{veh}}^{(1.1.1)}$ is used, because for this data (manually labeled) ground truth information for the exact start of all lane change maneuvers of the ego vehicle as well as of traffic vehicles exists. Therefore, with this data set the true lane change prediction performance can be evaluated - although for the intention recognition approach a different data set has been used for training due to the lack of lane change intention labels.

With a maximum informedness of $J_{\max} = 0.70$ ($w = 0.6$) the ego vehicle's behavior identification performs 26 % better compared to the pure maneuver-based lane change detection approach on the same data set. At average the ego vehicle's lane changes are anticipated 3.25 s before the center of the vehicle's front actually crosses the lane marking. This is an advantage of 0.92 s compared to using the maneuver detection only.

For predicting the traffic vehicles' lane change behavior a maximum informedness of $J_{\max} = 0.74$ ($w = 0.6$) is achieved, which is a performance of 117 % compared to the traffic vehicle's maneuver detection on the same data set. Thereby lane changes of traffic vehicles are anticipated at average 2.45 s beforehand, which is quite comparable to using solely the maneuver detection approach ($-0.06\,\text{s}$).

These results show that the vehicles' lane change prediction accuracy can significantly be improved when not only detecting the start of a maneuver reactively, but also making use of recognizing the driver's maneuver intention based on the predominant traffic situation.

In Figure 6.25 an exemplary cut-in maneuver of the traffic vehicle $\mathcal{V}_{\text{p}_1\text{r}_1}$ with respect to the ego vehicle is shown. It can be seen that while the lane change detection (LCD) predicts the lane change 1.81 s in advance to the actual lane assignment change, the lane change intention (LCI) is already recognized 11.41 s ahead. Eventually the intention recognition and maneuver detection lead to a lane change behavior identification already 6.41 s before the actual lane change.

An exemplary cut-out scenario is shown in Figure 6.26. Already 7.80 s seconds in advance to the lane change the driver of the vehicle in front of the ego vehicle is recognized having the intention to change to the right lane. The characteristic lane change movement is detected 5.00 s later. To sum up, the cut-out behavior of the subject vehicle is anticipated 6.40 s ahead of the lane assignment change.

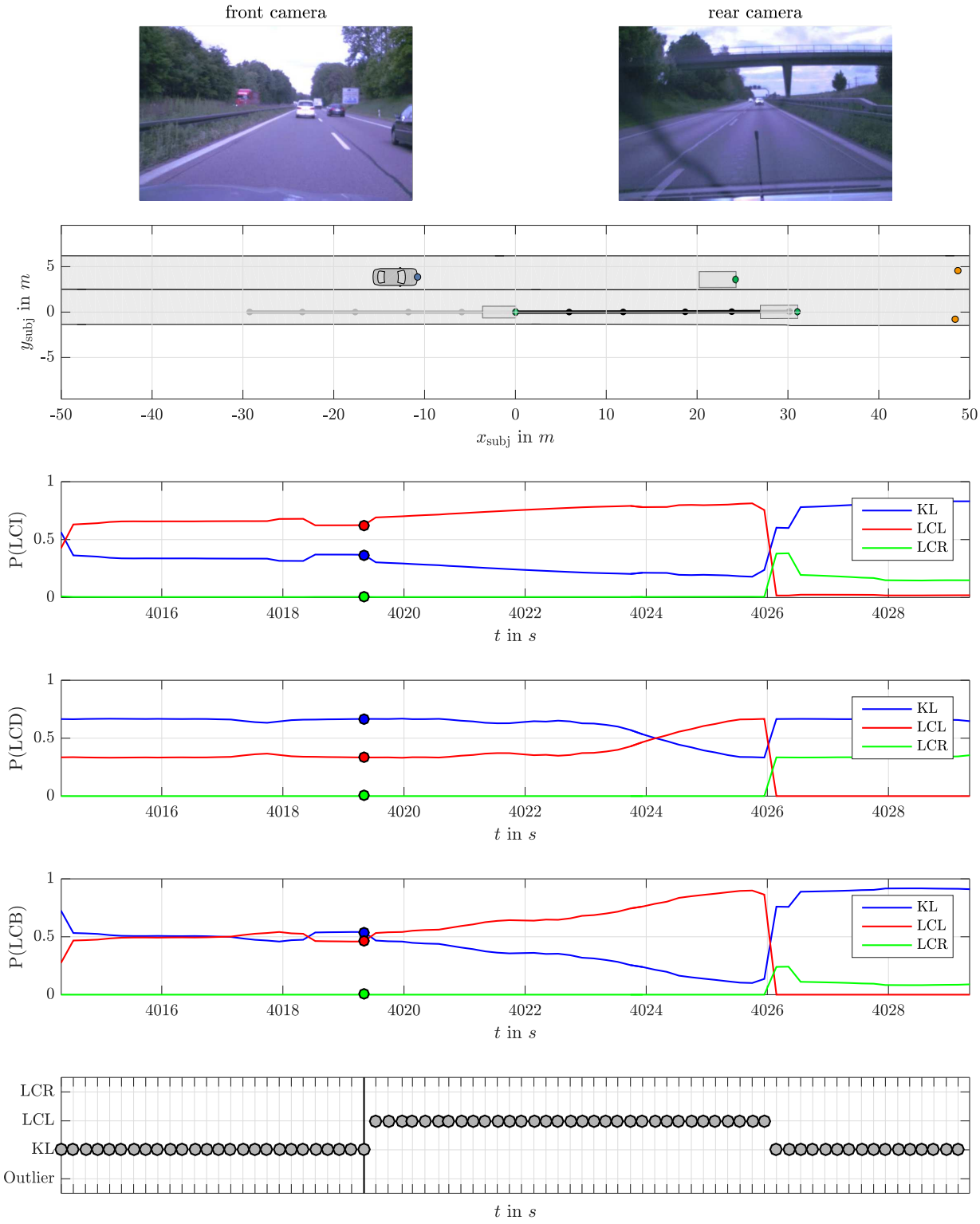


Figure 6.25: Example of a cut-in behavior identification in front of the ego vehicle. From top to bottom: 1: Top view of the traffic situation. 2: Course of inferred maneuver intention probability. 3: Course of inferred maneuver detection probability. 4: Course of inferred behavior identification probability. 5: Labels used for training and evaluating the maneuver classification.

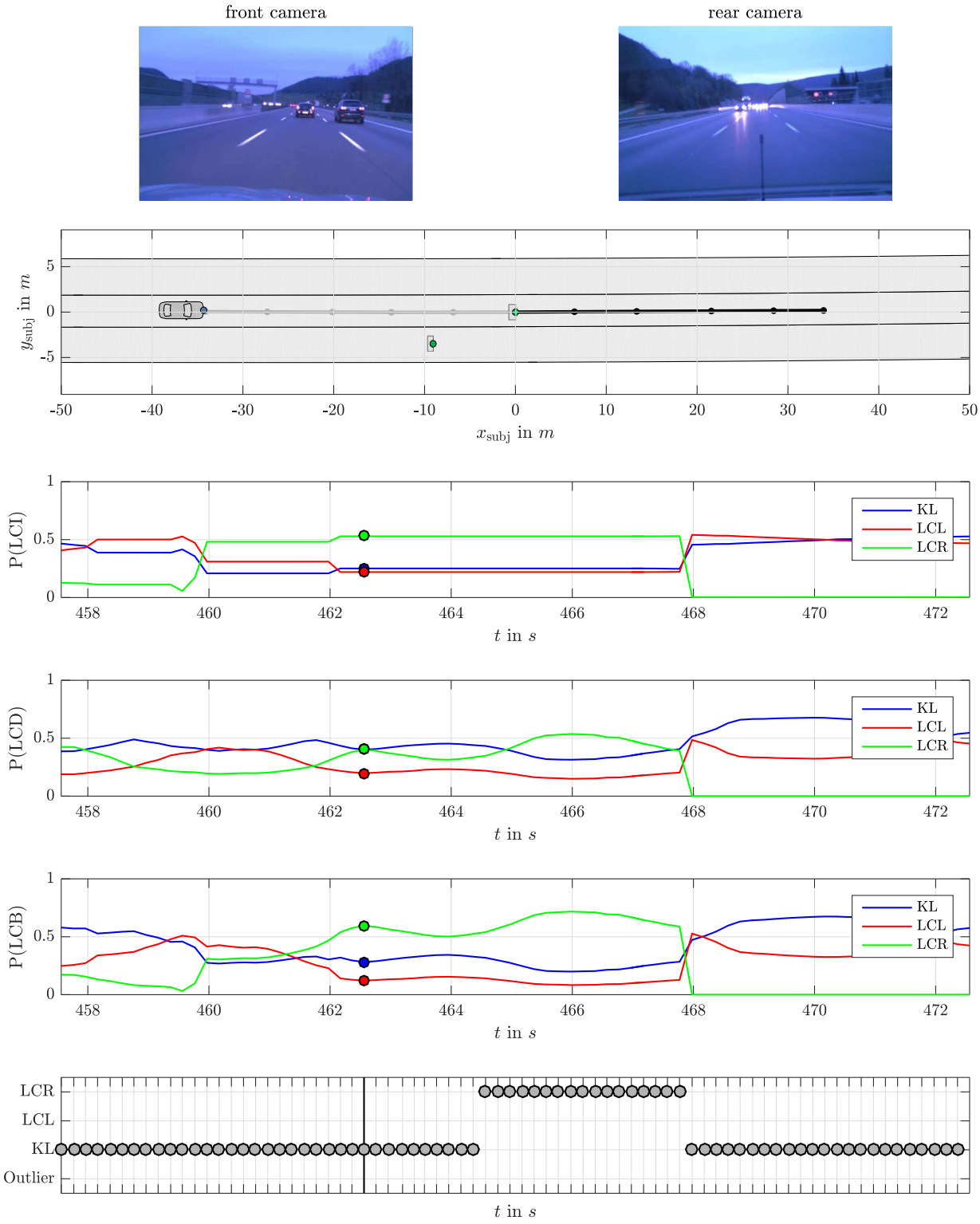


Figure 6.26: Example of a cut-out behavior identification in front of the ego vehicle. From top to bottom: 1: Top view of the traffic situation. 2: Course of inferred maneuver intention probability. 3: Course of inferred maneuver detection probability. 4: Course of inferred behavior identification probability. 5: Labels used for training and evaluating the maneuver classification.

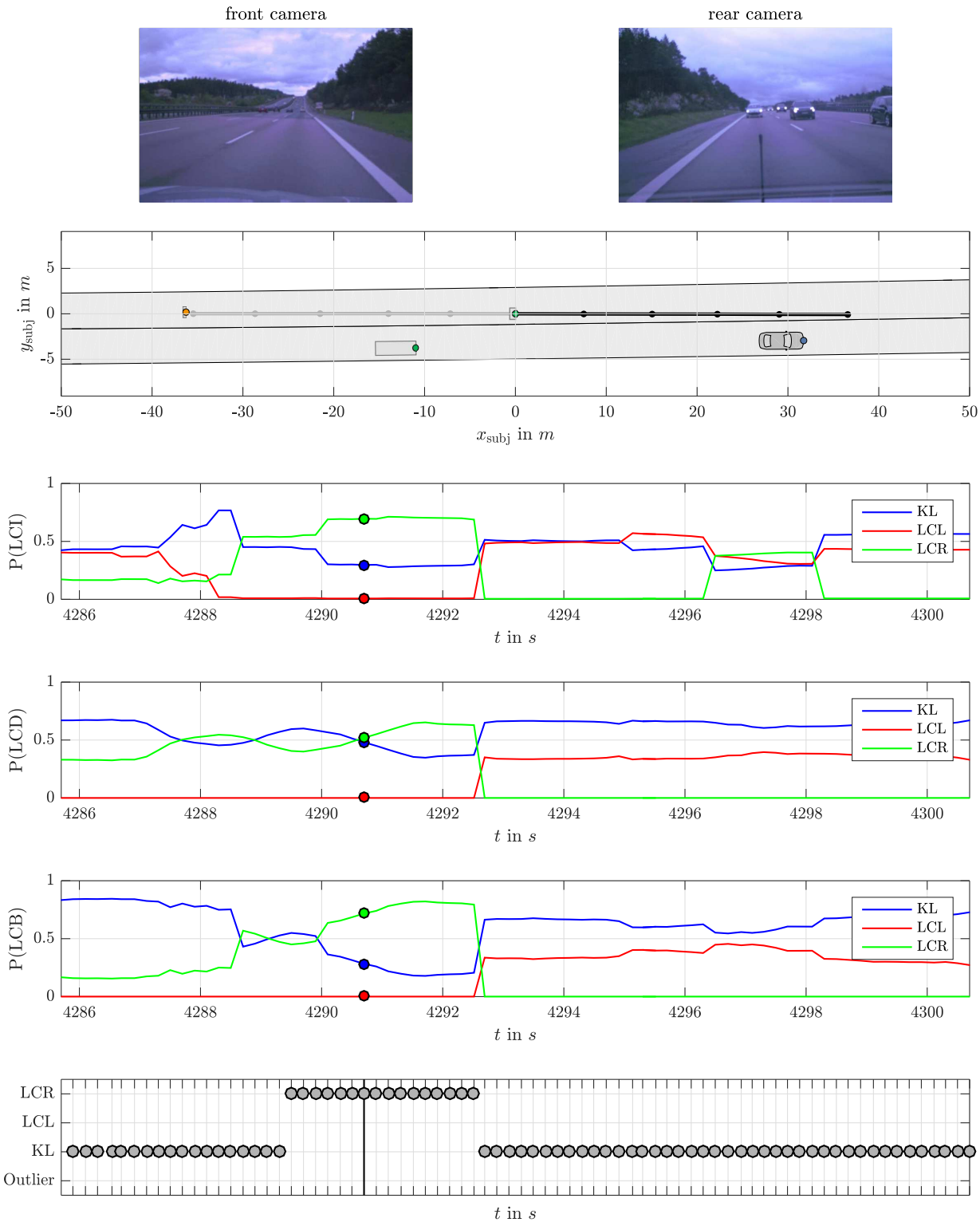


Figure 6.27: Example of a cut-in behavior identification behind the ego vehicle. From top to bottom: 1: Top view of the traffic situation. 2: Course of inferred maneuver intention probability. 3: Course of inferred maneuver detection probability. 4: Course of inferred behavior identification probability. 5: Labels used for training and evaluating the maneuver classification.

Another lane change scenario is shown in Figure 6.27. Here the traffic vehicle $\mathcal{V}_{s_1 l_1}$ changed into the gap behind the ego vehicle from the left after overtaking the ego vehicle's successor. It can be seen that the maneuver detection at first wrongly imputed a lane change from $t = 4287.70\,\mathrm{s}$ to $4288.90\,\mathrm{s}$. Only at $t = 4290.70\,\mathrm{s}$ the lane change is correctly detected. However, the driver's intention to change the lane is already correctly anticipated from $t = 4288.70\,\mathrm{s}$. Therefore the false positive lane change prediction only shows through in the behavior identification from $t = 4288.70\,\mathrm{s}$ to $4289.10\,\mathrm{s}$, but from $t = 4290.10\,\mathrm{s}$ on the lane change is correctly identified. This corresponds to a prediction time of $T_{\mathrm{pred}} = 2.42\,\mathrm{s}$. (The falsely recognized lane change intentions from $t = 4295.14\,\mathrm{s}$ to $4298.10\,\mathrm{s}$ originate from a defective lane marking detection that is just not in sight yet.)
For both the ego vehicle as well as the traffic vehicles the behavior identification performance has also been evaluated using automatic labeling: First on data set $\mathcal{D}_{\mathrm{veh}}^{(1.1.1)}$ (for comparison) and eventually on the larger data set $\mathcal{D}_{\mathrm{veh}}^{(1.1)}$.
On $\mathcal{D}_{\mathrm{veh}}^{(1.1.1)}$ for the ego vehicle the maximum informedness of the behavior identification based on automatically defined labels is $J_{\max} = 0.73$ ($w = 0.6$) with a mean prediction time of $T_{\mathrm{pred,mean}} = 3.49\,\mathrm{s}$. For the traffic vehicles' predictions the performance is $J_{\max} = 0.76$ ($w = 0.6$) and $T_{\mathrm{pred,mean}} = 2.45\,\mathrm{s}$. So like before (and thus expected), the evaluation actually shows better results when using automatically defined labels as ground truth compared to labels being manually defined by a human expert. In particular, for the ego vehicle the the maximum informedness has improved by 4.6 % and for the traffic vehicles at least by 2.8 %.
Finally the performance on the data set $\mathcal{D}_{\mathrm{veh}}^{(1)}$ is analyzed. With a maximum informedness of $J_{\max} = 0.74$ ($w = 0.5$) and a mean prediction time of $T_{\mathrm{pred,mean}} = 6.02\,\mathrm{s}$ for predicting the lane changing behavior of the ego vehicle the results of the smaller data set $\mathcal{D}_{\mathrm{veh}}^{(1.1.1)}$ can not only be confirmed, but it is also shown that the prediction time averaged over all predicted lane changes even increased.
However, the classification performance for identifying the traffic vehicles' behavior has decreased by 11.8 % to $J_{\max} = 0.68$ ($w = 0.5$) while the mean prediction time shows slightly improved results with $T_{\mathrm{pred,mean}} = 2.66\,\mathrm{s}$. This is surprising, because when evaluating the pure maneuver-based lane change detection approach in section 6.4 the performance actually got better on the larger data set. But in terms of absolute numbers the behavior identification is still superior to the maneuver detection by 11.5 %.

CHAPTER 7

Conclusion

7.1 Summary

For the task of assisted as well as highly or even fully automated driving it is crucial for the technical system to understand the current traffic situation and to anticipate the future positions not only of the ego vehicle, but also of the surrounding traffic participants. To contribute to this requirement, in this thesis a novel framework has been proposed that is able to infer not only maneuver intentions of the ego vehicle's driver (during assisted driving) but especially those of drivers in the vicinity. Also the future positions of their vehicles are projected.

In detail, first the Bayesian network presented in section 3.4 infers the probability of a driver's intention to change to a different lane mainly based on characteristics describing the inter-vehicle relations. As described in subsection 5.3.4 the BN has been pre-trained with simulation data to guide the optimization of the parameters representing the internal, hidden states of the model towards a meaningful representation.

Second, maneuver-based trajectory hypotheses describing the course of possible future positions of the vehicles are generated by solving an optimal control problem as stated in subsection 4.2.2. The underlying assumption is that human drivers always try to minimize costs when driving, for example in terms of the required time or acceleration of the vehicle to reach the desired state.

Finally, the individual future driving behavior is identified by observing the vehicle's movement and classifying the correctness of any generated trajectory hypothesis in the very next time steps. At the same time the information about the driver's maneuver intention is being taken into account to improve the quality of the predictions.

All models employed in the prediction framework have been trained and evaluated based on 1100 km of real world highway driving with a prototype test vehicle. Different labeling strategies have been investigated regarding the quality of the labels to be used as input

to the machine learning approaches as well as regarding the scalability of the labeling approaches towards an increasing amount of data.
Summarizing it could be confirmed that manually labeling driving data offline (i.e., after the data has been recorded) by a human expert is very time consuming and thus expensive. But interestingly transferring the labeling procedure to the vehicle and labeling lane changes online while driving did not change this fact and also didn't hypothesize better results of the prediction algorithm in the end. On the other hand, automatic labeling simplified and accelerated the labeling process a lot, but as the automatically defined lane change labels are only an approximation of the exact start of a maneuver the automatic labeling procedure led to the evaluation metrics showing a little bit too optimistic results. Nonetheless, with the help of the automatic labeling procedure the concept of inferring a driver's lane change intentions primarily based on the predominant traffic situation could be verified. A maximum informedness of $J_{\max} = 0.55$ (labeling with $T_{\text{intention}} = 3.0\,\text{s}$) demonstrates that the proposed hybrid Bayesian network is able to infer lane change intentions of the surrounding vehicles' drivers in the majority of the observed cases. With a mean prediction time of $T_{\text{pred,mean}} = 4.68\,\text{s}$ this allows for a safe and comfortable reaction to cut-in maneuvers of automated vehicles in the future.
It could also be shown that by pre-training the network's parameters with the help of the state of the art traffic simulation software Pelops the internal states of the model actually are meaningful, which not only helps to understand why particular lane changes are intended, but also to find the causes of false predictions. This is in contrast to a neural network, where even a human expert is not able to interpret the meaning of individual parameters - least of all being able to manually make reasonable adjustments after the training process.
In the prediction framework started maneuvers can be detected contemporary to their beginning by first generating trajectory hypotheses describing likely future positions of any vehicle and comparing the actually implemented trajectory in the subsequent time steps. Indeed, with $T_{\text{pred,mean}} = 2.53\,\text{s}$ the time this maneuver-based prediction approach anticipates lane changes in advance is shorter compared to the recognition of maneuver intentions via the Bayesian network, but with a maximum informedness of $J_{\max} = 0.61$ the approach is more accurate.
Finally, the behavior identification fuses the information of the intention recognition and maneuver detection, which results in an even more precise prediction with a maximum informedness of $J_{\max} = 0.68$. With a mean prediction time of $T_{\text{pred,mean}} = 2.66\,\text{s}$ the requirement of 2.4 s discussed in section 4.1 is fulfilled.

7.2 Discussion

Clearly the approaches for a driving behavior prediction presented in this thesis have limitations and the described investigations are based on assumptions.
Far and foremost, the training of the prediction framework and the evaluation of the prediction results presented in chapter 6 are all based on a data set which has been hand-

selected in terms of filtering sequences with faulty environment model information. So in order to obtain the presented results in the vehicle while driving the reliability of the environment perception has to be improved, especially for bad weather conditions like rain (see comparison of intention recognition results with and without data samples with faulty sensor detections in Figure 6.17). But it is expected that the maneuver prediction can be improved even further when more information about the vehicles' environment is being detected, e.g. the current speed limit or the activation of an object vehicle's indicator (see the composition of features for the existing lane contentedness factors and description of the omitted factors in section 3.4).

On the other hand, also algorithmic weaknesses have to be mentioned. When it comes to inference in Bayesian networks, indeed the employed clique tree algorithm (explicated in subsection 3.3.5) can handle inference also in a hybrid BN, avoiding discretization which can potentially lead to information loss. But unfortunately this advantage may come at the cost of large clique sizes, which implies that the inference procedure can become impractical. Therefore, the structure of the Bayesian network has to be chosen carefully to allow for a manageable inference in terms of the computational complexity.

With the current implementation of the exact inference method in MATLAB as an extension of the Bayes Net Toolbox (MURPHY, 2001) the inference of a single driver's maneuver intention with the network structure illustrated in Figure 3.15 takes 150 ms on a 2.6 GHz Intel Core i5 (I5-4288U). To be able to integrate the approach in the automated driving platform the algorithm implementation needs to be optimized and ported to C/C++. On the other hand the trajectory hypotheses generation via solving an optimal control problem and the classification of a hypothesis' correctness via a logistic regression classifier has a run-time of less than 15 ms per trajectory when using the (not optimized) implementation in MATLAB and is therefore not run-time critical.

Another assumption that has been made for the task of predicting driving maneuvers is that human behavior is independent of time series information and past decisions. While the good performance of the prediction framework shows that this assumption is justifiable in most cases, there also exist characteristics in the human driving behavior that can only be modeled by also accounting for information from the past. For example, it can be assumed that a human driver has the motivation to not change lanes often, so the intention of a driver to change lane depends on whether a lane change has already been performed in the near past. Another example is the acceptance of driving on a lane with lower contentedness compared to a neighboring lane. It is assumed that the longer a driver tolerates driving on an adverse lane, the higher is the motivation to eventually change to a more advantageous lane.

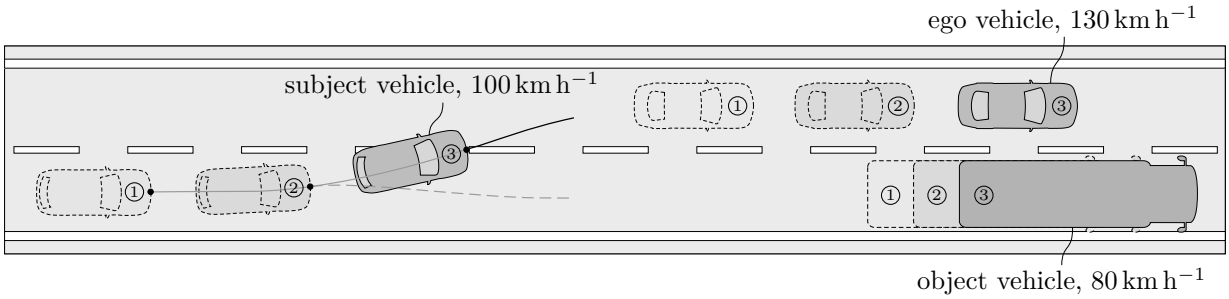
While a Bayesian network generally allows to be extended to a dynamic Bayesian network using the Markov property (i.e., the conditional probability distribution of future states, which are conditioned on both the past and present states, only depends on the present state), exact inference becomes impractical due to an exponential blow up of the number of states accumulating over time, see (LERNER, 2002, pp. 178 sqq.).

Finally it has to be emphasized that the evaluation methodology and metrics also have downsides. First of all, the maneuver prediction time metric has to be handled with

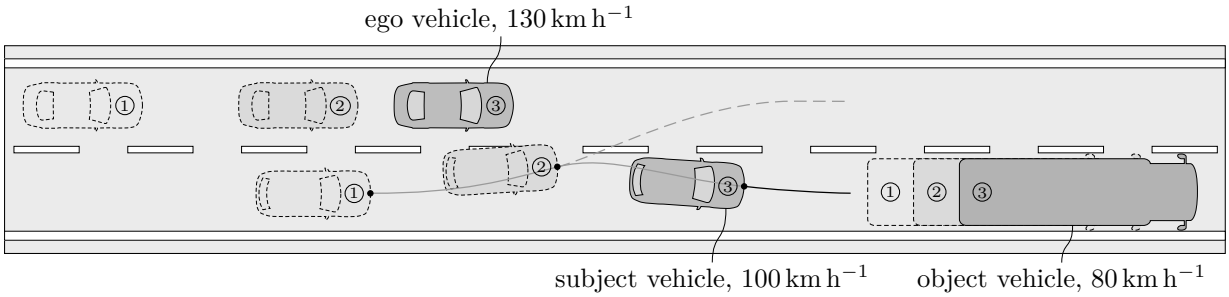
caution. As already explained in subsection 6.1.2 and seen in section 6.4, it is prone to overestimate the true maneuver prediction time in case of a high false positive rate.
Moreover, in this thesis all predictions have been evaluated solely using performance metrics quantifying the correctness of the predictions with respect to the true evolution of the traffic situation. But these metrics decrease even for scenarios where wrong predictions in turn may not effect the driving behavior of the technical system at all. For example, if a false negative lane change prediction (i.e., a lane change that is not predicted but actually happens) regards a subject vehicle which is driving behind the ego vehicle, the later is not affected by this (unpredicted) lane change in first place (see Figure 7.1a). On the other hand, there even exist situations in which the reaction of the driving automation system to a likely cut-in maneuver of a subject vehicle may be reasonable and desired behavior although no lane change is ever taking place. For example, if a subject vehicle attempts a close cut-in maneuver and aborts the lane change just before merging into the ego vehicle's lane, it is desirable for the automated driving ego vehicle to react to the risk of a potential collision. Moreover, in such cases even a human driver is often not able to anticipate the future evolution of the traffic situation correctly. So as the prediction has the aim to enable assisted or even fully automated driving, a better indication of an expedient prediction would be to measure an improvement of the driving behavior implemented by the final technical system when employing the proposed prediction framework. But unfortunately no automated driving function is available for testing the prediction framework in closed-loop behavior yet.

7.3 Future Directions

The assessment of the current and future traffic situation as presented in this thesis enables a variety of future investigations and improvements. For example, the long-term prediction of a vehicle's future trajectory can be used to improve the environment sensors' object tracking process as it allows to keep or re-associate the track of an observed object that is being occluded for a longer period of time (e.g. when driving next to a truck). As a consequence individual objects don't immediately vanish from the environment model when being occluded. Moreover, objects can be identified as unique vehicles for a longer period of time.
A longer tracking time of a uniquely identified vehicle again may enable identifying the driving characteristics of a particular driver, because any prediction algorithm has more time to observe and learn the individual driving behavior. In other words, the prediction model could take into account different driving styles.
Another improvement could be to not only use causal but also agnostic reasoning in the Bayesian network. While so far the inference of a lane change intention was queried with the cause of the different lane contentedness factors via evidence of the features characterizing the predominant traffic situation, an observed lane change could also be reasoned. In this case evidence would be given for the lane change and the cause of the lane change could be explained subsequently, for example allowing for an inference of an individual driver's



(a) Tolerable false negative prediction.



(b) Tolerable false positive prediction.

Figure 7.1: False predictions (visualized as dashed lines) that are tolerable. (a) False negative prediction of subject vehicle at t_2, which does not affect the driving behavior of the ego vehicle since the lane change takes place behind. (b) False positive prediction at t_2 due to the subject vehicle aborting a lane change attempt. In this case it is not only tolerable, but also desirable and safe behavior to brake behind the potentially cutting in subject vehicle.

desired velocity that may have motivated the maneuver.
For future investigations the proposed structure of the Bayesian network to infer a driver's lane change intention can easily be extended, e.g. by a contentedness factor regarding the driver's desired route. As input to a route contentedness factor an agnostic (i.e., explaining) reasoning path from the previous inferences could be used. For example, if a lane change is observed that was not motivated by one of the other contentedness factors, chances are that the driver changed the lane in order to take a highway exit.

APPENDIX A

Algorithm Pseudocode

A.1 Clique Tree Construction for Hybrid BN

Pseudocode for the clique tree construction in case of hybrid BN as presented in subsection 3.3.4.

```
1:  function CliqueTreeConstruction(𝒢)
2:      // connect parents of each node
3:      𝒢_M ← MoralizeGraph(𝒢)
4:      // choose strong elimination order π, virtually
5:      // eliminate nodes in order π while connecting all
6:      // nodes formerly connected to the eliminated node
7:      𝒢_T ← StronglyTriangulateGraph(𝒢_M)
8:      // for each discrete node with continuous parents,
9:      // connect all parents with the discrete neighbors
10:     // of the parents' continuous connected component
11:     𝒢_I ← AssureIntegrableDistributions(𝒢_T)
12:     // find maximal cliques 𝓒 in 𝒢_I
13:     𝓒 ← FindMaximalCliques(𝒢_I)
14:     // let 𝓒 be the nodes of a tree, connected by
15:     // the sepsets 𝓢 with S_{i:j} = C_i ∩ C_j
16:     𝒯_C ← BuildCliqueTree(𝓒)
17:     return 𝒯_C
```

A.2 Clique Tree Calibration for Hybrid BN

Pseudocode for incorporating evidence and running the message passing procedure (calibrating a clique tree) in case of hybrid BN as presented in subsection 3.3.5.

1: **function** CliqueTreeCalibration($\mathcal{T}_C, \boldsymbol{\theta}, \boldsymbol{d}$)
2: // initialize all CG potentials to their identity element
3: $\Phi_C \leftarrow$ "empty" $\forall \mathcal{C} \in \mathcal{T}_C$
4: // remove all (C)CD CPD from the clique tree
5: $\mathcal{T}_{C\setminus\mathcal{S}} \leftarrow$ BuildReducedCliqueTree($\mathcal{T}_C$)
6: **for** $i \leftarrow 1, ..., |\boldsymbol{\theta}|$ **do**
7: **if** evidence exists in $\boldsymbol{d}$ for scope of $\boldsymbol{\theta}_i$ **then**
8: // use evidence to initialize factor
9: $\phi_i \leftarrow$ InitializeFactorFromEvidence($\boldsymbol{d}$)
10: **else if** CPD type of $\boldsymbol{\theta}_i$ is (C)CD CPD **then**
11: // CG potentials cannot be initialized from (C)CD CPD
12: **continue**
13: **else**
14: // use CPD parameters to initialize factor
15: $\phi_i \leftarrow$ InitializeFactorFromCPD($\boldsymbol{\theta}_i$)
16: // multiply factor ϕ_i onto associated potential
17: $\Phi_C\langle\phi_i\rangle \leftarrow \Phi_C\langle\phi_i\rangle \otimes \phi_i$
18: // collect evidence and copy root potential
19: $\mathcal{T}_{C\setminus\mathcal{S}} \leftarrow$ CollectEvidence($\mathcal{T}_{C\setminus\mathcal{S}}$)
20: $\Phi_{\text{root,pre-calib}} \leftarrow \Phi_{\text{root}} \in \mathcal{T}_{C\setminus\mathcal{S}}$
21: // incorporate all (C)CD CPD using Equation 3.41b
22: $\mathcal{T}_C \leftarrow$ IncorporateSoftmaxDistributions($\mathcal{T}_{C\setminus\mathcal{S}}$, $\Phi_{\text{root,pre-calib}}$, $\boldsymbol{\theta}$)
23: // run message passing procedure
24: $\mathcal{T}_C \leftarrow$ CollectEvidence($\mathcal{T}_C$)
25: $\mathcal{T}_C \leftarrow$ DistributeEvidence($\mathcal{T}_C$)
26: $\mathcal{T}_{C,\text{calib}} \leftarrow \mathcal{T}_C$
27: **return** $\mathcal{T}_{C,\text{calib}}$, $\Phi_{\text{root,pre-calib}}$, ℓ

A.3 Expectation Maximization

Pseudocode for the expectation maximization (EM) algorithm as presented in subsection 5.3.3.

function ExpectationMaximization($\mathcal{G}$, $\boldsymbol{\theta}^0$, $\mathcal{D} = \{\boldsymbol{d}[1], ..., \boldsymbol{d}[M]\}$)
  // initialization
  $t \leftarrow 0$
  $\ell^0 \leftarrow -\infty$
  $\mathcal{T}_C \leftarrow$ CliqueTreeConstruction($\mathcal{G}$)
  repeat
    $t \leftarrow t + 1$
    for $m \leftarrow 1, ..., M$ **do**
      // enter evidence and run message passing
      $\mathcal{T}_{C,\text{calib}}$, $\Phi_{\text{root,pre-calib}}$, $\hat{\ell} \leftarrow$ CliqueTreeCalibration($\mathcal{T}_C$, $\boldsymbol{\theta}^{t-1}$, $\boldsymbol{d}[m]$)
      $\ell^t = \ell^t + \hat{\ell}$
      for $i \leftarrow 1, ..., |\boldsymbol{\theta}|$ **do**
        if CPD type of $\boldsymbol{\theta}_i$ is in exponential family **then**
          // infer expected values and update expected sufficient statistics
          $s(\mathcal{D})_i \leftarrow$ UpdateESS($\mathcal{T}_C$, $s(\mathcal{D})_i$)
        else if CPD type of $\boldsymbol{\theta}_i$ is (C)CD CPD **then**
          // copy raw data
          $\hat{d}_i[m] \leftarrow \boldsymbol{d}[m]$ w.r.t. scope of $\boldsymbol{\theta}_i$
    for $i \leftarrow 1, ..., |\boldsymbol{\theta}|$ **do**
      if CPD type of $\boldsymbol{\theta}_i$ is in exponential family **then**
        // compute ML estimate based on expected sufficient statistics
        $\boldsymbol{\theta}_i^t \leftarrow$ MaximizeParametersESS($s(\mathcal{D})_i$)
      else if CPD type of $\boldsymbol{\theta}_i$ is (C)CD CPD **then**
        // compute ML estimate based on raw data
        $\boldsymbol{\theta}_i^t \leftarrow$ MaximizeParametersSoftmax($\mathcal{D}$, $\Phi_{\text{root,pre-calib}}$)
  until $(\ell^{t-1} - \ell^t) > \epsilon$
  return $\boldsymbol{\theta}, \ell$

A.4 Maximum Entropy IRL

Pseudocode for the maximum entropy inverse reinforcement learning (IRL) algorithm, which has been presented in section 5.4.

function MaximumEntropyIRL($\tilde{\boldsymbol{x}}_1, ..., \tilde{\boldsymbol{x}}_n$)
  // compute empirical feature vector
  $\tilde{\mathbf{f}} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{N} \tilde{\boldsymbol{x}}_{i,k}^2$
  // initialization
  $\boldsymbol{Q} \leftarrow \text{diag}(1, 1, 1)$
  repeat
    for $i \leftarrow 1, ..., n$ **do**
      // solve quadratic program
      $\mathbf{u}^* \leftarrow \text{QuadProg}(2\boldsymbol{H}, 2\boldsymbol{x}_0^T, \boldsymbol{F})$
      $\mathbf{x}_i \leftarrow \boldsymbol{\mathcal{A}}\boldsymbol{x}_0 + \boldsymbol{\mathcal{B}}\mathbf{u}^*$
    // estimate feature vector w.r.t. current $\boldsymbol{Q}$
    $\mathbf{f} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{N} \mathbf{x}_{i,k}^2$
    // compute gradient
    $\boldsymbol{g} \leftarrow \mathbf{f} - \tilde{\mathbf{f}}$
    for $j \leftarrow 1, ..., \dim(\boldsymbol{Q})$ **do**
      if $\boldsymbol{Q}(j,j) - \alpha \boldsymbol{g}(j) > 0$ **then**
        // take gradient step
        $\boldsymbol{Q}(j,j) \leftarrow \boldsymbol{Q}(j,j) - \alpha \boldsymbol{g}(j)$
      else
        // limit parameters to lower bound
        $\boldsymbol{Q}(j,j) \leftarrow 0$
  until $||g|| > \epsilon$
  return $\boldsymbol{Q}$

APPENDIX B

Deduction of Time to Brake Metric

In the following the time to brake (T_{tb}) metric shall be deduced, which denotes the remaining time span for a subject vehicle until a braking maneuver has to be started in order to maintain a minimum safety distance $s_{\text{rel,safe}}$ to a slower object vehicle in front just in time. Thereby, for the preceding object vehicle constant velocity (i.e., $a_{\text{obj}} = 0$) is assumed. The subject vehicle on collision course also approaches with constant velocity in the beginning, but in the very last moment it decelerates with $a_{\text{subj,min}}$ instantly, so

$$a_{\text{subj}}(t) = \begin{cases} 0 & \text{if } t < t_{\text{brake}} \\ a_{\text{subj,min}} & \text{if } t_{\text{brake}} \leq t \leq t_{\text{end}}. \end{cases} \tag{B.1}$$

To only just avoid a collision between the subject and the preceding object vehicle it is required that

$$v_{\text{rel}}(t = t_{\text{end}}) \stackrel{!}{=} 0 \quad \text{and} \tag{B.2}$$

$$s_{\text{rel}}(t = t_{\text{end}}) \stackrel{!}{=} s_{\text{rel,safe}}. \tag{B.3}$$

In words, the relative distance s_{rel} (which according to Equation 3.3 is defined as clearance) between the vehicles has to be exactly the safety distance $s_{\text{rel,safe}}$ at t_{end}, which is the end of the maneuver. Moreover, the relative velocity between the vehicles must be equalized. With $v_0 = \dot{v}(t = t_0)$ as well as $a_0 = \ddot{s}(t = t_0)$, and with the equations of motion

$$v(t = t_s) = v_0 + \int_{t_0}^{t_s} a(t)\,\mathrm{d}t \quad \text{and} \tag{B.4}$$

$$s(t = t_s) = s_0 + v_0\,t_s + \iint_{t_0}^{t_s} a(t)\,(\mathrm{d}t)^2 \tag{B.5}$$

the postulations in Equation B.2 and B.3 become

$$v_{\text{rel},0} + a_{\text{subj,min}}(t_{\text{end}} - t_{\text{brake}}) \stackrel{!}{=} 0 \quad \text{and} \tag{B.6}$$

$$s_{\text{rel},0} + v_{\text{rel},0} t_{\text{end}} + \tfrac{1}{2} a_{\text{subj,min}}(t_{\text{end}} - t_{\text{brake}})^2 \stackrel{!}{=} s_{\text{rel,safe}}, \tag{B.7}$$

respectively. Solving Equation B.6 for t_{end} leads to

$$t_{\text{end}} = t_{\text{brake}} - \frac{v_{\text{rel},0}}{a_{\text{subj,min}}}. \tag{B.8}$$

Inserting Equation B.8 in Equation B.7 yields

$$\begin{aligned}
s_{\text{rel,safe}} &= s_{\text{rel},0} + v_{\text{rel},0}(t_{\text{brake}} - \frac{v_{\text{rel},0}}{a_{\text{subj,min}}}) + \frac{1}{2}\, a_{\text{subj,min}}((t_{\text{brake}} - \frac{v_{\text{rel},0}}{a_{\text{subj,min}}}) - t_{\text{brake}})^2 && \text{(B.9a)}\\
&= s_{\text{rel},0} + v_{\text{rel},0}\, t_{\text{brake}} - \frac{v_{\text{rel},0}^2}{a_{\text{subj,min}}} + \frac{1}{2}\, a_{\text{subj,min}}((t_{\text{brake}} - \frac{v_{\text{rel},0}}{a_{\text{subj,min}}})^2 \\
&\quad - 2\,(t_{\text{brake}} - \frac{v_{\text{rel},0}}{a_{\text{subj,min}}})\, t_{\text{brake}} + t_{\text{brake}}^2) && \text{(B.9b)}\\
&= s_{\text{rel},0} + v_{\text{rel},0}\, t_{\text{brake}} - \frac{v_{\text{rel},0}^2}{a_{\text{subj,min}}} + \frac{1}{2}\, a_{\text{subj,min}}(t_{\text{brake}}^2 - 2\, t_{\text{brake}} \frac{v_{\text{rel},0}}{a_{\text{subj,min}}} \\
&\quad + \frac{v_{\text{rel},0}^2}{a_{\text{subj,min}}^2} - 2\, t_{\text{brake}}^2 + 2\, \frac{v_{\text{rel},0}}{a_{\text{subj,min}}}\, t_{\text{brake}} + t_{\text{brake}}^2) && \text{(B.9c)}\\
&= s_{\text{rel},0} + v_{\text{rel},0}\, t_{\text{brake}} - \frac{v_{\text{rel},0}^2}{a_{\text{subj,min}}} + \frac{1}{2}\, a_{\text{subj,min}}\, t_{\text{brake}}^2 - v_{\text{rel},0}\, t_{\text{brake}} \\
&\quad + \frac{1}{2}\, \frac{v_{\text{rel},0}^2}{a_{\text{subj,min}}} - a_{\text{subj,min}}\, t_{\text{brake}}^2 + v_{\text{rel},0}\, t_{\text{brake}} + \frac{1}{2}\, a_{\text{subj,min}}\, t_{\text{brake}}^2 && \text{(B.9d)}\\
&= s_{\text{rel},0} + v_{\text{rel},0}\, t_{\text{brake}} - \frac{1}{2}\, \frac{v_{\text{rel},0}^2}{a_{\text{subj,min}}} && \text{(B.9e)}
\end{aligned}$$

Finally, solving Equation B.9e for t_{brake} results in

$$t_{\text{brake}} = \frac{v_{\text{rel},0}}{2\, a_{\text{ego,min}}} - \frac{s_{\text{rel},0} + s_{\text{rel,safe}}}{v_{\text{rel},0}}. \tag{B.10}$$

For the case of $t_0 = 0$ this point in time is equal to the time span T_{tb}, the so-called time to brake.

APPENDIX C

Road Model Estimation

As the environment model available in the car while driving contains only a road model with limited observation range (due to the sensor's limitations and the fact that no precise digital map is available yet), an algorithm to post-process all lane marking detections in a recorded measurement is presented. The goal is to fuse all lane marking detections provided by the camera in a sliding window approach by not only storing past lane marking information to keep track of lanes behind the vehicle (since there is only one camera mounted at the vehicle's front, see subsection 5.1.3), but also to look ahead in time and taking into account lane boundaries that the vehicle will pass by in the future.

Though this foresight information obviously is not available while driving and therefore cannot be used in a driving automation system, for the task of developing behavior prediction algorithms it is nevertheless inevitable to have an improved road observability over the detections provided by the camera at an individual measurement in time. Using the foresight information is justified by the expectation to have a precise digital map available in the car as the development of these systems evolves. Eventually in the future the digital map will replace the road model foresight processing presented in this section.

While Figure C.1 visualizes the road model generation process in a flow chart, in Figure C.2 the individual steps that are necessary to build the road model are illustrated in detail.

The first step of the road model generation is to process lane marking detections provided by the camera. In the robot operating system (ROS), it is either possible to receive sensor information online in the vehicle as soon as the camera sends it, or by iterating over all sensor detections saved in a measurement file offline. Because the road model algorithm presented here takes future information from the camera's lane marking detections into account that are not available online in the vehicle (as described before), the later method to receive the measurement information is used in the following.

The course of an individual lane marking detected by the camera system is represented by

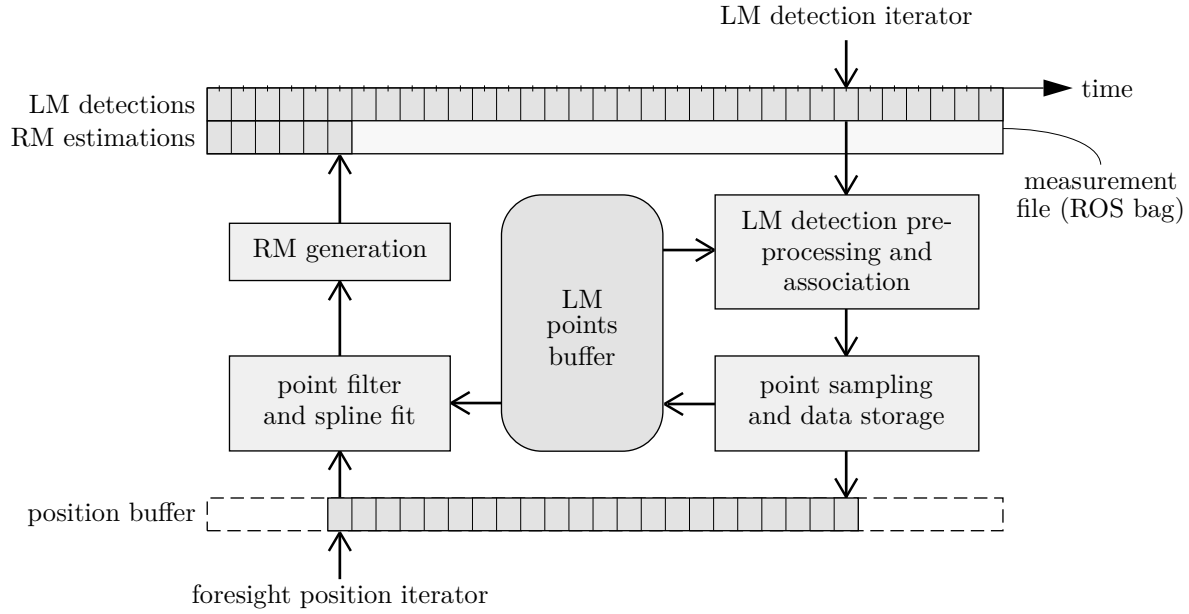


Figure C.1: Chart of processing steps to build the road model (RM) from the camera's lane marking (LM) detections.

a polynomial of 3rd order in the ego vehicle's coordinate system, so it holds

$$y(x) = ax^3 + bx^2 + cx + d \tag{C.1}$$

with x as the longitudinal position and y the lateral deposition as shown in Figure C.2a. Additionally, information about the detected marking type (whether the marking is solid or dashed etc.), the longitudinal observation range $x_{\max}$ and the detection quality is given. As the camera is able to reliably detect the lane markings of the ego lane only, the road model generation algorithm uses a simple heuristic to check if the existence of an undetected adjacent lane marking is likely: Iff the left (or right) marking of the ego lane is of type dashed and no adjacent lane marking is detected, a synthetic lane marking detection with a solid line type and an offset of a standard lane width $l_{w,\text{std}}$ is added. This is done by setting the polynomial coefficients a_{syn}, b_{syn} and c_{syn} of the synthetic lane marking equal to the original lane marking polynomial and using $d_{syn} = d_{orig} \pm l_{w,\text{std}}$, respectively. The standard lane width is chosen based on the German regulations for the structure of highways (German *Richtlinien für die Anlage von Autobahnen*, RAA) that provide for a lane with of 3.75 m or 3.5 m depending on the absolute lane number counted from the rightmost lane. Because the absolute lane number the ego vehicle is driving on can not reliably be obtained by the camera, a standard lane width of $l_{w,\text{std}} = 3.75\,\text{m}$ is assumed.
From each lane marking polynomial a number of points along the x-axis of the ego vehicle are sampled in the second processing step, visualized in Figure C.2b. To be able to use the sampled points at a later time step, all points are transformed from the ego vehicle's coordinate frame to a world-fixed odometry frame, otherwise the point's positions would become invalid as soon as the ego vehicle's position changes over time. For each individual lane marking a container is created that holds all points sampled from the marking.

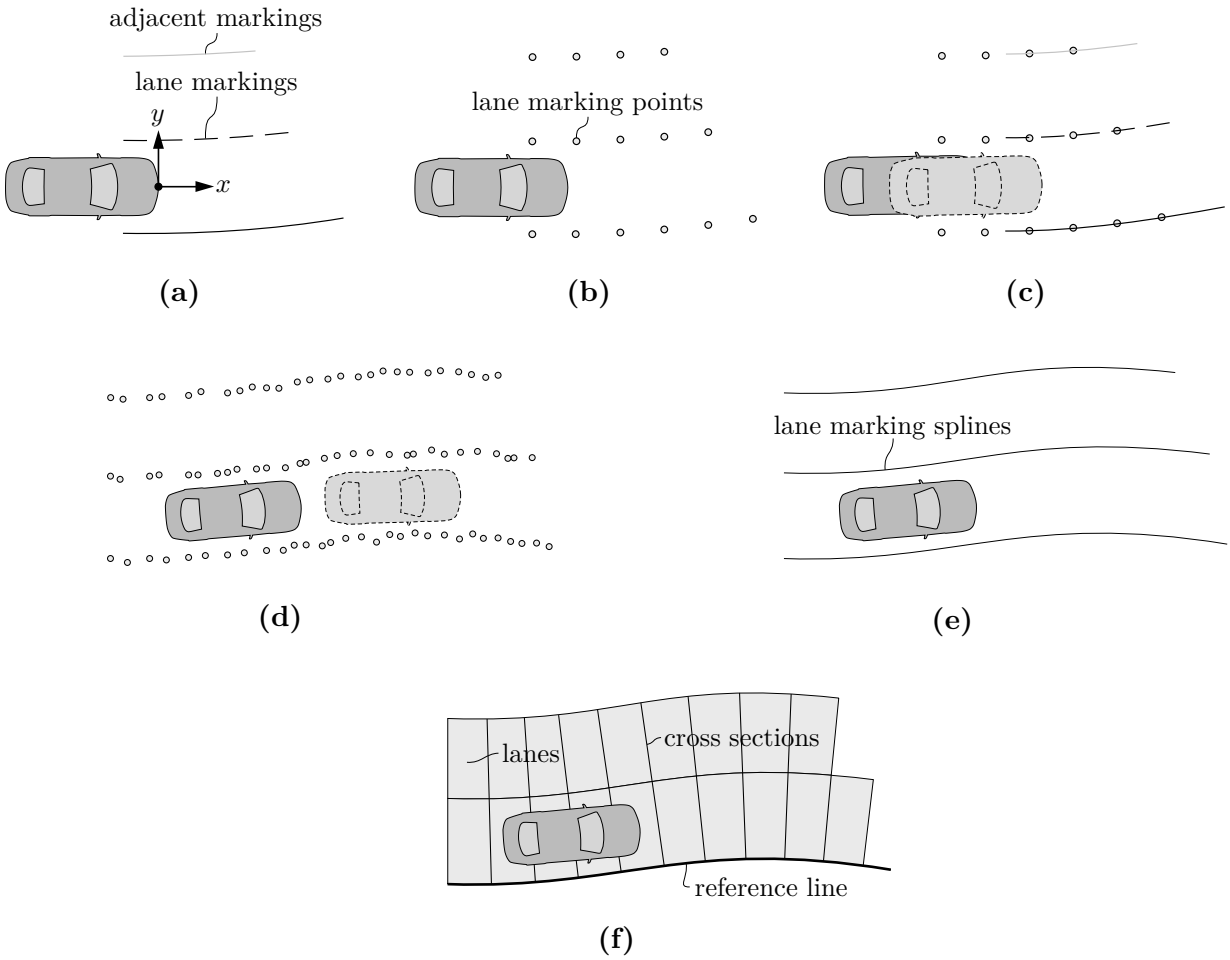


Figure C.2: Visualization of processing steps to build the road model (RM) with foresight information from the camera's lane marking (LM) detections. (a) Assuming adjacent lane markings when marking of ego lane is dashed. (b) Sampling points from lane markings and converting them to odometry frame. Saving all points belonging to an individual lane marking. (c) Associating new lane marking detections to known lane marking point containers. (d) Evolution of steps a - c until desired foresight distance is reached. (e) Fitting a spline to each individual lane marking. (f) Creating a road model from the lane marking splines in the OpenDRIVE format.

Also the ego vehicle's own position in the odometry frame is saved in a buffer at each time step to keep track of the positions at which lane marking detections have been taken into account already (see Figure C.1).
As now the ego vehicle drives ahead and new lane marking detections are provided by the camera in the next time step, the new markings are associated to known lane markings from previous time steps by checking the distance between the new lane marking and the point closest to the marking of each stored lane marking point container as shown in Figure C.2c. If the new marking is close enough to a known marking, points are again sampled from the new marking polynomial and added to the point container of the known marking. In case no association can be performed a new lane marking point container is created. As the process of sampling points from new marking polynomials and adding the points to known marking containers repeats with every new detection provided by the camera, the number of points in each lane marking container grows (see Figure C.2d).
Next, at each time step it is checked if sufficient foresight information is acquired by summing up all inter-point distances in the ego vehicle position buffer, which delivers the distance traveled by the vehicle during the lane marking information collection phase. As the detection of surrounding vehicles provided by the LIDAR sensor is limited to 100 m (see subsection 5.1.3) it is sufficient to limit the desired foresight distance and the maximum distance of old points behind the ego vehicle to 200 m.
When the desired foresight distance is reached a spline is fitted to all points of each lane boundary using a least squares fit. The result is illustrated in Figure C.2f.
Based on the lane marking splines all information needed to generate a road model in the OpenDRIVE road description format are computed (OpenDRIVE Format Specification, 2015). The final road model is visualized in Figure C.2f.

APPENDIX D

Environment Model Faults

In the following a number of exemplary environment model faults of the test vehicle are illustrated that had to be labeled as outlier in order to be ignored when training and evaluating the driving behavior prediction framework presented in this thesis.

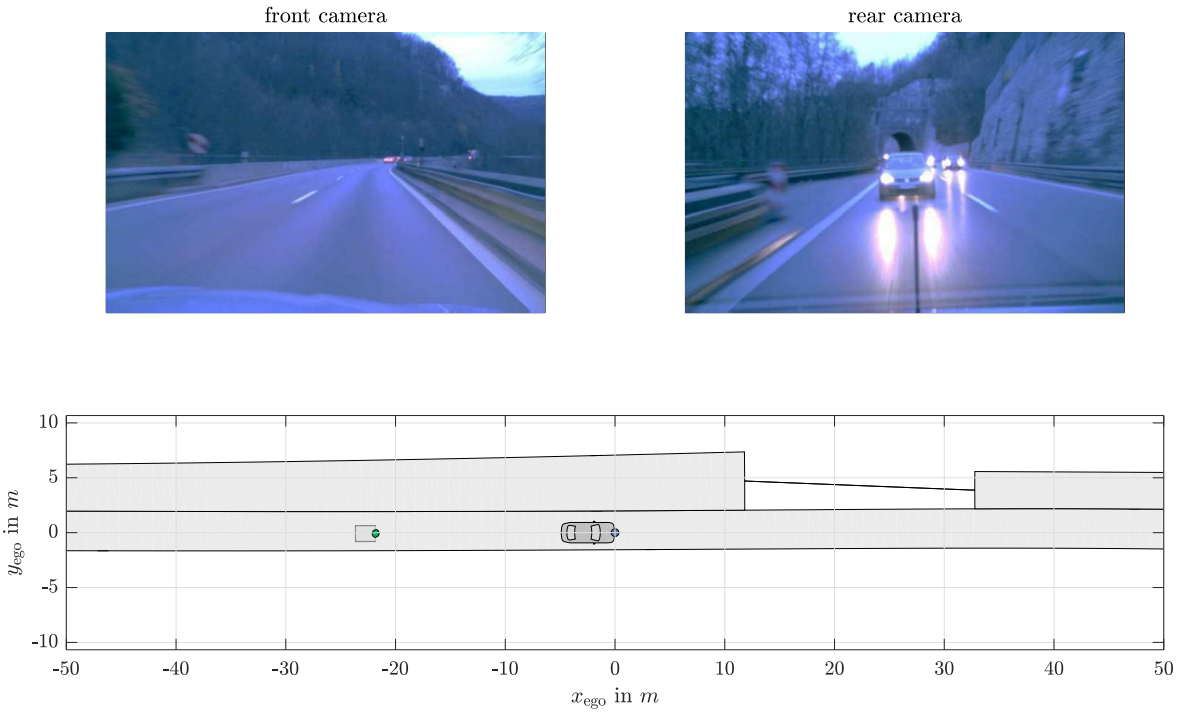


Figure D.1: Exemplary false negative detection of left lane boundary. First the lane width of the left lane is overestimated, then the detection of the left marking is missing completely, causing the lane model to be interrupted.

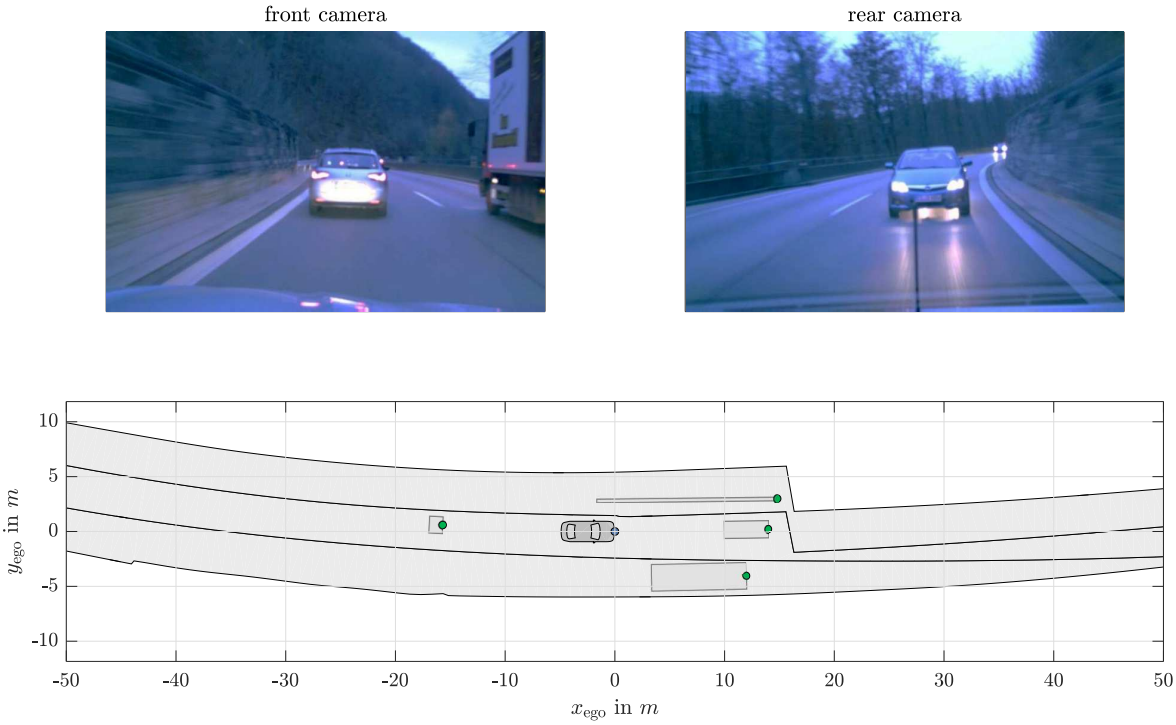


Figure D.2: Exemplary false positive detection of left lane boundary, because the computer vision system erroneously interprets the side wall as the left lane marking. Additionally the LIDAR sensors classify the peripheral development on the left side as dynamic traffic object by mistake.

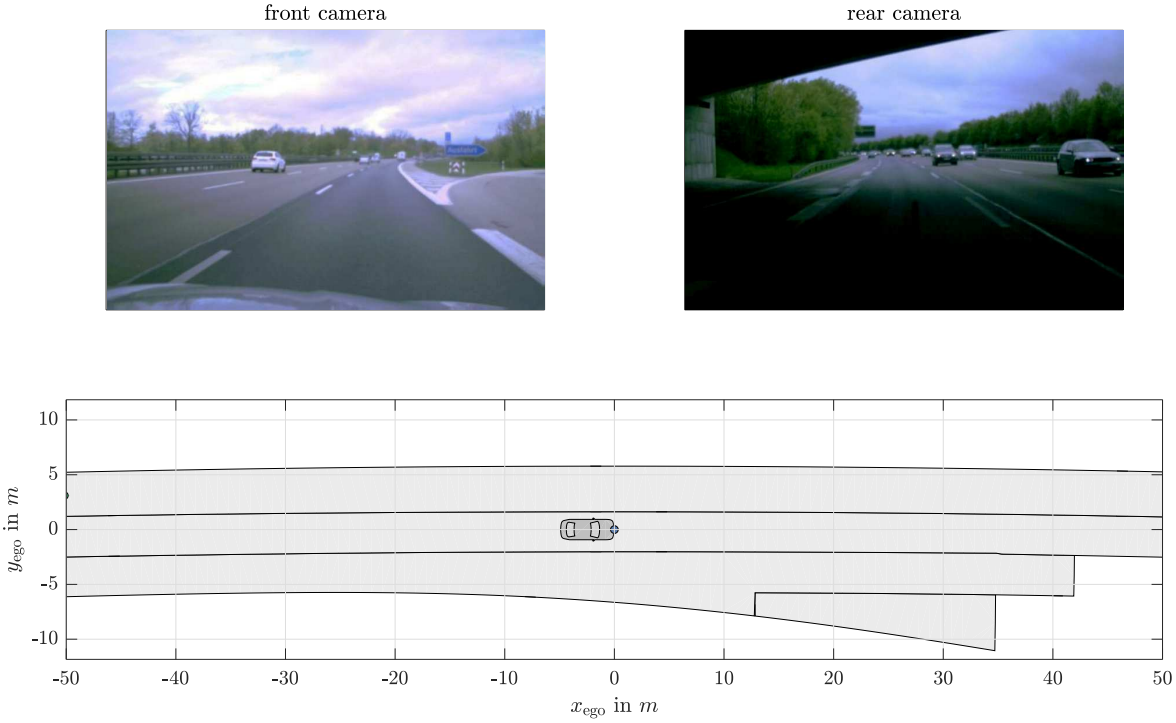


Figure D.3: Example of erroneous lane geometry detection at highway exits.

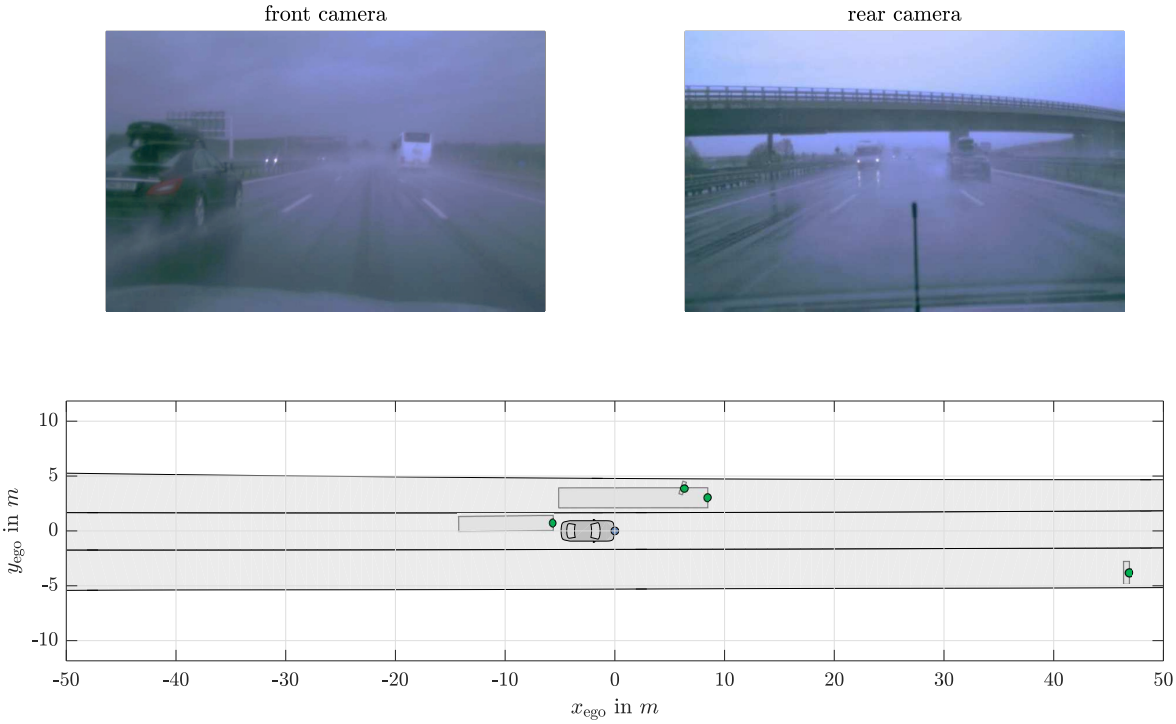


Figure D.4: Exemplary false positive detections of dynamic objects due to heavy rain (e.g., ghost object right behind the ego vehicle). Moreover, the length of the vehicle on the left lane is overestimated due to spray.

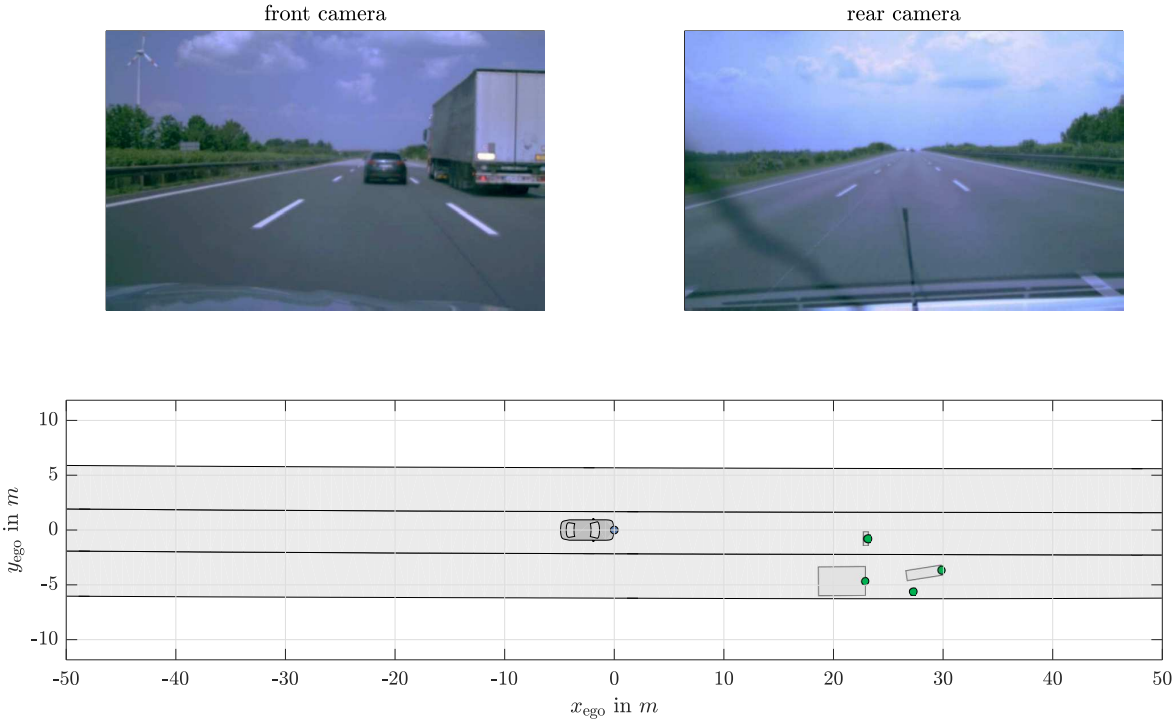


Figure D.5: Example of a truck detection being decomposed into multiple objects. When overtaking the truck and sensing the object more obtuse-angled the detections are fused to a single object.

List of Figures

List of Tables

Bibliography

Abbeel, Pieter, Ng, Andrew Y. (2004). „Apprenticeship Learning via Inverse Reinforcement Learning“. In: *Proceedings of the 21st International Conference on Machine Learning*. ICML '04. Association for Computing Machinery. DOI: 10.1145/1015330.1015430.

Ahmed, Kazi Iftekhar (1999). „Modeling Drivers Acceleration and Lane Changing Behavior“. PhD thesis. Massachusetts Institute of Technology, Dept. of Civil and Environmental Engineering.

Althoff, Matthias, Stursberg, Olaf, Buss, Martin (2009). „Model-Based Probabilistic Collision Detection in Autonomous Driving“. In: *IEEE Transactions on Intelligent Transportation Systems* 10.2. DOI: 10.1109/tits.2009.2018966.

Ata, Baris, Coban, Ramazan (2017). „Linear Quadratic Optimal Control of an Inverted Pendulum on a Cart using Artificial Bee Colony Algorithm: An Experimental Study“. In: *Cukurova University Journal of the Faculty of Engineering and Architecture* 32.2.

Bahram, Mohammad, Lohrer, Artur, Aeberhard, Michael (2014). „Generatives Prädiktionsmodell zur frühzeitigen Spurwechselerkennung“. In: *9. Workshop Fahrerassistenzsysteme*. URL: https://www.uni-das.de/images/pdf/fas-workshop/2014/FAS2014_Tagungsband.pdf (visited on 11/18/2018).

Barth, Alexander, Franke, Uwe (2008). „Where Will the Oncoming Vehicle be the Next Second?“ In: *IEEE Intelligent Vehicle Symposium*. DOI: 10.1109/ivs.2008.4621210.

Bengio, Yoshua (2009). „Learning Deep Architectures for AI“. In: *Foundations and Trends in Machine Learning*. Vol. 2. 1. DOI: 10.1561/2200000006.

Berndt, Holger, Dietmayer, Klaus (2009). „Driver Intention Inference with Vehicle Onboard Sensors“. In: *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. DOI: 10.1109/icves.2009.5400203.

Berndt, Holger, Emmert, Jörg, Dietmayer, Klaus (2008). „Continuous Driver Intention Recognition with Hidden Markov Models“. In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. doi: 10.1109/itsc.2008.4732630.

Bertele, Umberto, Brioschi, Francesco (1973). „On Non-serial Dynamic Programming“. In: *Journal of Combinatorial Theory, Series A*. Vol. 14. 2. doi: 10.1016/0097-3165(73)90016-2.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Information science and statistics. Springer. doi: 10.1198/tech.2007.s518.

Bonnin, Sarah, Weisswange, Thomas H., Kummert, Franz, Schmüdderich, Jens (2013). „Accurate Behavior Prediction on Highways Based on a Systematic Combination of Classifiers“. In: *IEEE Intelligent Vehicle Symposium (IV)*. doi: 10.1109/ivs.2013.6629477.

Bonnin, Sarah, Weisswange, Thomas H., Kummert, Franz, Schmüdderich, Jens (2014). „General Behavior Prediction by a Combination of Scenario-Specific Models“. In: *IEEE Transactions on Intelligent Transportation Systems* 15.4. doi: 10.1109/tits.2014.2299340.

Borelli, Francesco, Bemporad, Alberto, Morari, Manfred (2015). *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press. doi: 10.1017/9781139061759.

Boyraz, Pinar, Sathyanarayana, Amardeep, Hansen, John H. L. (2009). „Driver Behavior Modeling Using Hybrid Dynamic Systems For 'Driver-Aware' Active Vehicle Safety“. In: *Proceedings of the 21st (ESV) International Technical Conference on the Enhanced Safety of Vehicles*.

Bundesanstalt für Strassenwesen (BASt) (2015). *Verkehrs- und Unfalldaten - Kurzzusammenstellung der Entwicklung in Deutschland*. url: http://www.bast.de/DE/Publikationen/Medien/Dokumente/Unfallkarten-national-deutsch.pdf (visited on 11/18/2018).

Bundesrepublik Deutschland (Mar. 2013). *Straßenverkehrs-Ordnung (StVO)*.

Dagli, Ismail, Breuel, Gabi, Schittenhelm, Helmut, Schanz, Alexander (2004). „Cutting-in Vehicle Recognition for ACC Systems - Towards Feasible Situation Analysis Methodologies“. In: *IEEE Intelligent Vehicles Symposium (IV)*. doi: 10.1109/IVS.2004.1336509.

Dagli, Ismail, Reichardt, Dirk (2002). „Motivation-based Approach to Behavior Prediction“. In: *Proceedings of the IEEE Intelligent Vehicle Symposium* 1. doi: 10.1109/IVS.2002.1187956.

Deo, Nachiket, Rangesh, Akshay, Trivedi, Mohan M. (2018). „How Would Surround Vehicles Move? A Unified Framework for Maneuver Classification and Motion Prediction“. In: *IEEE Transactions on Intelligent Vehicles* 3.2. doi: 10.1109/tiv.2018.2804159.

Dickmanns, Ernst Dieter, Graefe, Volker (1988). „Applications of Dynamic Monocular Machine Vision“. In: *Machine Vision and Applications* 1.4. doi: 10.1007/bf01212362.

Dogan, Ueruen, Edelbrunner, Johann, Iossifidis, Ioannis (2011). „Autonomous Driving: A Comparison of Machine Learning Techniques by Means of the Prediction of Lane Change Behavior“. In: *Proceedings of the IEEE International Conference on Robotics and Biomimetics*. doi: 10.1109/robio.2011.6181557.

Donges, Edmund (1982). „Aspekte der Aktiven Sicherheit bei der Führung von Personenkraftwagen“. In: *Automobil-Industrie* 27.2. issn: 0005-1306. url: https://trid.trb.org/view/1044474.

Dupuis, Marius et al. (2015). *OpenDRIVE Format Specification*. url: http://www.opendrive.org/docs/OpenDRIVEFormatSpecRev1.4H.pdf (visited on 11/18/2018).

Ehmanns, Dirk (2002). „Modellierung des taktischen Fahrerverhaltens bei Spurwechselvorgängen“. PhD thesis. RWTH Aachen.

Ehmanns, Dirk, Hochstädter, Almut (2000). „Driver-Model of Lane Change Maneuvers“. In: *Proceedings of the 7th World Congress on Intelligent Systems*. url: https://trid.trb.org/view/733063.

Eichhorn, Andreas von, Werling, Moritz, Zahn, Peter, Schramm, Dieter (2013). „Maneuver Prediction at Intersections using Cost-to-go Gradients“. In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. doi: 10.1109/itsc.2013.6728219.

Endsley, Mica R. (1988). „Situation Awareness Global Assessment Technique (SAGAT)“. In: *Proceedings of the National Aerospace and Electronics Conference (NAECON)*. doi: 10.1109/naecon.1988.195097.

Endsley, Mica R. (1995). „Toward a Theory of Situation Awareness in Dynamic Systems“. In: *Human Factors* 37.1. doi: 10.4324/9781315087924-3.

Evans, Leonard, Rothery, Richard (1974). „Detection of the Sign of Relative Motion when Following a Vehicle“. In: *Human Factors* 16.2. doi: 10.1177/001872087401600208.

Fawcett, Tom (2005). „An Introduction to ROC Analysis“. In: *Pattern Recognition Letters* 27.8. doi: 10.1016/j.patrec.2005.10.010.

Fayyad, Usama M., Irani, Keki B. (1993). „Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning“. In: *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. url: https://www.ijcai.org/Proceedings/93-2/Papers/022.pdf (visited on 11/18/2018).

Federal Highway Administration (FHWA), U.S. Department of Transportation (2017). *Next Generation Simulation (NGSIM)*. url: https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm (visited on 11/18/2018).

Festner, Michael, Eicher, Alexandra, Schramm, Dieter (2016). „Beeinflussung der Komfort- und Sicherheitswahrnehmung beim hochautomatisierten Fahren durch fahrfremde Tätigkeiten und Spurwechseldynamik“. In: *11. Uni-DAS e.V. Workshop Fahrerassistenzsysteme und automatisertes Fahren*. url: https://www.uni-das.de/images/pdf/veroeffentlichungen/2017/07.pdf (visited on 11/18/2018).

Firl, Jonas, Tran, Quan (2011). „Probabilistic Maneuver Prediction in Traffic Scenarios“. In: *European Conference on Mobile Robots (ECMR)*.

Gasser, Tom M., Arzt, Clemens, Ayoubi, Mihiar, Bartels, Arne, Eier, Jana, Flemisch, Frank, Häcker, Dirk, Hesse, Tobias, Huber, Werner, Lotz, Christine, Maurer, Markus, Ruth-Schumacher, Simone, Schwarz, Jürgen, Vogt, Wolfgang (2012). *Rechtsfolgen zunehmender Fahrzeugautomatisierung*. Research rep. Bundesanstalt für Straßenwesen (BASt).

Gerdes, Arati (2006). „Automatic Maneuver Recognition in the Automobile: the Fusion of Uncertain Sensor Values using Bayesian Models“. In: *Proceedings of the 3rd International Workshop on Intelligent Transportation (WIT)*.

Ghahramani, Zoubin (2016). *A History of Bayesian Neural Networks*. url: http://bayesiandeeplearning.org/2016/slides/nips16bayesdeep.pdf (visited on 02/14/2018).

Gindele, Tobias, Brechtel, Sebastian, Dillmann, Rüdiger (2010). „A Probabilistic Model for Estimating Driver Behaviors and Vehicle Trajectories in Traffic Environments“. In: *IEEE 13th Annual Conference on Intelligent Transportation Systems*. doi: 10.1109/itsc.2010.5625262.

Gindele, Tobias, Brechtel, Sebastian, Dillmann, Rüdiger (2015). „Learning Driver Behavior Models from Traffic Observations for Decision Making and Planning“. In: *IEEE Intelligent Transportation Systems Magazine* 7.1. doi: 10.1109/mits.2014.2357038.

Gipps, P. G. (1980). „A Behavioural Car-Following Model for Computer Simulation“. In: *Transportation Research Part B: Methodological* 15.2. doi: 10.1016/0191-2615(81)90037-0.

Gipps, P. G. (1986). „A Model for the Structure of Lane-Changing Decisions“. In: *Transportation Research Part B: Methodological* 20.5, pp. 403–414. doi: 10.1016/0191-2615(86)90012-3.

Graf, Regine, Deusch, Hendrik, Fritzsche, Martin, Dietmayer, Klaus (2013). „A Learning Concept for Behavior Prediction in Traffic Situations“. In: *IEEE Intelligent Vehicle Symposium (IV)*. doi: 10.1109/ivs.2013.6629544.

Gutjahr, Benjamin, Pek, Christian, Gröll, Lutz, Werling, Moritz (2016). „Rechnereffiziente Trajektorienoptimierung für Fahrzeuge mittels quadratischem Programm (Efficient Trajectory Optimization for Vehicles using Quadratic Programming)“. In: *at - Automatisierungstechnik* 64.10. doi: 10.1515/auto-2016-0074.

Gwehenberger, Johann, Schwertberger, Walter, Daschner, Dieter (2006). „Wirkungspotenziale von Adaptive Cruise Control und Lane Guard System bei schweren Nutzfahrzeugen“. In: *Allianz Zentrum für Technik (AZT)* 4.10, 11.

Hayward, John C. (1972). „Near-Miss Determination through Use of a Scale of Danger“. In: *51st Annual Meeting of the Highway Research Board*.

He, Lei, Zong, Chang-fu, Wang, Chang (2012). „Driving Intention Recognition and Behaviour Prediction based on a Double-Layer Hidden Markov Model“. In: *Journal of Zhejiang University* 13.3. doi: 10.1631/jzus.c11a0195.

Heller, Uta, Letzner-Friedlein, Petra, Matuschek, Anna, Trede, Siegfried, Weber, Bernd (2015). *DAT Report 2015*. Tech. rep. Deutsche Automobil Treuhand GmbH.

Hidas, Peter (2005). „Modelling Vehicle Interactions in Microscopic Simulation of Merging and Weaving“. In: *Transportation Research Part C: Emerging Technologies* 13.1. doi: 10.1016/j.trc.2004.12.003.

Hidas, Peter, Behbahanizadeh, Kamran (1999). „Microscopic Simulation of Lane Changing under Incident Conditions“. In: *Transportation and Traffic Theory*.

Hillenbrand, Jörg (2007). „Fahrerassistenz zur Kollisionsvermeidung“. PhD thesis. Fakultät für Elektrotechnik und Informationstechnik der Universität Fridericiana Karlsruhe.

Hinton, Geoffrey, Deng, Li, Yu, Dong, Dahl, George, Mohamed, Abdel-rahman, Jaitly, Navdeep, Senrio, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Sainath, Tara, Kingsbury, Brian (2012). „Deep Neural Networks for Acoustic Modeling in Speech Recognition“. In: *IEEE Signal Processing Magazine* 29.6. doi: 10.1109/MSP.2012.2205597.

Hirsenkorn, Nils, Kolsi, Hakim, Selmi, Moez, Schärmann, Alexander, Hanke, Timo, Rauch, Andreas, Rasshofer, Ralph, Biebl, Erwin (2017). „Learning Sensor Models for Virtual Test and Development“. In: *11. Workshop Fahrerassistenzsysteme und automatisiertes Fahren.*

Hochstaedter, Almut, Ehmanns, Dirk, Neunzig, Dirk (1999). „PELOPS as a Tool for Development and Configuration“. In: *EUROmotor Telematic.*

Hornik, Kurt, Stinchcombe, Maxwell, White, Halbert (1989). „Multilayer Feedforward Networks are Universal Approximators“. In: *Neural Networks* 2.5. doi: 10.1016/0893-6080(89)90020-8.

Iglinski, Hubert, Babiak, Maciej (2017). „Analysis of the Potential of Autonomous Vehicles in Reducing the Emissions of Greenhouse Gases in Road Transport“. In: *Procedia Engineering* 192. doi: 10.1016/j.proeng.2017.06.061.

International Organization for Standardization (ISO) (2002). *15622: Transport information and control systems - Adaptive Cruise Control systems - Performance requirements and test procedures.*

International Organization for Standardization (ISO) (2007). *17361: Intelligent transport systems - Lane departure warning systems - Performance requirements and test procedures.*

International Organization for Standardization (ISO) (2008). *17387: Intelligent transport systems - Lane change decision aid systems (LCDAS) - Performance requirements and test procedures.*

International Organization for Standardization (ISO) (2009). *22179: Intelligent transport systems - Full speed range adaptive cruise control (FSRA) systems - Performance requirements and test procedures.*

International Organization for Standardization (ISO) (2010). *12100 - Safety of machinery - General principles for design - Risk assessment and risk reduction.*

International Organization for Standardization (ISO) (2013a). *15623: Intelligent transport systems - Forward vehicle collision warning systems - Performance requirements and test procedures.*

International Organization for Standardization (ISO) (2013b). *22839: Intelligent transport systems - Forward vehicle collision mitigation systems - Operation, performance, and verification requirements.*

International Organization for Standardization (ISO) (2014). *11270: Intelligent transport systems - Lane keeping assistance systems (LKAS) - Performance requirements and test procedures.*

Japkowicz, Nathalie (2000). „The Class Imbalance Problem: Significance and Strategies". In: *Proceedings of the International Conference on Artificial Intelligence (ICAI).*

Johansson, Roger (2009). „Vision Zero - Implementing a policy for traffic safety". In: *Safety Science* 47.6. doi: 10.1016/j.ssci.2008.10.023.

Jordan, Justus, Ruhhammer, Christian, Klöden, Horst, Kleinsteuber, Martin (2015). „Learning Driving Scene Prediction from Environmental Perception of Vehicle Fleet Data". In: *IEEE 18th International Conference on Intelligent Transportation Systems*. doi: 10.1109/itsc.2015.96.

Kammel, Sören, Ziegler, Julius, Pitzer, Benjamin, Werling, Moritz, Gindele, Tobias, Jagzent, Daniel, Schröder, Joachim, Thuy, Michael, Goebl, Matthias, Hundelshausen, Felix von, Pink, Oliver, Frese, Christian, Stiller, Christoph (2008). „Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge". In: *Journal of Field Robotics* 25.9. doi: 10.1002/rob.20252.

Kämpchen, Nico, Schiele, Bruno, Dietmayer, Klaus (2009). „Situation Assessment of an Autonomous Emergency Brake for Arbitrary Vehicle-to-Vehicle Collision Scenarios". In: *IEEE Transactions on Intelligent Transportation Systems* 10.4. doi: 10.1109/tits.2009.2026452.

Kasper, Dietmar, Weidl, Galia, Dang, Thao, Breuel, Gabi, Tamke, Andreas, Wedel, Andreas (2011). „Object-Oriented Bayesian Networks for Detection of Lane Change Maneuvers". In: *IEEE Intelligent Transportation Systems Magazine* 4.3. doi: 10.1109/mits.2012.2203229.

Kesting, Arne, Treiber, Martin, Helbing, Dirk (2007). „General Lane-Changing Model MOBIL for Car-Following Models". In: *Transportation Research Record* 1999.1. doi: 10.3141/1999-10.

Kjaerulff, Uffe B. (1990). *Triangulation of Graphs - Algorithms Giving Small Total State Space*. University of Aalborg, Institute for Electronic Systems, Department of Mathematics and Computer Science.

Knapp, Andreas, Neumann, Markus, Brockmann, Martin, Walz, Rainer, Winkle, T. (2009). *Code of Practice for the Design and Evaluation of ADAS*. Version 5.0. Preventive and Active Safety Applications (PReVENT), Integrated Project RESPONSE 3.

Koller, Daphne, Friedman, Nir (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press. isbn: 9780262013192.

Kopf, Matthias (1994). „Ein Beitrag zur modellbasierten, adaptive Fahrerunterstützung für das Fahren auf deutschen Autobahnen“. In: *VDI Fortschrittsberichte*. Vol. 203. Verkehrstechnik/Fahrzeugtechnik. isbn: 318320312X.

Krizhevsky, Alex, Sutskever, Ilya, Hinton, Geoffrey (2012). „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Communications of the ACM* 60.6. doi: 10.1145/3065386.

Kuchar, James K., Yang, Lee C. (2000). „A Review of Conflict Detection and Resolution Modeling Methods“. In: *IEEE Transactions on Intelligent Transportation Systems* 1.4. doi: 10.1109/6979.898217.

Kuderer, Markus, Gulati, Shilpa, Burgard, Wolfram (2015). „Learning Driving Styles for Autonomous Vehicles from Demonstration“. In: *Robotics and Automation (ICRA)*. doi: 10.1109/icra.2015.7139555.

Kuderer, Markus, Kretzschmar, Henrik, Sprunk, Christoph, Burgard, Wolfram (2012). „Feature-Based Prediction of Trajectories for Socially Compliant Navigation“. In: *Robotics: Science and Systems*. doi: 10.15607/rss.2012.viii.025.

Kuge, Nobuyuki, Yamamura, Tomohiro, Shimoyama, Osamu, Liu, Andrew (1998). „A Driver Behavior Recognition Method based on a Driver Model Framework“. In: *SAE Technical Paper Series*. SAE International. doi: 10.4271/2000-01-0349.

Kumar, P., Perrollaz, Mathias, Lefevre, Stephanie, Laugier, Christian (2013). „Learning-based Approach for Online Lane Change Intention Prediction“. In: *IEEE Intelligent Vehicles Symposium (IV)*. doi: 10.1109/ivs.2013.6629564.

Kumar, Vinodh E., Jerome Jovinthaand Srikanth, K. (2014). „Algebraic Approach for Selecting the Weighting Matrices of Linear Quadratic Regulator“. In: *International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*. doi: 10.1109/icgccee.2014.6922382.

Lauritzen, Steffen L., Spiegelhalter, D. J. (1988). „Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems“. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 50.2. doi: 10.1111/j.2517-6161.1988.tb01721.x.

Lawitzky, Andreas, Althoff, Daniel, Passenberg, Christoph F., Tanzmeister, Georg, Wollherr, D., Buss, Martin (2013). „Interactive Scene Prediction for Automotive Applications“. In: *IEEE Intelligent Vehicles Symposium (IV)*. doi: 10.1109/ivs.2013.6629601.

Lefevre, Stephanie, Vasquez, Dizan, Laugier, Christian (2014). „A Survey on Motion Prediction and Risk Assessment for Intelligent Vehicles“. In: *ROBOMECH Journal* 1.1. doi: 10.1186/s40648-014-0001-z.

Lenz, Barbara, Fraedrich, Eva (2016). „New Mobility Concepts and Autonomous Driving: The Potential for Change“. In: *Autonomous Driving*. doi: 10.1007/978-3-662-48847-8_9.

Lerner, Uri (2002). „Hybrid Bayesian Networks for Reasoning About Complex Systems“. PhD thesis. Stanford University, Department of Computer Science.

Lerner, Uri, Segal, Eran, Koller, Daphne (2001). „Exact Inference in Networks with Discrete Children of Continuous Parents“. In: *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI2001)*. url: http://arxiv.org/abs/1301.2289.

Liebner, Martin, Ruhhammer, Christian, Klanner, Felix, Stiller, Christoph (2013). „Generic Driver Intent Inference Based on Parametric Models“. In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. doi: 10.1109/itsc.2013.6728244.

Lin, Chiu-Feng, Ulsoy, A. Galip, LeBlanc, David J. (2000). „Vehicle Dynamics and External Disturbance Estimation for Vehicle Path Prediction“. In: *IEEE Transactions on Control Systems Technology* 8.3. doi: 10.1109/87.845881.

Ludmann, J. (2000). „Beeinflussung des Verkehrsablaufes auf Straßen - Analyse mit dem fahrzeugorientierten Verkehrsflusssimulationsprogramm PELOPS“. PhD thesis. Institut für Kraftfahrwesen, RTWH Aachen.

Lunze, Jan (2014). *Regelungstechnik 2*. Springer Vieweg. doi: 10.1007/978-3-662-52676-7.

Maas, Niko (2017). „Konzeptionierung, Auslegung und Umsetzung von Assistenzfunktionen für die übergabe der Fahraufgabe aus hochautomatisiertem Fahrbetrieb“. PhD

thesis. Universität Duisburg-Essen, Fakultät für Ingenieurwissenschaften, Abteilung Maschinenbau und Verfahrenstechnik.

Maerivoet, Sven, Moor, Bart de (2005). „Cellular Automata Models of Road Traffic“. In: *Physics Reports* 419.1. doi: 10.1016/j.physrep.2005.08.005.

Mandalia, Hiren M., Salvucci, Dario (2005). „Using Support Vector Machines for Lane-Change Detection“. In: *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting* 49.22. doi: 10.1037/e577512012-017.

Maurer, Markus, Stiller, Christoph (2005). *Fahrerassistenzsysteme mit maschineller Wahrnehmung*. Springer-Verlag. doi: 10.1007/b138667.

McCall, Joel C., Wipf, David P., Trivedi, Mohan M., Rao, Bhaskar D. (2007). „Lane Change Intent Analysis Using Robust Operators and Sparse Bayesian Learning“. In: *IEEE Transactions on Intelligent Transportation Systems* 8.3. doi: 10.1109/tits.2007.902640.

McCulloch, Warren S., Pitts, Walter (1943). „A Logical Calculus of the Ideas Immanent in Nervous Activity“. In: *Bulletin of Mathematical Biology* 5.4. doi: 10.1007/bf02478259.

Meyer-Delius, Daniel, Sturm, Jürgen, Burgard, Wolfram (2009). „Regression-Based Online Situation Recognition for Vehicular Traffic Scenarios“. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. doi: 10.1109/iros.2009.5354209.

Moller, Martin Fodslette (1993). „A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning“. In: *Neural Networks* 6.4. doi: 10.1016/s0893-6080(05)80056-5.

Moridpour, Sara, Rose, Geoff, Sarvi, Majid (2009). „Modelling the heavy Vehicle Drivers' Lane Changing Decision under heavy Traffic Conditions“. In: *Road and Transport Research* 18.4. issn: 1037-5783.

Morris, Brendan, Doshi, Anup, Trivedi, Mohan (2011). „Lane Change Intent Prediciton for Driver Assistance On Road Design and Evaluation“. In: *IEEE Intelligent Vehicles Symposium (IV)*. doi: 10.1109/ivs.2011.5940538.

Mossman, Douglas (1999). „Three-way ROCs“. In: *Medical Decision Making* 19.1. doi: 10.1177/0272989x9901900110.

Murphy, Kevin (1998). *Fitting a Conditional Linear Gaussian Distribution*. url: https://www.cs.ubc.ca/~murphyk/Papers/learncg.pdf (visited on 03/05/2018).

Murphy, Kevin (1999). „A Variational Approximation for Bayesian Networks with Discrete and Continuous Latent Variables“. In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI1999)*. URL: https://arxiv.org/abs/1301.6724.

Murphy, Kevin (2001). „The Bayes Net Toolbox for Matlab“. In: *Computing Science and Statistics* 33.2. URL: https://www.cs.ubc.ca/~murphyk/Papers/bnt.pdf (visited on 03/05/2018).

Murphy, Kevin (2012). *Machine Learning - A Probabilistic Perspective*. The MIT Press. ISBN: 978-0-262-01802-9.

Murray, R. M. (2006). *Introduction to Control Theory (CDS 110b)*. California Institute of Technology. URL: http://www.cds.caltech.edu/~murray/courses/cds110/wi06/lqr.pdf (visited on 11/18/2018).

Nagatani, Takashi (1994). „Traffic Jam and Shock Formation in Stochastic Traffic-Flow Model of a Two-Lane Roadway“. In: *Journal of the Physical Society of Japan* 63.1. DOI: 10.1143/jpsj.63.52.

Nagel, Kai, Wolf, Dietrich E., Wagner, Peter, Simon, Patrice (1998). „Two-lane Traffic Rules for Cellular Automata: A Systematic Approach“. In: *Physical Review E* 58.2. DOI: 10.1103/physreve.58.1425.

Neunzig, D., Breuer, K., Korthals, H. (2000). „PELOPS - Analyse, Simulation und Optimierung von komplexen Verkehrsabläufen“. In: *Logistik Forum Nürnberg*.

Ng, Andrew Y., Russell, Stuart J. (2000). „Algorithms for Inverse Reinforcement Learning“. In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*. URL: https://ai.stanford.edu/~ang/papers/icml00-irl.pdf (visited on 11/18/2018).

Oliver, Nuria, Pentland, Alex P. (2000). „Graphical Models for Driver Behavior Recognition in a SmartCar“. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/ivs.2000.898310.

Pearl, Judea (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Elsevier. DOI: 10.1016/c2009-0-27609-4.

Pentland, Alex P., Liu, Andrew (1999). „Modeling and Prediction of Human Behavior“. In: *Neural Computation* 11.1. DOI: 10.1162/089976699300016890.

Petrich, Dominik, Dang, Thao, Kasper, Dietmar, Breuel, Gabi, Stiller, Christoph (2013). „Map-based Long Term Motion Prediction for Vehicles in Traf-

fic Environments“. In: *IEEE Intelligent Transportation Systems Conference*. DOI: 10.1109/itsc.2013.6728549.

Platho, Matthias, Gross, Horst-Michael, Eggert, Julian (2013). „Learning Driving Situations and Behavior Models from Data“. In: *Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systes*. DOI: 10.1109/itsc.2013.6728245.

Powers, David M. W. (2011). „Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation“. In: *Journal of Machine Learning Technologies* 2.1. DOI: 10.9735/2229-3981.

Quigley, Morgan, Gerkey, Brian, Conley Ken andn Faust, Josh, Foote, Tully, Leibs, Jeremy, Berger, Eric, Wheeler, Rob, Ng, Andrew (2009). „ROS: an open-source Robot Operating System“. In: *ICRA Workshop on Open Source Software*. URL: http://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf (visited on 11/18/2018).

Rahman, Mizanur, Chowdhury, Mashrur, Xie, Yuanchang, He, Yiming (2013). „Review of Microscopic Lane-Changing Models and Future Research Opportunities“. In: *IEEE Transactions on Intelligent Transportation Systems* 14.4. DOI: 10.1109/tits.2013.2272074.

Rickert, M., Nagel, K., Schreckenberg, M., Latour, A. (1995). „Two Lane Traffic Simulations using Cellular Automata“. In: *Physica A: Statistical Mechanics and its Applications* 231.4. DOI: 10.1016/0378-4371(95)00442-4.

Rolnick, Arnon, Lubow, R. E. (1991). „Why is the Driver Rarely Motion Sick? The Role of Controllability in Motion Sickness“. In: *Ergonomics* 34.7. DOI: 10.1080/00140139108964831.

Salvucci, Dario D., Mandalia, Hiren M., Kuge, Nobuyuki, Yamamura, Tomohiro (2007). „Lane-Change Detection Using A Computational Driver Model“. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 49.3. DOI: 10.1518/001872007x200157.

Schlechtriemen, Julian, Wabersich, Kim Peter, Kuhnert, Klaus-Dieter (2016). „Wiggling Through Complex Traffic: Planning Trajectories Constrained by Predictions“. In: *IEEE Intelligent Vehicles Symposium (IV)*. DOI: 10.1109/ivs.2016.7535557.

Schlechtriemen, Julian, Wedel, Andreas, Breuel, Gabi, Kuhnert, Klaus-Dieter (2014). „A Probabilistic Long Term Prediction Approach for Highway Scenarios“.

In: *IEEE 17th International Conference on Intelligent Transportation Systems*. DOI: 10.1109/itsc.2014.6957776.

Schlechtriemen, Julian, Wedel, Andreas, Hillenbrand, Jörg, Breuel, Gabi, Kuhnert, Klaus-Dieter (2014). „A Lane Change Detection Approach using Feature Ranking with Maximized Predictive Power“. In: *IEEE Intelligent Vehicles Symposium (IV)*. DOI: 10.1109/ivs.2014.6856491.

Schlechtriemen, Julian, Wirthmueller, Florian, Wedel, Andreas, Breuel, Gabi, Kuhnert, Klaus-Dieter (2015). „When Will it Change the Lane? A Probabilistic Regression Approach for Rarely Occurring Events“. In: *IEEE Intelligent Vehicles Symposium (IV)*. DOI: 10.1109/ivs.2015.7225907.

Schreier, Matthias, Willert, Volker, Adamy, Jürgen (2014). „Bayesian, Maneuver-Based, Long-Term Trajectory Prediction and Criticality Assessment for Driver Assistance Systems“. In: *17th IEEE International Conference on Intelligent Transportation Systems (ITSC)*. DOI: 10.1109/itsc.2014.6957713.

Schubert, Robin, Richter, Eric, Wanielik, Gerd (2008). „Comparison and Evaluation of Advanced Motion Models for Vehicle Tracking“. In: *11th International Conference on Information Fusion*.

Singh, Karandeep, Li, Baibing (2012). „Estimation of Traffic Densities forMultilane Roadways Using a Markov Model Approach“. In: *IEEE Transactions on Industrial Electronics* 59.11. DOI: 10.1109/tie.2011.2180271.

Society of Automotive Engineers (SAE) (Jan. 2015). *Honda introduces 'industry first' intelligent adaptive cruise control*. URL: https://www.sae.org/news/2015/01/honda-introduces-industry-first-intelligent-adaptive-cruise-control (visited on 11/18/2018).

Society of Automotive Engineers (SAE) (Sept. 2016). *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. SAE Standard. URL: http://standards.sae.org/j3016_201609/ (visited on 11/18/2018).

Statistisches Bundesamt (Destatis) (2015). *Genesis-Online*. URL: https://www-genesis.destatis.de (visited on 11/18/2018).

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, Fergus, Rob (2014). „Intriguing Properties of Neural Networks“. In: arXiv: 1312.6199v4 [cs.CV]. URL: https://arxiv.org/abs/1312.6199.

Toledo, Tomer, Katz, Romina (2009). „State Dependence in Lane-Changing Models“. In: *Transportation Research Record: Journal of the Transportation Research Board* 2124.1. doi: 10.3141/2124-08.

Toledo, Tomer, Koutsopoulos, Haris N., Ben-Akiva, Moshe E. (2007). „Integrated Driving Behavior Modeling“. In: *Transportation Research Part C: Emerging Technologies* 15.2. doi: 10.1016/j.trc.2007.02.002.

Tomar, Ranjeet Singh, Verma, Shekhar (2012). „Safety of Lane Change Maneuver through A Priori Prediction of Trajectory Using Neural Networks“. In: *Network Protocols and Algorithms* 4.1. doi: 10.5296/npa.v4i1.1240.

Torkolla, Kari, Venkatesan, Srihari, Liu, Huan (2005). „Sensor Sequence Modeling for Driving“. In: *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference, FLAIRS - Recent Advances in Artifical Intelligence*.

Treiber, Martin, Hennecke, Ansgar, Helbing, Dirk (2000). „Congested Traffic States in Empirical Observations and Microscopic Simulation“. In: *Physical Review E* 62.2. doi: 10.1103/physreve.62.1805.

Unfallforschung der Versicherer (UDV) (2009). *Demonstration von Notbrems- und Auffahrwarnsystemen am Pkw*. Tech. rep. Gesamtverband der Deutschen Versicherungswirtschaft e.V. (GDV). url: https://m.udv.de/system/files_force/tx_udvpublications/UDV_Jahresbericht__Web_100604_0.pdf (visited on 11/18/2018).

Verband der Automobilindustrie e.V. (VDA) (2015a). *Automation - From Driver Assistance Systems to Automated Driving*. url: https://www.vda.de/dam/vda/publications/2015/automation.pdf (visited on 11/18/2018).

Verband der Automobilindustrie e.V. (VDA) (2015b). *Automatisierung - Von Fahrerassistenzsystemen zum automatisierten Fahren*. url: https://www.vda.de/dam/vda/publications/2015/automatisierung.pdf (visited on 11/18/2018).

Verband Deutscher Verkehrsunternehmen e. V. (VDV) (2015). *Zukunftsszenarien autonomer Fahrzeuge Chancen und Risiken für Verkehrsunternehmen*. url: https://www.vdv.de/position-autonome-fahrzeuge.pdfx (visited on 11/18/2018).

Vollarth, Mark, Briest, Susanne, Schiessl, Caroline, Drewes, Jörn, Becker, Uwe (2006). *Ableitung von Anforderungen an Fahrerassistenzsysteme aus Sicht der Verkehrssicherheit*. Tech. rep. Bundesanstalt für Straßenwesen (BASt).

Wagner, Peter, Nagel, Kai, Wolf, Dietrich E. (1997). „Realistic Multi-Lane Traffic Rules for Cellular Automata“. In: *Physica A: Statistical Mechanics and its Applications*.

Wallentowitz, H., Neunzig, D. (1999). „Assessment of new Vehicle and Traffic Technologies with the Simulation System PELOPS“. In: *Traffic and Mobility: Simulation - Economics - Environment*.

Waymo (2017). *On the Road to Fully Self-Driving*. URL: https://storage.googleapis.com/sdc-prod/v1/safety-report/waymo-safety-report-2017-10.pdf (visited on 11/18/2018).

Weidl, Galia, Madsen, Anders L. (2016). „Situation Awareness and Early Recognition of Traffic Maneuvers“. In: *9th EUROSIM Congress in Modelling and Simulation*.

Werbos, Paul (1974). „Beyond Regression: New Tools for Predicting and Analysis in the Behavioral Sciences“. PhD thesis. Harvard University, Cambridge, Massachusetts.

Werling, Moritz (2015). *Verhaltensgenerierung für Fahrzeuge*. Lecture notes. URL: https://www.mrt.kit.edu/lehre_SS_Verhaltensgenerierung_Fahrzeuge.php (visited on 11/18/2018).

Wiedemann, Rainer (1974). „Simulation des Straßenverkehrsflusses“. Habilitation. Universität Karlsruhe.

Wiest, Jürgen, Höffken, Matthias, Kressel, Ulrich, Dietmayer, Klaus (2012). „Probabilistic Trajectory Prediction with Gaussian Mixture Models“. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/ivs.2012.6232277.

Winkle, Thomas (2016). „Safety Benefits of Automated Vehicles: Extended Findings from Accident Research for Development, Validation and Testing“. In: *Autonomous Driving*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-662-48847-8_17.

Winner, Hermann, Hakuli, Stephan, Lotz, Felix, Singer, Christina (2016). *Handbook of Driver Assistance Systems*. Springer. ISBN: 9783319123523.

Winner, Hermann, Hakuli, Stephan, Wolf, Gabriele (2012). *Handbuch Fahrerassistenzsysteme*. 2nd ed. ISBN: 9783834886194.

Worrall, R. D., Bullen, A. G., Gur, Y. (1970). „An Elementary Stochastic Model of Lane-Changing on a Multilane Highway“. In: *Highway Research Record*.

Wu, Jianping, Brackstone, Mark, McDonald, Mike (2000). „Fuzzy Sets and Systems for a Motorway Microscopic Simulation Model“. In: *Fuzzy Sets and Systems* 116.1. DOI: 10.1016/s0165-0114(99)00038-x.

Yang, Qi, Koutsopoulos, Haris N. (1996). „A Microscopic Traffic Simulator for Evaluation of Dynamic Traffic Management Systems“. In: *Transportation Research Part C: Emerging Technologies* 4.3. doi: 10.1016/s0968-090x(96)00006-x.

Zhang, Nevin Lianwen, Poole, David (1994). „A Simple Approach to Bayesian Network Computations“. In: *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*.

Zheng, Zuduo (2013). „Recent Developments and Research Needs in Modeling Lane Changing“. In: *Transportation Research Part B: Methodological* 60. doi: 10.1016/j.trb.2013.11.009.

Ziebart, Brian, Maas, Andrew, Bagnell, J. Andrew, Dey, Anind K. (2008). „Maximum Entropy Inverse Reinforcement Learning“. In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*.

Own Publications

Rehder, Tobias, Georgiev, Zdravko, Louis, Lawrence, Schramm, Dieter (2015). „Effektive Nutzung von hochdimensionalen kontinuierlichen Umfelddaten zur Prädiktion von Fahrverhalten mit Bayesschen Netzen“. In: *Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel (AAET)*. ITS Niedersachsen. isbn: 9783937655345.

Rehder, Tobias, König, Alexander, Göhl, Michel, Louis, Lawrence, Schramm, Dieter (2019). „Lane Change Intention Awareness for Assisted and Automated Driving on Highways“. In: *IEEE Transactions on Intelligent Vehicles (Early Access)*, pp. 1–12. doi: 10.1109/TIV.2019.2904386.

Rehder, Tobias, Maas, Niko, Louis, Lawrence, Schramm, Dieter (2015). „Merkmalselektion zur Prädiktion von motivationsbasiertem Fahrverhalten“. In: *7. Wissenschaftsforum Mobilität: Nationale und internationale Trends in der Mobilität*. Springer Fachmedien Wiesbaden. doi: 10.1007/978-3-658-14563-7_13.

Rehder, Tobias, Münst, Wolfgang, Louis, Lawrence, Schramm, Dieter (2016a). „Influence of different Ground Truth Hypotheses on the Quality of Bayesian Networks for Maneuver Detection and Prediction of Driving Behavior“. In: *Advanced Vehicle Control (AVEC)*. Crc Press. doi: 10.1201/9781315265285-49.

Rehder, Tobias, Münst, Wolfgang, Louis, Lawrence, Schramm, Dieter (2016b). „Learning Lane Change Intentions through Lane Contentedness Estimation from Demonstrated Driving“. In: *IEEE Conference on Intelligent Transportation Systems (ITSC)*. doi: 10.1109/itsc.2016.7795661.

Co-authored Publications

König, Alexander, Rehder, Tobias, Hohmann, Sören (2017). „Exact Inference and Learning in Hybrid Bayesian Networks for Lane Change Intention Classification“. In: *IEEE Intelligent Vehicles Symposium (IV)*. DOI: 10.1109/ivs.2017.7995927.

Maas, Niko, Rehder, Tobias, Louis, Lawrence, Schramm, Dieter (2014). „Empirische Evaluation von Prädiktionsmethoden am Beispiel der Vorhersage eines Spurwechsels aufgrund der Verkehrssituation“. In: *6. Wissenschaftsforum Mobilität: Decisions on the Path to Future Mobility*. Springer Fachmedien Wiesbaden. DOI: 10.1007/978-3-658-09577-2_13.

Münst, Wolfgang, Rehder, Tobias, Bögner, Fabian, Louis, Lawrence, Icking, Christian (2017). „Decision Frameworks for using Uncertain Predictions for Cut-In Detections in (Semi-) Automated Driving“. In: *IEEE Vehicular Technology Conference (VTC)*. DOI: 10.1109/vtcspring.2017.8108279.

Münst, Wolfgang, Rehder, Tobias, Louis, Lawrence, Icking, Christian (2016). „A Novel Method for Providing Situational Awareness to Longitudinal Controllers in (Semi-) Automated Vehicles“. In: *IEEE Intelligent Transportation Systems Conference (ITSC)*. DOI: 10.1109/itsc.2016.7795675.

Student Projects

Georgiev, Zdravko (2014). „Evaluation of Features for Efficient Supervised Classification“. MA thesis. Fakultät für Informatik, Technische Univerität München.

Göhl, Michel (2016). „Fahrverhaltensprädiktion auf Basis individuell geplanter Trajektorienhypothesen für unterschiedliche Fahrverhaltenstypen“. MA thesis. Karlsruher Institut für Technologie.

Patent Applications

Gresser, Klaus, Louis, Lawrence, Han, Jiawei, Rehder, Tobias (July 8, 2014). „Fahrerassistenz-Teilsystem für ein Kraftfahrzeug zum Ermitteln eines Steuerparameters für ein Fahrerassistenzsystem“. German pat. 102014213259.

König, Alexander, Rehder, Tobias (Dec. 6, 2016). „Verfahren zur Prognose der Bewegung eines benachbarten Verkehrsteilnehmers“. German pat. 102017200180.

Louis, Lawrence, Münst, Wolfgang, Rehder, Tobias (May 7, 2015a). „Verfahren und Vorrichtung zur Reduzierung des Kollisionsrisikos eines Fahrzeugs“. German pat. 102015208530.

Louis, Lawrence, Münst, Wolfgang, Rehder, Tobias (June 11, 2015b). „Verfahren zur frühzeitigen Erkennung von potentiellen Gefahren im Straßenverkehr“. German pat. 102015210672.

Al-Nuaimi, Anas, Rehder, Tobias, Khakhutskyy, Valeriy, Martinek, Philipp, Sivalingam, Udhayaraj (Sept. 5, 2018). „Verfahren zum Bestimmen einer Spurwechselangabe eines Fahrzeugs, ein computerlesbares Speichermedium und ein Fahrzeug“. German pat. 102018215055.

Rehder, Tobias, Göhl, Michel (Jan. 24, 2017). „Objektklassenspezifische und fahrerindividuelle Fahrverhaltensprädiktion mittels Inversem Reinforcement Learning“. German pat. 102017212629.