Probabilistic Freespace Prediction in Structured Traffic Environments for Trajectory Planning

DISSERTATION
zur Erlangung des Grades eines Doktors
der Ingenieurwissenschaften

vorgelegt von
Dipl.-Inform. Julian David Schlechtriemen

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen
Siegen 2020

Betreuer und erster Gutachter
Prof. Dr.-Ing. Klaus-Dieter Kuhnert
Universität Siegen

Zweiter Gutachter
Prof. Dr. Manfred Reichert
Universität Ulm

Tag der mündlichen Prüfung
23.02.2021

Zusammenfassung

In der vorliegenden Arbeit wird ein methodisches Framework entwickelt, welches einem hochautomatisierten Fahrzeug auf Autobahnen im Fehlerfall die Ausführung eines sicheren Rückfallverhaltens ermöglicht. Das vorgestellte Framework kann dabei Fehlerfälle der Umgebungserfassung, der Situationsanalyse sowie eine ausbleibende Fahrerübernahme behandeln. Der Schwerpunkt dieser Arbeit liegt jedoch nicht auf der Darstellung eines Gesamtsystems. Vielmehr wird auf die Entwicklung der fehlenden funktionalen Bausteine fokussiert, die zur Umsetzung eines solchen Rückfallsystems benötigt werden.

Im Verlauf der Arbeit wird nach einem Überblick über den inhaltlichen Kontext und einer Einführung in die benötigten theoretischen Grundlagen eine Systemarchitektur vorgestellt, in der die in der Arbeit entwickelten Methoden eingesetzt werden können. Die Methoden umfassen dabei Verfahren zur Manövererkennung, Bewegungsprädiktion und Trajektorienplanung. Wichtige Ziele der Systemarchitektur liegen in der Wiederverwendbarkeit und Erweiterbarkeit der entwickelten Methoden für weitere Anwendungsfälle.

Der Hauptteil der Arbeit beschäftigt sich mit der Umsetzung der in der Systemarchitektur identifizierten Funktionsbausteine, die zur Planung einer Rückfalltrajektorie benötigt werden. In einem ersten Schritt werden dazu Klassifikationsverfahren zur Erkennung und Prognose von Fahrmanövern entwickelt. Das Klassifikationsproblem beschränkt sich hierbei auf die Erkennung von Spurwechseln und Spurfolgeverhalten. Da das zukünftige Verhalten von Fahrern nicht eindeutig bestimmbar ist, werden hierbei nur Verfahren betrachtet, die eine probabilistische Bewertung der ausgeführten Manöver ermöglichen. Den aktuellen Stand der Technik erweiternd, werden hierbei der Situationskontext und mögliche Einflussfaktoren systematisch analysiert. In zwei Experimenten werden die ausgewählten Verfahren vorgestellt und evaluiert.

Für die Prognose zukünftiger Aufenthaltsorte der Fahrzeuge wird eine Methode vorgestellt, welche den physikalischen Bewegungsstatus und die Ergebnisse der Manöverklassifikation berücksichtigt. Die Methode liefert hierbei nicht nur Informationen über den wahrscheinlich prognostizierten Ort, sondern eine zeitabhängige Aufenthaltswahrscheinlichkeit der Verkehrsteilnehmer. Zur besseren Einordung der Güte der Ergebnisse erfolgt ein Vergleich mit bestehenden klassischen Verfahren zur Bewegungsprognose.

Zur Bestimmung einer Rückfalltrajektorie wird unter Nutzung der Bewegungsprognosen des umgebenden Verkehrs wird abschließend ein neues Trajektorienplanungsverfahren vorgestellt. Hierzu wird eine Repräsentation des dynamischen Freiraums vorgestellt, die ein effizientes Sampling einer Schar von Trajektorien ermöglicht. Aus dieser Trajektorienschar wird anschließend anhand eines Optimierungskriteriums die am besten passende ausgewählt. Zur Erzeugung der Trajektorien wird ein flachheitsbasierter Ansatz eingeführt, der Trajektorien durch stückweise Polynome modelliert. Ohne Durchführung eines weiteren Optimierungsschritts garantiert die vorgestellte Methode, dass die Gesamttrajektorie unter Berücksichtigung der Randbedingungen ruckminimal ist.

Zusammenfassend werden in der vorliegenden Arbeit neue Verfahren vorgestellt, die wesentliche Verbesserungen in den Bereichen Manöverklassifikation und Bewegungsprädiktion realisieren. Die entwickelte Methode zur Trajektorienplanung erlaubt die Berücksichtigung von Prädiktionsdaten und trägt dazu bei, dass vorgestellte Problem der Erzeugung einer Rückfalltrajektorie zu lösen. Die Wirksamkeit der in der Arbeit vorgestellten Methoden wurde neben den theoretischen Untersuchungen auch im realen Fahrbetrieb nachgewiesen.

Abstract

This thesis addresses the problem of providing a methodical framework for highly automated driving on highways, which allows an automated vehicle to execute a safe fallback behavior in case of a system failure. The presented framework is able to handle failures of the environment perception, situation analysis and a missing take over by a driver. The thesis is however not focusing on the implementation of the whole system. Instead, it focuses on the missing functional components which are needed for the implementation of such kind of fallback system.

Within the thesis in a first step an overview on the context of the problem and an introduction into the theoretical foundations is presented. To provide the reader with a holistic understanding, a system architecture is presented, in which the methods developed in this thesis can be applied. The methods include algorithms for maneuver recognition, position prediction and trajectory planning. Important goals of the architecture are reusability and extendability of the developed methods for further use cases.

The main part of the thesis deals with the implementation of the functions which were identified in the system architecture and are needed for planning fallback trajectories. To do so, in a first step classification methods are developed for the recognition of driving maneuvers. The classification problem in the context is limited to the recognition of lane changing and lane following behavior. Because future behavior of human drivers is not deterministic, the set of methods was restricted to algorithms which provide a probabilistic output. Extending the state of the art, the situation context and possible influencing factors are analyzed systematically. The presented approach is evaluated in two experiments.

To predict future positions, a method is introduced which on the one hand considers the physical state of a vehicle and on the other hand also takes account of the situation context. This is realized by using the results of the method developed for maneuver recognition. The presented algorithm does not only deliver information about the future position, but a time dependent probability distribution of future whereabouts. To provide an insight in the prediction performance, a comparison to existing classic methods of position prediction is presented.

In order to plan a fallback behavior, a new method for trajectory planning is introduced, which allows incorporating future positions of other vehicles. To do so, a representation of the dynamic free space is introduced, which allows to sample a family of trajectories efficiently. From this family the best trajectory according to a chosen optimization criterion is selected. For the generation of trajectories a flatness-based planning approach is presented, which models the trajectories using composite polynomials. Without the execution of an additional optimization step, the method guarantees, that the whole trajectory is jerk minimal given the defined constraints.

Summarizing, this thesis introduces new methods, which implement meaningful improvements in the field of maneuver classification and position prediction. The method presented for planning trajectories allows incorporating prediction informa-

tion and is an enabler to solve the problem of planning fallback trajectories. Besides the theoretical investigations presented in thesis, the methods have proven to be applicable in prototypical vehicles in real traffic.

Contents

List of Figures

List of Tables

List of Abbreviations

ACC Adaptive Cruise Control

ADAS Advanced Driver Assistance Systems

AIC Akaike Information Criterion

ASIL Automotive Safety Integrity Level

AUC Area under the curve

BASt Bundesanstalt für Straßenwesen

BIC Bayesian Information Criterion

CA constant acceleration

CAD computer-aided design

CBR Case Based Reasoning

CDF Cummulative Density Function

CFS Correlation based Feature Selector

CMS Collision Mitigation System

CPU Central Processing Unit

CRP Chinese Restaurant Process

CV constant velocity

DARPA Defense Advanced Research Projects Agency

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DDT Dynamic Driving Task

DP Dirichlet Process

DPGMM Dirichlet Process Gaussian Mixture

EE Electrical / Electronic

ECU Electronic Control Unit

EM Expectation Maximization

FFNN Feed Forward Neural Network

Flw Lane Following

FN false negatives

FNR false negative rate

FP false positives

FPR false positive rate

GNB Gaussian Naive Bayes

GMM Gaussian Mixture Model

GMR Gaussian Mixture Regression

GPU Graphics Processing Unit

HMI Human Machine Interface

HMM Hidden Markov Model

IDM Intelligent Driver Model

ISO International Organization for Standardization

KDE Kernel Density Estimator

LcL Lane Change Left

LcR Lane Change Right

LDA Linear Discriminant Analysis

LOOB Leave One Out Bootstrapping

LOOC Leave One Out Cross-validation

LSTM Long Short Term Memory

MAD Mean Absolute Deviation

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

MLE Maximum Likelihood Estimate

MLP Multi Layer Perceptron

MSE Mean Squared Error

NGSIM Next Generation Simulation [1]

NHTSA National Highway Traffic Safety Administration

ODD Operational Design Domain

PCA Principal Component Analysis

PDF Probability Density Function

PPV Postive Predictive Value

RMSE Root Mean Squared Error

RBF Radial Basis Function

ReLu Rectified Linear Activation Function

RF Random Forest

RMSE Root Mean Squared Error

RNN Recurrent Neural Networks

RRT Rapidly-exploring Random Tree

ROC Receiver Operating Characteristics

SAE Society of Automotive Engineers

SVM Support Vector Machine

TN True negatives

TNR True negative rate

TP True positive rate

TPR True positive rate

TTC Time to collision

VBGMM Variational Bayesian Gaussian Mixture Model

V2X vehicle to X communication

VC Vapnik Chervonenkis

WCRP Weighted Chinese Restaurant Process

Acknowledgments

This PhD thesis would not have been possible without the help and support of many people throughout the last years. First and foremost, I would like to thank Prof. Dr. Dieter Kuhnert for supervising me and making this Dissertation possible at the University of Siegen. I also want to express my gratitude to the members of the audit committee, Prof. Dr. Manfred Reichert, Prof. Dr. Frank Gronwald and Prof. Dr. Malte Lochau.

I would also like to thank Andreas Wedel for shaping the path when starting this work. Being my supervisor in the first years of my work as PhD student, Andreas provided me with inspiring ideas and his support on so many levels. I have learned a lot about enthusiasm, and how to build the pathways and bridges from research ideas to publishable research items. I would also like to thank my friends and former colleagues of Situation Analysis Group at Böblingen and the Vehicle Intelligence Group in Sunnyvale for many fruitful and intense technical discussions. This especially holds true for my former PhD colleagues Dominik Petrich and Viktor Gomer.

I particularly like to thank Florian Wirthmüller and Kim Wabersich which not only were co-authors of my papers but also closely collaborated on actually doing the research. I also want to thank Jörg Hillenbrand who not only supported me in getting my PhD position at Daimler, but also helped me to pave the way to my assignment to Sunnyvale, CA. Furthermore, I would like to personally thank Gabi Breuel, Matthias Schell, Christoph Keller, Thao Dang, Jochen Hipp, Florian Kerber, Jens Desens and Galia Weidl for their collaboration.

Last but not least I like to thank my family and my former schoolmates and fellow students, who all contributed to what I am now in their very own way. My deepest thanks goes to Anja for always supporting me, no matter when or where. Thank you for your patience, understanding and encouragement.

There are many more people who supported me on my journey and whom I owe gratitude, but there are too many to name. Without all of you, I would not have been able to master the challenges of this PhD thesis.

Thank you!

Chapter 1

Introduction

Automated vehicles are becoming reality. There are many reasons to increase the degree of automation. On the one hand automation systems like adaptive cruise control or lane-keeping are features increasing the comfort for the driver. On the other hand fully automated robo-taxis may not only be a technical revolution, they may also fundamentally change the way how mobility in our society will look like in the next decades.

This thesis presents a concept for a non-fail-operational, but fail-safe automated driving system. The goal of this first chapter is to introduce into the problem domain which this thesis is attacking. Based on this understanding the contributions are presented. The chapter starts with an overview of how the different levels of automation are defined focusing on systems in which a human driver and the automation system have strong interactions, see Sec. 1.1. The problem of executing a handover from automated driving back to a human driver is investigated in Sec. 1.2. Motivated by this handover problem the contribution of this thesis follows in Sec. 1.3. The contribution presents a toolkit of methods enabling automated vehicles to fulfill the mandatory safety requirement of having a fallback performance available whenever needed. The chapter concludes with Sec. 1.4, where the structure of this thesis is presented in detail.

1.1 The Different Levels of Automation

On the journey to autonomous driving starting from standard 'driver only' modes, passing driver assistance systems like Adaptive Cruise Control (ACC) and heading straightforward to fully automated vehicle (-fleets) one will find repeating definitions of automation levels. These levels will be briefly explained in this section. The different levels do not only define what a vehicle is capable of, they also have a strong impact on liability issues. A first definition of driving modes which is widely used was defined in [3] by a project-group of the German Bundesanstalt für Straßenwesen (BASt). The counterpart in the United-States, the National Highway Traffic Safety Administration (NHTSA) caught up with an own set of definitions. The most popular definitions however were published by the SAE (formerly known

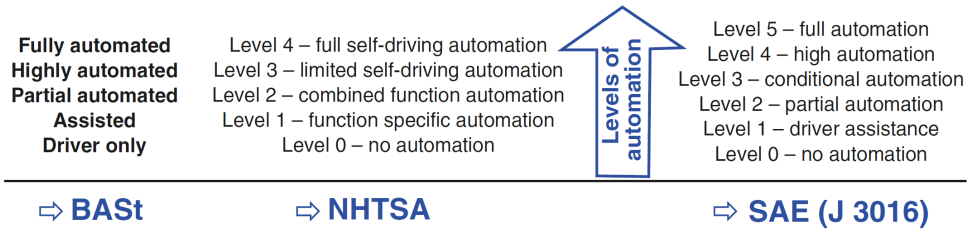


Fig. 1.1: *Standardization process of nomenclature of automation levels for autonomous driving, see also [2].*

as 'Society of Automotive Engineers') in 2014. Its definitions do fully match the ones of the BASt despite the definition of level five which is not available in the defined set of the BASt, see also Fig. 1.1. To eliminate confusion the standardized terminology of the SAE J 3016 will be used in this thesis. The definitions are starting with *driver only* at *Level-0*. The higher the number of the automation level, the more responsibility is handed over to the vehicle. Finally this ends up at *Level-5* full automation, see Fig. 1.2 for a detailed overview how the responsibilities are shifted from the driver towards the autonomous driving system. Warning and momentary intervention systems (like collision mitigation systems) do not change the role of the driver and therefore do not automate the Dynamic Driving Task (DDT), see also [4]. In the following subsections a more elaborate overview over the capabilities of the different driving modes will be given, while skipping *Level-0: No Automation*, where there is no automation system, which capabilities can be explained.

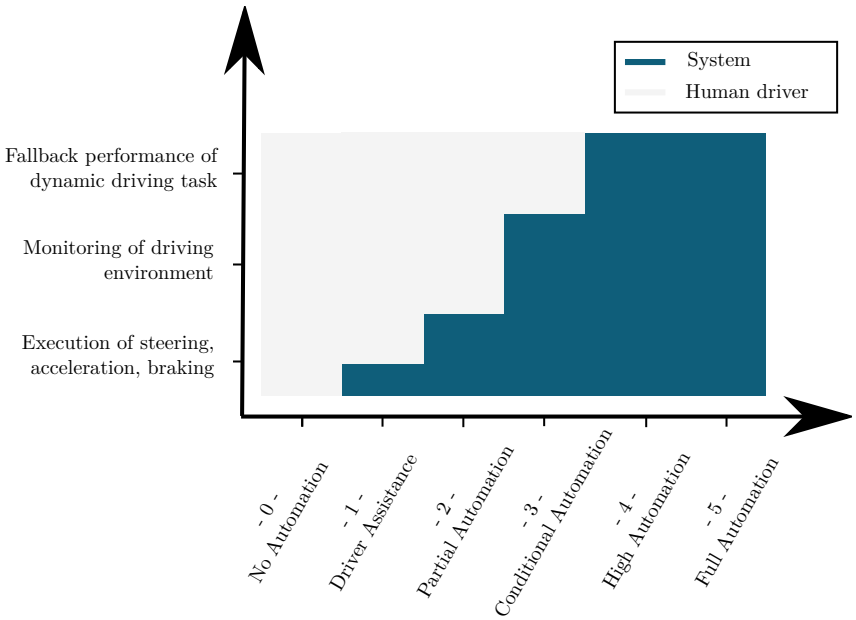


Fig. 1.2: *Responsibilities of the driver and an automated driving system at the different SAE automation-levels.*

1.1.1 Level-1: Driver Assistance

> *The driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the DDT.*

Definition of *Driver Assistance* by the SAE [4]

Driver Assistance systems have been the first steps towards driving automation. First prototypical implementations of an adaptive cruise control were developed in the 1980s [5]. These ACC systems extend the functionality of standard cruise control systems by ensuring a safe distance to the vehicle ahead, while trying to keep the velocity as close as possible to the value set by the driver. The system can be realized using any sensor which is capable of measuring the distance to the vehicle ahead, being it radar, lidar or stereo-cameras, see [6]. For a more detailed description of an implementation of such a system see [7]. Collision Warning/Mitigation/Avoidance systems are not a part of the SAE's definition of automation, due to only being intended to help the diver in emergency situations. Nevertheless, automated systems have to be at least as good in emergency situations as state-of-the-art collision avoidance systems. A major challenge in the development process of those systems is the unknown plan of the (human-) driver, who is executing the DDT. While the first systems on the market focused on vehicles ahead of the system-vehicle, more recent system help to avoid crashes at intersections and are capable to react to pedestrians [8].

1.1.2 Level-2: Partial Automation

> *The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the DDT*

Definition of *Partial Automation* by the SAE [4]

While Level-1 systems can be realized by implementing either steering *or* longitudinal assistance, a Level-2 partial automated system is the combination of longitudinal and lateral assistance. In terms of liability for this mode it has to be ensured, that the driver is in the loop and is therefore monitoring the driving environment all the time to allow the driver to act in an appropriate way if needed. Practically this is the first mode where the human driver is disengaged from the physical driving task. Still the monitoring whether the driver is in the loop and the transition between system-operation and driver-operation is a challenging task. In many systems this is solved by the requirement to the driver to keep his hands on the wheel. This can can be implemented by deactivating the system if no steering force was measured after a fixed take-over time. While Naujoks in his study [9] states, that this take-over

time can be longer than a few seconds, accidents showed, that long take-over times can lead to drivers, for which is hard to distinguish whether the system operates in Level-2 or Level-3 mode, see [10].

1.1.3 Level-3: Conditional Automation

> *The driving mode-specific performance by an automated driving system of all aspects of the DDT with the expectation that the human driver will respond appropriately to a request to intervene*

Definition of *Conditional Automation* by the SAE [4]

Conditional Automation is defined as the first automation level, where the system is fully responsible of monitoring the driving environment while staying in its system limits. When the system limits are reached, the driver needs the so called *take-over time* t_{warn} to get back into the loop to execute the 'Fallback performance of the dynamic driving task' [4]. The remaining question what amount of time is needed to get back into the driving loop in an appropriate manner is investigated in multiple publications, see for example [11], [12]. Hereby the question arises whether the take over time can also determined beforehand by taking into account the gaze of the driver [13]. It is however questionable if the take-over time is the only measure needed, or one should also take into account the take-over quality [14]. According to gasser [3], one can only think of accidents in which the automated system is involved which are caused by a third party. In addition, one can also think of (rare) system failures or force majeure as possible reasons for accidents. A major point is, that the driver does not have to recognize the need to take over control, this task has always to be executed by the system by warning the driver.

1.1.4 Level-4: High Automation

> *The driving mode-specific performance by an automated driving system of all aspects of the DDT, even if a human driver does not respond appropriately to a request to intervene*

Definition of *High Automation* by the SAE [4]

In difference to lower levels of automation one cannot rely on a driver as fallback operator when operating a vehicle in Level-4. According to the definition of the SAE a high automated system is always able to revert to a state of minimum risk, if the driver does not take over control. This can be a challenging tasking and is not possible under all circumstances in all situations and traffic environments.

For example, it may be hard to define a safe state on an expressway if no hard shoulder is available. This is the reason why a Level-4 system may be limited to specific traffic environments under a set of defined conditions (weather, daytime, etc.), see [15]. Highly constrained Level-4 systems may only be capable of following

Fig. 1.3: *Automation level-4 vehicle EZ10 of the company EasyMile in Helsinki. The system is capable to follow learned routes and is used to for last-mile transportation with a maximum velocity of 12 km/h [16].*

specific routes with limited speed, e.g. the systems presented in [16] and[17]. The set of all restrictions are called system borders and vice versa the capabilities and conditions in which a Level-4 system shall be operated the Operational Design Domain (ODD). There exist various visions on how Level-4 automation should look like, e.g. low-speed shuttles for the last-mile of travel [16] and on demand ride hailing fleets [18]. This results in the possible challenge of needing specific licenses for the different Level-4 systems in the respective context in which they are operated. How state authorities will handle this vehicle specific licensing process remains unclear [15].

1.1.5 Level-5: Full Automation

> *The full-time performance by an automated driving system of all aspects of the DDT under all roadway and environmental conditions that can be managed by a human driver*
>
> Definition of *Full Automation* by the SAE [4]

Reaching Level-5 means that vehicles are capable of operating vehicles in all conditions in which an human can operate a vehicle. In addition to capabilities already known by Level-4 systems, Level-5 systems would also implement for example the capability to operate off-road vehicles in undeveloped areas. This would therefore require an almost human-like artificial intelligence.
The definition of Level-5 can be interpreted as a reminder, that Level-4 systems always have limitations, see Fig. 1.4 for a visual explanation. To the best of the knowledge of the author of this thesis there are no companies or institutions aiming to realize a Level-5 system. For these reasons the definition of Level-5 is only of theoretical interest at the current point in time.

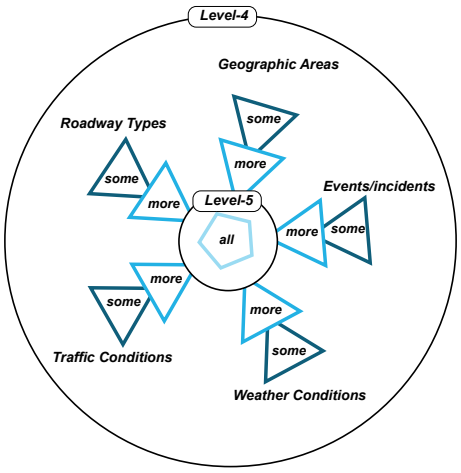


Fig. 1.4: *Transition from Level-4 to Level-5. It is not sufficient for a Level-5 vehicle of being capable to handle all aspects of one subdomain like being able to handle all Roadway types. A full automated vehicle has to be able to be operated safely under all conditions and situations in which a human can operate it.*

1.2 Fallback from Level-3 Driving

On the one hand the definition of Level-3 systems simplifies the automated driving task, because a driver is always available as fallback when reaching the limits of the automated system. On the other hand the implementation of such kind of systems raises the human-machine-interaction question how control can be handed back to the driver. This includes the mechanism how a driver is warned, informed and brought back into the loop but also the aspect how much time a driver needs to fulfill this task.

1.2.1 Human Machine Interaction

Consequently, the questions which kind of second task can be executed by the driver and further how this affects the take-over quality and take-over time comes up. In [14] the effect of drivers executing a second task vs. drivers not doing so is investigated. The data presented in the study shows that a non-driving related second task, for example watching video/reading newspaper/email impairs the reaction of a driver negatively. Answers to the question of how take-over systems shall be implemented to maximize take-over quality and minimize take-over time are delivered in [12]. The result is, that visual-auditory warnings are highly superior to pure visual Human-Machine-Interaction HMI approaches, regarding time and quality. In [19] it is investigated how the take-over time varies depending on the traffic situation and non-driving related tasks the driver is executing. As result the study suggests, that the take-over time shall increase with the complexity of the traffic scene, while a variation of the non-driving related task seems to have only minor impact. The investigations of [20] show that it is hard to provide a Human Machine Interface (HMI) supporting drivers in the take-over phase to support situational

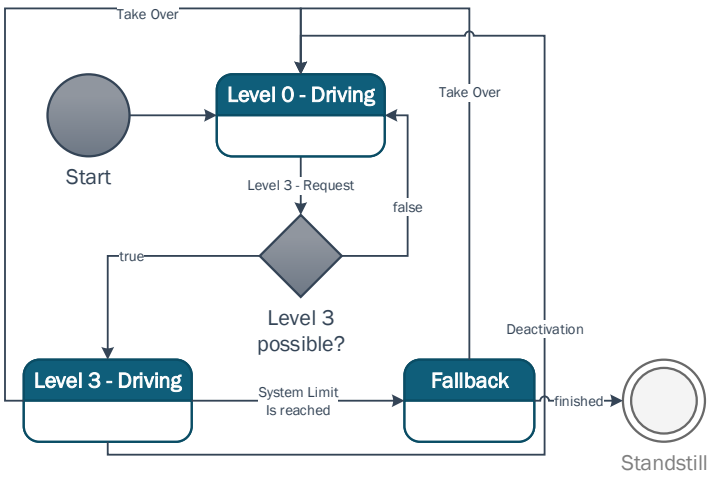


Fig. 1.5: *State transitions between fallback behavior, Level-3 and Level-0 driving.*

awareness and safety. Another finding is, that drivers tend to perform non-driving related tasks if driving with an automated system. As a conclusion, from the different research items, it is hardly possible to define one fixed value for the take-over time. Its definition heavily depend on the take-over quality and the design of the HMI system.

1.2.2 State transitions of a Level-3 System

The methods envisioned in this thesis are developed with a focus on problems which need to be solved to realize Level-3 systems. One main issue in this context is how to handle situations, in which the Level-3 system detects, that it is not operating within its system limits at the current time-point. For these kinds of situations a fallback-behavior needs to be available in the system, bringing the driver back into the loop. To handle control back to the driver, the corresponding state-transitions have to be designed carefully, see Fig. 1.5 for a flowchart visualizing the approach used in this thesis, and [21]. In the envisioned system the activation of the Level-3 automation system has to be requested actively by the driver of the vehicle ('Level-3 Request' in Fig. 1.5). However, it can only be activated in case its system inherent constraints are fulfilled, see Sec. 1.1. The constraints may consist but are not limited to the type of the road, the speed-limit and the health of the system. There are two possibilities to get out of the Level-3 state. The first is by an active request of the driver, for example by taking over the steering wheel or executing a braking maneuver (see 'Take Over' in Fig. 1.5). The second possibility is, that the system detects, that it is not operating within its system-limits anymore (for example a dropout of a sensor, or an immediate change of weather conditions). In this case a fallback behavior has to be triggered ('System Limit is reached' in Fig. 1.5), which will be executed until a standstill is reached or the driver takes over control.

1.3 Research Contribution

The central question leading to the subsequent questions answered in this thesis is the following:

> *How can an activated Level-3 system plan and execute a fallback-performance if the environment perception drops out until the driver is back in the loop?*

This question however can be generalized. Every situation, in which a Level-3 system detects it is not operating within its system limits anymore raises the same issue to be solved. To attack the problem, the contribution of this thesis is threefold. The first contribution is a system concept implementing the fallback performance of a Level-3 system. This concept is based on the second and third contribution, the methods to provide probabilistic long-term predictions of vehicles and a trajectory planner, which is able to take this prediction information into account. Overall the methods provided here shape the path from Level-2 Partial Automation to Level-3 Conditional Automated systems. More pictorial the questions answered in this thesis are:

- What are the future positions of other traffic participants?
- If we know those future positions, how can a safe fallback-maneuver be planned?

This thesis introduces a system design and a toolkit of methods capable of solving the described problem. Besides being tailored for Level-3 systems and highway environments, the methods are designed in a way that they can be used for different levels of vehicle automation.

1.3.1 System Design

1.3.1.1 System Design Problem

As described in Sec. 1.1, the main step from Level-2 to Level-3 can be seen in the fact, that the driver does not have to monitor the traffic anymore. He only has to be available for getting back into the loop after a few seconds. For the automated vehicle this means that a fail-safe strategy needs to be available, which is capable of keeping the vehicle in a safe state until the driver is able to take over control. Therefore a safe automated vehicle needs to have at least one of those two capabilities:

- Detecting possible system limits at least t_{warn} beforehand
- Executing the fallback performance until driver takes over control or vehicle is in a safe state

While the first capability is a demanding challenge to the in-vehicle environment sensing and recognition systems, the second solution raises high requirements to the

concept and performance of the fallback system. Still, it may be hard guaranteeing to detect system limits beforehand in every situation, which makes the availability of a fall-back behavior realizing fail-safety mandatory.

1.3.1.2 Contribution in System Design

This thesis presents a concept for a non-fail-operational Level-3 system. To implement fail-safety the components of the automated driving system are continuously monitored. On reaching a system limit or detecting a failure, the automated system switches to the fallback performance. On the one hand the problem how early a driver shall be warned is discussed thoroughly in the scientific community. On the other hand the question of how a technical solution guaranteeing the take-over time within a Level-3 system can be implemented is not widely discussed, see also section 1.2 of this chapter.

The approach proposed in this thesis resolves the aforementioned issue. Using the last sample of valid input data, consisting of dynamic and static objects and the traffic infrastructure (e.g. lane markings and endings), the position of all objects are predicted as a function of time. The knowledge of future traffic scenes allows deriving the free-space as function of time. Using a representation of this prediction information, a method to plan trajectories is presented allowing the vehicle to execute collision free maneuvers. The major contribution of the concept developed in this thesis is therefore providing a consistent concept to safe maneuver-planning in fallback scenarios. This especially holds true for planned trajectories in situations when no new input-data is available, for example due to a drop-out of the sensor or problems in the sensor-fusion system.

1.3.2 Maneuver recognition

1.3.2.1 Maneuver Recognition Problem Description

To achieve an abstract understanding of how other traffic participants will act in the future it is crucial to have an understanding of their plans and intention. The argument, that vehicle to x communication (V2X) can solve this issue does not hold true until 100% market penetration of such kind of technology is available. The only chance therefore is to estimate the behavior of other vehicles based on observations. For example in highway scenarios one may observe a vehicle approaching a slower truck. Inferring from former observations, one may predict, that the respective vehicle will cut-out to overtake. Because different drivers act in different manners, such kind of predication are always uncertain. This results in a basic assumption in this thesis, which is that predictions of vehicle (and therefore also human) behavior are always uncertain and thus have to be modeled probabilistically. To determine these probabilistic predictions also the question, which observable features contain the most information about future human behavior, needs to be answered.

1.3.2.2 Contribution in Maneuver Recognition

The contributions in terms of maneuver recognition are twofold. The main contribution is the systematic investigation of the most relevant features for lane change recognition in highway scenarios.

In a first step, a superset of features is systematically constructed based on a model of the traffic scene, which on the one hand provides a simple representation of the environment, but also contains the relevant information influencing vehicles. This is quite unlike previous strategies [22] [23] [24], where the feature set was chosen by so called 'expert knowledge'. Using this superset of features the relevance of each feature as function of time until a vehicle changes its lane assignment is evaluated.

The second contribution concerns about the use of machine learning techniques for feature-selection and intention recognition with highly unbalanced classes. The experimental results proved the strategy clearly superior to many former approaches which were purely focused on a fixed feature set in combination with machine learning techniques or explicit models, see also section 4.2. Using simple classification algorithms the methods proposed in this thesis showed superior results compared to former approaches, see chapter 4 for a detailed discussion.

1.3.3 Prediction of Future Vehicle Positions

1.3.3.1 Position Prediction Problem

Given the situation, that a probabilistic prediction of the maneuver class is already available, see Sec. 1.3.2, one still has no knowledge about future positions of other traffic participants. As already the knowledge about the maneuver class is uncertain, the information about future positions has to be represented in a probabilistic fashion, too. Due to complexity of real-world traffic a probabilistic position prediction shall on the one hand be done with respect to vehicle dynamics, but on the other also reflect the behavior of human drivers, which are interacting with each other. To make this problem even more challenging, a function generating this kind of information needs to be processed efficiently in real-time on in-vehicle Electronic Control Units (ECU).

1.3.3.2 Contribution in Position Prediction

The main contribution regarding position prediction presented in this thesis is a real time capable solution for the probabilistic position prediction of object vehicles in highway scenarios. Unlike previous strategies, see for example[25], [26], the behavior uncertainty of vehicles is taken care of. The output of the provided method is a distribution for the future vehicle positions function of time. This distribution is estimated with respect to the current vehicle state and the estimated maneuver class. Compared to former publications non-measurable features like 'desired velocity' [25] of a vehicle are not used. All inputs of the proposed algorithms (see chapter 4) are measurable with current automotive sensors. Compared to [27] future behavior is not derived from past behavior, e.g. the past trajectory. Instead, the Markov

assumption is used, which basically says that information of past states is aggregated in the current state, where the state of a vehicle also includes the relation to its environment, see [28] for a tutorial on Markov Models.

The result is a real-time capable algorithm which is parametrized by machine learning techniques allowing to represent the complexity of driver reactions in a probabilistic fashion. Hereby the probability density functions of future positions represent the uncertainty as observed in real-world situations. The main advantage of using this output for situation analysis and trajectory planning is not only to have precise estimate of the positions of other traffic participants. It is also to have the knowledge in difficult situations that the future position of a vehicle is highly uncertain.

1.3.4 Trajectory Planning in Structured Dynamic Environments

1.3.4.1 Trajectory Planning Problem

Experienced drivers understand the traffic scene which can be interpreted as the extraction of a feature vector. Based on this feature vector they predict other vehicles by estimating their probable future position and use this information as constraints in combination with the static environment for decision making and trajectory-planning. While methods for predicting are part of the contribution of this thesis, see Sec. 1.3.2 and Sec. 1.3.3, the problem remains to provide a planning method, capable of dealing with the infinite number of traffic configurations. Hereby a trajectory planning method shall incorporate arbitrary predication information and also the constraints of the static environment, e.g. the shape of the road and static obstacles.

1.3.4.2 Contribution in Trajectory Planning

In the scope of this thesis a trajectory planner is presented, which incorporates prediction information of other traffic participants more explicit into account compared to former methods, see chapter 6 for a detailed overview.

From a system-architecture perspective, the trajectory planner is only loosely coupled to software modules which are executing their tasks at an earlier stage in the functional chain (e.g. vehicle prediction) and is callable using a defined geometric interface.

The contribution of this approach is threefold: First a generic interface between scene predictions, higher level maneuver decisions and the trajectory planner is presented based on a formal problem definition. This interface provides the hard constraints for the planning problem and is independent of the underlying trajectory planning method. The second contribution covers an increased flexibility of trajectory shapes compared to former sampling based planning approaches, which were limited to a fixed number of motion primitives. The third contribution is a method for

algebraic optimal sampling of trajectories based on sampled trajectory points, which enables the approach to be computed efficiently in real time.

1.4 Outline

This thesis is structured as follows. In chapter 2 the methodical foundations needed to gain an understanding in the following chapters are explained. This includes an introduction to safety engineering definitions, the used coordinate systems, the needed fundamentals of machine learning, statistical evaluation and important vehicle models. In order to achieve an understanding which functions need to be developed to implement a fail-safe Level-3 system the architecture of the envisioned system is described on different level of abstraction in chapter 3. Within this chapter, three major functional blocks are identified, which will be discussed in detail in the following three chapters.

Chapter 4 provides a deeper insight into how maneuvers of traffic participants can be recognized from a conceptual a methodical perspective. The presented algorithms are evaluated in two experiments where the first experiment is focusing on establishing a meaningful benchmark using simple methods while the second is aiming on maximizing the prediction horizon. Using the generated information of the methods of maneuver recognition, chapter 5 introduces an approach how future positions of vehicles can be predicted in a probabilistic fashion in longitudinal and lateral direction. The presented method is evaluated in two experiments. To be able to process the prediction information, of the previous two chapters, in chapter 6 a sampling based method of trajectory planning is introduced which is able to handle long term prediction information. Therefore, a technique to represent dynamic freespace information and a flatness based method for the generation of composite trajectories are introduced. The thesis concludes with chapter 7 which summarizes the achieved results and provides an insight into possible research perspectives.

Chapter 2

Background

The following chapter provides the theoretical foundations for the research conducted in the chapters 3 - 6. Methods and algorithms, which are only used within a single chapter are introduced in the respective chapter, all other background information needed is covered in the following sections.

This chapter starts with a brief overview on what safety means for automated driving functions in Sec. 2.1. The focus of the chapter then changes towards more technical topics. In Sec. 2.2 the coordinate systems used throughout this thesis are explained. The chapter continues with the needed foundations of machine learning in Sec. 2.3 and concludes with an overview on the vehicle models used in this thesis in Sec. 2.4.

2.1 Safety of Automated Driving Functions

The basic requirement to driving automation systems is, that they are safe. The definition of *safe* and how safety can be quantified and assessed is however not intuitively clear. In this section an overview of the relevant safety definitions and the context in which they are embedded will be given. This also includes liability, which becomes relevant in cases, in which the safety of an automation system is called into question.

2.1.1 Definitions

Safe describes the property of a system, that it causes no, or only minimal harm to itself, people and its environment, see [29]. *Fail-safe* systems are described as systems, which, in the case of failure, are still *safe*, see [30]. *Fail-operational* systems are described as systems which in the case of failure can continue their operation. This is typically implemented by double- or three-time redundancy of components, see [31].

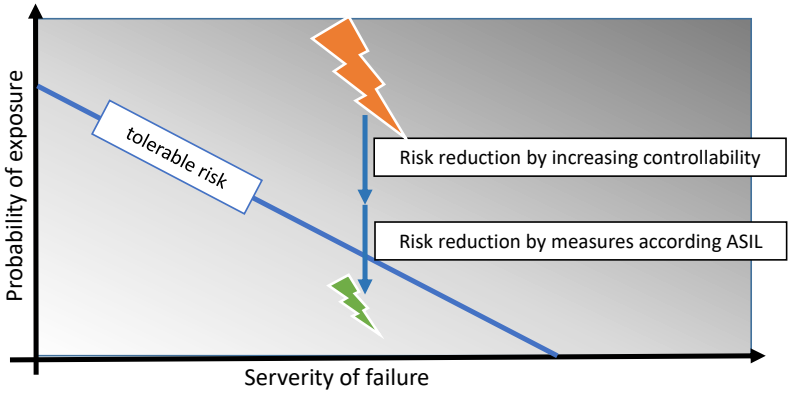


Fig. 2.1: *Strategy how a a non-tolerable risk is transformed to a tolerable one by increasing the controllability of the situation and by taking measures according to ASIL.*

2.1.2 Functional Safety

More formal the part of safety, which depends on the correct functionality of the safety relevant system and other risk minimizing functions is called *Functional Safety*. Requirements and processes for Functional Safety for ADAS and Autonomous Driving are defined in the ISO 26262, which is derived from the IEC 61508. While the IEC61508 is the basic functional safety standard for all industries and different kind of products, the ISO26262 is intended for functional and electrical safety of automotive products, see [32]. To comply to the processes defined in the ISO26262, a product among other things has to be assessed in hazard and risk analysis. The main goal of this is to:

- Recognize and detect possible dangerous situations
- Take countermeasures to avoid, or if not possible to mitigate dangerous situations

This process is visualized in Fig. 2.1, see [33]. Within this figure the task of a system design is always to come up with solutions which are below a 'tolerable risk', where risk is defined as the combination of 'severity of failure' and the 'exposure'. A dangerous situation with a high probability of exposure is not tolerable for an automated driving system. In case of a Level-1, Level-2 and Level-3 system the manufacturer can still rely on a human driver, who may be able to resolve those situations. To do so, the controllability by the driver has to be ensured. For example in case of a sensor dropout it may help to increase take over time, which describes the time until the driver needs to take over control. For Level-3 systems it is important allowing the driver to assess the situation carefully. Additional measures to be taken in the Risk & Hazard analysis may then be able to bring the risk to a tolerable level.

2.1.3 Safety of Use

Safety of Use includes all risks, which users and third parties are faced with in case of intended use and foreseeable miss-use. This however does not include gross misuse. Precondition for all considerations is, that all components are working fault-free as intended. To gain an insight of the importance of this safety aspect, one has to know that human error and failure provokes a major part (86%) of accidents. Only in 11% of the cases the cause is not clearly recognizable and only 3% of the cases are about technical errors or failure, see [34]. Main reasons for accidents are: speeding, distance under-runs, right of way errors and insufficient visibility. While major parts of these risks can be minimized by automated systems, new risks may occur. For example drivers in a Level-2 system may try to not fulfill their task to monitor the traffic environment, see [35]. To analyze the safety of use, one shall consider the following four aspects of safety related systems:

- comprehensibility
- predictability
- controllability
- potential of misuse

see [36]. Major parts of the challenges which have to be resolved to improve the 'Safety of Use' are part of the Human-Machine-Interface which is not in the scope of this thesis.

2.1.4 Liability

To achieve legal security for the manufacturer of a product, its development process shall comply to the state of the art. For automotive safety relevant systems, this means that all developments and development processes have to comply to the ISO2626, which is the relevant norm for the development of in-vehicle systems. These also includes the task to document the design of the product. In case of recourse or liability, this is a guideline for the lawsuit of a damaged party. Following [37] a plaintiff according to the malfunction doctrine only has to prove that:

- "the product malfunctioned
- the malfunction occurred during proper use, and
- the product had not been altered or misused in a manner that probably caused the malfunction."

The plaintiff can force the manufacturer to provide all relevant documentation to avoid a sentence due to discovery (which is inadmissible under German civil law), see [38]. The manufacturer in this case has to prove, especially for a safety related system, that the method of development complies with the state of the art. This leads to high requirements regarding the processes for development and validation and also the measures reducing risks, see [37], [39].

2.2 Coordinate Systems

To construct a model of the environment and to create plans for a vehicle one needs coordinate systems, in which information can be represented. For different calculations it is however handy to use different coordinate systems. In case of calculating the time until a collision may happen it may make sense to do all calculations for example using a reference point on the front of a system vehicle. When planning trajectories using a model of the vehicle kinematic, it is handy to use a reference point around which the vehicle rotates(e.g. rear axis in case of a bicycle model). In a different use-case, when analyzing how different vehicles relate to each other, it is useful to use a curvilinear coordinate system. In this section an overview will be given, how the different coordinate systems are defined and how they can be transformed into each other.

2.2.1 Vehicle Coordinate Systems

Within ISO 8855, there exists the definition of two main coordinate systems, the earth fixed and the vehicle fixed one, see [40]. Within the earth fixed one, the x_e and y_e axis are parallel to the ground plane, while the z_e direction is pointing upwards contrary to the gravitational force. This earth fixed coordinate system may be fixed in an arbitrary location on the ground plane.

In contrast, the vehicle coordinate system is fixed in a vehicle reference point. For different applications, different reference points are used in industry and science. Within this thesis the vehicle reference point is fixed in the middle of the rear axis. In the vehicle coordinate system, the x_v-axis is pointing forwards, the y_v-axis orthogonal to the left and z_V is pointing upwards. The vehicle coordinate system is a Cartesian coordinate system. For many applications however, the definitions given beforehand are not handy. This mainly stems from the fact, that every vehicle defines its own vehicle coordinate system. Whenever for example relations between two vehicles shall be investigated, it is handy to use a within ISO 8855 so called 'Intermediate axis system'. Within this system, x and y are projections of x_v and y_v on the ground plane. The z-axis is defined starting in the vehicle reference point pointing upwards while being orthogonal to the ground plane, see Fig. 2.2 for a visual explanation.

2.2.2 Curvilinear Coordinates

Roads in general are not straight. Thus, for many calculations it is more feasible to use a orthogonal coordinate system which is following the curvature of the road to 'rectify' traffic scenes. The coordinate system used in this thesis will be called 'lane-coordinate system'. The longitudinal distance along the curvature of a reference line will be denoted as s, the lateral offset as d. The transformation from the lane-coordinate system to the euclidean vehicle coordinate system and vice versa cannot be solved analytically in general. Within this thesis it is assumed, that the reference line L_{ref}, which is the basis for the transformation between the lane and the euclidean coordinate system is represented as a polyline. To solve the transformation

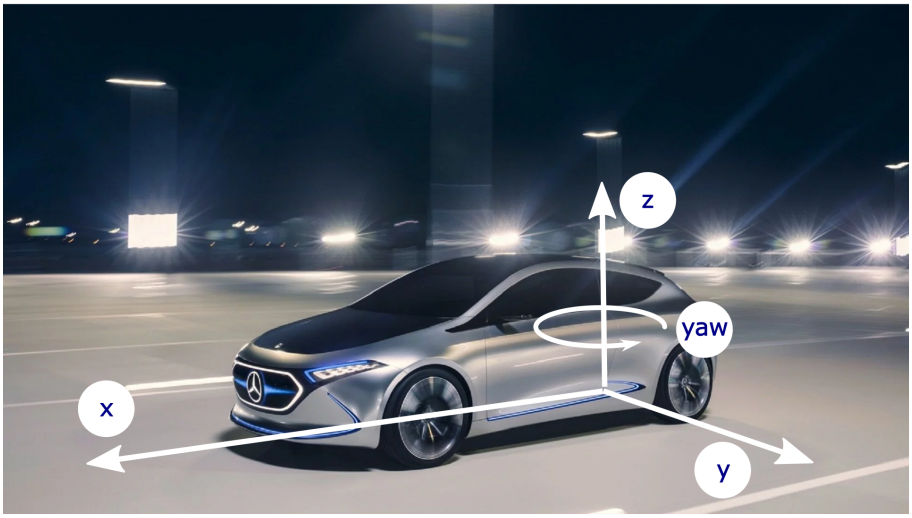


Fig. 2.2: *Visualization of the intermediate axis system as defined by ISO 8855 and used within this thesis.*

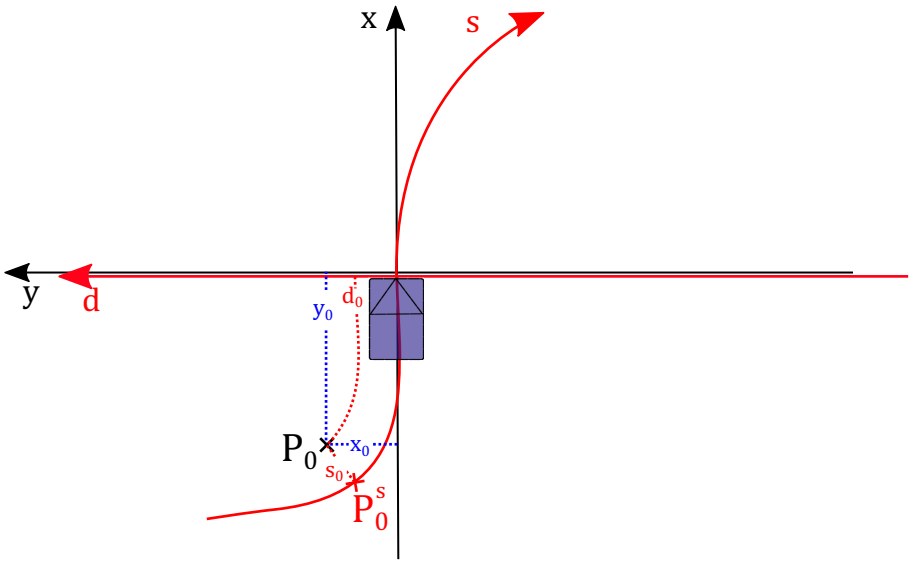


Fig. 2.3: *Using the transformation function Γ and Γ^{-1} one can transform between the Cartesian coordinate system and the curvilinear lane-coordinate system. P_0 is transformed from the Cartesian coordinate system by being projected onto the red solid reference line of the lane-coordinate system. The distance between P_0 and the projected point P_0^s represents the lateral component d_0. The corresponding s value is determined by the integrated length of the reference line until the origin of the lane coordinate system.*

task, a straightforward algorithm will be used throughout the thesis, using the reference line L_{ref} along which s is measured. The algorithm is limited to the case, where the distance of a Point P to this line d_{lat} is small compared to the radius of the road defined by its curvature. This always holds true on highways. A polyline of n points P in the euclidean space is defined as:

$$L_{ref} = [P_1, .., P_{n-1}, P_n], \text{ where} \tag{2.1}$$

$$P_m = \begin{pmatrix} x_m \\ y_m \end{pmatrix} \tag{2.2}$$

In order to transform between the euclidean and the curvilinear space a function Γ is introduced, which is able to do so, see also Fig. 2.3. The function transforming back to the Cartesian coordinate system is called Γ^{-1} respectively:

$$P^{sd} = \Gamma(P^{xy}) \quad \text{and} \quad P^{xy} = \Gamma^{-1}(P^{sd}) \tag{2.3}$$

For a short survey on how to compute this transformation in general see [41]. While different reference lines can be used for different lanes, which is especially needed in case of merging roads, intersections and roundabouts, throughout the thesis only a single reference line is used. This limitation is introduced due to the focus on highway scenarios, where only one driving direction is considered to be relevant for the Level-3 automated vehicle. Please see Alg. 1 of how a naive algorithm implementing Γ can be realized.

Alg. 1 High Level definition of the transformation function Γ

procedure FINDCLOSESTPAIRNAIVE($P, \mathbf{L}$) ▷ brute force
  $i \leftarrow 1$
  $d \leftarrow \infty$
  for $i < numElements(\mathbf{L})$ **do**
    $d_{cur} = |\mathbf{L}[i-1] - P| + |\mathbf{L}[0] - P|$
    if $d_{cur} < d$ **then**
      $i_{min} \leftarrow i$
      $d \leftarrow d_{cur}$
    end if
  end for
  return$(\mathbf{L}[i_{min} - 1], \mathbf{L}[i_{min}])$
end procedure

procedure PROJECTONLINESEGMENT(P_{in}, P_{s1}, P_{s2}) ▷ Segement defined by P_{s1} & P_{s2}
  Calculate line through P_{s1} & P_{s2}
  Calculate intersection with perpendicular through P_{in}
  $d =$ distance from intersection to P_{in}
  $s_{s1} =$ distance from P_{s1} to intersection
  $s =$Length integrated up to $P_{s1} + s_{s1}$
end procedure

procedure $\Gamma(P^{xy}, \mathbf{L}^{xy}_{ref})$
  $P_{prev}, P_{next} \leftarrow findClosestPairNaive(P^{xy}, \mathbf{L}^{xy}_{ref})$
  $s, d \leftarrow projectOnLineSegment(P^{xy}, P_{prev}, P_{next})$
  $P^{sd} \leftarrow makePoint(s, d)$
  return(P^{sd})
end procedure

2.3 Machine Learning

There are many real-world problems in which one wants to know future values based on current observations. For example a classification of available information is needed, e.g. whether an email is spam or not. To do so, in many applications computers are used to automate such kind of tasks. Based on an observation $\mathbf{x}$ output values containing the information of interest $\mathbf{y}$ shall be generated. Machine learning in this context describes various algorithms to learn models or their parameters using statistical techniques together with training data. This approach differs to so called 'Expert System' which emulate human decision making, by having decision rules explicitly programmed into them.

This underlines the main difference: in expert systems decisions are based on human interpretations, in machine-learning they are data driven, by having built a model which is based on a set of training-data. One of the key advantages of machine

learning techniques is, that even pretty complicated and high-dimensional problems can be solved straightforward if a sufficient number of training-data is available. There exist a huge number of related terms which partly overlap or are fully included in the definition of machine learning, being it big-data, data mining, computational statistics or predictive analytics. Within various fields, be it engineering, finance industries, and research machine learning is used to generate (maybe hidden) insights, classify, predict and to detect trends and values using historical and current data. Within machine learning, there exist mainly two different approaches to learn output-values $\mathbf{y}$ from input-data $\mathbf{x}$. So called *Generative Models* generate a model of the joint distribution of input- and output-data which results in a probability distribution $p(\mathbf{x}, \mathbf{y})$. The second approach is learning *discriminative* models by training the posterior of $\mathbf{y}$ given $\mathbf{x}$ resulting in a distribution $p(\mathbf{y}|\mathbf{x})$. Various studies observed that discriminative models outperform generative ones in classification applications. On the other hand generative models may converge faster with fewer data samples, especially when using parametric distribution models, see [42]. Intuitively one can understand that learning the desired result directly in discriminative models produces superior prediction performance when compared to learning a distribution model containing all in and output dimensions. On the other hand a handy property of generative models is, that it is possible to sample pairs of $\mathbf{x}, \mathbf{y}$ from a trained model.

However when dealing with Machine Learning techniques terminologies are defined differently in literature. In this thesis terminologies are defined according to [43]:

- *Target Function* as the (sadly) unknown underlying function one wants to approximate by a hypotheses
- *Learning Algorithm* as the method capable of inducing a model of the target function given the training data
- *Hypotheses* as a function approximating the target function
- *Model* analogous to hypotheses
- *Model Parameters* as the parameters defining the 'behavior' of the model itself when fed with data
- *Hyperparameters* as the tuning parameters of the model, i.e. the number of k in k-means or the topology of a Neural Network

Within this section an overview of how models can be fitted to data, how the 'best' model can be selected from the various ones possible, how performance can be measured and a introduction to some of the most important basic learning techniques applied within this thesis will be given.

2.3.1 Model Selection

A main problem when generating models is the so called Bias-Variance Dilemma or Bias-Variance Tradeoff. The term *Bias* is defined as the average error of predicted

values vs. the correct values. Models with high bias have high error in the training and evaluation dataset. Intuitively they can be understood as models which are oversimplified, which is called *underfitting*. In contrast, models with high *Variance* have a pretty accurate fit on the training data but fail to generalize on evaluation-data, which is called *overfitting*, see also Fig. 2.4 for a visual explanation. The

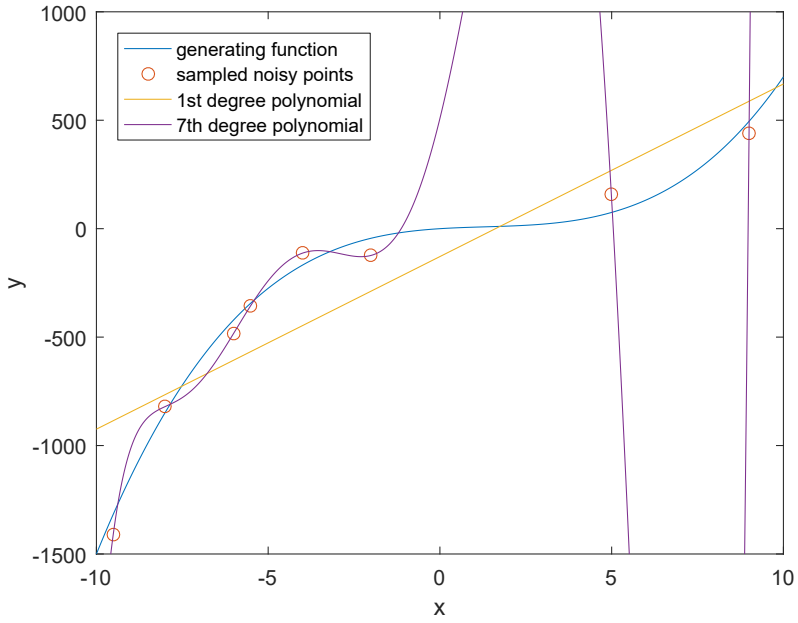


Fig. 2.4: *Visualization of the Bias-Variance Tradeoff. Samples (red) are sampled randomly from the generating polynomial* $x^3 - 4 * x^2 + 10 * x$*. When fitting a 1st-order polynomial the samples of the generating function are not approximated well. The function which tries to approximate has not a sufficient number of degrees of freedom. In the context of model fitting this is called high bias or underfitting. On the other hand a polynomial of 7th degree has to many degrees of freedom and fits the sampled points well but clearly is not a good fit for the generating function, which is called high variance or overfitting.*

reason why these both problems are coupled, stems from the fact that a reduction of the complexity of a model, for example reducing the degree of a polynomial, results in an increased bias (depending on the dataset). Vice versa increasing the capacity of a model by adding additional degrees of freedom may lead to a higher error due to variance. To minimize the total error an optimum has to be found. Therefore, the task of model selection is to find the right or *optimum model complexity*, see [44].

Given for example a binary classification problem, for which it is assumed that there is a unknown function f which generates training data and therefore maps samples x to output variables y. A specific output instance y_n in this case would be always one of the two class labels:

$$f : x \rightarrow y. \tag{2.4}$$

Assuming that a set of hypotheses functions $h \in \mathcal{H}$ are available, which are theoretical

also able to map input variables x to output variables $y \in Y$:

$$h_i : x \to y\,, \tag{2.5}$$

where h_i is one specific hypothesis instance within H. In this case a learning algorithm $\mathcal{I}$ has the task to choose a model representing a hypotheses h_s out of $\mathcal{H}$ minimizing the total error, in order to approximate f as close as possible. Given a set of data used for training $\mathcal{X}_n = \{x_1, x_2 ... x_n\}$, the empirical error E_{emp} of a hypotheses h_i after training can be measured by:

$$E_{emp}(h_i) = \frac{1}{N} \sum_{n=1}^{N} |h_i(x_n) - f(x_n)| \tag{2.6}$$

The empirical error measures how well the hypothesis performs on the training dataset. However one may be interested in the performance on the overall dataset. This Generalization Error can be denoted as:

$$E_{gen} = P(|h(x) - f(x)|) \tag{2.7}$$

From the formula provides the insight, that minimizing the error on the training dataset alone is not sufficient to minimize the Generalization Error. Using machine learning techniques the goal is to get E_{gen} in an ideal world close to zero. Even if this would be possible, it would not be possible to verify this, because in the general case f and E_{gen} are unknown. In his research Vapnik [45] investigated under which conditions the following can be approximated:

$$E_{gen} \approx E_{emp}. \tag{2.8}$$

Even if one knows that the empirical error can be used as an approximation of the generalization errors, the question arises under which conditions it approximates well and whether there are error bounds which guarantee the quality of the approximation. The work in [46] shows that with a probability P_{bound} the following bound can be guaranteed for a classifier:

$$E_{gen} \leqslant E_{emp} + \sqrt{\frac{h(\ln(\frac{2n}{h}) + 1) - \ln(\frac{1-P_{bound}}{4})}{n}} \tag{2.9}$$

where h denotes the VC (Vapnik–Chervonenkis) dimension of the data and n the number of training data. Please see [47] for a detailed explanation. Unluckily the VC dimension can only be obtained for a very small number of functions and has only gained practical benefit when being used for linear discriminant functions. To answer whether a hypothesis will generalize well on unseen data, in real-world applications evaluation methods (for example: cross-validation) are widely applied. An overview of how these methods work is given in the next subsection.

2.3.2 Evaluation Methods

When questioning how good the performance of a trained model is, one may come up with the question of why this point is particular of interest. There exist three major use-cases why one may be interested in measuring a models performance:

- Estimating the generalization performance on data not used for training
- Selecting the best performing hypotheses from the set of model hypotheses
- Comparing different machine learning techniques, where each of them shall be tuned with the optimal hyper-parameters and the optimal hypotheses

In this context hyper-parameters describe the parameters which determine the possible complexity of model, but not the parameters of the model, which are inferred by the learning algorithm itself.

2.3.2.1 Holdout and Stratification

The most simple technique to evaluate a model is the holdout method. When applying this technique the dataset is split into a training and a evaluation set, see Fig.2.5 for a visualization. While the first one is selected for training of model, on the second one the performance of this trained model is evaluated, see [48]. It is important to note that the performance shall not be measured on the dataset used for training, since this may result in way too optimistic performance estimates. This is due to the fact that in this case one can not distinguish whether the model generalizes well or it is just the perfect fit through the samples of the training data, see also Fig. 2.4.

In order to split the data, the most common technique, especially when it comes to deep learning applications, is to randomly select one third of the whole dataset as evaluation data while the rest remains for training, see [43]. This however may raise issues in case of high (class-) biased machine learning problems and small datasets. By doing random sampling from the dataset a class imbalance may increase, or in the worst case no samples of the minor represented class are available in the evaluation set. A workaround for this issue is making use of stratification techniques. Those techniques take care that in the sub-sampling process the proportion of classes are maintained. Another trade-off one has to tackle when applying the holdout method are the proportions of the training versus the evaluation set. Assigning only a small proportion of the data to the training set may result in models which make not optimal use of the power of the hypothesis. A small the size of the evaluation set however may result in less precise measures of the hypothesis performance.

A method to deal with this issue is the so called *Monte Carlo Cross-Validation*. To do so, the dataset is repeatedly splitted k times into a training and a evaluation set. To get a robust overall prediction of the models performance the average of the k performance measures is computed to get the needed robustness. The value of k in this case can be quite large compared to the naive holdout method. This benefit however is attended by the k-times increased training time of the model. In

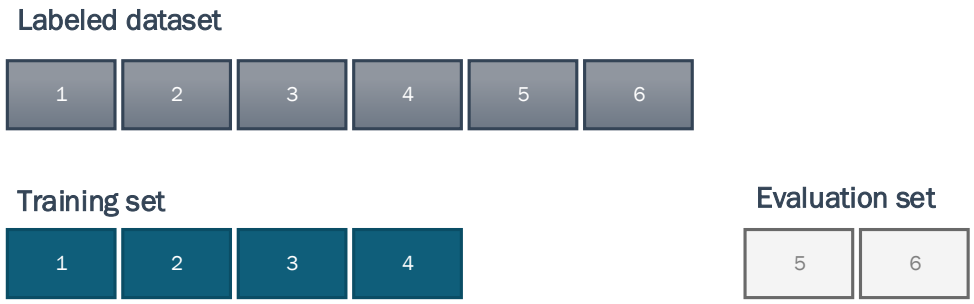


Fig. 2.5: *The Holdout method splits training and evaluation dataset straightforwardly.*

the last years the holdout method became increasingly popular for deep learning applications, often executed as 3-fold holdout. This is mainly due to the fact that deep learning methods are only chosen whenever large datasets are available. Because of the high number of samples, the risk of high variance is minimized. The reduced computational effort compared to other methods makes the holdout method to the method of choice in such cases.

2.3.2.2 Bootstrapping

The idea behind bootstrapping is the generation of sampled distributions which are 'good' estimates of the distribution of the whole dataset. The main difference to the Monte Carlo Holdout method is, that in every round of bootstrapping samples are selected from the overall dataset with replacement. This means that individual samples in the sampling process can be selected multiple times. A popular technique allowing statistically good performance estimates is the *leave one out bootstrap* method (LOOB). In difference to the original definition of bootstrapping in [49], samples not selected for training are used for the evaluation of the models, which are trained in every bootstrap round, see Fig. 2.6. Using the evaluation results the statistical properties of the bootstrapping process given the hyper-parameters can be computed. Applied in multiple rounds of bootstrapping using different sets of hyper-parameters, one may for example prefer a hyper-parameter set with a sufficient good averaged accuracy but also a relatively low variance. These properties, for example the mean and variance of accuracy, are meaningful estimates of the performance of the hypotheses given the hyper-parameters when being deployed to real-world applications.

2.3.2.3 Cross-Validation

A major issue of holdout techniques is that only a small part of the data is selected for training, which often results in too conservative hyper-parameter sets causing a to pessimistic bias. In contrast, in k-fold cross-validation (depending on the choice of k) more data can be used for example hyper-parameter selection which may lead to superior bias values compared to holdout techniques, see [48]. However, this

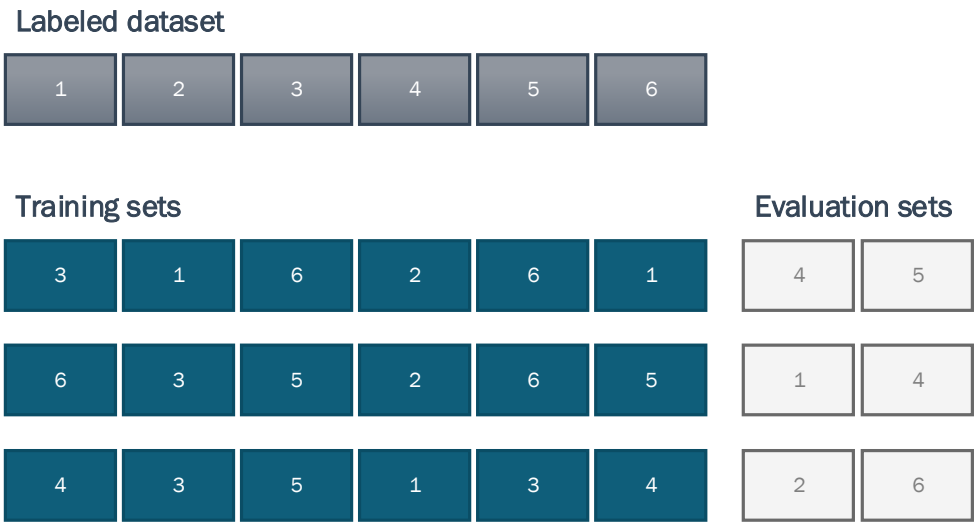


Fig. 2.6: *Within the leave one out bootstrap process for every training run random samples are sampled from the overall dataset.*

advantage comes with a k-times increased training and evaluation time. To perform k-fold cross-validation, k models with partly overlapping training datasets are trained. Evaluation of the models is then executed on parts of the data, not overlapping with the respective training dataset, see Fig. 2.7 for a visual explanation. A second drawback, when k gets close (or even equal in Leave One-Out-Cross-Validation) to the number of n samples available in the dataset, is the increase in variance. The reason why the variance may increase is the similarity of the k training sets, so overfitting cannot be detected in the different folds, see [50]. Summarized one can say large values of k reduce the bias, but increase the variance and also the computational cost.

2.3.3 Evaluation Measures for Discrete Data

Given a trained classifier one is usually interested in measuring its performance. Given a binary decision problem and the knowledge of the labels within a test set, every sample can be be evaluated according to the following scheme called *confusion matrix*. In the following measures are presented which are able to quantify the performance of a classifier, which are based on the values defined in the confusion matrix, see [51] for more details.

2.3.3.1 Rates

The numbers defined in the confusion matrix are heavily used being normalized to the actual class. Therefore, the *true positive rate* also written as TPR and its

Fig. 2.7: *Within the Leave One Out Cross-validation method every sample of the original dataset is selected for evaluation. Only one part of the data is selected for evaluation in every iteration. This is the reason why it is equal to k-fold cross-validation with* $k = 1$.

Tab. 2.1: *Confusion matrix: Each column of the matrix represents the instances as the class determined by the classifier, while each row represents the instances as labeled actual class. Using the table,it can be derived whether and how the classifier 'confuses' the two classes.*

	Classified positive	Classified negative
actual positive	true positives (TP)	false negatives (FN)
actual negative	false positives (FP)	true negatives (TN)

analogous terms are defined as follows:

$$TPR = \frac{TP}{TP + FN} \tag{2.10}$$

$$FNR = \frac{FN}{FN + TP} \tag{2.11}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.12}$$

$$TNR = \frac{TN}{TN + FP} \tag{2.13}$$

2.3.3.2 Sensitivity

Sensitivity refers to the classifiers ability to correctly detect positives which are actually positive. The term Sensitivity is also often known as *Recall* and is identical to the TPR.

$$sensitivity = \frac{TP}{TP + FN} = TPR \tag{2.14}$$

$$= \text{probability of a sample classified positive given that it is postive} \tag{2.15}$$

2.3.3.3 Specificity

Specificity describes the ability of the classifier to correctly reject negatives. Specificity of a classifier is the probability of a negative being classified as negative and is identical to the TNR. Mathematically, this can also be written as:

$$specificity = \frac{TN}{TN + FP} = TNR \tag{2.16}$$

$$= \text{probability of a negative classification given a negative sample} \tag{2.17}$$

2.3.3.4 Accuracy

The term accuracy describes the proportion of correct predictions of a classifier vs. all predictions made and therefore describes the 'trueness' of the results. More formally one writes:

$$accuracy = \frac{TN + TP}{TP + TN + FP + TP} \tag{2.18}$$

$$= \text{probability that the classification results are correct} \tag{2.19}$$

2.3.3.5 Precision

The precision value describes the probability that a sample classified as positive is actual positive. Often it is also referred to as the positive predictive value (PPV).

$$precision = \frac{TP}{TP + FP} \tag{2.20}$$

$$= \text{probability that positive classifications are correct} \tag{2.21}$$

2.3.3.6 F1 score

The F1 score is defined as the harmonic average of the precision and sensitivity. The F1 score reaches its maximum at 1 and is always positive.

$$F1 = 2 * \frac{precision * sensivity}{precision + sensitivity} \tag{2.22}$$

$$= \text{harmonic average of precision and sensivity} \tag{2.23}$$

2.3.3.7 Balanced Measures

In case of imbalanced classes the measures defined formerly can produce non-intuitive results, where in the worst case a model always providing a prediction of the majority class can result an accuracy of close to 1. To handle such kind of issues there exist measures which normalize the skew between classes. For example the balanced accuracy for a multi-class problem is defined according to [52] as

$$ACC^{bal} = \frac{1}{|N|} \cdot \sum_{n \in N} \frac{TP_n}{P_n} \,. \tag{2.24}$$

where N defines the set of classes and the n a specific class. The balanced precision can be defined accordingly as:

$$precision^{bal} = \frac{TPR}{TPR + FPR} \,, \tag{2.25}$$

and the balanced F_1 measure as:

$$F_1^{bal} = 2 * \frac{precision^{bal} * recall}{precision^{bal} + recall} \,. \tag{2.26}$$

All three measures are independent from class skew and are directly comparable to the results of *accuracy*, *precision* and F_1 of classification problems with balanced class distributions.

2.3.3.8 Receiver Operating Characteristics

The aforementioned performance measure are not measures characterizing the inherent predictive power of a classification algorithm. The numerical values heavily depend on the choice of threshold values, for example of a specific probability for

which a sample is accepted as positive. By varying this threshold value very different values of the aforementioned measures will be the result. This leads to the conclusion that the former defined measures of classifier performance are lacking the invariance against threshold values and are not the best choice when assessing the 'predictive power' of a model. A solution to overcome this issue are the Receiver Operating Characteristics (ROC) of a classifier. The ROC is a plot visualizing the performance of a classifier in a binary decision problem. The curve visualizes the TPR as a function of the FPR, see Fig. 2.8. It is generated by varying a discrimination threshold

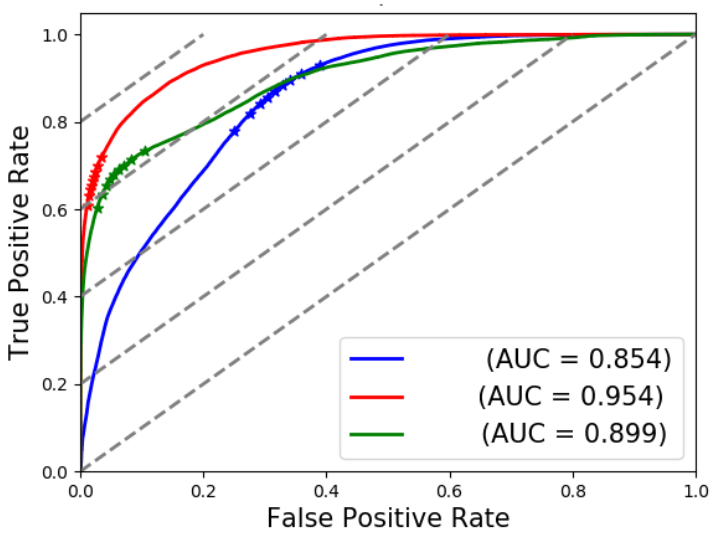


Fig. 2.8: *ROC curves of three classifiers to be compared. The points on the curves denote the thresholds of classifying a value as true in 10% steps.*

T. Given is for example a classification algorithm which outputs class probabilities. To compute the aforementioned measure (for example the true positive rate), one needs to define a threshold $0 \leq T \leq 1$ from which on a classification result is defined 'positive'. Using this definition of T a classifier can be tuned to increase the true positive rate, which sadly also increases the number of false positives and vice versa. Every point of the ROC curve corresponds to a specific value of T. The predictive power of the classifier can therefore be expressed as the so called *Area under the Curve* or *AUC*, which is as its name says the area under the ROC curve. For a detailed explanation of the ROC and efficient computation algorithms to generate the curves please see [53].

2.3.4 Scoring Methods for Continuous Data

The methods presented in the former subsection are tailored for classification problems. In this section measures of performance for continuous values are introduced. For example a rain predictor in the discrete case could predict whether it may rain on the next day, while a predictor for the continuous case would estimate the amount

of water which it expects. It is obvious, that also for the second case one needs methods to judge and to compare the performance of different predictors.

2.3.4.1 Mean Squared and Absolute Error

The most common measures when assessing a model given a dataset y is the Root Mean Squared Error (RMSE). It is defined by:

$$RMSE = \sqrt{\sum_{i=1}^{m}(f(x_i) - y_i)^2} \; . \tag{2.27}$$

Even though, the $RMSE$ is widely used there are some drawbacks using it. It is highly sensitive to outliers and has the property, due to its quadratic term, that a higher variance of the errors results in a higher RMSE. Some researchers therefore prefer to use the Mean Absolute Error (MAE), which is defined by:

$$MAE = \sum_{i=1}^{m} |(f(x_i) - y_i)| \tag{2.28}$$

For a more detailed discussion including the pro and cons of both measures please see [54] and [55].

2.3.4.2 Maximum Likelihood

Given the situation that one has collected data from an unknown distribution, one then often is interested in identifying the distribution that is most likely to have generated this collected data. Using a statistical model class (for example a Gaussian distribution), each distribution is identified by a parameter or parameter set. By varying this parameter(s), different distributions are generated from the family of distributions. The probability density function PDF $f(y|\Theta)$ defines the probability of observing a data vector y given the parameter Θ, where Θ can have n-dimensions. Given a data vector $\mathbf{y} = y_1, y_2, ..., y_m$ and the property of statistically independence of each single observation of y, the probability density function P of the observations can be rewritten by:

$$P(\mathbf{y}|\Theta) = P(y_1|\Theta) * P(y_2|\Theta)...P(y_m|\Theta) \tag{2.29}$$

In many applications $\mathbf{y}$ and the model-class are known, but Θ needs to be estimated. The task is to maximize the likelihood of Θ given the data. Less formally, the task is to find the value of Θ which parametrizes the distribution model that most probably produced the data $\mathbf{y}$. Therefore, the Likelihood function $L(\Theta)$ is defined as

$$L(\Theta|\mathbf{y}) = P(\mathbf{y}|\Theta) \tag{2.30}$$

needs to be maximized to find the Maximum Likelihood Estimate (MLE). The most important difference between the Probability Density Function (PDF) and

the Likelihood function is therefore, that the Likelihood is a function of the model-parameters given the data while the PDF is a function of the data given the model-parameters. The process of Maximum Likelihood estimation in this context is not more than the maximization of the likelihood function by varying Θ. In practical applications this is mostly done by maximizing the Log-Likelihood, because both functions are related monotonically and a maximum of the one function also relates to a maximum of the other. For a more detailed explanation please see [56]. The MLE however can be used to compare the goodness of fit of different models. A practical application for example would be the maximization of the Likelihood for multiple distribution models or hyperparameter sets to find out which model class fits the existing data best.

2.3.4.3 Information measures

Akaike [57] found out that there is a relation between the Kullback Leibler Distance (a measure from information theory) and the Likelihood of a model given the data and model parameters. In his research he presented the Akaike Information Criterion (AIC) which is defined according to [57] as:

$$AIC = -2\log(L(\Theta|\mathbf{y})) + 2K \tag{2.31}$$

where K is the asymptotic bias correction term and represents the number of parameters which were estimated to determine the model. The value of the AIC therefore describes the information loss introduced by the model compared to the data. Please keep in mind that from the perspective of information theory every model is only an approximation of the reality. When used for model fitting, usually one is interested in the model with the least information loss. Especially interesting for practical applications in this context is K, because it penalizes the number of free parameters and thus can also be interpreted as measure to prevent overfitting, see also [58] for more detailed insights. The Bayesian Information Criterion (BIC) introduced by Gideon Schwarz et al. in [59] is not related to information theory, see [58]. It is defined as:

$$BIC = -2\log(L(\Theta|\mathbf{y})) + Klog(N) \tag{2.32}$$

where K is defined accordingly to the AIC and N is the number of data. When comparing the BIC of different models, the one with the lowest BIC is preferable. There is a philosophical debate in the scientific community about which one the two measures is the 'right' one for model selection, see [58]. However, research shows that there are good reasons for both measures to be used. For example [60] states, that the model chosen to be the best in evaluation is often the one which is selected by both metrics. In cases the BIC and AIC pick different models, mostly the model chosen by the BIC tends to have bias, while the model chosen by AIC often suffers from variance.

2.3.5 Supervised Learning

The definition of Supervised learning describes the task of learning a function that calculates output using given input data. The behavior of this function is shaped in the training phase based on example input-output pairs. Therefore, labeled data needs to be available to train a model. Each sample of the training set consists of an input vector and desired output vector. This output can be discrete when doing classification or continuous when doing regression. In the training phase the function is inferred based on the training data. More formally a *supervised learner* $\mathcal{F}$ is a function that maps an input-instance $\boldsymbol{x} \in \mathcal{X}$ to an output instance $\boldsymbol{y} \in \mathcal{Y}$ using internal functions and data structures. Accordingly, one defines a *learning algorithm* $\mathcal{I}$ as an algorithm, which builds a learner from a dataset $\mathcal{D}$. The dataset $\mathcal{D}$ is defined as the space in which the mapping between input and output instances is known:

$$\mathcal{D} = \mathcal{X} \times \mathcal{Y}, \tag{2.33}$$

see [48]. A trained learner is capable to produce a predicted output instance $\boldsymbol{y}_n$ for every new input-instance $\boldsymbol{x}_n$. This prediction is based on the knowledge of the training dataset $\mathcal{D}_t$ and the used induction algorithm $\mathcal{I}_t$. This dependency can be expressed more formally as:

$$y_n = \mathcal{F}(x_n | \mathcal{D}_t, \mathcal{I}_t). \tag{2.34}$$

Different supervised learning techniques use different methods to infer a model based on the training data. The problem of training a model however, can always be understood as an optimization problem. In classification one may minimize the number of wrongly predicted classes. For regression tasks the sum of squared errors between prediction and actual values is a typical measure which one wants to minimize. A very general algorithm for solving such kind of optimization tasks is Gradient Descent.

2.3.5.1 Gradient Descent

One of the most basic methods for training many supervised learners is Gradient Descent. Given is a simple problem of fitting a continuous regression curve to data points x. Using a learning algorithm one likes to generate a model for the distribution from which x was drawn. The system of equations is overdetermined for all cases, in which the number of data points is higher than the degrees of freedom of the polynomial. The problem of fitting the polynomial can therefore be understood as a minimization problem of finding a function minimizing the sum of squared distances between data points and the polynomial one is looking for.

An often used technique to tackle this problem is the method of least squares, which is typically solved using iterative gradient descent methods. The method of Gradient Descent is an important method to train Neural Networks using Back-Propagation and polynomial classifiers. Therefore, in the first step a cost function J which is computed using the N points of the training dataset is defined, which shall

be minimized:

$$\min_{x \in \mathbb{R}^n} J(a_0, a_1 ... a_m) = \sum_{n=0}^{N} (f(x_n) - y_n)^2 \tag{2.35}$$

where a_m denote the m-th parameter of the polynomial f of degree M:

$$f(x) = \sum_{m=0}^{M} a_m x^m \quad . \tag{2.36}$$

Starting in a randomly selected vector $A_0 = \begin{pmatrix} a_0 & a_1 & ... & a_m \end{pmatrix}^T$ the gradient is computed by:

$$d_i = -\nabla J \left(a_0, a_1 ... a_n \right) \tag{2.37}$$

where $\nabla J(A_i)$ determines the gradient of J given the parameters A_i. One then progresses in the direction of the (negative-) descending gradient via:

$$A_{i+1} = A_i + \alpha d_i. \tag{2.38}$$

This procedure is repeated continuously until there is no more improvement in further steps and the algorithm is converged. The learning rate α affects how many iterations are needed in order to converge. A small value of α increases the number of needed iterations and therefore slows down the process of finding a minimum. A large value however means that the algorithm will fluctuate around the minimum of the cost function, which means that the algorithm does not converge. To speed up the performance and to deal with the problem that the naive version of Gradient Descent may converge in local minima, various approaches and more elaborate methods exist, see [61] for an overview. A more general problem of gradient descent is, that it is not able to handle non-convex cost functions. When applied this is typically handled by starting the algorithm multiple times with multiple parameter seeds A_i.

2.3.6 Unsupervised Learning

In contrast to Supervised Learning in Unsupervised Learning no label information is available. Applications are the identification of commonalities and outliers for data analytic applications and density estimation. The hypotheses generated in the training process are trying to model the relationship between multiple observations, e.g. multiple feature-vectors. Typically this is done in a two-step process. First one reduces the dimensionality of the data by applying dimension-reduction algorithms, e.g. Principal Component Analysis(PCA), Linear Discriminant Analysis (LDA) or Locally Linear Embedding, see [45]. Alternatively, also feature-selection techniques can be used instead, see [62]. In the second step one tries to learn the representation of the data. This can incorporate clustering techniques like k-Nearest-Neighbour or density-based clustering algorithms like DBSCAN, see [63]. Other Unsupervised Learning techniques try to learn a representation of the data by modeling it as a probability density function. In the following the most important algorithms applied within in the thesis are presented.

2.3.6.1 K-Means

Given a dataset $\mathcal{D}$ consisting of N data points $\mathbf{x}$, from which it is known that K clusters can be separated (maybe due to being generated by different distributions), one may be interested which data point is generated from which cluster. A cluster in this case can visually be understood as an accumulation of data points whose inter-point distances within the cluster are small compared to distances to data points not belonging to the cluster. Therefore, the learning problem can be formulated as the minimization of inter-point distances within all K clusters. More formally the 'cost' J which shall be minimized can be written by:

$$J = \sum_{k=1}^{K} \sum_{n=1}^{N} I_{k,n} ||\mathbf{x}_n - \mu_k||^2 \tag{2.39}$$

where $I_{k,n}$ is the indicator variable being 1 if a data point $\mathbf{x}_n$ belongs to the k-th cluster and 0 if not. The current cluster center μ_k of each cluster can be calculated using

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} I_{k,n} \mathbf{x}_n \tag{2.40}$$

where N_k denotes the number of data points assigned to the k-th cluster. The calculation of μ_k is sometimes called maximization step and is closely related to the maximization step of the EM-Algorithm, which is explained in a following subsection. Analogous to the EM again in the expectation step the data is now assigned to the newly derived cluster centers μ_k. This is done by calculating the euclidean distance $d_{k,n}$ of each $\mathbf{x}_n$ to each μ_k. Using these distances the indicator variables are recomputed using:

$$I_{k,n} = \begin{cases} 1 & if\ k = \operatorname{argmin} d_{k,n} \\ 0 & else \end{cases} \tag{2.41}$$

For the basic code structure of the algorithm as described by Lloyd (see [64]), please see Alg. 2. The clustering process however is not guaranteed to find the global minimum of J, but can converge in local minima. To overcome this, the algorithm is typically executed multiple times with different initial values for μ_k where the clustering result with the lowest J will be taken as final result. There also exist multiple extensions and tweaks for the K-Means algorithm, see also [65] for an overview. One of the major difficulties of K-Means however is, that the number of clusters has to set as a prior beforehand. To resolve this issues multiple approaches are known, for example by comparing the models using AIC and/or BIC. Another approach popular in real-word applications is the so called 'elbow method', where the decrease of the objective function J is plotted versus the number of clusters. When an increasing number of clusters does not lead to a significantly reduced objective value of the objective function, this number of k where the 'elbow' is, is selected. However, in most cases it is hard to find an unambiguous value for k.

Alg. 2 Execution steps of the K-Means algorithm

```
1: procedure K-MEANS(𝒟, K)
2:     set random initial values for μ_k for all k ∈ K
3:     while true do
4:         compute I_{k,n}^{cur} for all n ∈ N
5:         if I_{k,n}^{cur} == I_{k,n} then          ▷ If cluster assignement did not change
6:             break
7:         end if
8:         compute μ_k for all k ∈ K
9:     end while
10:    return(I_{k,n})
11: end procedure
```

2.3.6.2 EM-Algorithm for Gaussian Mixtures

Gaussian Mixture Model (GMM) are parametric models which can be chosen for modeling continuous multidimensional distributions. A GMM can be intuitively understood as the weighted sum of multiple Gaussians. In most applications, the EM-Algorithm is selected for the training of GMMs. A Gaussian Mixture distribution G is defined by:

$$G \sim \sum_{k=1}^{n} w_k \mathcal{N}(\mu_k, \Sigma_k) \tag{2.42}$$

where w_k is the weight, $\mu_k \in \mathbb{R}$ the mean and $\Sigma_k \in \mathbb{R}$ the covariance matrix of each of the n Gaussians. These parameters will be denoted as Θ in the following sections:

$$\Theta_k = \begin{pmatrix} w_k & \mu_k & \Sigma_k \end{pmatrix} \tag{2.43}$$

with the constraint:

$$\sum w_k = 1 \tag{2.44}$$

While it is now clear how continuous distributions can be modeled using a mixture of Gaussians, one needs a method which is able to fit these distributions to a dataset $\mathcal{D}$. Fitting a Gaussian Mixture to $\mathcal{D}$ is a maximum-likelihood parameter estimation problem. Defining a short written version of the probability density function for Gaussian Mixtures p_{gmm} by:

$$p_{gmm}(\mathbf{x}|\Theta) = \sum_{k=1}^{n} w_k p_{\mathcal{N}}(\mu_k, \Sigma_k) \tag{2.45}$$

the aggregated density for all n samples of x in $\mathcal{D}$ is defined as:

$$p_{gmm}(\mathcal{D}|\Theta) = \prod_{n} p_{gmm}(x|\Theta) \tag{2.46}$$

which therefore defines the likelihood $\mathcal{L}$ of the parameters given the data of a Gaussian mixture

$$\mathcal{L}(\Theta|\mathcal{D}) = p_{gmm}(\mathcal{D}|\Theta) \tag{2.47}$$

To maximize the likelihood $\mathcal{L}$ therefore the parameters Θ have to be varied. Unluckily there exists no analytical expression, helping to solve this maximum-likelihood problem directly in case of a Gaussian Mixture. The steps of the EM Algorithm for Gaussian Mixture distributions can be separated into three main functionalities: initialization, expectation and maximization. In the initialization step all K components are initialized with their mean $\mu_{\mathbf{k}}$, weight w_k and covariance matrices Σ_k. The initialization can be done purely randomized. To speed up the learning process, the initialization with the values computed by k-means is widely used. The expectation step when learning Gaussian Mixtures is the derivation of the responsibility $\gamma_k(\mathbf{x}_i)$ of every component to each data sample i in D. To understand the definition of this responsibility value, one can remember that every component of a Gaussian Mixture with K components has a weight w_k, which can also be interpreted as prior probability. Based on this insight, on can also define the posterior of a data sample $\mathbf{x}$ given a single component of the mixture. This posterior can be interpreted as responsibility $\gamma_k(\mathbf{x})$ of a component k for a data sample $\mathbf{x}$:

$$\gamma_k(\mathbf{x}) = p(k|\mathbf{x}) = \frac{w_k p(\mathbf{x}|k)}{\sum_{j=1}^{K} w_j \mathcal{N}(\mathbf{x}, \mu_j, \Sigma_j)} \tag{2.48}$$

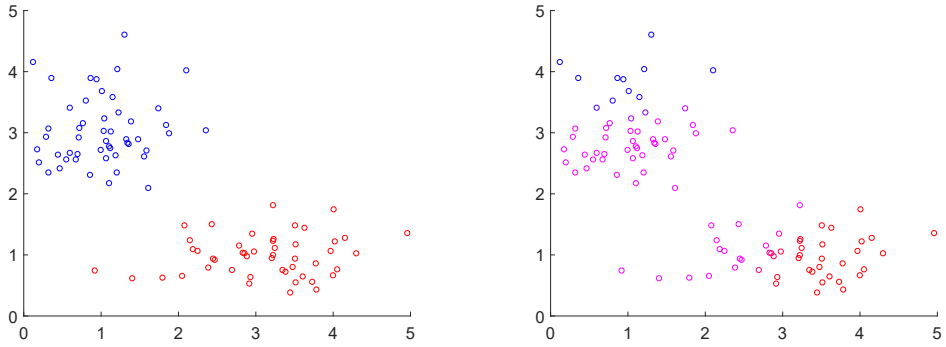
In the maximization-step the new values $\Theta^{updated}$ are derived. This can be done by:

$$\mu_k^{updated} = \frac{1}{N} \sum_{i=1}^{N} \gamma_k(\mathbf{x}_i)\mathbf{x}_i \tag{2.49}$$

$$\Sigma_k^{updated} = \frac{1}{N} \sum_{i=1}^{N} \gamma_k(\mathbf{x}_i)(\mathbf{x}_i - \mu_k^{updated})(\mathbf{x}_i - \mu_k^{updated})^t \tag{2.50}$$

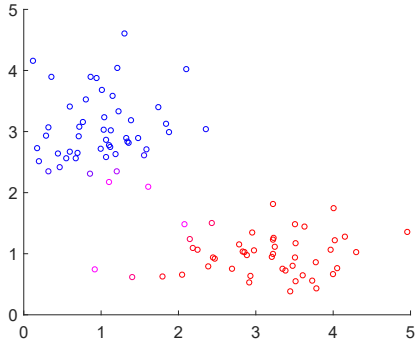
$$w_k^{updated} = \frac{1}{N} \sum_{i=1}^{N} \gamma_k(\mathbf{x}_i) \tag{2.51}$$

Using the newly derived parameters $\Theta^{updated}$, the likelihood of the model given the data can be computed. Now iteratively the expectation and maximization step are recomputed until the value of the likelihood converges within a predefined range or an upper bound of iterations is reached, see also fig. 2.9 for a visualization of the EM converges. In real-world implementations, often both criteria are used in combination.

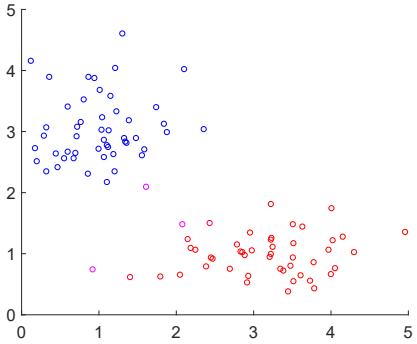


(a) *Samples drawn distribution-wise*

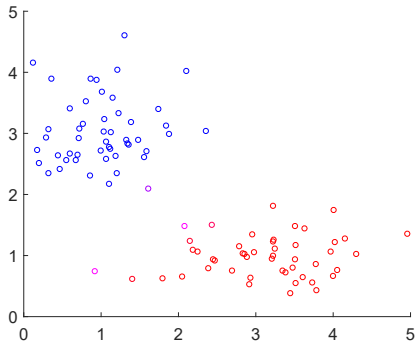
(b) γ_k *after 1st iteration*



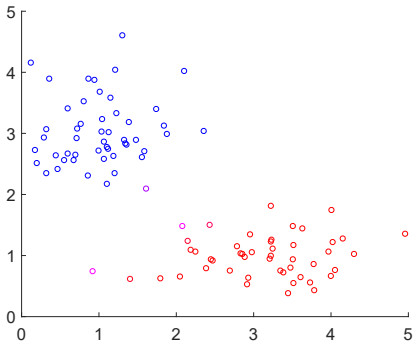
(c) γ_k *after 2nd iteration*



(d) γ_k *after 3rd iteration*



(e) γ_k *after 4th iteration*



(f) γ_k *after 5th iteration*

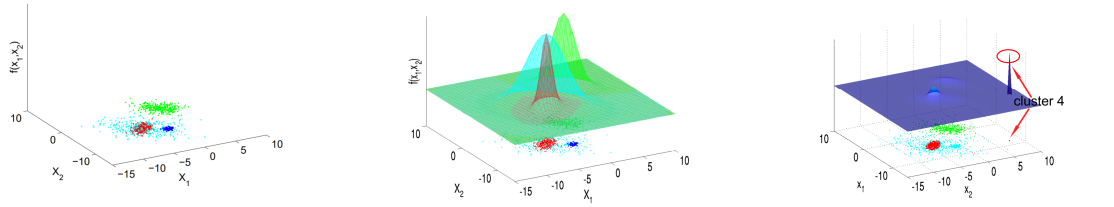
Fig. 2.9: *The figure shows how the EM Algorithm is executed. In subfigure 2.9a you can see the data points generated by random sampling two Gaussians. The color (red and blue) indicates the Gaussian the sample is generated from. In the following subfigures the responsibilities γ for each data point are visualized color-coded. With a randomized initialization the EM Algorithm is executed for the distribution model of a GMM with two components. In subfigure 2.9f the algorithm has almost converged after the fifth step.*

2.3.6.3 Gibbs Sampling for Learning Dirichlet Process Gaussian Mixtures

For the presented approach of learning Gaussian Mixtures using the EM-algorithm a couple of problems arise, see also [66]:

- The 'best' number of components is hard to determine
- When having a non optimal choice of the number of components, the method is prone to overfitting
- The EM-algorithm only converges in local optima

In case of having only one or two closely located outliers beside the main component which can be approximated well with a Gaussian, there is a high probability, that these outliers are modeled as a component with very low variance and therefore very high values of the PDF, which may lead to problems when using the model for applications like Naive-Bayes. To overcome this issues, an alternative way of how to



(a) *Samples drawn by three Gaussian distributions*

(b) *EM solution for two Gaussians*

(c) *EM solution for four Gaussians*

Fig. 2.10: *Based on samples generated in Fig. 2.10a different values of k are used to fit the mixture. While the fit using two components in Fig. 2.10b gives a quite smooth approximation of the distribution, using four components results in overfitting and a peak in the PDF, see Fig. 2.10c. Source of pictures: [67].*

train Gaussian Mixture distributions was developed using a Dirichlet Process, see [66]. The main properties of this approach are:

- The number of components is not predetermined, but is only limited by an upper bound
- The number of components is not a hyper-parameter anymore, it is estimated within the training process
- The approach is based on a Dirichlet Process, which generated as distribution of distributions

To understand how the Dirichlet Process is used for training it is crucial to understand how the Chinese Restaurant Process (CRP), which is closely related to the Dirichlet Process (DP), works.

Chinese Restaurant Process (CRP)
Assuming there is a restaurant with K nonempty and an infinite number of total tables. Each table has an infinite number of seats. Every non-empty table has a number denoted as c_i, which is assigned to it. The number of customers is defined as $N-1$ which will be denoted as $y_1, y_2, ...y_{n-1}$ in the following. A new customer arriving at the restaurant assigned with the index N has now two possibilities:

- Joining one of the already seated $k = 1, ..., K$ tables
- Open a new table k^* which increases K by one

The probability of joining a table is defined within this model according to [68] as

$$p(c_i = k|c_{-i}, \alpha) = \frac{N_{-i,k}}{N + \alpha - 1} \tag{2.52}$$

and the probability of opening a new table (see also [68]) is defined by

$$p(c_i = k^*|c_{-i}, \alpha) = \frac{\alpha}{N + \alpha - 1} \tag{2.53}$$

In this context c_{-i} is the assignment of all guests but y_i and $N_{-i,k}$ the number of all guests (but y_i), which are sitting at table k. The concentration parameter α is used to control the probability of opening new tables. Given for example a constellation of $N = 5$ guests, where the two first guests are seated at table $c_1 = c_2 = 1$ and the two other already seated guests are sitting at table $c_3 = 2$ and $c_4 = 3$. By setting the parameter $\alpha = 1$ the probabilities to join a nonempty table are $p(c_i = 1|c_{-i}, \alpha = 1) = \frac{2}{5}$ and $p(c_i = 2|c_{-i}, \alpha = 1) = \frac{1}{5}$. The probability to open a new table is $p(c_i = k^*|c_{-i}, \alpha = 1) = \frac{1}{5}$. When increasing the value $\alpha = 100$ however the probabilities change significantly. For a nonempty table one calculates $p(c_i = 1|c_{-i}, \alpha = 100) = \frac{2}{104}$ and $p(c_i = 2|c_{-i}, \alpha = 100) = \frac{1}{104}$ and the probability of opening a new one is $p(c_i = k^*|c_{-i}, \alpha = 100) = \frac{100}{104}$. From these observations, the following can be derived:

- As long as $\alpha \gg N$ a lot of new tables are opened
- When $N \approx \alpha$ new guests are assigned to already seated tables. Hereby tables which are 'fuller' are preferred. Murphy describes this in [66] as the 'get richer phenomenon'.

When looking now at the CRP, it is intuitively clear, that this method cannot be used for clustering, because its result only depends on the number of guests on the already seated tables and the parameter α. The individual properties of each data point (=guest) do not influence the result anyhow. Therefore, the *Weighted Chinese Restaurant Process* (WCRP) is introduced. To do so, every of the $k = 1, ..., K$ tables is associated with a parameter Θ_k. Using the already computed prior probabilities $p(c_i = k|c_{-i}, \alpha)$ for every table k and k^*, one needs to calculate how good the new guest does fit to the already seated persons $y_{-i,k}$ at a table k. This value is called *Posterior Predictive* $p(y_{-i,k}, \beta)$ and can be computed according to the math

defined in [68]. The probability of being seated at a table k can be computed by a multiplication of of prior and posterior predictive (see also [68]) by:

$$p(c_i = k|c_{-i}, y, \alpha, \beta) \propto p(c_i = k|c_{-i}, \alpha) \cdot p(y_{-i,k}, \beta) \tag{2.54}$$

Collapsed Gibbs Sampler for DPGMMs
The formulas defined in the last paragraph can be used in a *Collapsed Gibbs Sampler* defined according to [68] for training a Dirichlet Process Gaussian Mixture Model (DPGMM) see Alg. 3. Basically the algorithm works as follows: In every iteration every sample is removed once from its former (randomly assigned) cluster and is reassigned to a new one according to the prior probabilities of the CRP and the posterior predictive probability of the WCRP. However, this assignment is a probabilistic one to the different tables. This results in a discrete distribution over the table indices for every single value of k. For every iteration (each loop of *iterator*)

Alg. 3 Collapsed Gibbs sampler for DPGMM according to [68]

procedure COLLAPSED GIBBS SAMPLER(y, α)
  set random initial values c_i for all data values y_i
  for *iterator* $= 1:1:T$ **do**
    for $i = 1:1:N$ **do**
      Delete the table assignment of y_i
      Update parameters of the table y_i was removed from
      Remove any empty table and decrease K accordingly
      for $k = 1:1:K$ **do**
        $p(c_i = k|c_{-i}, \alpha) = \frac{N_{-i,k}}{N+\alpha-1}$
        $p(y_{i,k}|y_{-i,k}, \beta)$ according to [68]
        $p(c_i = k|c_{-i}, y, \alpha, \beta) \propto p(c_i = k|c_{-i}, \alpha) \cdot p(y_{i,k}|y_{-i,k})$
      end for
      //determine the likelihood and prior in case of opening a new table
      $p(c_i = k^*|c_{-i}, \alpha) = \frac{\alpha}{N+\alpha-1}$
      $p(y_{i,k^*}|\beta)$ according to [68] (set $N = 1$)
      $p(c_i = k^*|c_{-i}, y, \alpha, \beta) \propto p(c_i = k^*|c_{-i}, \alpha) \cdot p(y_{i,k^*}|y_{-i,k^*})$
      Normalize and then sample a new value c_i from $p(c_i|c_{-i}, y, \alpha, \beta)$.
      Use the new value c_i to calculate an update.
      Do $K++$ if a new table is opened
    end for
  end for
end procedure

in Alg. 3 of the algorithm a set of parameters is 'sampled' including the number of k. By repeating this sampling step multiple times a 'smooth' distribution of values over their value range can be generated. From these parameter-values the distribution parameters of a non-infinite GMM can be easily derived.

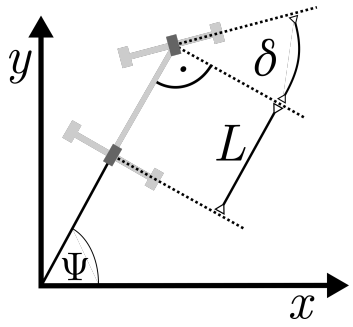


Fig. 2.11: *Parameters of the bicycle model. In dark gray the two wheels of the model are depicted. In bright gray the axis and wheels of a road vehicle with two axis and four wheels, simplified by the bicycle model, are depicted.*

2.4 Vehicle Dynamics

To predict and analyze the behavior of moving objects different kind of models are used. For use cases like tracking of vehicles in sensor fusion systems for example, which have a high update frequency, often simple models are sufficient. Whenever vehicle dynamics are getting more important for the problem to solve, e.g. trajectory planning or even motion control the models applied need to be more realistic and get more sophisticated. An overview of the most important models and their limitations will be given within this section.

2.4.1 Point Models

Given information of the location and the state of an object within a coordinate system the most simple approach to model its movement is by assuming its velocity and acceleration staying constant. This means for every future position $\mathbf{x_f}$ at a time-point t_f in its future the following is assumed:

$$\mathbf{x_f} = \mathbf{x_o} + \mathbf{v_0}(t_f - t_0) + \frac{1}{2}\mathbf{a_0} * (t_f - t_0)^2 \tag{2.55}$$

where t_0 denotes the time-point at which the position $\mathbf{x_0}$ was determined. In many publications the movement hypothesis is called constant acceleration model (CA). The vector of acceleration $\mathbf{a_0}$ is set to 0 in the most simple realizations of this dynamic model, which is sometimes called constant velocity model (CV).

2.4.2 Kinematic Bicycle Model

In the Bicycle Model (often also: Ackermann Model) the two pairs of wheels of a vehicle are lumped together to a single wheel, which is located in the middle of the front, respective the rear axis. The model has two inputs to control its dynamics, the acceleration a and the steering angle δ. The model is defined by:

$$\dot{\boldsymbol{x}}(t) = \begin{pmatrix} a(t) \\ \cos(\Psi(t))v(t) \\ \sin(\Psi(t))v(t) \\ \frac{1}{L}\tan(\delta(t))v(t) \end{pmatrix} \tag{2.56}$$

where

$$\boldsymbol{x}(t) = [v(t), x(t), y(t), \Psi(t)]^T \tag{2.57}$$

$$\boldsymbol{u}(t) = [a(t), \delta(t)]^T \tag{2.58}$$

$$\boldsymbol{y}(t) = [x(t), y(t)] \tag{2.59}$$

as depicted in Fig. 2.11. The vehicles position x,y, its velocity v and the acceleration a are defined with respect to the rear axis of the model. The parameter Ψ denotes the orientation in a global coordinate system, while L defines the distance between the front and the rear axis. The function $\boldsymbol{x}(t)$ denotes the time-dependent vehicle state with respect to the bicycle model, $\boldsymbol{u}(t)$ the time-dependent control input and $\boldsymbol{y}(t)$ the time-dependent position of the vehicle in the Cartesian coordinate system. For convenience within this thesis the function $h(x(t))$ is introduced, which allows the extraction of the position in the Cartesian coordinate system $\boldsymbol{y}(t)$ out of arbitrary $\boldsymbol{x}(t)$.

$$\boldsymbol{y} = h(\boldsymbol{x}) \tag{2.60}$$

However when using the model for control, planning and analysis purposes, one needs to be aware of the limitations of the model simplifications. According to [69] the kinematic bicycle model is a valid choice with only low error for dry road-conditions. Planned trajectories are feasible as long as the lateral acceleration $a_y \lessapprox 0.5g$ and for low friction scenarios as long as the $a_y \lessapprox 0.5\mu g$. When analyzing scenarios where the vehicle is close to the handling limits, a more realistic vehicle model needs to be chosen. For highway scenarios, which are in the focus of this thesis the model provides a good balance between realism and model complexity.

Chapter 3

System Concept and Architecture

The functionality developed in this thesis is tailored to generate safe fallback behavior of automated vehicles. Within this chapter a possible system context is developed in which the identified functions can be embedded. However, the specific functions even if explained here in a top down manner can also be used in different use-cases as defined in this chapter. Please denote, that the structure of this chapter is inspired by the architecture framework ARC42, see [70].

3.1 Introduction and Goals

Within a Level-3 automated system for highway scenarios, see also Sec. 1.1, a fallback functionality implementing a fallback behavior is needed in case a driver does not take back control or the sensed data cannot be trusted anymore. In order to implement such kind of system the following issues need to be considered:

- Future positions of other vehicles surrounding the system-vehicle need to be anticipated where eratic driver behavior shall be modeled, too
- Based on the dynamic information and the predicted future positions of other vehicles the system-vehicle needs to be able to plan feasible trajectories
- The computational effort needs to be reasonable and no additional hardware shall be needed compared with the hardware a Level-3 system is already equipped with
- The system shall be real-time capable
- The system shall show deterministic behavior
- The system behavior shall feel 'natural' to the driver in the vehicle

3.1.1 Requirements Overview

The main purpose of the functionality developed within this thesis is the prediction of other traffic participants and using this information to plan kinematic and dynamic

feasible trajectories which can be executed by an automated vehicle. The research target is to evaluate which information available in a traffic scene can be used best to infer the (future) behavior of other traffic participants, while keeping computing effort reasonable. The main functions to be provided are:

- The extraction of measures possibly influencing the behavior of every vehicle to be predicted in the traffic scene
- A methodology to select the most relevant features from this list of overall measures
- A real-time capable algorithm which induces the current maneuver from these most relevant features
- A real-time algorithm which is able to predict future positions using the features and the maneuver class while handling uncertainty
- A real-time algorithm which is able to plan feasible and smooth trajectories for the system-vehicle while considering the prediction information expressing the future positions of other vehicles

Regarding the interfaces between the different functions, which implement the aforementioned requirements, there is a special focus on keeping the information flow linear. This means for example no feedback loop using the planned trajectory as input for vehicle prediction purposes may be used. The combination of functions is supposed to enable a Level-3 system to plan a behavior implementing the most safe and comfortable trajectory when a fallback behavior is needed.

3.1.2 Quality Goals

In order to be usable in safety relevant systems, see Sec. 2.1, the functions developed within this thesis need to fulfill several requirements regarding there quality.

Tab. 3.1: *Quality goals of functions to be implemented to fulfill the functional requirements, see Sec. 3.3.*

Quality measure	**Explanation**
Testability	The functional blocks shall be testable individually in order to find faults easier and therefore reduce the number of faults of the overall system.
Reusability	The functional blocks to be deployed in the vehicle shall be reusable (with slight adaptations) for different scenarios.
Modularity	The functional blocks shall be usable independently, e.g. a trajectory planner shall not depend on an a specific algorithm solving the prediction task.

3.2 Architecture Constraints

When designing the Level-3 automation function in this thesis, the freedom of design was limited to the design of the subsystem predicting traffic participants and subsystem planning trajectories. Besides this, both subsystems are embedded in a real-time, safety relevant real-time-system. This puts tough constraints of computation time on the subsystems to be developed.

Tab. 3.2: *Constraints which influence the design of functions within the system.*

Quality measure	**Explanation**
Constant Runtime	The functional blocks to be deployed in the vehicle shall finish their computation in an almost constant time time-budget in order to comply with the underlying software-framework. Additionally the time-budget is highly limited.
Efficiency	The functional blocks shall have minimal requirements towards computational effort and memory consumption, in order to be executed on automotive embedded control units.
Determinism	In order to be able to trace and understand possible unwanted system behavior, non-determinism is not acceptable for a safety relevant system. Therefore all techniques, in which random numbers influence the output of a functional block at runtime, are not usable.

3.3 System Scope and Context

Having functions which predict the behavior of traffic participants and plan trajectories, the question arises how these functions are embedded into the overall automated vehicle. To clarify, which kind of data is needed as input for the system and which are its outputs, the system scope needs to be clear. The overall system is described in a building block view as follows in Fig. 3.1. As documented in this building block view, the Level-3 system needs to react in accordance to the (sensed) environment, a map as an additional model of the world and to the input of the driver. The vehicles reactions however are limited to the putting force on the road by steering, deceleration and acceleration, informing the driver and exterior light output, i.e. flashing its indicators. All functionality within therefore needs to be deployed in the most middle block, the *conditional automated vehicle*. This thesis mainly tackles how the fallback behavior of a conditional automated vehicle can be implemented within its technical context. In order to do so, in a first step the desired behavior of the vehicle needs to be defined. From former studies it is known that

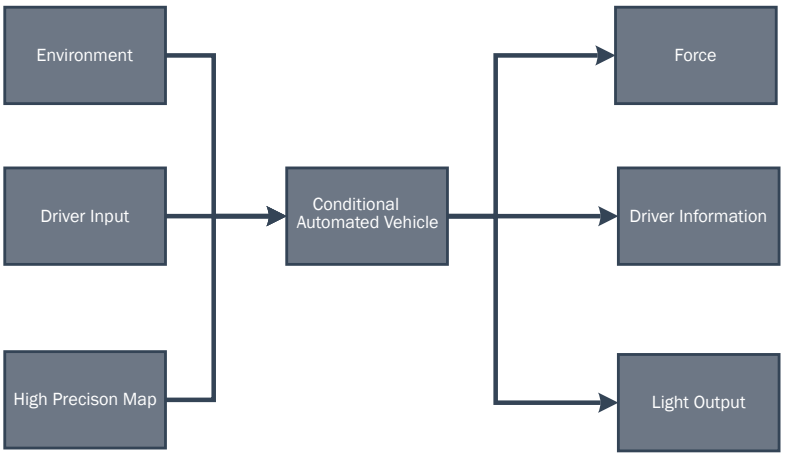


Fig. 3.1: *Building block view of the context in which the conditional automated driving system perceives and acts on architecture level (-1).*

a driver is able to take over control after a few seconds, see for example [14]. This take over time-period is an upper-bound for the duration of the fallback maneuver. However, as a conclusion from the studies, it cannot be guaranteed, that a driver gets back into the loop in-time. Therefore, at the end of the fallback maneuver, the automated vehicle needs to be in a safe state. Assuming the possibility, that a driver does not take over control, therefore the vehicle needs to break down to a standstill. In Fig. 3.2 exemplary scenarios and how the envisioned maneuver by the fallback behavior are defined.

3.3.1 Business Context

The system as described in this thesis needs comply to the requirements of vehicles sold to end-customers. Therefore, the use of expensive sensors or less reliable non-automotive-grade sensor setups and computing units is not possible. For the techniques developed within this thesis, this guides the way into the direction of using simple and computational inexpensive methods.

3.3.2 Architecture Level (0) - Technical Context

The system developed within this thesis assumes a number of challenges on the way to automated driving to be solved. All system inputs representing the environment perception and map depicted on the top right in Fig. 3.3 need to deliver sufficient output quality enabling a Level-3 automation. The system output, e.g. the braking system and the relevant components of the active steering subsystem are assumed to be implemented fail-safe, see Sec. 2.1. This also holds true for the connection in between the subsystems. To achieve this different strategies can be pursued. One

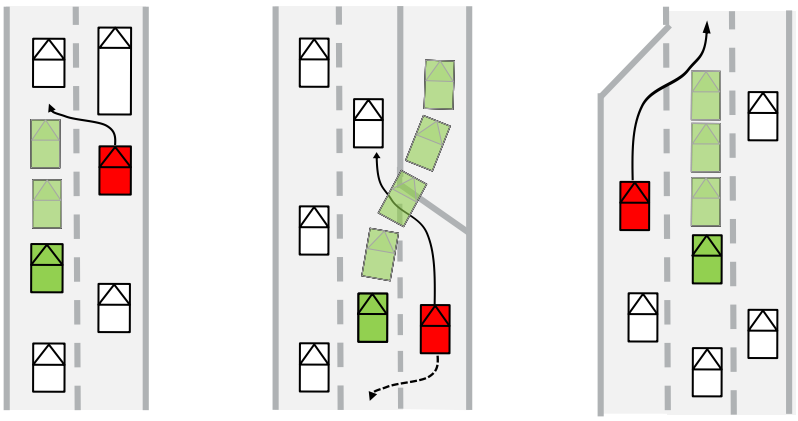


Fig. 3.2: *Fallback Behavior which to be executed by the Conditional Automated Vehicle in case of a system failure. The green vehicle depicts the automated vehicle, the slightly transparent vehicles depict the path it is supposed to take. The red vehicle depicts the most critical vehicle for the designated fallback maneuver. Due to the limited duration of the fallback behavior, multilevel maneuvers cannot be executed. The desired behavior is limited to stopping in the current lane or a lane change to the hard shoulder in case the automated vehicle is already on the most right lane.*

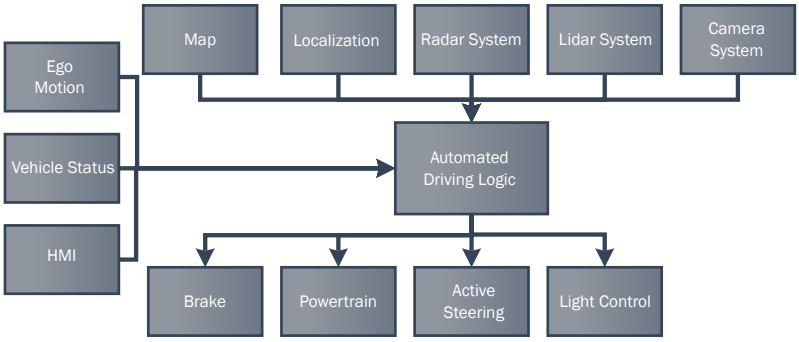


Fig. 3.3: *Architecture level (0) building block view of the conditional automated driving logic within its technical context and neighboring systems.*

possibility is to design relevant parts of the subsystem redundantly, the other one is to have a valid fallback strategy in-place. Both variants however require a monitoring system which is either able to switch to the second redundant component or to trigger a fail-safe strategy. Within this thesis these challenges are out of scope the research presented. Nevertheless, it is important to understand the technical context and the interfaces between the different subsystems of the presented conditional automated system. Therefore, please see the following table, which describes the interfaces depicted in Fig. 3.3:

Tab. 3.3: *Interfaces to neighboring system in the technical context of the Automated Driving Logic on architecture level (0).*

Source	Sink	Explanation
Map	Automated Driving Logic	Data provided here contains at least geometry of roads, lanes, their connectivity and other static entities of the environment as requested by the Automated Driving Logic.
Localization	Automated Driving Logic	Data contains the information enabling a consumer to transform between global (e.g. map coordinates) and local coordinate systems (e.g. data sensed by radar or Lidar systems).
Radar System	Automated Driving Logic	Object representation of the dynamic environment measured by radar sensor(s), where each object contains at least sensor relative information of position, velocity and acceleration.
Lidar System	Automated Driving Logic	Object representation of the environment measured by Lidar sensor(s), where each object contains at least sensor relative information of position, velocity and geometry.
Camera System	Automated Driving Logic	Object representation of the environment measured by camera sensor(s), where each object contains at least sensor relative information of position, velocity, geometry and object class.
Ego Motion	Automated Driving Logic	Information describing the ego motion of the automated vehicle. This contains at least the yaw rate, steering angle, velocity and acceleration.
Vehicle status	Automated Driving Logic	Health state of the vehicle and its systems. In case preconditions in this system are not met, the driver is requested to come back into the loop.
HMI	Automated Driving Logic	Data generated by input of the driver (e.g. activation, deactivation of automation system). Human-Machine Interface (HMI) is also responsible to monitor the awareness of the driver to take over control when needed.
Automated Driving Logic	Brake	Commands used to control the Brake as needed to execute a planned trajectory. The data generated in the block Automated Driving Logic needs to consider the state and capabilities of the brake when sending control signals.

Source	Sink	Explanation
Automated Driving Logic	Powertrain	Commands used to control the Powertrain as needed to execute a planned trajectory. The data generated in the Automated Driving Logic needs to consider the state and the capabilities of the Powertrain when sending control signals.
Automated Driving Logic	Active Steering	Commands used to control the steering system as needed to execute a planned trajectory. The data generated in the Automated Driving Logic needs to consider the state and the capabilities of the steering system when sending control signals.
Automated Driving Logic	Light Control	Commands used to control the light system as needed to inform other road users and to improve the sensing capabilities of the camera and the driver.

To gain a better understanding of how the different functions work together, an overview of the functional chain is given in Fig. 3.4. Within this view, the interaction between the automated driving function, its fallback-behavior and its neighboring systems is defined.

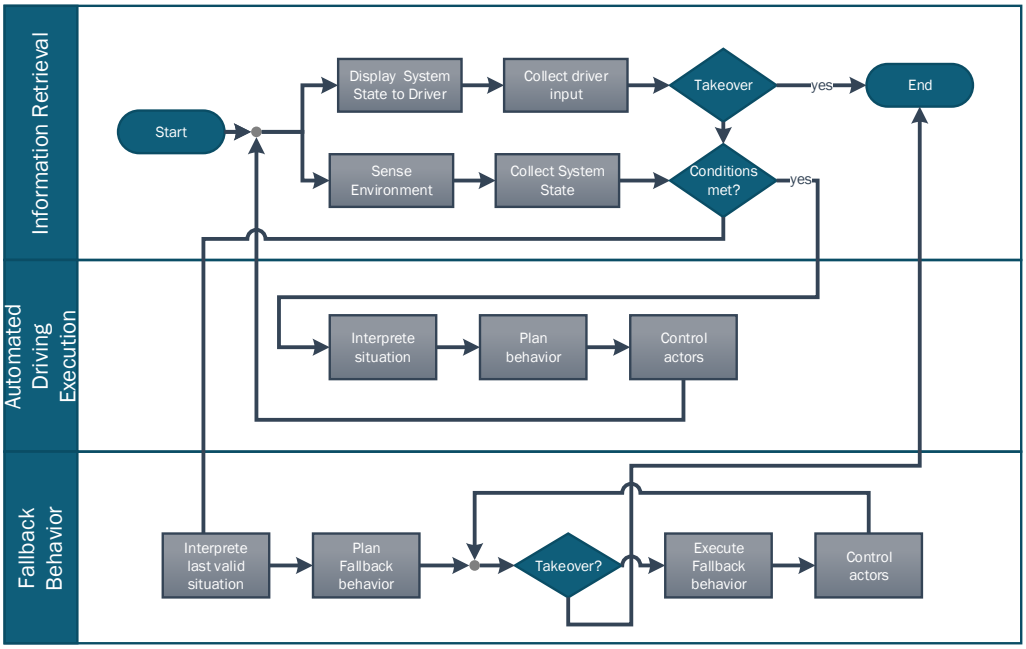


Fig. 3.4: *View on the functional chain of the automated driving logic and its interactions with its neighboring systems and functions.*

3.4 Solution Strategy

In order to implement a fallback behavior for a conditional automated driving system, which considers the quality and architecture constraints described in the former subsections, a number of design decisions were made:

- To increase testability, the planning and prediction system were decoupled. This decision also fosters a more loose coupling of software systems, see Fig. 3.5. This loose coupling also increases the reusability and modularity of the functions, so they can be implemented in different systems and use-cases.

- In order to have reproducible system behavior all methods do not use random number at execution time. This ensures that same numeric input data always results in the same numeric system output.

- In order to have a real-time capable system, all methods need to be executed within a constant (and limited) run-time budget. This decision has an impact on the choice of trajectory planning methods. In many optimization techniques runtime is mainly influenced by initialization values and varies a lot. Therefore it was decided to use sampling based methods for trajectory planning.

- To be deployed in safety critical systems algorithms need to be tested extensively in order to detect bugs and potential critical (sub-) system behavior. In order to handle such kind of problems, black box end-to-end learning approaches (e.g camera input is used as input for a machine learning model which directly generates control outputs for the actors) were not taken into consideration as possible solution for the problem of generating a fail-safe strategy for the conditional automated driving system.

- For the prediction of other traffic participants, the design decision was made to model their behavior in a probabilistic fashion. This decision was driven by the observation, that different drivers act in a different manner. This also results in the decision to use machine learning for modeling driver behavior, lead by the insight, that human driving strategies, technically speaking, are influenced by high dimensional input vectors and cannot be approximated well by low dimensional models like for example constant acceleration.

- The whole prediction system additionally was designed (and is limited to) to handle motorized vehicles in highway (or similar) traffic environments. Therefore the dynamic environment of the system developed in this thesis is limited to be modeled as distinct objects and not for example in a grid.

3.5 Building Block View

This section provides a decomposition of the system *Automated Driving Logic* into its subsystems and interfaces, which is called *Level 1* in the following. A more detailed

white box view of the building blocks is only defined for the subsystem Fallback Behavior Generation which is the context of the contributions within this thesis. This detailing of the subsystem into its components in called *Level 2* view and can be found in Sec. 3.5.2.

3.5.1 Architecture Level (1) - Automated Driving Logic

The white box view of the block *Automated Driving Logic* in Fig. 3.5 contains the building blocks computing the values as needed to comply to the requirements of the output interfaces using the input interfaces. Within this white box it also clarified,

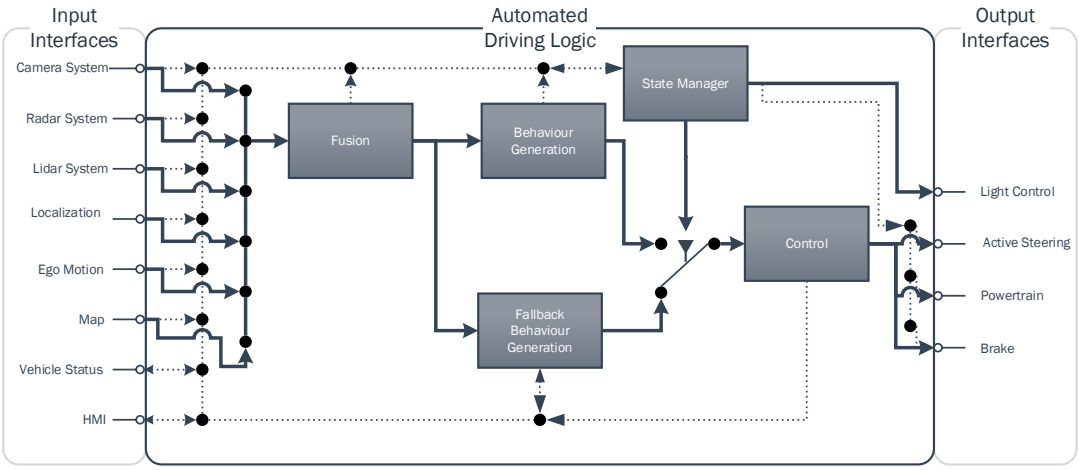


Fig. 3.5: *Whitebox block view of the Automated Driving Logic including the interfaces to the neighboring systems. Dashed lines denote the non-functional status and error information of systems and subsystem. Solid lines represents the functional data needed to compute the output signals for the actors. Based on the state as chosen by the State Manager, the fallback behavior or the standard behavior is forwarded to the Control block.*

which external interfaces are used for the individual blocks. The block State Manager implements the mediator pattern as described in [71]. Depending of the state of the subsystems within the Automated Driving Logic subsystem and the states of the neighboring systems as provided by the Input Interfaces, the State Manager is able to activate Conditional Automated driving. This is done by forwarding the output of the Behavior Generation Subsystem to the Control block. By request of the driver implemented by the external HMI system, the driving automation can be deactivated. In case of a critical fail state of one or multiple sensor subsystems, the data generated by the Fusion module may not be sufficient trustworthy to continue the automated ride. In such cases the driver can be alerted using the HMI and the output of the Fallback Behavior Generation block is forwarded to the Control block. Within this thesis only the block Fallback Behavior Generation and its interfaces will be explained in more detail in the next subsection. Please see the following table for an explanation of the functionality implemented in each block:

Tab. 3.4: *Architecture level (1) subsystems and their responsbility within the system Automated Driving Logic.*

Name	**Responsibility**
Fusion	Fuses dynamic object information as provided by the Camera, Lidar and Radar System into one representation in one ego-centered coordinate system, using Ego Motion information. Based on Localization data and Ego Motion Information, Map data is transformed into the same local coordination system as sensed by the other sensors. The information is then fused with the static object information provided by Camera, Lidar and Radar. The output of the fusion block is an object oriented representation of the static and dynamic world in one coordinate system.
State Manager	Manages the overall conditional automation state of the automated vehicle. Therefore, it consumes the states of all functional relevant neighboring systems and subsystems and implements an overall state logic.
Behavior Generation	Plans feasible trajectories to be executed by the Control Block. Trajectories are supposed to be driveable, collision free, comfortable, and be compliant to traffic rules. Therefore, this module is not only in charge to analyze the current situation but also handle the Dynamic Driving Task (DDT).
Fallback Behavior Generation	Continuously plans a fallback driving maneuver and trajectory for a given output of the Fusion module. The main difference to the Behavior Generation Module is, that replanning is not a solution strategy on how to interact with other traffic participants. Trajectories need also to be driveable, collision free and comply to traffic rules.

Name	Responsibility
Control	Controls the actors, namely brake, powertrain and active steering. To do so it needs to consume the planned trajectory provided by Behavior Generation or Fallback Behavior Generation. In order to compute feasible outputs detailed models describing the dynamic behavior and the limitation of the powertrain, braking system and active steering are considered.

3.5.2 Architecture level (2) - Fallback Behavior Generation

In this subsection a more detailed insight in the structure of the block *Fallback Behavior Generation* will be given. All other blocks defined on architecture level (1) (see Fig. 3.5) will not be described, due to not being part of the contribution of this thesis. The block Fallback Behavior Generation decomposes into three functional

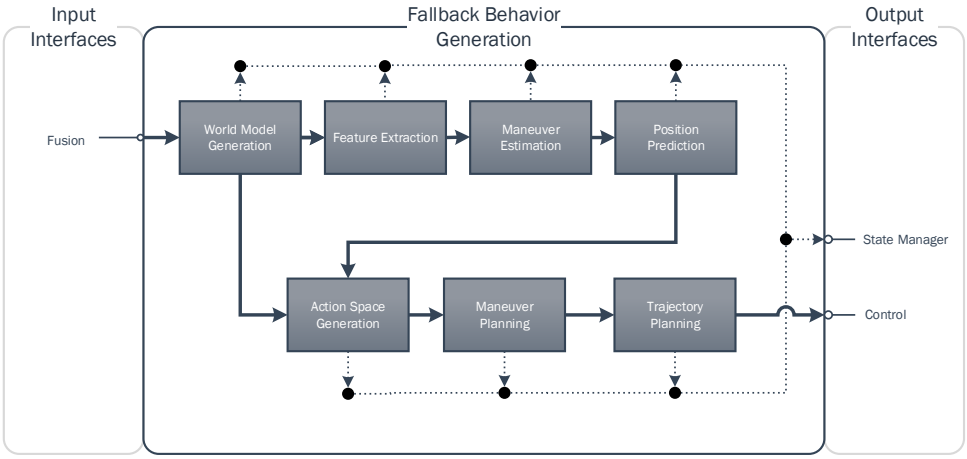


Fig. 3.6: *Architecture level (2) white box block view of the Fallback Behavior Generation block including the interfaces to the neighboring systems. Dashed lines denote the non-functional status and error information. Solid lines represent the functional data needed to compute the output signals for the actors.*

clusters. The first cluster consists of only one block, the World Model Generation. In this block all entities delivered by fusion are set into relation, e.g. vehicles are assigned to lanes. The second cluster consists of the block Feature Extraction, Maneuver Estimation and Position Prediction. Within this cluster the way how others traffic participants will act is estimated. The third cluster consists of the Action Space Generation, the Maneuver Planning and the Trajectory Planning block. It computes the fallback trajectory of the ego vehicle using the information of the two other blocks.

Tab. 3.5: *Architecture level (2) Subsystems and their responsibility within the subsystem Fallback Behavior Generation.*

Name	**Responsibility**
World Model Generation	Generates relations between the different semantic entities of a scene for example lanes, roads, vehicles. Relations may contain multiple measures, e.g. the relative position or the right of way. Please see Sec. 4.5 for more details.
Feature Extraction	Computes the inputs which are needed to estimate the maneuver class and the future trajectories of other traffic participants. Which features are the ones which are relevant is determined beforehand offline. Please see Sec. 4.8.2 and Sec. 4.9.3 for more details.
Maneuver Estimation	Estimates the probabilities of the different maneuvers a vehicle may be executing at the moment. Maneuvers can be for example lane-following or a lane change to the right. Please see Sec. 4.8.3 and Sec. 4.9.4 for more details.
Position Prediction	Based on the estimated maneuvers, the kinematic state vector of a vehicle and its relations to the environment future positions as a function of prediction time are estimated within this block. Please see chapter 5 for more details.
Action Space Generation	Based on the environment and the information generated out of the Position Prediction block a model of the dynamic freespace is generated here. Action Space in this context is defined as a kind of gap on a lane as a function of time. Please see Sec. 6.6 for a more detailed explanation.
Maneuver Planning	Based on the dynamic freespace model generated in the former block the possible maneuvers are estimated. The output is a sequence of $1..n$ action spaces. Please see Sec. 6.6.
Trajectory Planning	Out of the sequence of action spaces defined by Maneuver Planning, in this block a controllable collision-free trajectory is computed which is then forwarded to Motion Control. Please see chapter 6.

The main contributions of this thesis are within the building block Feature

Extraction, Maneuver Estimation (see chapter 4), Position Prediction (see chapter 5)and Trajectory Planning, see chapter 6. A brief sketch how the building block Action Space Generation can be implemented is described in chapter 6. The functin of the block Maneuver Plannings is limited to two one-step maneuvers, namely braking in the own lane or performing a lane change to the hard shoulder, see Fig. 3.2. Due to this limitation, it is not described in more detail in this thesis. A more detailed view on the respective functions can be found in the respective chapters.

3.6 Runtime View

The runtime behavior of every component of the described conditional automated system when applied in a product is determined by the underlying architecture of AUTOSAR, see [72]. AUTOSAR in this context is a standard defined to achieve a standardized architecture for automotive computing units by various OEMs and suppliers. Therefore, in this section only a brief insight in the system interactions on architecture level (1) will be given. An overview how the conditional automated system is activated and how the interactions work in a fallback scenario are depicted in Fig. 3.7 and Fig. 3.8. In Fig. 3.7 the signals which need to be available in order to

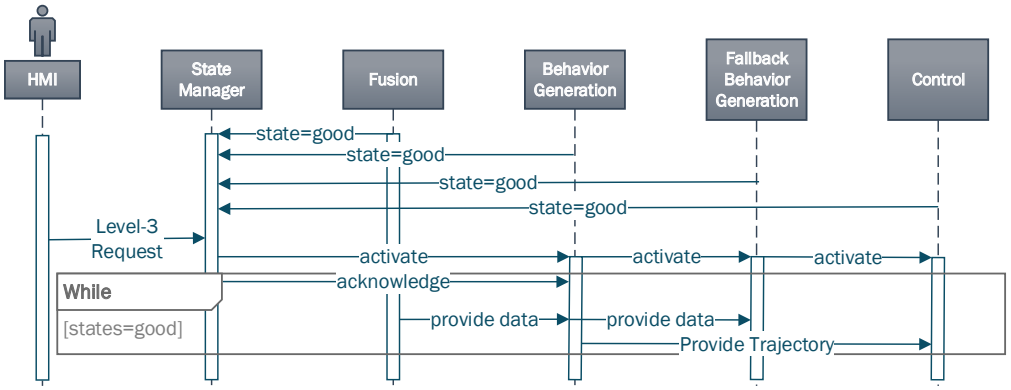


Fig. 3.7: *Simplified view of activation of conditional automated system upon request of Driver. To increase understandability only the most relevant interactions are depicted.*

activate the automated driving are documented. It is important to understand, that in every cycle both, the Behavior Generation and the Fallback Behavior Generation block compute a trajectory, but only the trajectory of the non-fallback module is forwarded to the Control block.

The reason to do so gets more clear when looking at the sequence diagram, describing the interaction in case the Fallback Performance needs to be activated, see Fig. 3.8. If this is the case, no valid information is available anymore from the

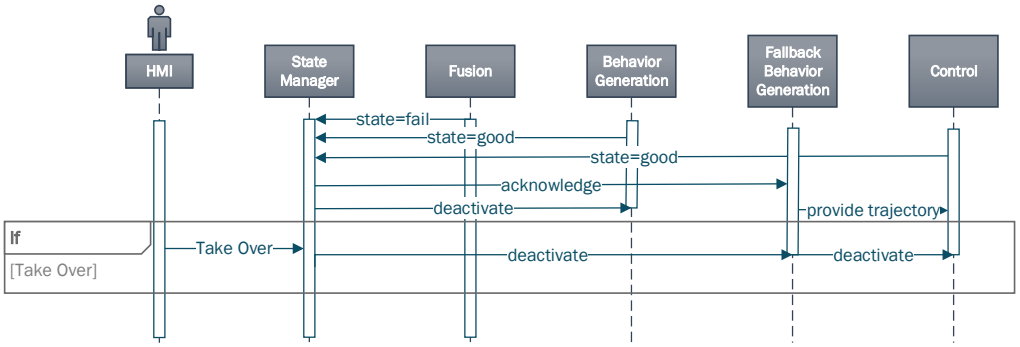


Fig. 3.8: *Simplified view of activation of the fallback performance of the conditional automated system upon failure of the Fusion module. To increase understandability only the most relevant interactions are depicted.*

Fusion block and no updated trajectory can be computed. Therefore, the Fallback Behavior Generation module relies on the last valid information available, which is typically not older than $20ms - 100ms$. In case the need for the fallback trajectory is detected, the trajectory is selected as input to the Control block.

3.7 Risks and Technical Debts

In the following section, a brief overview on the technical risks is given. These risks result from the technical design defined in the former sections.

The first risk for the reusability of the presented approach is the missing interaction between the prediction and the planning part. Not explicitly modeling this interaction is a valid approach when planning a single level maneuver like a fallback trajectory, which typically takes not more time than a few seconds. The simplifying assumption may not hold true when exploring multilevel maneuvers like an overtaking. In such cases interactions between different participant may play a larger role compared to fallback maneuvers. Trajectories planned and executed by the ego vehicle strongly influence the maneuvers executed by other traffic participants, which again has an impact on the planning of the system vehicle. In case one wants to reuse the existing functionality, the architecture needs to be changed and the Behavior Planning module needs an interface to request object behavior for multiple ego-trajectory hypothesis.

The second risk is the limitation of the whole approach to a 2D representation. This limits the reusability of functions and architecture especially in city scenarios where steep roads may exist. To deal with a full 3D representation, the features space and the dimensionality of the position prediction output needs to be increased. For the trajectory planning part presented in chapter 6, the simple 2D approach presented in this thesis may be sufficient to be used for strategic and tactical planning. However, for the trajectory planning part new functionality needs to be developed

when such use cases are present.

The third identified risk deals with the generalization of machine-learning techniques. The prediction system may produce non-feasible results when trying to handle situations which are not (or only in a very homeopathic number) available in the training datasets collected in real world. There are multiple measures to handle this. One would be to add synthetic data generated by simulated scenarios. To do so, the situation space to be handled by the system needs to be defined beforehand. Data can be then extracted from the simulations and added to the training data. Another option is to compute the confidence of the methods for every prediction. Confidence in this context can be intuitively be understood by how similar the data is compared to data seen in the training phase of the model. Whenever this value is below a threshold to be defined, the Behavior Generation may be deactivated by the State Manager, and the fallback behavior planned in the last cycle can be executed.

Chapter 4

Maneuver Recognition

Observations of traffic participants and the environment enable humans to drive road-vehicles safely. However, when being driven by a second person, one recognizes differences between a non-experienced in contrast to an experienced driver. One may get the feeling, that the latter one anticipates what may happen in the next few moments and considers this foresight in his driving behavior.

According to the system architecture defined in Sec. 3.5.2 a functional block to implement such kind of skill needs to be available in a Level-3 automated system. Besides the data-flow depicted in the system architecture in this thesis, there are several use-cases where such kind of information is useful. Be it risk estimation, object selection for adaptive cruise control, behavior planning purposes or as input of motion prediction algorithms.

The chapter starts with the definition of the problem to be solved in Sec. 4.1. Related to the problem definition an overview on related work is presented in Sec. 4.2. Referencing the related methods, the contribution of this chapter is presented in Sec. 4.3. To provide the reader with a high-level overview of the method, the architecture and design decisions envisioned to solve this problem are detailed in Sec. 4.4. As the basis for all investigations in this chapter, the concept of the environment model is presented in Sec. 4.5. In order to be able to select the most meaningful features, the techniques applied for feature selection are explained in Sec. 4.6. Accordingly, the methods to generate classification models are presented in Sec. 4.7. Two experiments attacking this problem are presented in this chapter. The first one in Sec. 4.8 applies a heuristic feature selection strategy to solve the defined problem. Within the first experiment models are generated using a small dataset. The evaluated models were selected with the focus of having minimal requirements regarding runtime and memory. The second experiment in Sec. 4.9 focuses on maximizing the performance for a prediction horizon of $5.0\,s$ using a large dataset while still maintaining acceptable requirements regarding memory and runtime budget on automotive hardware.

The work presented in this chapter is mainly based on already published work in [73]–[76]. The references are documented also in the introduction of the respective sections.

4.1 Problem Definition

Lane change recognition or prediction has already been investigated in various publications, see Sec. 4.2. Many of these approaches only focus on the motion-state-vector of the vehicle to be predicted and only use local information like the distance to a marking to predict lane changes. However, global information clearly influences the behavior of human drivers. Such kind of information can be for example the relation to the vehicle in front, the traffic flow or the distance the next exit ramp. Such kind of information is not in considered in most approaches. Therefore, the problem to be

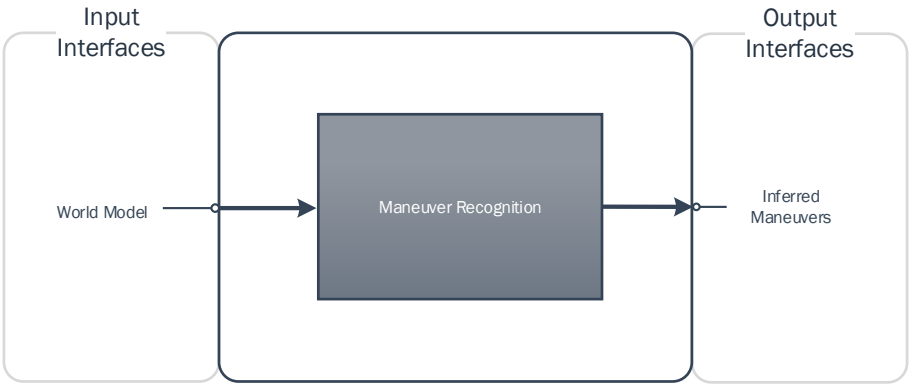


Fig. 4.1: *Architecture level (3) black box diagram depicting the in and output interfaces of the functions to be defined in the context of maneuver recognition.*

solved is the generation of a classification method, which is capable to estimate the probability of a maneuver execution by a vehicle based on all information available and measurable in a traffic scene. Within the investigations also answers to the following questions are presented.

- How early and precise can a lane change be detected based on all the information available in a typical traffic scene?
- Which subset of all conceivable features shows the best trade-off between classification performance and a desirable small number of features?

Due to the fact that vehicles follow the lane most of the time, the classification problem is also highly unbalanced. In comparison to the time vehicles follow the lane, the time period in which a lane change is executed is pretty short. For the definition of a lane change time point please see Fig. 4.2.

4.2 Literature

The term 'recognition of driving maneuvers' is strongly linked to terms like prediction, intention recognition and situation assessment. In this section a short overview of approaches attacking the problem of inferring the executed maneuver of traffic participants is given. In [22], a SVM is used by Kumar et al. with a feature vector

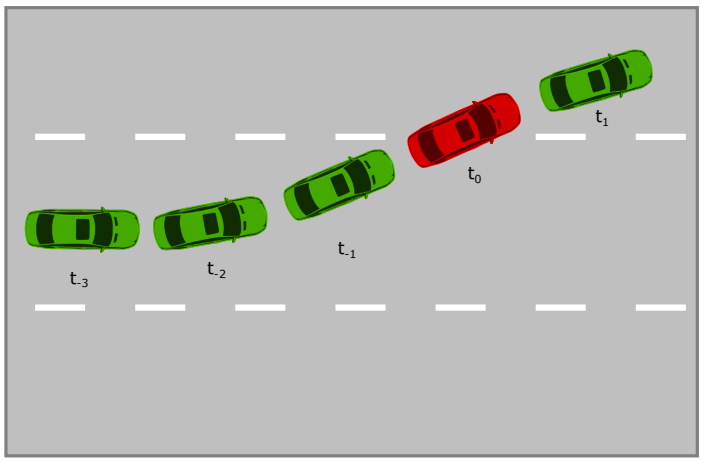


Fig. 4.2: *The time point of a lane change is defined as the point, at which the middle of a vehicle crosses the middle of the lane marking to the neighboring lane. Within the figure this is pictured by the red colored vehicle at the lane change time point* t_0.

consisting of the lateral relative position of a car in the lane, the steering angle relative to the road curvature and the first derivatives of these two features. By using a Bradley Terry Model, a probabilistic output is generated from the result of the SVM. This probabilistic output is then processed by a Bayesian filter for the final lane change intention prediction. By using this filter the precision of the classification algorithm is significantly improved. In [77], Feed Forward Neural Networks FFNN, Recurrent Neural Networks RNN and SVMs are compared by Dogan et al.. The research uses different combinations of features consisting of lateral relations to the corresponding lane, steering angle, the Time To Collision Time to collision (TTC) to the preceding car and the curvature of the road. The evaluation of the method demonstrates, that the SVM achieved the best results followed by the Recurrent Neural Networks (RNN). Furthermore, the usage of the TTC to the car in front reduced the false positive and false negative rate and increased the prediction time. In [23], an object oriented Bayesian network is used for the recognition of lane changes by Kasper et al.. The feature set consists of the movement in relation to the assigned lane and a free-space representation. This approach predicts a lane change 0.6 s earlier than a standard adaptive cruise control system. In [24], a Case-based reasoning approach is used by Graf et al. to detect cut-in maneuvers of vehicles. The approach uses a feature set consisting of the relative distance and velocity between the vehicle in front and the system vehicle. The final features were chosen using temporal abstraction and consisted of trend and level information of the relative distance and velocity. The evaluation shows that the approach has the ability to detect lane change maneuvers until 2.3 s before a car is in the target lane with a percentage of correct classifications of 79 %. Weidl et al. introduces [78] a system being capable of detecting lane changes with high accuracies (>99 %), approximately 1 s before their occurrence. For this purpose, dynamic Bayesian networks are used. The method in [79] presented by Wissing et al., is capable of detecting lane changes

approximately 1.5 s before their occurrence. To do so the lane change probability is determined using a situation- and a movement-based component, where an F_1-score of more than 98 % is achieved. In the research [80] of Bahram et al. an interaction-aware heuristic model is combined with an interaction unaware learned model. The heuristic model, which is based on game theory, implements a classifier based on Bayesian networks. In the evaluation the method demonstrated its ability to detect lane changes on average 1.8 s in advance, with an AUC better than 93 %.

4.3 Contribution

The contribution of this chapter is a strategy how the problem of maneuver recognition can be solved in structured traffic environments systematically. This strategy enables the development of models 'understanding' traffic situations. Thus, the developed models are able to detect intentions of other road vehicles several seconds in advance.

The key contribution within the overall strategy is the development of a scheme how the superset of potential relevant features can be derived systematically. The presented approach in this aspect differs strongly to prior work, in which features are mostly selected based on 'expert knowledge'. The superset of features derived by the presented method contains on the one hand a representation of the traffic situation and the relations to potentially interacting vehicles. On the other hand the superset contains information of the static environment and its semantics. Using state of the art techniques for feature selection, the most relevant features were selected for the respective classifications methods. This investigation provides the second contribution, which is the identification of features which are relevant to determine whether a vehicle may execute a lane change. The third contribution of this chapter are the developed models for lane change intention recognition. The evaluation shows, that the application of the proposed strategy, hand in hand with the application of lightweight machine learning models, provides superior results compared to prior work.

4.4 Solution Design

To solve the problem outlined in Sec. 4.1 a design tailored to the problem domain and complying to the black box architecture defined in Fig. 5.1 is presented in this section. The problem is a probabilistic classification problem which can be treated as black box problem by machine learning techniques. For the work presented in this chapter, four design decisions are documented in the following, which provide guidance regarding methods and experiments.

Accordingly, the first design decision is not to limit the number of features beforehand. Similar to human stimulus processing, in a first step the environment shall be sensed and a superset of possible relevant features shall be extracted. As in popular models of human perception chains (e.g. [81]), the information, which is important has to be selected in a second step. In machine learning, this can

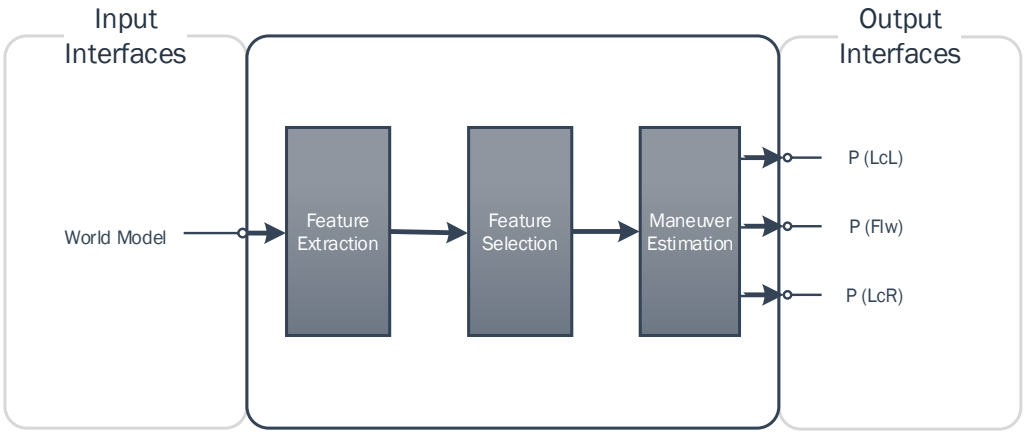


Fig. 4.3: *White box view on architecture level (3) on the maneuver recognition approach presented in this thesis, see also Fig. 3.6 for the embedding in the system context.*

be implemented by feature selection techniques. Selecting only the meaningful features typically does not only improve the prediction performance of a classification algorithm, it also reduces the complexity of the model to be trained. This property helps to reduce the runtime which is important for the envisioned use case.

The second design decision is to use the Markov assumption for determining maneuver probabilities. This means for the case of maneuver recognition, that the probability of executing a maneuver at the next time-point in future only depends on the maneuver executed at the current time-point and the current situation. This explicitly excludes, that the sequence of events which preceded the current situation needs to be considered.

The third design decision is to limit the maneuvers to be recognized to lane change to the right (LcR), lane change to the left (LcL) and lane following (Flw). This decision is based on the envisioned highway use-case. All maneuvers in the lateral domain (e.g. overtaking) can be modeled as a sequence of these basis maneuvers.

Finally, the fourth design decision limits the complexity of the environment model defined for the investigations. As comparably cheap series and close to automotive series sensors are used, the range of sight for which features can be derived is limited. Thus, all features which are based on relations to other vehicles are only derived for direct neighboring vehicles, see Sec. 4.5.

The design decisions are reflected in the white box building block view, depicting the design on architecture level (3) in Fig. 4.3.

4.5 Environment Model

When approaching the problem of predicting traffic participants, the question arises: *What are the relevant features, which have a major influence on the behavior of traffic participants in highway-scenarios?* To attack this issue a method to systematically decompose the traffic environment of vehicles in highway scenarios is presented here. The approach distinguishes mainly between three classes of relations, see Fig. 4.4.

R_{infra} includes all features to the infrastructure (e.g. distance to next ramp), $R_{vehicle}$ describes the relations to the vehicles surrounding the vehicle of interest. R_o includes the features which relate to the vehicles own status or the relations to the lane its located in, e.g. the lateral distance to the center of the lane. Please note, that the

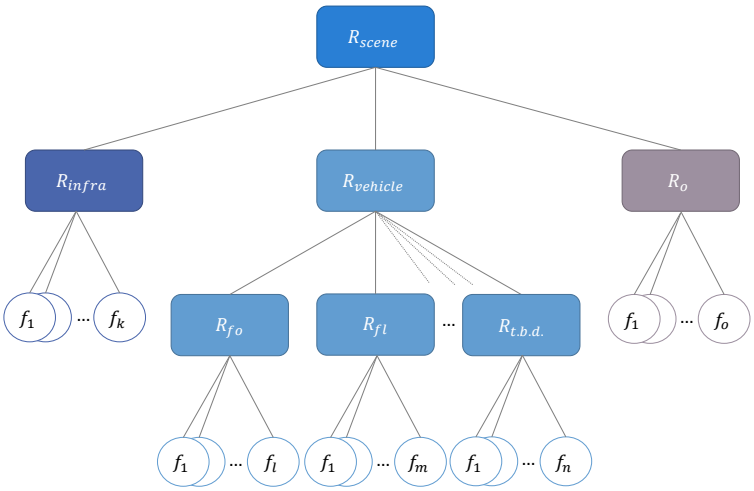


Fig. 4.4: *Feature hierarchy used for the derivation of features for predicting vehicles in highway scenarios.*

relation $R_{vehicle}$ is decomposed into different subclasses including the features to each relating vehicle taken into account for prediction, see Fig. 4.5. When using the technique for the feature derivation as described in Fig. 4.4 an unbounded number of relations to vehicles surrounding the vehicle to be predicted can be defined. Within the experiments however, the number of vehicles is bounded to the eight nearest neighboring vehicles, as can be seen in Fig. 4.5. There are two important reasons for this limitation. On the one hand the complexity of the model shall be kept reasonable. In case there is evidence missing information is limiting the prediction performance significantly, the model is extend-able straightforwardly in future investigations. On the other hand the range of the sensors available in the experiments is limited. For example the second predecessor of a vehicle cannot be measured and tracked with sufficient stability in most cases.

The environment model defined for the derivation of the features is simplified by using a curvilinear coordinate system along the curvature of the road, in which all of the following features f are computed, see section 2.2.2. Using the feature hierarchy of Fig. 4.4 the feature vector F_{sit} is therefore defined as:

$$F_{sit} = \begin{pmatrix} R_{vehicle} & R_o & R_{infra} \end{pmatrix}^\top \tag{4.1}$$

where $R_{vehicle}$ is the concatenation of the relations to the neighboring vehicles,

$$R_{vehicle} = \begin{pmatrix} R_{fl} & R_{fo} & R_{fr} & R_l & R_r & R_{bl} & R_{bo} & R_{br} \end{pmatrix}^\top . \tag{4.2}$$

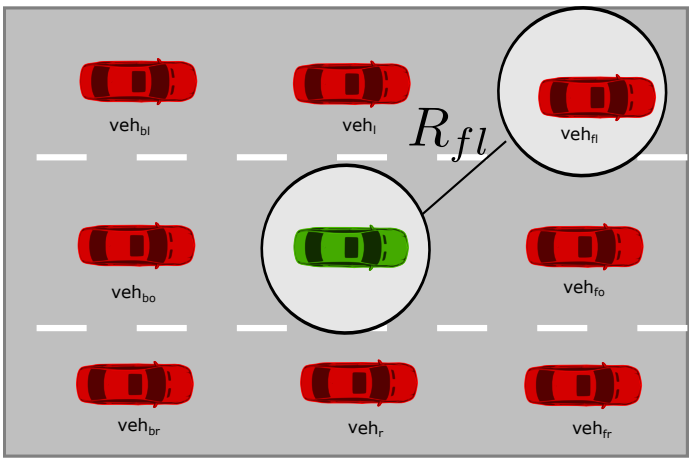


Fig. 4.5: *Relation vectors R_r are generated to the cars which directly surround the observed vehicle o, for which one wants to determine the maneuver probabilities P_{LcL} P_{LcR} and P_{Flw}. In this example the vector R_{fl}, is shown, which is the relation between observed vehicle o and the vehicle positioned relative in (f)ront on the (l)eft lane.*

For the experiment in this chapter it was assumed, that the defined set of features contains all measurable information allowing to infer about the future behavior of other traffic participants. Even if some of those features are highly dependent, the number of features shall not be limited beforehand, but in the feature selection process in the respective experiments.

4.6 Feature Selection Techniques

To improve the prediction performance of the algorithm detecting maneuvers and to reduce the calculation effort for predicting on devices with limited computational power, a subset of features shall be selected from the available superset of possible features. Many approaches how feature selection in machine learning can be done is documented in literature. In the implementation of the different strategies the findings of [62] are used. In the following an overview on filtering and wrapper strategies for feature selection will be given.

4.6.1 Filtering

Filtering methods are one of the most simple way to do feature selection. Instead of using a trained model for the selection of features, heuristics as for example correlation measures are used. Consequently, filtering can be executed faster than wrapper methods and thus the methods scale well even for a lot of features. Used for natural data they can provide good results, see [62]. One distinguishes between methods computing correlation measures, methods computing information theoretic measures, single variable classifiers and learning algorithms as a filter for other learning methods (e.g. tree based methods can be applied for feature selection).

Within this thesis, when using filtering techniques mainly correlation based methods are applied. According to [82] the goal of correlation based feature selection methods can be formulated like this:

> A good feature subset is one that contains features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other.

The weakness of methods purely focused on correlation is on the one hand, that the properties of the classification algorithm are not taken into account. On the other hand correlation based methods are not able to identify meaningful features which according to [62] are: 'completely useless by itself but can provide a significant performance improvement when taken with others.' However, experiments [83] show strong evidence, that on natural data correlation based methods for feature selection improved or at least provided comparable prediction performance when compared to predictors using the full feature-set. A decrease of performance can be expected in cases where features are predictive in only small areas of the available data [82]. One can think of different heuristic strategies to select features based on their correlation with the target value and their feature cross-correlation. A more systematic approach is the Correlation based Feature Selector (CFS) introduced by [83]. For all feature-sets S the 'merit' M_S as a measure of the predictive performance is computed by:

$$M_S = \frac{n\overline{r_{cf}}}{\sqrt{n + n(n-1)\overline{r_{ff}}}} \tag{4.3}$$

where n describes the number of features, $\overline{r_{cf}}$ is the mean correlation of all features with the class label. The variable $\overline{r_{ff}}$ describes the mean feature-feature inter-correlation of all features within S. As can be seen in (4.3), strongly correlated features in a feature-set S will reduce the value of M while a stronger correlation with the class label $\overline{r_{cf}}$ will enlarge it. All these computations rely on the assumption, see also [83], that there are no strong feature interactions present in the dataset, but instead that every relevant feature alone is at least weakly correlated with the class label. To avoid only measuring linear dependencies between a variable and an outcome, as when using Pearsons correlation coefficient, within this chapter rank correlation criteria are used. See [84] for an empirical survey and Fig. 4.6 for a visual explanation why rank correlation are able to provide the more meaningful results.

Another possibility for filtering according to [62] are Single Variable Classifiers. The main idea behind this approach is to measure the predictive performance of each feature by itself by building a classifier for every single feature. Based on a threshold of a performance measure, for example the accuracy, features are selected or not.

4.6.2 Wrapper Techniques for Feature Selection

The most popular wrapper techniques for feature selection are according to [62] *forward selection* and *backward elimination*. Both techniques in general require a higher computational effort compared to filtering methods. While forward selection

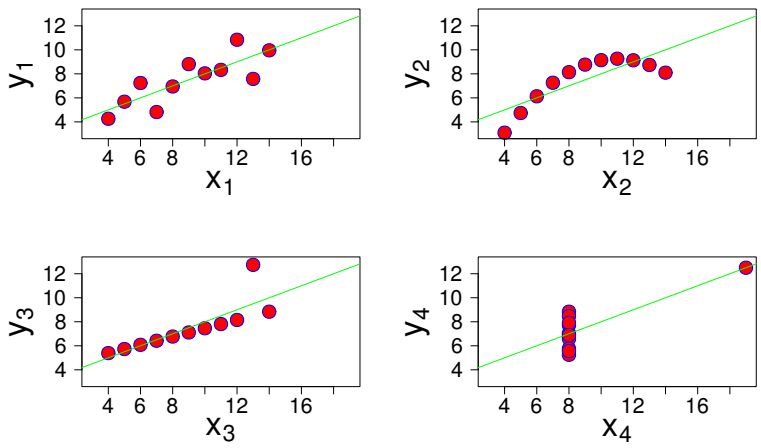


Fig. 4.6: *Anscombe's Quartet shows the weakness of Pearson's linear correlation coefficient by not taking into account non-linear dependencies and being non robust against outliers. All four datasets are constructed in a way, that they own the same value for Pearson's correlation coefficient, even though the dependencies are strongly different.*

techniques stepwise include the most meaningful features into the feature subset, backward elimination starts with the whole featureset and subsequently removes the features having the least contribution to the performance of the model. Speaking from a high-level perspective, forward selection tends to be computational more efficient to generate a set of variables maximizing the predictive power of a classifier. On the other hand backward elimination tends to select stronger sets of variables, mainly because the importance of each variable is investigated considering to possible dependencies to other variables within the feature-set [62]. The main advantage of both wrapper techniques is, that the learner itself is handled as a black box, but its properties are considered implicitly within the learning process. When striving for a really small number of features backward elimination may remove a feature with the best predictive power on its own in favor to a set of variables which together outperform this single feature [62].

4.7 Classification Methods

Several methods and techniques were used to estimate maneuver classes and class probabilities within the experiments of this thesis. In this section a short survey of the methods applied in this chapter is presented.

4.7.1 Naïve Bayes

The Naïve Bayes algorithm is a generative learning algorithm. Its naivety is reasoned in the assumed conditional independence between the different features. While

it has a higher asymptotic error than discriminative models, it approaches its asymptotic error faster than a discriminative classification model when the number of training samples increases [42]. Even if its model assumptions do not hold true, the classification performance on real world applications is often surprisingly good [85]. For a multi-class problem the application of the Naïve Bayes algorithm can be explained as follows: The probability of a sample $Z = f_1, f_2...f_n$ with the features f_i belonging to a class c with $C = \{c_1...c_m\}$ is:

$$p(c|Z) \propto p(Z|c)p(c) \tag{4.4}$$

and

$$p(Z|c) = \prod_{i}^{n} p(f_i|c) \tag{4.5}$$

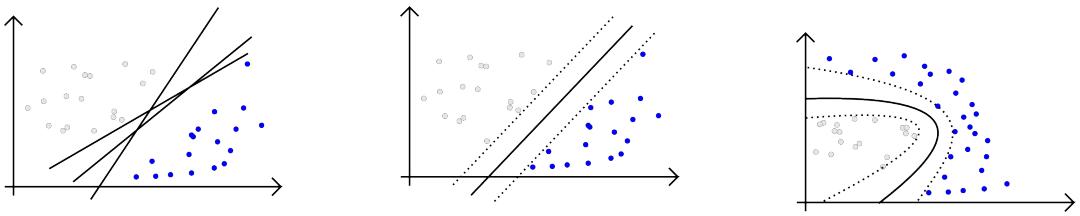
Using these calculations, the probabilities of a sample belonging to the different classes can be determined. When applied as classification technique, different decision threshold strategies can be used to decide for a class to which a sample belongs. Although most implementations of the Naïve Bayes Classification algorithm assume a normal distribution of the variables, this is obviously not a valid assumption in general, thus an appropriate distribution model shall be selected to improve its classification performance.

4.7.2 Support Vector Machines

A popular discriminative classification algorithm which works well even when fed with little data, is the method of Support Vector Machines (SVM). The advantage of the method are the comparable high precision and good generalization abilities on 'realworld data'. When applied within this thesis, the popular LibSVM implementation was chosen, see [86]. The reason for the good performance of SVMs can be seen in the structural risk minimization principle, which also provides an upper bound on the risk of misclassifications. This bound, which implicitly characterizes the generalization ability of the classifier, is determined not only by the training error R_{emp}, but also by the complexity of the model used, [87]:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h\left(\log\frac{2l}{h} + 1\right) - \log\frac{\eta}{4}}{l}} \tag{4.6}$$

Hereby, with $p = 1 - \eta$, the probability of the validity of the upper error barrier can be described, where l denotes the number of training samples. The variable h represents the Vapnik Chervonenkis VC dimension of the function class (in the linear case, for example, in R^n at most $h = n + 1$ points can be arbitrarily separated). As the number of training samples increases, an increasingly complex function class can be chosen, while the risk barrier remains the same. In its basic form, a SVM is a linear classifier for a two-class problem with the labels $y \in \{-1, 1\}$, which maximizes the width of the 'margin' to the boundary points of the data of the respective class,



(a) *The SVM solves the problem of which of the possible separating planes is the 'optimal' one.*

(b) *This is done by selecting the plane with the largest margin to the support points of the samples.*

(c) *By applying the kernel trick also a non-linear separation of the data can be realized.*

Fig. 4.7: *Visualization of the fundamental idea of Support Vector Machines.*

the 'support vectors', see Fig. 4.7. To apply the method on multi-class problems a one vs-all strategy can be implemented. It can be shown that maximizing the width of the margin is equivalent to minimizing the magnitude of the normal vector w, which defines the position of the plane in space. However, for this optimization problem, there are l constraints for all training samples x:

$$y * (\langle w, x \rangle) - b) \geq 1 \tag{4.7}$$

where b is the support vector of the separating plane. If one wants to apply this technique to nonlinear problems, one uses the 'kernel trick', where the scalar product in the above formula may be replaced by a Radial Basis Function (RBF), a polynomial, or the like. This implicitly realizes the scalar product after a transformation into a higher-dimensional space in which data can be separated linearly. In all implementations in this thesis an RBF is applied for doing so. Please find a visual explanation of the method in Fig. 4.7. The additional, free parameter γ which is the free parameter of the RBF, as well as the parameter c, were optimized by means of a 5-fold cross validation when used in this thesis, see the documentation of [86] for an explanation. The parameter c herein characterizes the weighting of misclassifications on the parting line, the so called 'soft margin'.

4.7.3 Random Forests

When classification problems get more complex and an ideal separating plane in a high-dimensional data-space is shaped in a non-trivial fashion, independence between the features cannot be assumed in general. Thus, linear classifiers like the Naïve Bayes may produce non-optimal results. A popular method to attack such kind of non-trivial classification problems are Random Forests [88], which have gained popularity because of their good classification performance even for high dimensional data. The method is an ensemble of several Decision Tree Classifiers. Each Decision Tree of a Random Forest (as shown in Fig. 4.8) splits at each stage all presented samples into two groups based on a threshold of a specific feature. The free parameters of the model to be optimized prior to training of the final models are num_e as the

number of trees and the maximum tree depth max_d. Additional advantages of the Random Forest (RF) method are re their fast learning and prediction process, their easy parallelization capability, their high accuracy, and the property that they are white box models, meaning that each result can be retraced easily, see [89].

4.7.3.1 Training of Random Forests

In the usual training process Decision Trees are stepwise generated as follows: In each step a subset of the training data is randomly selected. For n randomly selected features the best threshold to split all samples in node q into two parts, defined by the lowest Gini impurity $H(X_q^g)$ is calculated by:

$$H(X_q^g) = \sum_m P_{q,m}(1 - P_{q,m}) \tag{4.8}$$

where n is a parameter which is constant for the whole Random Forest and has to be lower than the number of all features. $P_{q,m}$ describes the probability of class m in node q, see (4.9). Out of these thresholds the split with the lowest $H(X_q^g)$ is selected as a new split. This splitting step is repeated until the maximum tree depth is reached or there are not enough remaining samples for a further split, see Fig. 4.8 for an example consisting of two trees. For a more detailed discussion, see [88].

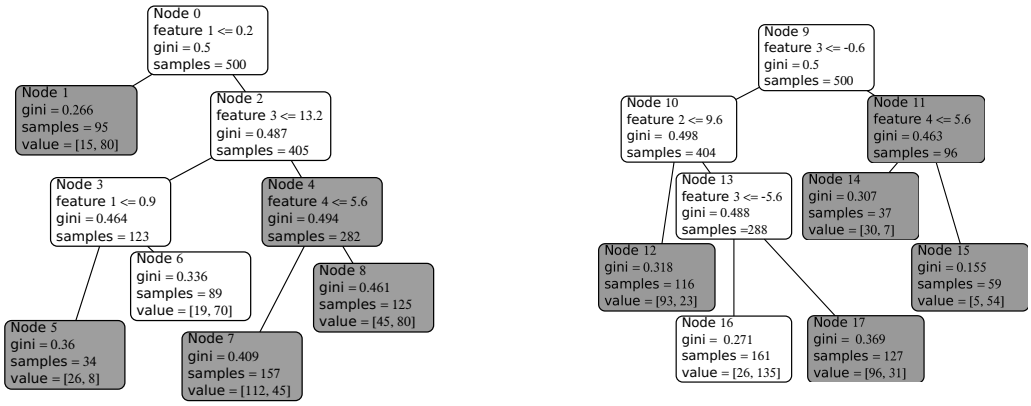


Fig. 4.8: *Simple example of a Random Decision Forest, which is an ensemble of* $num_e = 2$ *decision trees. The decision process for an example dataset* $d = [3.2, 9.7, -7.1, 0.2]$ *is highlighted in white, see [75].*

4.7.3.2 Classification using Random Forests

For each single tree of the Random Forest the probabilistic prediction result of a class m can be determined as the probability of this maneuver in the corresponding leaf node q as $P_{q,m}$, see (4.9). Within this thesis and consistent with [75] the final probability estimates P_m of the Random Forest are computed by computing the average of all trees of the forest, see [89]. This contrasts with the original publication

of Breiman [88], where the predicted class is chosen by the maximum number of votes of the single trees.

$$P_{q,m} = \frac{N_{q,m}}{N_q} = \frac{1}{N_q} * \sum_{x_i \in R_q} I(y_i = m) \tag{4.9}$$

The probability $P_{q,m}$ of a class m in a node q can be calculated as the proportion of $N_{q,m}$ to N_q. $N_{q,m}$ is calculated as the sum of all samples $x_i \in R_q$ with the label y_i which belong to class m. A sample is an element of a data-region R_q if it fulfills all split criteria leading to node q. N_q in this context describes the overall number of samples in node q, see [89]. I denotes the identity function in this context. The probability P_m for a class m can be straightforward calculated by:

$$P_m = \frac{1}{|Q|} \sum_{q \in Q} P_{q,m} \tag{4.10}$$

where Q denotes the set of the leaf nodes for this sample.

4.7.4 Feedforward Neural Networks

While the method of Feed Forward Neural Networks FFNN became popular in the late 1980s, it did not become the dominant technique in many machine learning applications until the 2000s. The advantages of using Neural Networks were identified far before they were widely applied in science and industry. The key advantage according to [90] is, that Feed Forward Neural Networks can be used as 'universal approximators' and that standard multilayer feedforward networks are capable of 'approximating any measurable function to any desired degree of accuracy'. The application of the method for complex problems was however limited by the difficulties in training the networks, see [91]. Especially the huge training effort became feasible to handle with the introduction of CPU/GPU clusters, which accelerated the training process by magnitudes, see [92]. Another issue which limited the training of deep neural networks is the vanishing gradient problem, where in the training changes in the last layers only have a minor impact on the first layers of the neural network. This issue however can be handled by modifying the activation function to the Rectified Linear Activation Function (ReLu), see [93]. The main principle behind the Feedforward Networks however still stays the same.

According to [94] Feed Forward Neural Networks can implement an arbitrary mapping of one vector space into another. A Multi Layer Feedforward Neural Network consists of multiple layers, where the last one is called the *output layer* in the following, and the first one the *input layer*. The one or multiple layers in between are called *hidden layers*, see Fig. 4.9 for a visual explanation. The training of neural networks is usually done by back-propagation. This training method can be divided into two steps. In the forward phase an input vector is fed to the input layer and the corresponding result at the output layer is determined using the current parameters of the network. At the output layer the error to the known desired output is calculated. Using this error value basically a gradient descent algorithm is

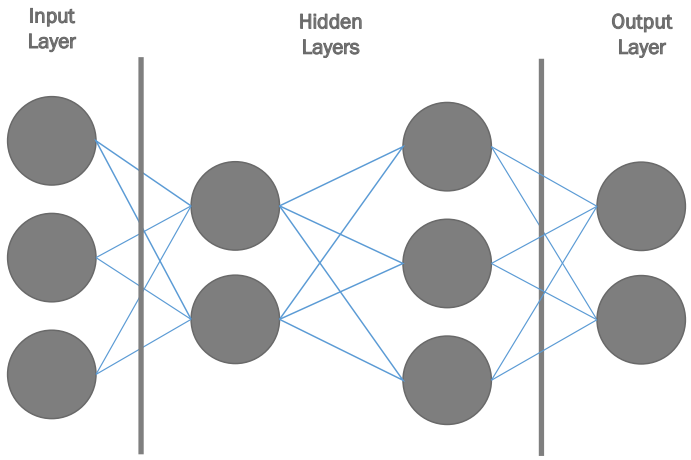


Fig. 4.9: *Example of a Feedforward Neural Network with three input neurons, two output neurons and two hidden layers.*

used, to propagate the error back and to adapt the weights w, where the influence of those parameters is explained in the following. Please see [95] for a tutorial on how backpropagation works in detail. According to [94] the output o_l of a single neuron l is computed by:

$$o_l = f(\xi_l) \tag{4.11}$$

where f is the chosen activation function which was selected, which can be for example a softmax or a sigmoid, see [91] for a survey on possible activation functions. The ξ function for the corresponding neuron can be expressed as:

$$\xi_l = \sum_{k=1}^{n} x_k w_{kl}. \tag{4.12}$$

Herein n is the number of the inputs of the neuron and w_{kl} the weight which is multiplied with each of the inputs x_k. Basically this means that ξ builds a weighted sum of all inputs of the neuron, where the weights w_{kl} are the parameters which need to be determined in the training phase. For a more detailed introduction into Feed-Forward Neural Networks, see [96].

4.7.4.1 Hidden Markov Model and Gaussian Filtering

The presented methods up to here only present 'one shot' predictions for the estimated maneuver class based on a single observation. Depending on the noise of the data and using the knowledge, that the maneuver class executed by a human typically does not change cyclic one may be interested in a method which is able to consider the history of observations. The task therefore is to compute the conditional probability $p(m|Z_{0:t})$, where the index $0:t$ denotes the time period starting at 0 until the time-point t. This conditional probability can be computed using a Hidden Markov Model (HMM) [97], see Fig. 4.10. The model uses the Markov Property, which assumes that all information is aggregated in the current state S and future states

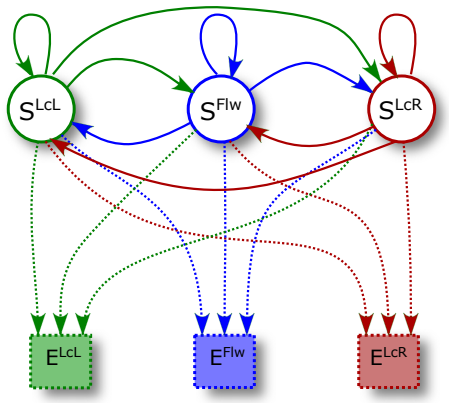


Fig. 4.10: *Hidden Markov Model for the three maneuver classes LcL, Flw and LcR.*

only depend on new observations and the current state. Thus the following holds true (by setting the probabilistic output of a classifier as emissions E of the HMM):

$$E^m = p(m|Z_t) \tag{4.13}$$

The prediction step to evaluate the probability of a state S^k at a time-point t is then defined as:

$$\bar{p}(S_t^k) = \sum_M p(S_t = S^k|S_{t-1} = S^m)p(S_{t-1}^m) \tag{4.14}$$

and the update step:

$$p(S_t^k) = \eta p(E_t^\Sigma|S_t = S^k)\bar{p}(S_t^k) \tag{4.15}$$

with:

$$p(E_t^\Sigma|S_t = S^k) = \sum_M p(S_t = S^k|E_t^m)p(E_t^m) \tag{4.16}$$

The conditional state transition probabilities of (4.14) are stored in the matrix $\boldsymbol{T}$ and the conditional emission probabilities of (4.16) in the matrix $\boldsymbol{E}$. For the proposed approach of Gaussian filtering in [22], $\boldsymbol{E}$ is set to the identity matrix $\boldsymbol{I}$.

4.8 Experiment I

The first experiment focuses on the investigation how good lane changes of surrounding traffic can be recognized with close to series sensors using small models generated by machine learning techniques. To be deployable on typical in-vehicle computing units, these algorithms need to be computational inexpensive. All methods applied in the investigation provide a probabilistic output, which is required by the architecture defined in chapter 3. The work presented in this section was firstly published in [73], [75] and [74].

4.8.1 Setup & Dataset

All data used in the experiment was measured by a system vehicle which was equipped with a front facing stereo camera and two mid/long-range sensors the front and back, see Fig. 4.11 for a visualization of the sensor setup. Furthermore, as described in Sec. 2.2.2 the environment model is simplified by using a curvilinear coordinate system along the curvature of the road, in which all features were computed. The data

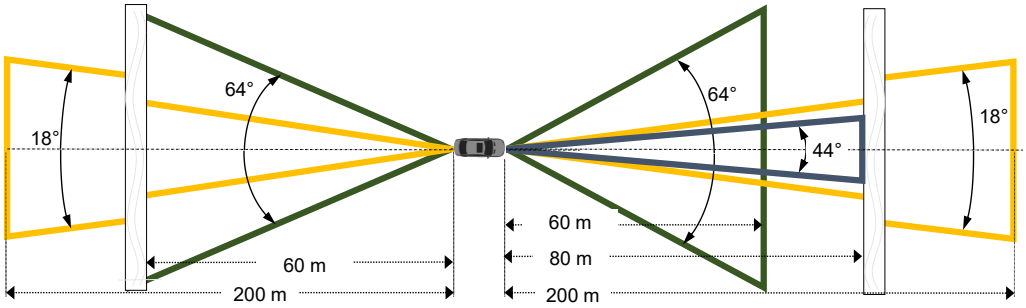


Fig. 4.11: *Sensor setup of the research vehicle used for the first experiment. Blue denotes the area sensed by the stereo camera, yellow the long-range beam of the radar sensor and green the mid-range beam of the radar sensor.*

used in the experiment was collected on a test drive along a route from Sindelfingen to Brussels. In the data-collection process the feature vector F_{sit} was computed every *100ms* for every measured and tracked vehicle with sufficient measurement confidence of the system-vehicle and written into a database. For vehicle tracking the method proposed by [98] was used. In total the database consists of 160 781 samples. In an offline and automated post-processing step for each collected sample the time for the next lane change to the left and to the right was determined. To detect whether a lane change happened, the distance to the left and right lane marking was analyzed. Whenever the waveform of the respective distance increased or decreased with a step size comparable to the width of a lane, a lane change was labeled and the time until the next lane change to the left T_{LcL} and the right T_{LcR} was added for each sample, Fig. 4.12 for a visual explanation. Based on the time until the next lane change the

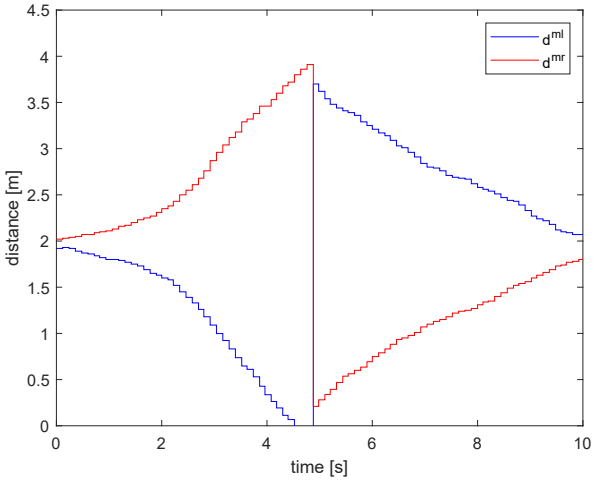


Fig. 4.12: *Labeling based on distance to lane markings. At approximately 5 s the distance to the left marking d^{ml} rapidly increases while at the same time the distance to the right marking d^{mr} decreases, which indicates that a lane change to the left was executed.*

maneuver M was determined using the following rules:

$$M = \begin{cases} LcL, & \text{if } (T_{LcL} \leq T_h) \wedge \\ & \quad (T_{LcL} < T_{LcR}) \\ LcR, & \text{if } (T_{LcR} \leq T_h) \wedge \\ & \quad (T_{LcR} < T_{LcL}) \\ Flw, & \text{otherwise} \end{cases} \tag{4.17}$$

where T_h denotes the time horizon of the prediction. Labeling samples $2\,s$ before a lane change event as positives, namely setting $T_h = 2\,s$ results in prior class probabilities of $P_{LcL} = 0.0051$, $P_{LcR} = 0.0037$ and $P_{Flw} = 0.9911$.

4.8.2 Model Generation

To satisfy the requirements as described in chapter 3, the algorithm to detect lane changes needs to provide a probabilistic output. To derive a meaningful featureset, Sec. 4.8.2.1 introduces a straightforward approach for feature selection. To solve the outlined classification problem, the methods used in the experiment are introduced in Sec. 4.8.2.2. The strategies to optimize the respective hyperparameters are described in Sec. 4.8.2.3.

4.8.2.1 Feature Selection

Within the feature selection process a small subset of features shall be determined which maximizes the predictive power. This is done in order to implement a classification algorithm which can be executed in real-time with minimal computational

Tab. 4.1: *Description of the evaluated features f for an observed vehicle.*

$\boldsymbol{R}$	$\boldsymbol{f}$	**description**	**constraint**
R_r	s_r^{rel}	longitudinal distance between o vs. related vehicle r	
	$v_{s,r}^{rel}$	longitudinal relative speed between o vs. related vehicle r	
	$a_{s,r}^{req}$	longitudinal deceleration required for o to avoid a collision with the related vehicle r	constant acceleration
	$ttc_{s,r}$	time to a longitudinal collision between o vs related vehicle r	constant acceleration
	$\tau_{s,r}$	time gap between o and related vehicle r	
	$v_{d,r}$	lateral velocity of a related vehicle r	
	car_r	existence of vehicle r	(0 = false, 1 = true)
R_o	d^{ml}	lateral distance between the center of o and left marking	
	d^{mr}	distance between the center of o and right marking	
	d_{cl}	distance between center of o and centerline of assigned lane	
	$ttcr_d^{mr}$	time to cross the right marking of assigned lane for o	constant velocity
	$ttcr_d^{ml}$	time to cross the left marking of assigned lane for o	constant velocity
	a_d^{req}	required acceleration which is needed to stay in the current lane	constant acceleration
	ψ	angle of the observed vehicle relative to the direction of the lane	
	v_d	lateral speed of the observed vehicle relative to the lane	
	$nlane_r$	number of lanes on the right side of observed the vehicle	
	$nlane_l$	number of lanes on the left of observed the vehicle	
	t_l^m	type of marking left	(0 = dashed, 1 = solid)
	t_r^m	type of marking right	(0 = dashed, 1 = solid)
	$a_{neg,l}^{req}$	required decc. for lane change left	
	$a_{neg,r}^{req}$	required decc. for lane change right	
	$a_{pos,l}^{req}$	required acc. for lane change left	
	$a_{pos,r}^{req}$	required acc. for lane change right	
R_{infra}	c_0	curvature of the road	clothoid model
	s^a	distance to the next approach to the highway	
	s^e	distance to the next exit of the highway	
	v_s^a	speed-limit of the current highway section	

effort while optimizing the results of the prediction models at the same time. To select this subset of useful features, different techniques can be applied [62], where one mainly differentiates between filtering and wrapper techniques. The feature selection problem comes hand in hand with the problem how early a lane change maneuver of an opponent vehicle can be detected before it crosses the lane markings based on all available information in F_{sit}.

To attack this issue the strategy is as follows. The maximum prediction horizon of each feature which is derived in the presented environment model shall be estimated by Single Variable Classifiers using a Naiv̈e Baysian approach. Based on these prediction horizons, the best performing, uncorrelated features shall be selected in order to build a small but powerful model. For the investigation the Area under the curve (AUC) of the Receiver Operating Characteristics (ROC) was used, which is useful for skewed distributions, because it is insensitive to changes in class distributions, see also the discussion in [53]. To deal with the problem, that *ROC* graphs can only handle two-class problems, a *ROC* graph was generated for every class against the remaining two classes. The *AUC* for multi-class problems can be calculated according to [53] by

$$AUC_{total} = \sum_{c \in M} AUC_c * p(c) \tag{4.18}$$

where M is the aggregate of the maneuvers LcL, LcR and Flw and $p(c)$ describes the prior of a class c. To get a statement over the time, data at different time intervals before a lane marking is crossed was selected. Using this metric a function for a feature vector F which describes the predictive power of the classifier C at a

time instance t_m before a lane change maneuver can be denoted:

$$AUC_t^C(t_m) = AUC_{total}^C(F^{t=t_m}) \tag{4.19}$$

This basically means that only feature vectors are taken into account, for which the time t is equal to the time t_m at which a maneuver m is executed. To select the desired small feature-set, the probability density functions for every feature was approximated and its predictive power analyzed using a Naïve Bayesian approach. Using single variable classifiers the value of $AUC_t^{C_f}(t_m)$ for every single variable classifier C_f for the time interval $[0\,s, 15\,s]$ was computed. By selecting only values larger than $AUC_t^C(t_{max}) > AUC_{min}$, a time point t_{max} for which a specific feature loses its predictive power was computed. For the result in Tab. 4.2, $AUC_{min} = 0.7$ was set and the features were sorted according to their prediction horizon t_{max}. In

Tab. 4.2: *Predictive power of features f and their Spearman's rank correlation coefficient ρ to the chosen features, see also feature definitions in Tab. 4.1.*

f	t_{max}	$\rho\left(v_{s,fo}^{rel}\right)$	$\rho\left(v_d\right)$	$\rho\left(d_{cl}\right)$
$v_{s,fo}^{rel}$	**2.2**	1	**-0.05**	**-0.07**
v_d	**2.0**	**-0.05**	1	**0.15**
$ttcr_d^{ml}$	1.8	0.03	-0.65	-0.19
a_d^{req}	1.8	0.05	-0.96	-0.17
d_{cl}	**1.0**	**-0.07**	**0.15**	1
d^{mr}	1.0	-0.09	0.14	0.88
d^{ml}	1.0	0.06	-0.12	-0.88
$ttcr_d^{mr}$	0.8	-0.06	0.69	0.2
$a_{s,fo}^{req}$	0.8	0.62	-0.06	-0.06

the results, the relative velocity to the front vehicle, $v_{s,fo}^{rel}$, contributes significantly already $2.2\,s$ before the lane change event and the lateral velocity of the vehicle with respect to its lane, v_d, contributes $2.0\,s$ prior to the lane change. The following features $ttcr_d^{ml}$ and a_d^{req} are clearly correlated to the lateral velocity and therefore it is not surprising, that they contribute similarly. The lateral displacement d_{cl} contributes significantly up to $1.0\,s$ before the lane change event. All remaining features show less predictive power. Therefore, to build a minimal classifier, the three most valuable uncorrelated features $v_{s,fo}^{rel}$, v_d, and d^{cl} were selected as input features for the SVM and Naïve Bayes algorithm. To investigate the validity of the approach, a Random Forest (RF) model was selected. In contrast to SVMs and Naïve Bayes, they method includes an implicit feature selection, as presented in Sec. 4.7.3. The featureset, which was derived in the training process of the RF, is documented in Tab. 4.3. For the training process, the data was labeled using a prediction horizon of $5.0\,s$. Because this prediction horizon was chosen purely on expert guess, the results

of the RF are compared in detail to the results of the two other models in Sec. 4.8.3.

Tab. 4.3: *Featureset which was derived in the training process of the Random Decision Forrest.*

f	description	f	description
car_l	boolean if there is a car on the left side of the observed vehicle	$v_{s,bl}^{rel}$	relative speed to the object on the back left lane
car_r	boolean if there is a car on the right side of the observed vehicle	$v_{s,br}^{rel}$	relative speed to the object on the back right lane
$nlane_l$	number of lanes on the left of the observed vehicle	$a_{s,f}^{rel}$	relative acceleration to the object in front
$nlane_r$	number of lanes on the right of the observed vehicle	$a_{s,fl}^{rel}$	relative speed to the object on the front left lane
t_l^m	boolean if marking left is dashed	$a_{s,fr}^{rel}$	relative acceleration to the object on the front right lane
t_r^m	boolean if marking right is dashed	a_d^{req}	required lateral acceleration to stay in the current lane
v_d	lateral velocity of the observed vehicle	$a_{pos,l}^{req}$	req. acc. for collision avoidance for a lane change left
v_d^{smo}	smoothed v_d	$a_{pos,r}^{req}$	req. acc. for collision avoidance for a lane change right
$ttcr_d^{ml}$	predicted time to a lane change left using a c.a. assumption	$a_{neg,l}^{req}$	req. decc. for collision avoidance for a lane change left
$ttcr_d^{mr}$	predicted time to a lane change right using a c.a. assumption	$a_{neg,r}^{req}$	req. decc. for collision avoidance for a lane change right
d_{cl}	distance between vehicle center and assigned centerline	ttc_f	time to collision with the object in front
d^{ml}	distance between vehicle center and the left marking	$ttc_{s,b}$	time to collision with the object in the back
d^{mr}	distance between vehicle center and the right marking	$ttc_{s,fl}$	time to collision with the object on the front left lane
s_f^{rel}	relative distance to the object in front	$ttc_{s,fr}$	time to collision with the object on the front right lane
s_b^{rel}	relative distance to the object in the back	$ttc_{s,bl}$	time to collision with the object on the back left lane
s_{fl}^{rel}	relative distance to the object on the front left lane	$ttc_{s,br}$	time to collision with the object on the back right lane
s_{fr}^{rel}	relative distance to the object on the front right lane	$\tau_{s,f}$	timegap to the object in front
s_{bl}^{rel}	relative distance to the object on the back left lane	$\tau_{s,b}$	timegap to the object in the back
s_{br}^{rel}	relative distance to the object on the back right lane	$\tau_{s,fl}$	timegap to the object on the front left lane
s_f^{rel}	relative distance to the object in front	$\tau_{s,fr}$	timegap to the object on the front right lane
s_{fl}^{rel}	relative distance to the object on the front left	$\tau_{s,bl}$	timegap to the object on the back left lane
$v_{s,f}^{rel}$	relative velocity to the object in front	$\tau_{s,br}$	timegap to the object on the back right lane
$v_{s,b}^{rel}$	relative velocity to the object in the back	$v_{d,f}$	lateral speed of the object in front
$v_{s,fl}^{rel}$	relative speed to the object on the front left lane	$v_{d,fl}$	lateral speed of the object in front left

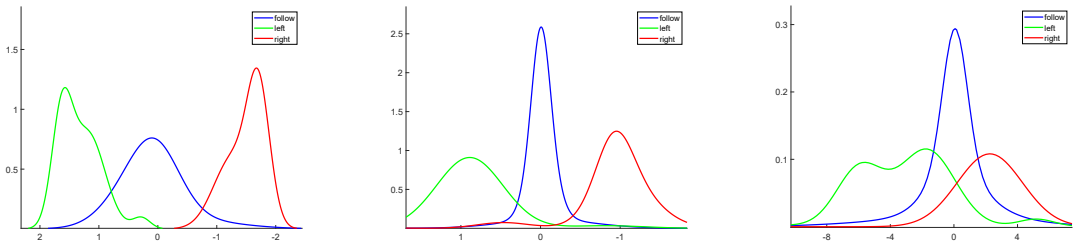
4.8.2.2 Algorithms

Within the experiment three different classification methods were used, the Naïve Bayes Algorithm, Support Vector Machines and Random Forests, see also Sec. 4.7 for a detailed explanation. The Naïve Bayes is the most straightforward classification approach providing probabilistic output as a baseline. SVMs were chosen to compare this baseline to a state-of-the-art approach for low dimensional datasets. Random Forests were chosen to investigate whether a model using an extensive feature set provides superior performance. All three methods, when applied as proposed in this experiment, have low requirements regarding runtime and memory. In accordance to the prediction horizons which were derived by the single variable classifiers in Tab. 4.2, the data for the training of the SVM and the distributions of the Naïve Bayes was labeled using a prediction horizon of $2\,s$. To validate whether larger prediction horizons are possible using the full featureset, the data used for the training of the RF was labeled using a prediction horizon of $5\,s$.

4.8.2.3 Hyperparameter Optimization

For the selected methods in Sec. 4.8.2.2 the respective hyperparameters need to be optimized to achieve the best possible performance and to ensure comparability. For the Naïve Bayes Classification preliminary investigations showed, that an unimodal normal distribution model is not a valid assumption. Many features have a strong non-symmetrical value distribution, see Fig. 4.13. Thus, Gaussian Mixture distributions

were chosen as a distribution model. To determine the best performing number of components of the Gaussian Mixture, the BIC was chosen as optimization criterion, see Sec. 2.3.4.3. To increase the robustness of the Gaussian Mixtures against singularities and noise, the DB-Scan [63] algorithm is used to estimate one single Gaussian mixture distribution for every value range where a continuous distribution exists. The method to do so is explained in detail in [73]. For the optimization of



(a) d_{cl} **(b)** v_d **(c)** $v_{s,fo}^{rel}$

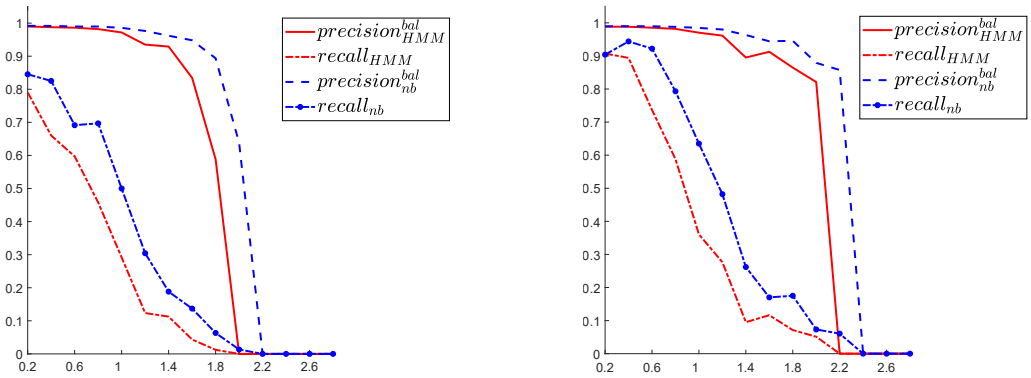
Fig. 4.13: *Probability density functions of the different features. The classes visualized are LcL (green), Flw (blue) and LcR (red). A good separation between the three classes is possible for each of the chosen features, see [73].*

the hyperparameters of the SVM and the Random Forest a grid search approach was chosen. For the Random Forest the parameters to be varied are the number of trees num_e and the maximum tree depth max_d, see Sec. 4.7.3. The remaining parameter f_r which denotes the number of features taken into account for a new split in the training process, was chosen according to [88], as the square-root of the total number of features. For the SVM accordingly, the parameters to be optimized are γ as the free parameter of the RBF and c parameterizing the soft margin, see Sec. 4.7.2. For each method the respective best performing hyperparameters were selected to train the model used in Sec. 4.8.3

4.8.3 Evaluation

In the following subsection the results of the first experiment are presented. For the evaluations, which allowed to generate Fig. 4.14 a winner-takes-all strategy was chosen. In Fig. 4.15 the probability of detecting a lane change increases as closer the time point of a lane change gets. When analyzing the plot of the precision values, almost every time a sample is classified as positive, it is also positive in reality. The reason for the variation over time is mainly the decreasing number of true positives as the time horizon increases. For the evaluation a winner-takes-all strategy was chosen. When applied in Advanced Driver Assistance Systems (ADAS), one may be interested in finding a more elaborate decision strategy. This can include different thresholds to decide when a sample is truly positive.

For example in the application in an ACC system which only implements highly limited acceleration and deceleration maneuvers, the decision threshold can be kept pretty low to implement a cooperative behavior. When deployed in a collision

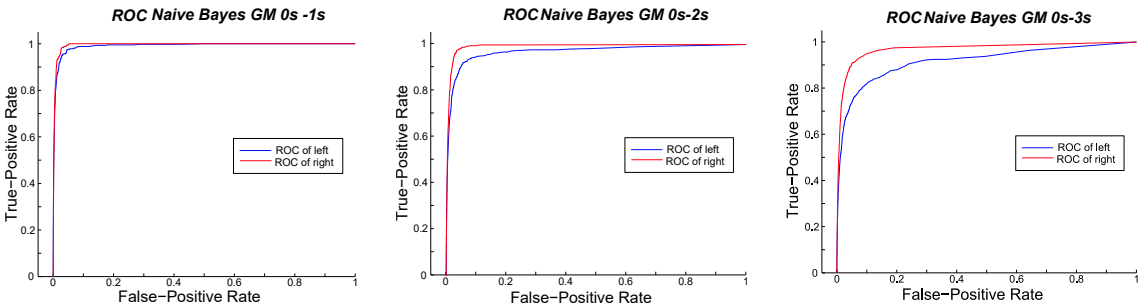


(a) *LcL(HMM)* **(b)** *LcR(HMM)*

Fig. 4.14: *Comparison between the HMM (red) and the Naïve Bayes Algorithm (blue) in (a) and (b) for the recognition of the maneuver classes LcL and LcR plotted against the time before a lane change occurs on the x-axis.*

avoidance or Collision Mitigation System (CMS) which allows strong braking one will only decide for the execution of an emergency maneuver for high values of the cut-in probability.

As can be seen in Fig. 4.14, the Naïve Bayes Algorithm shows better results without the use of an additional HMM, see Sec. 4.7.4.1 for the explanation of the method. Thus, there is no performance increase when applying additional filtering in the domain of maneuver probabilities. For further analysis only the unfiltered Naïve Bayes was taken into account. Its classification performance characterized by its ROC curve is depicted in Fig. 4.15. Each of the three figures in Fig. 4.15 was generated by selecting samples from the test-dataset in the time interval according to the captions. When extending the time horizon up to 3 s the predictive power decreases significantly, which indicates that lane changes can be recognized only up to 2 s with high reliability.



(a) *ROC 0 s-1 s* **(b)** *ROC 0 s-2 s* **(c)** *ROC 0 s-3 s*

Fig. 4.15: *Receiver Operating Characteristic showing the predictive power of the proposed Naïve Bayes Algorithm for samples in different time intervals before a lane change, see [73]. Blue denotes the maneuver class LcL, red the maneuver class LcR.*

To assess the performance of the Naïve Bayesian model it was compared against

a SVM as state of the art method. The results of this experiment are depicted in Fig. 4.16. The classification performance of both models is comparable up to 1.5 s with lower performance of the Naïve Bayes at higher time horizons. This indicates, that the assumption of independence between the different features may not hold true for prediction horizons larger than 1.5 s. Using the existing dataset

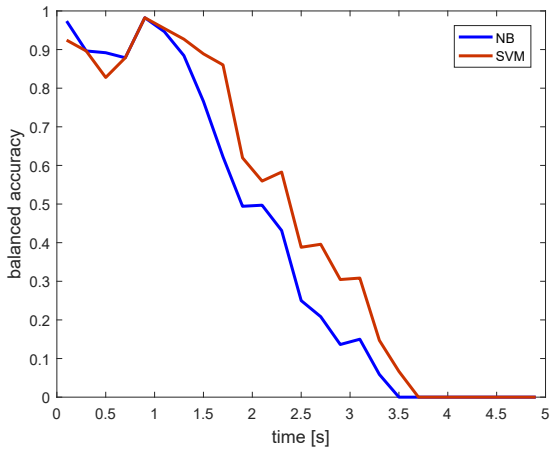


Fig. 4.16: *Comparison of the Naïve Bayes Classifier (blue) as proposed in experiment versus a Support Vector Machine(red) using the same data.*

it was investigated whether it is possible to increase the prediction time horizon significantly using the power of the whole featureset. This investigation is motivated by the insight, that the assumption of feature independence does not hold true for larger time horizons, see Fig. 4.16.

Thus, a RF was trained as described in Sec. 4.8.2.3 with a labeling time horizon of 5 s. The ROC curve for each class using the derived RF Model was computed based on the probability estimates of the decision trees, for the results see Fig. 4.17.

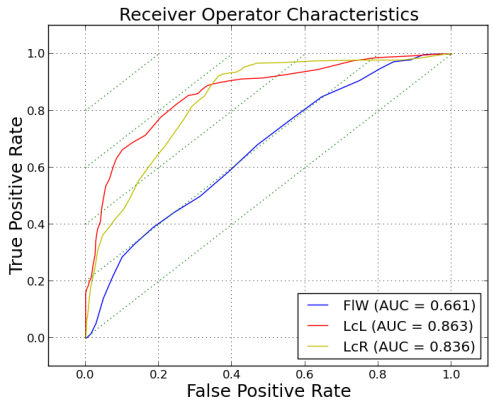
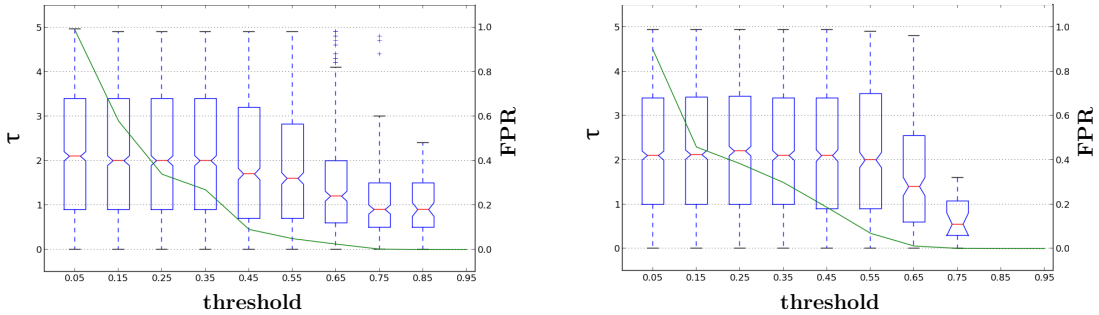


Fig. 4.17: *Receiver Operating Characteristic of the Random Forest for the classification task of recognizing the intent of a vehicle 5 s before its lane assignment changes.*

The results in Fig. 4.17 show that the prediction performance for a time horizon of 5 s is noticeable worse when compared the Naïve Bayes with a (short) prediction



(a) *Maneuver recognition time as a function of threshold on P_{LcL}*

(b) *Maneuver recognition time as a function of threshold on P_{LcR}*

Fig. 4.18: *Evaluation of the Random Forest model with the task to recognize the intent of a vehicle 5 s before its lane assignment changes. Because not only the value of the True Positive Rate is in the focus of interest, but also how early the lane change can be recognized, this property is visualized vs. a decision threshold using a boxplot in Fig. 4.18a and 4.18b. The corresponding value on the y-axis corresponds to the time τ at which this decision threshold is overrun for the first time and never undershot until the vehicle has changed its lane-assignment. The green line visualizes the corresponding the false positive rate.*

horizon of 2 *s*. It is still considerable, that 65% of the lane change samples to the left side can be detected in a time interval 5 s before the lane-assignment changes with a false alarm rate of 10%. To get a deeper insight into the properties of the approach, the prediction time via a threshold for a one against all decision is visualized in Fig.4.18a and Fig. 4.18b. As expected by increasing the decision threshold, for which a sample is classified as positive, the frequency of the true positive decreases. However one may be not only interested in the number of true positives, but also in the time horizon when a lane change can be detected. In Fig.4.18a at a decision threshold of 0.75 the number of false positives decreases close to zero, while there are still detections up to 3 *s*.

4.8.4 Conclusion

Using the available data collected in this first experiment, the lean approach using the Naïve Bayesian Classifier showed impressive results for prediction horizons up to 2 s (Sec. 4.8.3). This is especially remarkable, as only the relative velocity to the preceding vehicle, the lateral distance to the lane center and the lateral velocity were selected as features (Sec. 4.8.2.1) from the presented environment model (Sec. 4.5). Interestingly the prediction performance is also superior to more sophisticated models, see for example [23]. The experiments indicated, that with the number of data available and the presented method prediction horizons up to 2 s can be achieved with high accuracy. Whether the prediction horizon can be increased even further, cannot be answered reliably based on the data collected in this first experiment. However, the evaluations in which the prediction time horizon was extended up to

5 s indicate that there may be potential for improvements. To verify this assumption a second experiment was conducted in Sec. 4.9.

4.9 Experiment 2

In the second experiment the focus shifts towards the question of how good the prediction of traffic participants can get assuming high quality sensors are available. To predict maneuvers for longer time horizons, it is convenient to have a dataset which also contains tracks of the vehicle to be predicted for sufficient long observation time horizons. To collect such kind of data, a fleet of close to series vehicles equipped with comparable cheap camera and radar systems was used in the second experiment. The environment data of these vehicles was collected and the vehicles themselves were selected as prediction targets. This experiment setup heavily depends on the assumption, that future sensor generations can determine the features derived in the experiment for other vehicles comparable good as close to series vehicles today for themselves. All investigations presented in the section were presented firstly in [76], where the contribution of the author of this thesis is mainly in the research strategy and the conceptual part.

4.9.1 Setup

By a fleet of test-vehicles which were deployed in the area of Stuttgart a dataset consisting of round about 40 000 situations, collected on more than 30 000 km of highway driving was used for the second experiment. The vehicles of the fleet were operated by different drivers at varying weather conditions and different times.

The vehicles which collected this data were close to series vehicles equipped with a front facing camera, a long range front radar sensor and radars sensors covering the area left and right next to the system vehicle. When comparing the setup to the first experiment, the visibility of the traffic scene is limited regarding the area behind the system vehicle. In contrast to the work presented in Sec. 4.8 the system-vehicle itself was selected as prediction target. As the method focuses on the prediction of surrounding vehicles, only features were used, which can also be measured for surrounding vehicles. This strategy is also applied in other research, see for example [99] or [78]. Therefore, only a subset of the available data is available and information like the status of the driver or the steering wheel angle are excluded from the dataset. The main advantages compared to the setup in the first experiment is, that situation development can be observed continuously without occlusion artifacts and that sensor range is a less limiting factor. This way of data handling is widespread in literature, see for example [79]. Future sensor setups including multiple lidar sensors are expected to be more precise minimizing the data quality gap between sensed object data and the data available for the system vehicle in the experiment. The investigations rely on the environment model presented in Sec. 4.5. The vehicle behind the system vehicle on the same lane could not be sensed and thus the corresponding features are not present in the feature set. In Tab. 4.4 an overview of all features used in the experiment is documented.

Tab. 4.4: *Description of the evaluated Features f for an Observed Vehicle o in the second experiment, see Fig. 4.5 for a visual explanation of the vehicle relations.*

R	f	Description	Unit (Continuous) Range of Values (Nominal)	Element of B (40 Elements)	C (29 Elements)	D (MLP) (24 Elements)	D (GNB) (48 Elements)
R_r	general information describing the related vehicle r						
	$actv_r$	activity status	{0: inactive, 1: active}	{f, l}	{f, fl, fr, l}	{fr, r}	{fl, fr, l, r}
	mov_r	movement status	{0: standing, 1: moving}	{f, l, br}	{f, fl, fr, l, br}	{r, br}	{fl, fr, r, br}
	$class_r$	object class	{0: bicycle, 1: motorbike, 2: car, ..., 14: no class}	{f, l}	{f, fl, l}		{fl, fr, r}
	$cutinlvl_r$	cut-in level	{0: $P \leq 0.5$, 1: $P > 0.5$, 2: $P > 0.66$, 3: $P > 0.9$}	{l}			{r}
	relation between observed vehicle o and related vehicle r in o's cartesian coordinate system						
	$d_{x,r}^{rel}$	longitudinal distance	m	{f, l}	{f, l}	{f, r}	{fr, r}
	$d_{y,r}^{rel}$	lateral distance		{f, l, bl}	{f, fr, bl}	{r}	{fr, r, bl, br}
	$v_{x,r}^{rel}$	relative longitudinal speed	$^m/_s$	{f, r}		{r}	{f, fr, r}
	$v_{y,r}^{rel}$	relative lateral speed		{f, fl, l, r}	{f}	{f, fr, r}	{f, fr}
	$a_{x,r}^{rel}$	relative longitudinal acceleration	$^m/_{s^2}$			{fr}	{f, fl, fr, r}
	relation between observed vehicle o and related vehicle r in lane coordinates						
	$d_{s,r}^{rel}$	longitudinal distance	m	{f, l}	{l}	{fr}	{fl, fr, r}
	$d_{d,r}^{rel}$	lateral distance		{f, l}	{fr}	{r}	{fr, r}
	$v_{s,r}^{rel}$	relative longitudinal speed	$^m/_s$	{f, r}		{f}	{f, fl, fr, r}
	$v_{d,r}^{rel}$	relative lateral speed		{f, fl, l, r}		{l}	{fr, r}
R_o	fog^f	status of the front fog lamp	{0: off, 1: on}				
	fog^r	status of the rear fog lamp					
	fog^{rl}	status of the rear left fog lamp					
	fog^{rr}	status of the rear right fog lamp					
	wpr	wiper level	{0, ..., 15}				
	d^{ml}	distance between the center of o and the left marking	m	✓	✓	✓	✓
	d^{mr}	distance between the center of o and the right marking		✓	✓		
	d_{cl}	distance between the center of o and the centerline of the assigned lane		✓	✓		
	v_s	longitudinal speed of the observed vehicle	$^m/_s$				
	v_d	lateral speed of the observed vehicle		✓	✓		✓
	a_s	longitudinal acceleration of the observed vehicle	$^m/_{s^2}$	✓		✓	
	a_d	lateral acceleration of the observed vehicle		✓	✓	✓	✓
	ψ	angle of the observed vehicle relative to the direction of the lane	$^\circ$	✓	✓	✓	✓
R_{infra}	t^{ml}	type of the left marking	{0: no marking, 1: continuous, 2: broken}	✓	✓	✓	✓
	t^{mr}	type of the right marking		✓	✓	✓	✓
	c^{ml}	color of the left marking	{0: no marking, 1: white, 2: yellow}				✓
	c^{mr}	color of the right marking					✓
	$nlanes_{cam}$	number of parallel lanes observed via the camera	{0: 0, ..., 3: 3+}				✓
	$nlanes_{map}$	number of lanes stored in the map	{0, ..., 5}				
	$cntr$	country	{0: GER, 1: US, ...}				
	tnl	indicator if the situation takes place in a tunnel	{0: False, 1: True}				✓
	brd	indicator if the situation takes place on a bridge					
	v^{lim}	speedlimit of the current highway section	{1: $> 130\frac{km}{h}$, ..., 8: $< 11\frac{km}{h}$}				
	t^a	type of next approach to the highway	{0: unknown, 1: on ramp, 2: highway merge}				
	t^e	type of next exit of the highway	{0: unknown, 1: ramp, 2: highway divider}				
	w^{ml}	width of the left marking	m	✓	✓		
	w^{mr}	width of the right marking		✓	✓		
	w^{lane}	width of the lane				✓	
	d_x^a	distance to the next approach to the highway					
	d_x^e	distance to the next exit of the highway					
	c_0	curvature of the road	$^1/_m$				
	c_1	derivation of the curvature	$^1/_{m^2}$				

4.9.2 Dataset

To be able to test and develop algorithms on the environment model presented in Sec. 4.5, data from three different origins was fused. The first source of data was collected by the testing fleet. This data was enriched with information from a navigation map (containing information of e.g. bridges, tunnels and distances to highway entries). The third kind of data was calculated based on the first and second kind of data. This includes for example lateral and longitudinal distances in a curvilinear coordinate system along the road which were computed accordingly to the first experiment, see Sec. 2.2 and Fig. 2.3. Please see Tab. 4.4 for an overview on the features used for the experiment.

According to Sec. 4.1 all samples were assigned to the three maneuver classes LcL , Flw and LcR . The labeling process in accordance to Sec. 4.8, works as follows: First for each data-sample the time to the next lane change to the left and right neighboring lane is calculated. For the experiment a prediction horizon of 5 s was chosen. The data was labeled accordingly. It is important to understand that this labeling strategy can result in pessimistic performance values, as detections of lane changes earlier than 5 s before a lane change are evaluated to false positive, because being labeled as Flw.

The data is splitted into several parts after executing the aforementioned preprocessing stages. The first split divides the data in one part for the lane change classification and another one for the trajectory regression problem, see chapter 5 for a detailed description. The dataset used for the experiment of lane change recognition is then splitted into six parts. Five parts are selected in Sec. 4.9.3 for the design and parametrization of the algorithms. The sixth part is only used for evaluation purposes in Sec. 4.9.4. The split is performed on situations according to [75], to ensure that no optimistic results are produced because of similar samples from the same time series. To achieve an even proportion of the three maneuver classes, the number of samples within each fold is balanced using a random undersampling strategy. As the prediction problem is unbalanced, classifiers would mainly focus on the most frequent maneuver class (Flw) otherwise.

In the collected data approximately 94 % of the samples were lane following samples (cf. also [100]). The increased share of lane change samples in comparison to the first experiment can be partially explained by the increased prediction horizon. The increase based on the change of 2.0 s to 5.0 s can be expected with a factor of 2.5. The share of lane changing examples in the second experiment is however more representative when being compared to the first experiment, due to the larger amount of samples which were collected in a greater diversity of environment conditions. For the experiment additionally only situations in which samples were collected continuously up to the prediction horizon of 5 s were taken into account. This ensures, that the folds are balanced over time. This is necessary to perform fair evaluations, as the prediction task is a more demanding task for higher time horizons as for short ones. The remaining data contains approximately 8 hours of highway driving where each maneuver class is represented equally.

4.9.3 Model Generation

The following subsection gives an overview of the different algorithms used for feature selection, classification and the techniques to tune the respective hyperparameters for the maneuver classification in the second experiment.

4.9.3.1 Feature Selection

In the implementation of the feature selection process of the second experiment, three different feature selection methods were compared. Including the superset of features, this results in four datasets A, B, C and D. The findings of [62] were used in the application of the respective methods. The strategy to tackle the feature selection problem is to start with simple techniques and to continue with more sophisticated and computational expensive ones. As a baseline the classifiers were tested with the whole superset of features, where A denotes this superset in the following. The second feature set B was derived by applying a threshold on Spearman's rank correlation value (see [101, p. 133 ff]) of the feature with the respective class label. The threshold value to perform the selection was set to 0.15. Feature set B still contains many features which are highly cross-correlated. In order to remove this potentially redundant information, a third method was applied to the feature selection problem. The technique Correlation based Feature Selector (CFS) introduced by [83] was used for the generation of this third feature set C. The main difference compared to the naive strategy in B is, that correlation values for whole feature sets are determined. See also Sec. 4.6.1 for a more detailed explanation of the method. Again Spearman's correlation was used for the computations. To generate C backward selection with a 5-fold cross-validation was applied, because the computation of the CFS was not feasible for all combinations of features. The filtering methods for

Tab. 4.5: *Examined Feature Selection Variants in experiment 2.*

Variant	Description
A	Baseline feature set containing all features
B	Feature set based on correlation threshold of feature with class label
C	Correlation based Feature Selector (CFS)
D	Wrapper approach for selection of feature set

feature selection applied for the derivation of B and C, are typically fast to evaluate. However, wrapper techniques can use the advantage incorporating the properties of the respective classification algorithm in the feature selection process. Accordingly, for the generation of feature set D for every classification algorithm except the Random Forrest, which includes an implicit feature selection in its training, the best feature set was derived. Using a hyperparameter set which was optimized on C a backward selection strategy was applied. For each iteration one of the folds of Sec. 4.9.2 was used for training and one for validation. This limitation was introduced mainly to limit the time needed for feature selection, where one needs so see that for each of the 5 000 possible subsets a classifier needs to be trained and evaluated.

Tab. 4.5 summarizes the examined variants and their abbreviations. The resulting elements of the respective feature sets can be found in Tab. 4.1.

4.9.3.2 Algorithms

For the task of maneuver classification three different methods were selected for evaluation, where the first two were already part of the investigations in Sec. 4.8. The first method is a Gaussian Naïve Bayes approach using GMMs. The second method applied to the problem are Random Forest (RF). As third method a Multi Layer Perceptron (MLP) implementing a Feedforward Neural Network, see Sec. 4.7.4, similar to the approach proposed in [102] was chosen. This selection implements a direct comparison of the approaches of the first experiment and a state-of-the-art approach on a representative dataset. In contrast to [102], a modified labeling and a partly automated strategy was chosen to identify the best possible model structure, where the model was restricted to one hidden layer. In contrast to the first experiment SVMs were not used, because the server cluster used for training did not support parallelized training, which was an exclusion criteria due to non tolerable computation time requirements.

4.9.3.3 Hyperparameter Optimization

To achieve the best performance and to enable a comparison of the chosen classification algorithms, the respective hyperparameters were optimized. For the Gaussian Naive Bayes (GNB) this means to find the optimal number of Gaussian components for each feature and class. In contrast to the first experiment a Variational Bayesian Gaussian Mixture Model (VBGMM) was selected for this task, where this technique was already successfully applied by [27]. For the RF and the MLP the parameter optimization was executed for each feature set using a grid search. The performance of each parameter set was measured using the average balanced accuracy ACC^{bal}, derived using leave-one-out cross-validation with five folds of training and validation. For the RF the parameters to be optimized are the number of parallel trees, the maximum number of splits in each tree and the minimal number of samples that are necessary for a split. For the MLP the parameters accordingly are the step size which controls how fast the weights of the network are adapted towards the direction of the gradient and the structure of the network using the number of neurons in each layer. Using the described techniques, different feature sets were selected and the respective hyperparameters for the different classification algorithms were optimized. In a second step the respective best combinations were selected to derive the models which were used for further evaluation. Please see Tab. 4.4 for the set of features derived for the respective models.

4.9.4 Evaluation

In the following subsections the results of the second experiment are presented. Subsection 4.9.4.1 introduces the chosen metrics for the evaluation. Sec. 4.9.4.2

concludes with the listing and discussion of the results measured with the test dataset.

4.9.4.1 Metrics

To asses the performance of the classification algorithms multiple metrics are needed, as multiple objectives are of interest, namely the correctness and the prediction time horizon. For measuring the correctness and to deal with the bias of the class distribution, the balanced accuracy ACC^{bal} is selected, see Sec. 2.3.3.7. To quantify the predictive power of the classifier in accordance with the first experiment the ROC and the AUC are used. To quantify the prediction time horizon, which is the second objective, the time when a lane change is recognized is an important performance metric. As the exact definition of this time is essential for a comparison and most sources (see e.g. [78], [102] and [99]) are not very exact in that point, and as a finding of the first experiment the following two metrics are defined:

- τ_1: time of the first correct maneuver detection before lane assignment changes, see for example [79].
- τ_2: time of the correct maneuver classification before lane assignment changes, where the classification does not change anymore until the lane assignment changes.

The definition of τ_2 is a considerable stricter definition compared to the definition of τ_1.

4.9.4.2 Results

For each of the chosen methods the formerly defined metrics were computed using the respective best performing feature set and hyperparameters, please see Tab. 4.6 for a summary of the results. In Fig. 4.19 the respective ROC of the chosen models are depicted.

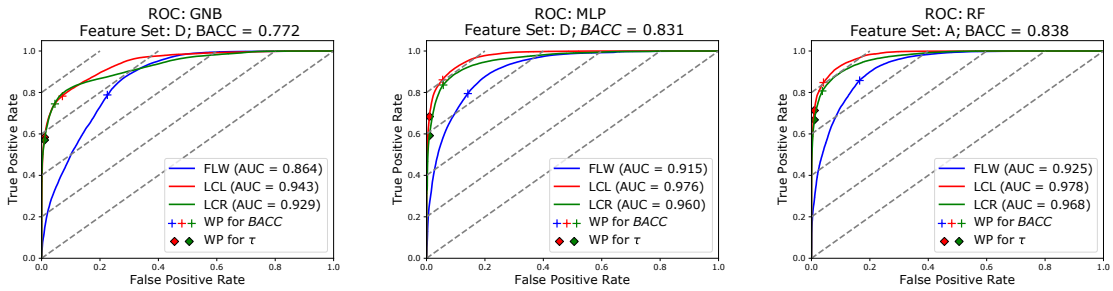


Fig. 4.19: *ROC-Curves for the developed lane change predictors with their respectively best parameter sets and hyperparameters.*

The classification of the lane following class *Flw* is notable worse when compared to the lane-changing maneuvers, as can be seen in Tab. 4.6, which is neglected in

Tab. 4.6: *Summary of Examined Classifiers with Preferred Hyperparameters.*

Classifier	Feature Set	ACC^{bal}	Performance on Test Data per Class (AUC; $\overline{\tau_1}$; $\overline{\tau_2}$) LcL	Flw	LcR
GNB	A	0.704	0.924 2.86±1.46 s 2.18±1.26 s	0.815 - -	0.905 2.92±1.42 s 2.21±1.21 s
	B	0.692	0.910 2.82±1.38 s 2.11±1.13 s	0.801 - -	0.895 2.82±1.32 s 2.09±1.06 s
	C	0.651	0.874 2.57±1.31 s 2.02±1.11 s	0.770 - -	0.884 2.73±1.31 s 2.02±1.07 s
	$\boldsymbol{D}_{GNB}$	**0.772**	**0.943** **3.26**±1.28 s **2.61**±1.13 s	**0.864** - -	**0.929** **3.11**±1.14 s **2.69**±0.94 s
MLP	A	0.823	0.973 3.67±1.26 s 2.95±1.25 s	0.909 - -	**0.961** 3.34±1.18 s 2.82±0.97 s
	B	**0.831**	0.974 3.74±1.07 s **3.08**±1.04 s	0.912 - -	0.959 **3.60**±1.16 s **2.86**±1.06 s
	C	0.798	0.966 3.44±1.07 s 2.86±0.91 s	0.891 - -	0.953 3.46±1.11 s 2.82±0.89 s
	$\boldsymbol{D}_{MLP}$	**0.831**	**0.976** **3.78**±1.16 s **3.08**±1.10 s	**0.915** - -	0.960 3.35±1.18 s 2.66±0.99 s
RF	$\boldsymbol{A}$	**0.838**	**0.978** **3.81**±1.14 s **3.31**±1.13 s	**0.925** - -	**0.968** 3.60±1.19 s **3.13**±1.08 s
	B	0.834	0.976 3.73±1.13 s 3.28±1.10 s	0.918 - -	0.959 **3.61**±1.17 s 3.08±1.00 s
	C	0.799	0.964 3.45±1.07 s 2.95±0.87 s	0.893 - -	0.953 3.49±1.10 s 2.95±0.90 s

most related research. This finding is consistent to the results of the first experiment, see Sec. 4.8.3. A reason may be, that every sample which is not within a close time horizon to an executed lane change maneuver is assigned to the Flw class. This includes therefore early detections of lane changes but also interrupted lane change maneuvers. On the other hand the confusion between the other two classes are very rare. In this context the findings of [80] can be confirmed, which state that lane changes to the left are easier to predict.

Another interesting finding is, that the classification problem remains resolvable even with a significantly decreased number of features. This is demonstrated by the MLP with feature set D, which includes only 24 features, where the lower dimensional input space improves the classification performance. When comparing the predictive performance of the different classification methods on the different feature sets in Tab. 4.6, the Random Forrest approach on the unfiltered feature-set A performs slightly best. The MLP on the feature set D which was derived using wrapper techniques follows closely. For all classification models the application of the CFS in feature set C results in a decrease in the predictive performance. When compared to the naive correlation based filter technique applied to derive B, this is interesting, as one may expect the more sophisticated technique to perform better. When focusing on the classification results the GNB approach interestingly does not profit from correlation based feature selection strategies, even though the chosen features are not independent, see Tab. 4.4. However, the prediction performance can be increased significantly by using wrapper techniques. The MLP on the other hand slightly improves its predictive performance, if wrapper techniques are applied. At least it does not show a decrease if uncorrelated features are removed from the superset. When comparing the result of the Random Forest on the different feature sets, the method on the examined dataset does not profit from correlation based feature selection strategies. When assessing the prediction horizon,the both best performing models achieve average lane change detection times of round about $3 - 4\,s$. This indicates that not all, but when taken together with the ROC, most lane changes can be detected as early as 3 s or earlier. One may however not draw the conclusion, that $3.81\,s + 1.14s\,s = 4.95\,s$ is the maximum prediction horizon which is technically possible, because the models were only trained for a time horizon up to 5 s. As a consequence, all evaluations were limited to the time horizon of 5 s.

4.9.5 Conclusion

The second experiment demonstrates, that the proposed strategy to derive features (Sec. 4.5) in combination with a best practice machine learning techniques is able to outperform sophisticated models presented in related work, see Tab. 4.7. Although the presented method has a significantly higher prediction horizon compared to the listed approaches, still its predictive power is superior. Simple approaches, which mainly focus on the lateral movement, accordingly are clearly outperformed, too. By using the proposed method and models, many maneuvers can be predicted before they are executed by interpreting the situation context. Some questions, which came up in the experiments could be addressed by further work. This is for example the

Tab. 4.7: *AUC values of experiment 2 compared related work.*

Approach	*AUC*		Prediction Horizon
	LcL	LcR	
GNB as described in 4.8.3	0.970	0.991	2.0 s
Interaction aware approach of Bahram et al. [80]	0.947	0.942	2.5 s
Hybrid approach of Wissing et al., see [103]	0.934	0.993	2.0 s
MLP	0.976	0.960	5.0 s
RF	0.978	0.968	5.0 s

investigation of the maximum feasible prediction time horizon. Another direction worth to investigate, is a systematic extension of the environment model. Especially the investigation of an "optimal" tradeoff between the complexity of the environment model and the achievable prediction performance would be helpful to design future generations of maneuver recognition models.

Chapter 5

Probabilistic Position Prediction

In order to avoid accidents and find a safe way to their destination, human drivers use their intuition and knowledge about formerly experienced situations when driving a vehicle. Thus, humans are taking into account future positions of other road users, where those positions are estimated based on their experience and the anticipated intent of others. This estimation skill is developed in the childhood. For example children younger than 12 cannot handle complex traffic situations safely, see [104]. When automating the driving task, the skill to estimate future positions needs to be part of an automated driving system.

Accordingly, in the system concept in Sec. 3.5.2 a function block *Position Prediction* is specified. The output of this block is needed to implement a fallback behavior, please see chapter 3 for details. Besides this use case such kind of output can also be used for other vehicle automation purposes, e.g. Collision Mitigation Systems (CMS), ACC or Level-4 automated systems.

In this chapter a method how future positions can be predicted is presented and compared with classical and state-of-the-art approaches. This novel method features a probabilistic output which can be derived efficiently in real time.

The chapter starts with the definition of the problem to be solved in Sec. 5.1. Based on this problem definition a brief overview of related work is presented in Sec. 5.2. The contribution is explained in Sec. 5.3, and the solution design is presented in Sec. 5.4. The representation of the input data used for the training of the models is presented in Sec. 5.5. Using this input data the methods to solve the outlined problem are introduced in Sec. 5.6. According to the scope of this thesis, the methods are applied and evaluated in highway scenarios. In the following the metrics which are used in evaluation are discussed briefly in Sec. 5.7. The proposed methods are evaluated in two experiments. The results of chapter 4 on maneuver recognition are reused for the experiments, which are presented in Sec. 5.8 and Sec. 5.9. While the first experiment investigates the principal applicability of the proposed method to the problem domain, the second experiment is a representative study where the method is evaluated on a large number of data.

The work presented in this chapter is mainly based on already published work in [75], [76], [105]. The references are documented again in the introduction of the respective sections.

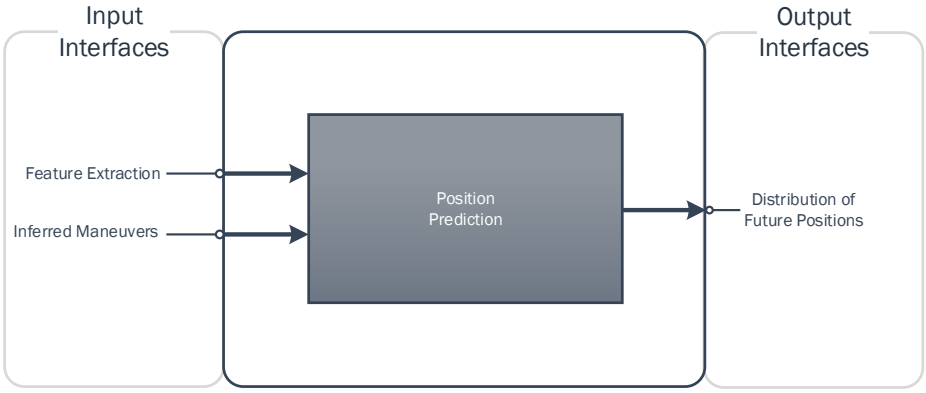


Fig. 5.1: *White box view of the functional block Position Prediction depicting the consumed information and the output to be computed, see also Fig. 3.6.*

5.1 Problem Definition

Recognizing the intent of other drivers as discussed in the former chapter is a high dimensional classification problem. A challenge of this problem is that human behavior is eratic sometimes. When it comes to position prediction, the problem in contrast is constrained by physics limiting the reachability of future positions. The available knowledge about the executed maneuver class additionally constrains future whereabouts.

Besides these constraints different drivers execute maneuvers differently. This basically means that the problem to be solved is to predict a time-dependent probability distribution and not a single trajectory. Thus, the objective of the learning problem is to generate an estimator which models the probability distribution of future vehicle positions. This shall be done in a way, such that the probability distribution reflects the frequency of the observed positions, assuming the same situation would be handled by different drivers.

Another problem is that when collecting data on highways the number of lane change samples is magnitudes smaller compared to the number of samples collected for lane following behavior, see also the findings of Sec. 4.8.3. This states a challenge especially relevant in the context of lane change prediction in highway scenarios.

From an architectural point of view, the problem to be solved in the block of *Position Prediction* as defined in Sec. 3.5.2 is the definition of a method, capable to estimate a distribution of future positions. This distribution shall reflect the physical state and inferred maneuver estimate of the vehicle to be predicted, see Fig. 5.1 for a visual explanation.

5.2 Literature

While maneuver recognition has already been investigated in numerous publications (see Sec. 4.2), only a comparable small number of researchers investigated the prediction of future whereabouts or even the distribution of future positions. The approaches mainly can be categorized into expert based models and machine learning approaches which is reflected in the following two subsections.

5.2.1 Expert Based Models

For many applications such as motion planning or collision avoidance systems, the information of future vehicle positions needs to be estimated. Especially for collision avoidance where only short prediction horizons are needed and motion dynamics are mainly described by physics, expert models are a valid choice. In [26] an approach for collision mitigation is described, which predicts the lateral movement of relevant vehicles with a constant yaw angle and the longitudinal movement with a constant acceleration model.

A more sophisticated approach to tackle the problem of longitudinal motion prediction are Intelligent Driver Models (IDM), which are developed in [25] for the simulation of traffic flows in highway scenarios. The IDM computes the estimated acceleration of a vehicle, where the value depends on the current velocity, the distance to the vehicle in front and the desired velocity. When applied to the problem defined in Sec. 5.1, the output in contrast to the defined inputs depends on immeasurable quantities, for example the desired velocity of other traffic participants. In [106] the IDM approach is transferred to assess the driver intent at urban intersections where it demonstrated its ability to infer the intention of a right turn.

Another model driven approach is the map based prediction algorithm introduced by [107]. For each possible driving direction (the method is mainly tailored for intersections) a motion hypotheses is defined, where a Kalman-Filter generates uncertainty estimates by using pseudo update steps based on the road geometry. Using this technique, a pseudo probabilistic output can be generated. The approach assumes future position to be Gaussian distributed, where the estimates depend on the geometric configuration of the predicted vehicle within the road geometry and the dynamic state of the vehicle. More global information of the traffic situation is however not considered in the derivation of the position estimates.

5.2.2 Learning Based Models

Most expert models either lack to provide probabilistic information, a global scene understanding or the predictive power to handle time horizons larger than 2 s. To extend the prediction horizon, various machine learning approaches exist in related work.

For example [27] estimates a distribution of possible trajectories. The approach uses a Variational Gaussian Mixture Model, where the density function of future trajectories is determined by computing the conditional distribution of future trajectory

snippets given past trajectories. The evaluation of this algorithm shows promising prediction results. The work presented in [108] extends this approach by using a Mixture of Experts approach, which allows incorporating categorical information. The latter includes for example the topology of a road intersection.

A more general approach for scene understanding using trajectory clustering is described in [109], where a three-stage hierarchical learning process is implemented. By using video data motion patterns are learned and abnormalities of behavior are detected. The work presented in [110] proposes the use of a Hidden Markov Model based maneuver recognition, which distinguishes between ten different maneuver classes. On this basis a module for trajectory prediction and an Interacting Multiple Model to consolidate the trajectories are described.

Another method presented in [111] proposes the use of a Long Short Term Memory (LSTM) network. The authors show improvements compared to their previous work using the NGSIM dataset for evaluation. Please see [1] for more details on the NGSIM dataset. Also based on the NGSIM data for evaluation, [100] proposes the use of a LSTM network for predicting trajectories. The presented method provides single shot predictions with an average error of approximately 0.4 m at 5 s.

Using a fully-connected Deep Neural Net for learning the parameters of a two-dimensional Gaussian Mixture Model, another approach is presented in [112]. For each situation, an adapted Gaussian Mixture distribution models the probability density in the output dimensions a_x and v_y (see also Tab. 4.1). This distribution is then sampled to estimate trajectories. In the evaluation an average lateral error of approximately 0.5 m at a prediction horizon of 5 s was achieved.

Wissing et al. present in [113] a two-step approach. In the first step a regression technique based on Random Forests is used to estimate the time to the next lane change. In [103], this approach is extended and combined with findings of [79]. The estimated times to the next lane changes to the left and to the right are used as input for a cubic polynomial to predict future trajectories. In the evaluation an average lateral error of approximately 0.5 m at a prediction horizon of 3 s is achieved, assuming a perfect maneuver classification. The authors of [102] also present a two-step approach: In a first step, a MLP is used to estimate the future lane of a vehicle. In a second step, a trajectory realization is estimated using an additional MLP. The evaluation of the module which predicts the trajectories results in an average lateral error of approximately 0.23 m at a prediction horizon of 5 s. For a more extensive survey on position prediction see also [114].

5.3 Contribution

The main contribution of this chapter is the presentation of a real-time capable, interaction-aware, probabilistic, data-driven approach for vehicle position prediction. The estimates of the expected uncertainty are derived using real-world observations in the training phase of the model. These uncertainty estimates of future vehicle behavior are modeled explicitly in contrast to prior work, see for example [25] and [26].

To predict the distribution of future positions, besides the kinematic state of a vehicle, the maneuver recognition estimates are incorporated into the prediction of future positions. See chapter 4 for a survey how such kind of estimates can be derived. To predict future positions non-measurable features like 'desired velocity' [25] are not required. All inputs of the proposed algorithms (please see chapter 4) are measurable with current automotive sensors. Furthermore, the Markov Assumption, that all information is aggregated in the current state serves as a basis of the research which differs to many prediction approaches, where past behavior is used to compute future trajectories, see for example [27].

The method presented in this chapter is also able to handle the statistical underrepresented lane change case explicitly. Using the proposed strategy, important but rare occurring corner cases can be handled reliably, where conventional learning methods will run into problems, see as well [115].

The result is a novel method which is able to handle the task of probabilistic position prediction considering the numerous factors which may affect the future position of an object. A major advantage of this method is, that it not only derives precise estimates of future positions. It also provides the knowledge in which cases future positions are highly uncertain. This information is especially interesting for trajectory planning and risk assessment methods.

5.4 Solution Design

To solve the problem outlined in Sec. 5.1 a design tailored to the problem domain and complying to the black box architecture defined in Fig. 5.1 is presented in this section. The outlined problem is a probabilistic regression problem which can be treated as black box problem by machine learning techniques. Still, the problem of position prediction on highways can be decomposed based on human problem understanding in a divide-and-conquer manner.

Accordingly, the first design decision is to decouple the lateral and longitudinal movement of vehicles to be predicted. This decision is based on the a priori knowledge that the lateral behavior of a vehicle in highway situation is mainly influenced on the driving intention of a driver and the lateral movement state of a vehicle. The longitudinal behavior in highway situations on the other hand mainly depends on the current longitudinal movement state and the presence of a preceding vehicle and the desired velocity. This design decision additionally reduces the dimensionality of the problem to be solved, which reduces the computational effort at runtime. Reducing the dimensionality also allows models to generalize faster in the training process due to the increased density of data.

The second design decision is to split the lateral prediction problem into three sub-problems according to the maneuver classes defined in chapter 4. This means that the prediction of positions is solved for each maneuver class individually which results in three regression models. Using the probabilistic information of the inferred maneuver, the results can be combined afterwards. This decision is based on the observations of Sec. 4.8.3 on the frequency of maneuvers, where most samples are lane

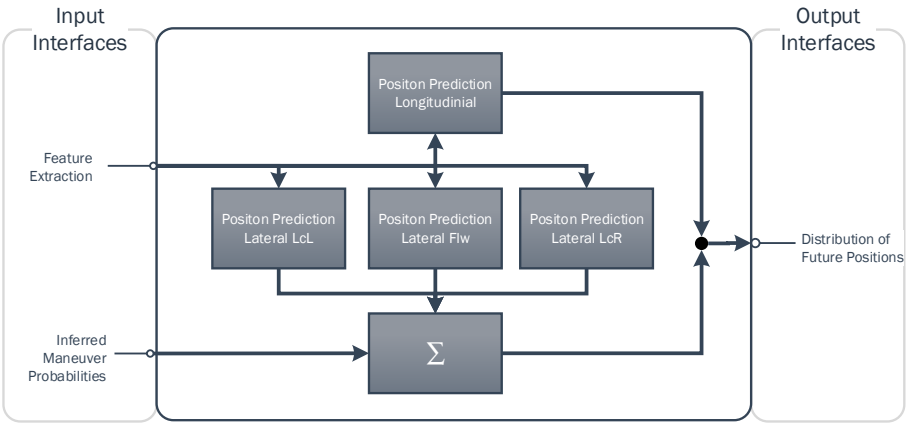


Fig. 5.2: *White box view on architecture level (3) on the position prediction method presented in this thesis, see also Fig. 3.6 for the embedding in the system context.*

following samples. The main goal of regression techniques is however to minimize the overall prediction error. In the case of lane change prediction, the data which is of interest covers only 1% of the data space. This is based on the insight, that only 1% of the data actually contains lane changes. This results in the problem of handling unbalanced class distributions, which is a problem which is widely studied in literature, see [116], [117]. The goal of the proposed method is to improve prediction results in lane change situations by explicitly generating models in the relevant data space. This is highly relevant for the fallback functions described in chapter 3. In addition, the second design decision also reduces the number of model dimensions needed, which again allows the models to converge faster with less risk of overfitting.

Both design decisions are reflected in the white box building block view, depicting the design on architecture level (3) in Fig. 5.2. To implement maneuver class specific regression models and fuse them afterwards a Mixture of Experts approach is chosen. This method is explained in detail in Sec. 5.6.2.

5.5 Features and Data Model

The features used for learning models which are able to predict future positions is a subset of the features defined in chapter 4. In order to reflect the architecture defined in Sec. 5.4, the features and the setup was done separately for the longitudinal and lateral dimension. The used approach is documented in the two following subsections.

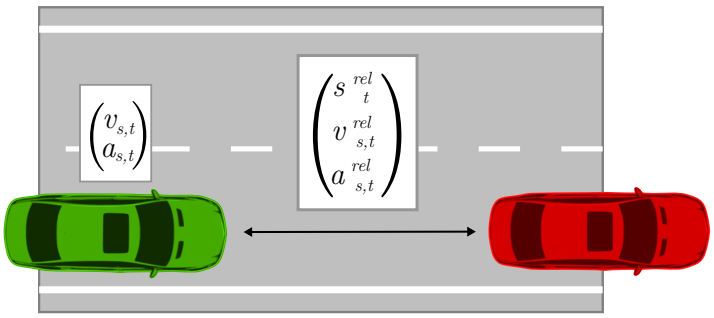


Fig. 5.3: *Input dimensions for longitudinal regression model in experiment 1.*

5.5.1 Data Model for Longitudinal Position Prediction

To train the longitudinal model the available data needs to be splitted into situations in which a leading vehicle was sensed, and into situations without a leading vehicle. According to the experiments in chapter 4, a curvilinear coordinate system along the curvature of the road was used, see Sec. 2.2.2.The longitudinal discretization chosen for the experiment was 1 m. For the generation of the data model the longitudinal measured state Z_t^{lon} of a vehicle at a time point t was defined.

$$Z_t^{lon} = \begin{pmatrix} t & v_{s,t} & a_{s,t} & s_t^{rel} & v_{s,t}^{rel} & a_{s,t}^{rel} \end{pmatrix}^T \tag{5.1}$$

In this context t is the current time point, $v_{s,t}$ the longitudinal velocity in the curvilinear coordinate system and $a_{s,t}$ the longitudinal acceleration. The relative measures to the vehicle in front on the same lane contain the relative distance s_t^{rel}, the relative velocity $v_{s,t}^{rel}$ and the relative acceleration $a_{s,t}^{rel}$. All relative measures refer to the difference of values between the defined pair of leading and following vehicle. For the sake of simplicity, the suffixes denoting the respective vehicle for which measures are computed are not denoted. The one vehicle of relevance is the preceding vehicle, see Fig. 5.3 According to the definitions of Sec. 5.6.1, an input vector $I_{t_0}^{lon}$ can be extracted from Z_t^{lon}.

$$I_{t_0}^{lon} = \begin{pmatrix} v_{s,t_0} & a_{s,t_0} & s_{t_0}^{rel} & v_{s,t_0}^{rel} & a_{s,t_0}^{rel} \end{pmatrix}^T \tag{5.2}$$

The containing parameters are the desired input of the prediction algorithm at the measurement time point t_0. The relative measures were removed from the vector in case no preceding vehicle was available in the data. In order to create a relation to future positions the output part O_t^{lon} needs to be defined.

$$O_t^{lon} = \begin{pmatrix} s_t & t \end{pmatrix}^T \tag{5.3}$$

For the generation of the states the traveled distance s_t at a specific time point $t + \delta t$ is computed using the naive Euler method. To do so δt and v_t and a piece wise constant velocity are assumed, where $s_0 = 0$ and $t_0 = 0$.

$$s_{t+\delta t} = s_t + \delta t \cdot v_t \tag{5.4}$$

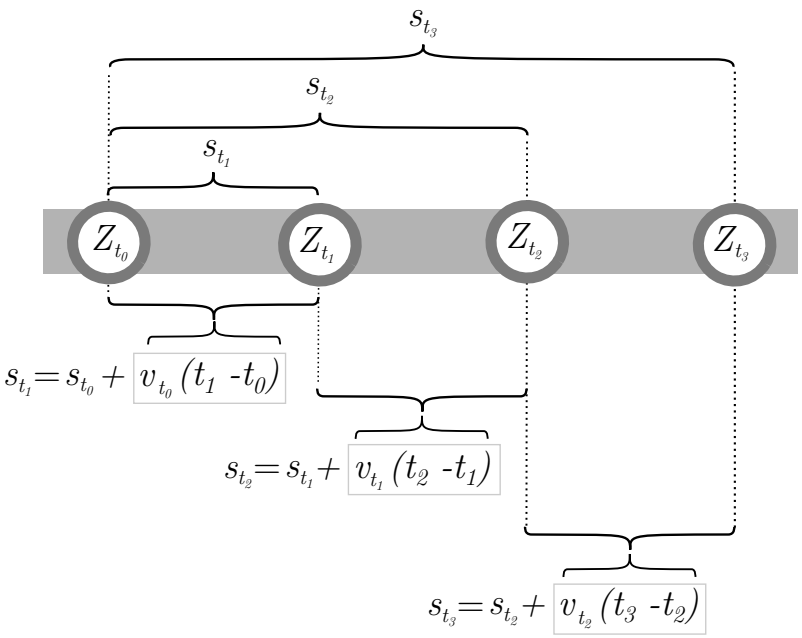


Fig. 5.4: *Generation of traveled distance s_t using the naive Euler method. The velocity values can be extracted from the respective measurements Z_t.*

The process of how the respective traveled distances were derived is also depicted in Fig. 5.4. The vectors $G_{\delta t}$ which are the input to the learner to estimate a distribution on in- and output dimensions can be defined using the respective vectors of the in- and output dimensions. Thus, $G_{\delta t}$ describes the relation between a given situation described by I_{t_0} and the output $O_{t_0+\delta t}$ at a time point $t_0 + \delta t$.

$$G_{\delta t} = \begin{pmatrix} I_{t_0} \\ O_{t_0+\delta t} \end{pmatrix} \tag{5.5}$$

Using this approach to model the dependencies, the number of samples is increased significantly. Given a number of n_z measurements and n_t number of future positions (at n_t times points), the total number of training data n_{train} is determined by multiplication of n_t and n_z.

5.5.2 Data Model for Lateral Position Prediction

In order to train the three experts (see also Fig. 5.2) for predicting the lateral positions the measurement vector Z_t^{lat} is defined as:

$$Z_t^{lat} = \begin{pmatrix} t & d_t^{cl} & d_t^{req,l} & d_t^{req,r} & v_{d,t} \end{pmatrix}^T \tag{5.6}$$

In this equation t denotes the current time point, d_t^{cl} the distance to the centerline of the lane the vehicle is currently assigned to, $d_t^{req,l}$ the distance to the centerline of the left neighboring lane, and accordingly $d_t^{req,r}$ the distance to the centerline to the right neighboring lane. The lateral velocity in the curvilinear coordinate system

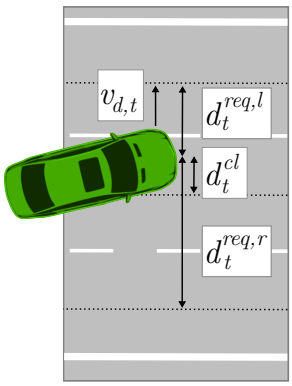


Fig. 5.5: *Input dimensions for lateral expert nodes in experiment 1. The doted black line visualizes the centerline of the respective lane.*

is denoted as $v_{d,t}$. Given a measurement $Z_{t_0}^{lat}$ at a time point $t = t_0$ three input vectors for the three experts $I_{t_0}^{lat,Flw}$, $I_{t_0}^{lat,LcL}$ and $I_{t_0}^{lat,LcR}$ can be extracted. These three vectors are defined by:

$$I_{t_0}^{lat,Flw} = \begin{pmatrix} d_{t_0}^{cl} & v_{d,t_0} \end{pmatrix}^T \tag{5.7}$$

$$I_{t_0}^{lat,LcL} = \begin{pmatrix} d_{t_0}^{cl} & d_{t_0}^{req,l} & v_{d,t_0} \end{pmatrix}^T \tag{5.8}$$

$$I_{t_0}^{lat,LcR} = \begin{pmatrix} d_{t_0}^{cl} & d_{t_0}^{req,r} & v_{d,t_0} \end{pmatrix}^T \tag{5.9}$$

where in accordance to Sec. 5.5.1 the containing parameters are the desired input of the prediction algorithm at the measurement time point t_0. The second part of the data, which is needed to train the models is the output vector O_t^{lat}. It is generated according the output vector used for the longitudinal prediction part.

$$O_t^{lat} = \begin{pmatrix} d_t & t \end{pmatrix}^T \tag{5.10}$$

To generate states of the traveled lateral distance, d_t can be computed in the same fashion as presented in Sec. 5.5.1 using the naive Euler method. The combined states $G_{\delta t}$, which are input for a learner which estimates a distribution over in- and output dimensions can be computed accordingly.

5.6 Methods

To predict distributions of future positions, a regression approach using Gaussian Mixtures as distribution model was chosen. The regression method is introduced in Sec. 5.6.1. As already outlined in Sec. 5.4, a Mixture of Expert design for the prediction of lateral positions is used. A brief overview on how this method works is given in Sec. 5.6.2. The section is completed by providing an overview of the techniques used for predicting the longitudinal behavior in Sec. 5.6.3.

5.6.1 Gaussian Mixture Regression

Regression methods are used for multiple applications. The most popular ones are the determination of dependencies of variables and predicting future values based on observed data. For the problem of position prediction the latter one is of interest. To solve the problem two things are basically needed:

- An input vector describing the current traffic situation and the vehicle state.
- An output vector containing future positions for respective time horizons.

These two vectors need to be set into a relation in order to estimate a distribution which models the dependencies between in- and output dimensions. A possible implementation is described within this subsection. The situation description and the vehicle state correspond to the input dimensions i. Future positions with the respective time points are included in the output dimensions o. When deciding for a probabilistic parametric method as required due to memory limitations, computational effort and architecture constraints Gaussian Mixtures are a handy choice as distribution model. To generate predictions, a conditional distribution in the output dimensions o can be computed based on a vector input in the input dimensions i.

To do so for a Gaussian Mixture distribution consisting of n components, where k denotes a single component from the set of components, the decomposition of the covariance matrices and means can be used.

$$\Sigma_k = \begin{pmatrix} \Sigma_{k,i} & \Sigma_{k,i,o} \\ \Sigma_{k,o,i} & \Sigma_{k,o} \end{pmatrix} \tag{5.11}$$

$$\mu_k = \{\mu_{k,i}, \mu_{k,o}\} \tag{5.12}$$

The index i again denotes the input- and the index o the output-dimensions. To compute predictions, one can derive the covariance matrices of the conditional output.

$$\Sigma_{k,o|i} = \Sigma_{k,o} - \Sigma_{k,o,i}\Sigma_{k,i}^{-1}\Sigma_{k,i,o} \tag{5.13}$$

The output mean of each component can be derived using:

$$\mu_{k,o|i} = \mu_{k,o} + \Sigma_{k,o,i}\Sigma_{k,i}^{-1}(I - \mu_{k,i}) \tag{5.14}$$

The conditional weights of the Gaussian Mixture can be computed by:

$$w_{k|i} = \frac{w_k p(I|\mathcal{N}(\mu_k, \Sigma_k))}{\sum_{k=1}^{n} w_n p(I|\mathcal{N}(\mu_n, \Sigma_n))} \tag{5.15}$$

The described regression algorithm provides the required probability density function in the output dimensions o. To benchmark the results, the value of the most probable value is of interest. This is particularly true for the comparison with classic

approaches. This can be realized according to [118] by computing the mean and the variance of the Gaussian Mixture.

$$\mu_{\sigma,o|i} = \sum_{k=1}^{n} w_{k|i} \cdot \mu_{k,o|i} \tag{5.16}$$

$$\Sigma_{\sigma,o|i} = \sum_{k=1}^{n} w_{k|i}^2 \cdot \Sigma_{k,o|i} \tag{5.17}$$

By using a machine learning approach for position prediction purposes, the question arises how trustworthy the output of the algorithm is. This is especially relevant in situations which are not similar to samples in the training dataset. To get an indication the size of the confidence interval γ can be used for modeling the similarity in the input dimensions i. In case of a Gaussian distribution, the size of a confidence interval can be approximated using the property, that the squared Mahalanobis distance is distributed according to χ^2 in the d_f dimensional space. Thus, by integrating the χ^2 distribution up to the squared Mahalanobis distance $m_{d_f,x}$ of a sample X, the minimum size of the confidence interval in which the value x is included can be derived. The size of the prediction interval γ_n can be computed by:

$$\gamma_n = F(m_{d_f,x}|v) \leq \int_0^{m_{d_f,x}} \frac{t^{\frac{v-2}{2}} e^{\frac{-t}{2}}}{2^{\frac{v}{2}} \Gamma\left(\frac{v}{2}\right)} dt \tag{5.18}$$

where $v \in \mathbb{N}^*$ are the degrees of freedom and Γ denotes the Gamma function. The value of γ_n corresponds to the probability enclosed by the unimodal multivariate Gaussian distribution with the index i. To extend the concept to a mixture of Gaussians, one has to weight the probabilities according to the their weights w_n. The probability γ_Σ enclosed by all Gaussian components can be computed by

$$\gamma_\Sigma = \sum_{n=1}^{k} w_k \gamma_k \tag{5.19}$$

and represents the uncertainty of a prediction.

5.6.2 Mixture of Experts

Multiple variants of the Mixture of Experts approach are popular, i.e. Hierarchical Mixture of Experts [119], Mixture of Experts for adaptive Kalman-Filters [120], and Mixtures of Experts of classification or regression models [121]. All of those methods have one main idea in common: They want to ensure that local information in the data is not *optimized out* by a global optimization process. Therefore, they are a solution for highly unbalanced classification problems and regression applications where local information in different parts of the dataset should be maintained. This is ensured by allowing the individual expert to specialize on smaller parts of larger problems, which is basically a divide and conquer strategy. According to [122] Mixture of Experts became popular for nonlinear classification problems, where the data contains natural distinctive subsets of patterns. The Mixture of Experts

approach implements a tree-structured architecture, where all Mixture of Experts models have three main components:

- Expert Functions which may be classification or regression models.
- Gating Functions which define data regions in which the individual Expert Functions are trustworthy.
- A probabilistic summation which combines the results of the experts based on the judgment of the Gating Functions.

For the problem of lateral position prediction as outlined in Sec. 5.1 the dataset can be divided along the maneuver classes $M = \{Flw, LcL, LcR\}$. In accordance with the definitions in chapter 4 Flw denotes lane-following, LcL lane change to the left and LcR lane change to the right situations. This is done in order to predict a distribution of lateral position $p(d)$ for each maneuver class, where d is the lateral position in a curvilinear coordinate system, see Sec. 2.2.2. To fuse the

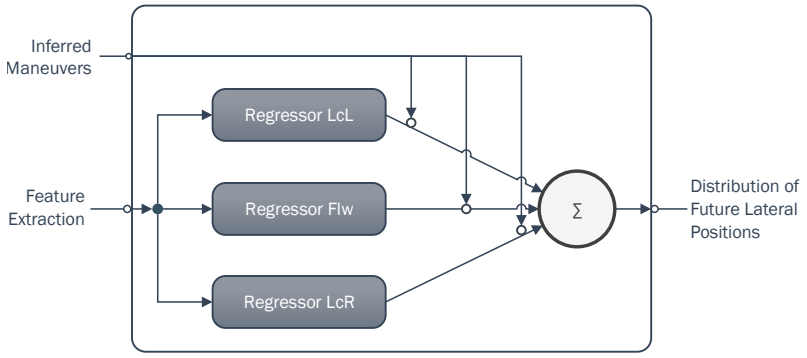


Fig. 5.6: *Mixture of Experts design as implemented for the lateral position prediction task in this thesis.*

three predicted distributions, the probability estimates P_m for each maneuver class (see (4.10)), which are computed by the algorithm presented in Sec. 4.7 are used. The combination of the lateral maneuver specific distributions $p_m(d)$ to one lateral distribution of positions $p(d)$ can be computed straightforward.

$$p(d) = \sum_{m}^{M} P_m * p_m(d) \tag{5.20}$$

In this context P_m is the probability of the inferred maneuver m out of the set of Maneuvers M and $p_m(d)$ the lateral probability distribution of the regression model for maneuver m.

5.6.3 Longitudinal Position Prediction Methods

The motivation behind generating a method for learning longitudinal future positions is mainly to achieve better results than naive techniques as for example the constant

velocity model. When solving the problem of probabilistic position prediction, the Gaussian Mixture Regression as described in Sec. 5.6.1 can be used. However, the problem, as described in Sec. 5.4 is quite different in the lateral and in the longitudinal direction. When observing real traffic, one may come to the conclusion that in many situations on highways the assumption of constant velocity is quite precise for a limited time horizon. To use this insight, a second method for predicting the longitudinal behavior of vehicles is presented in this thesis. Its novel idea is to learn the deviations to the constant velocity assumption, where the change to a constant velocity behavior is mainly the information one is interested in. To reflect this insight, a second output part $O_t^{\kappa,lon}$ was defined.

$$O_t^{\kappa,lon} = \begin{pmatrix} s_t - (t-t_0)v_{t_0} & t \end{pmatrix}^T. \tag{5.21}$$

This output is used for the same purpose as O_t^{lon}. However, the notation κ refers to a formulation of the prediction problem, where deviations are predicted instead of full future positions. From a data perspective the motivation behind this strategy is twofold. On the one hand one may hope for a reduction of the spread of the data in the 7-dimensional space. By having a more dense data representation, the task of learning probability density functions may become easier. On the other hand, the approach focuses on the part of the information, which is intuitively of interest.

To be able to compare the different approaches, the four models which are investigated are defined explicitly in the following. The first and most simple model is the constant velocity (CV) model, which is defined by:

$$s_{cv} = s_{t_0} + v_{t_0} \cdot (t - t_0) \tag{5.22}$$

When taking acceleration into account, the constant acceleration (CA) model can be defined accordingly.

$$s_{ca} = s_{t_0} + v_{t_0} \cdot (t - t_0) + \frac{a_{t_0}}{2} \cdot (t - t_0)^2 \tag{5.23}$$

The function which predicts future positions based on data vectors $G_{\delta t}$, as explained in Sec. 5.6.1, is denoted as:

$$s_{gmr} = \mu_{\sigma,o|i} \tag{5.24}$$

Accordingly the method which predicts deviations to the CV model is defined by:

$$s_{gmr^\kappa} = \mu_{\sigma,o^\kappa|i} + s_{cv} \tag{5.25}$$

where $\mu_{\sigma,o|i}$ and $\mu_{\sigma,o^\kappa|i}$ are the derived means of the predicted Gaussian Mixture distributions as defined in Sec. 5.6.1.

5.7 Metrics

There are many possibilities for an error metric to evaluate the results of a regression algorithm. Popular approaches are the Mean Squared Error (MSE), the Mean

Absolute Percentage Error (MAPE) or the Mean Absolute Error (MAE). To reduce the dependency of the achieved result to outliers, in this chapter the MAE is used. Because the problem of outliers also holds true when computing the standard deviation, the more robust Mean Absolute Deviation (MAD) is used in the following accordingly.

However, all these methods have a disadvantage for the envisioned application when used naively. By computing:

$$MAE = \frac{1}{l}\sum_{j=1}^{l}|f_j - y_j| \tag{5.26}$$

with predictions f_j and measured values y_j for l samples, all information about the temporal distribution of the errors are removed. To handle this issue, the errors can be separated according to their prediction horizon. To allow statistical computations, e.g. calculating the median, the errors can be cumulated in equidistant bins. Using these bins the statistical properties of a prediction model as a function of time can be investigated.

Besides measuring the statistical properties of the predicted means, the goodness of the predicted distribution has to be evaluated, too. The main challenge in evaluating the predicted distributions is, that for each ground truth value one predicted distribution is available. Thus computing the likelihood is not a feasible approach to assess the prediction performance. What can be done however, is the use of the properties of the confidence intervals. Having a measurement for the traveled way at a specific time point after the prediction, for example $s_{\delta t}$ one can compare it with the Gaussian Mixture defined by n values of the $\Theta_{k,o|i}$ parameter sets. The confidence values γ_{eval} corresponding to the 'enclosed probability' are defined as follows:

$$\gamma_{eval} = \sum_{k=1}^{n} w_k \gamma_{k,s_{\delta t}} \tag{5.27}$$

$\gamma_{k,s_{\delta t}}$ can be computed again, using the property that the squared Mahalanobis distance m_k^2 for an unimodal Gaussian k is distributed according to the χ^2 distribution.

$$m_k^2 = (x_{\delta t} - \mu_{k,o|i})^{-1}\Sigma_{k,o|i}(x_{\delta t} - \mu_{k,o|i}) \tag{5.28}$$

When evaluating the confidence values derived based on a look-up at the χ^2 distribution, the confidence values should be distributed linearly. For example, the percentage of values with $\gamma_{eval} \leq 0.2$ should be 20%.

5.8 Experiment 1

The first experiment on position prediction focuses on whether and how good the proposed method is able to solve the outlined prediction problem presented in Sec. 5.1. The section is structured as follows. In Sec. 5.8.1 a brief overview on the setup and the steps to generate the models is presented. The derived models are evaluated in Sec. 5.8.2. The findings of the experiment are discussed in Sec. 5.8.3.

5.8.1 Setup and Training

The investigations in the first experiment are based on the same dataset which was also used to execute the first experiment on maneuver recognition in Sec. 4.8. This means that series-production ready automotive sensors were used to collect the data.

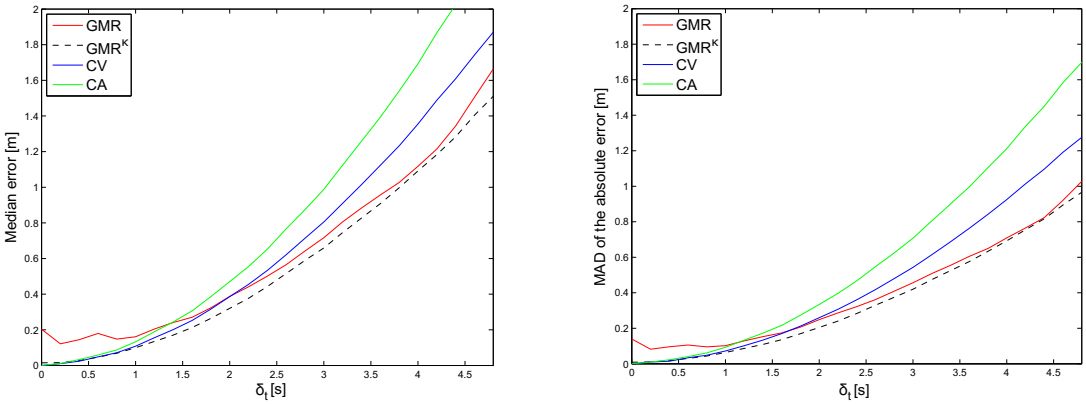
For training and the evaluation, each sample of the database was assigned to a maneuver-class $m \in \{Flw, LcL, LcR\}$. Within this process the dataset was divided into situations. Each situation consists of only one maneuver-class and one vehicle but multiple samples. The time-horizon, which was used for the labeling was defined for both lane change classes as the time-interval $0 - 5s$ before the vehicle is assigned to the neighbor lane (see Fig. 4.2).

5.8.1.1 Training of Models for Longitudinal Position Prediction

For the data setup of the longitudinal prediction model, only the measured object vehicles with a preceding second object vehicle were selected. Accordingly, $G_{t_0+\delta t}$ for δt up to 5s was computed for all vehicles, which were tracked long enough by the sensor setup. This resulted in an amount of 3.128.266 samples. From this data, two third were selected as training dataset. The minimum in the structural risk for learning the distribution from all samples G resulted in $n_{lon} = 90$ components for the GMR and in $n_{lon}^{\kappa} = 23$ components for the GMR_{lon}^{κ} model.

5.8.1.2 Training of Lateral Expert Nodes

The maneuver probabilities which were used as output of the Gating Nodes (see also Sec. 5.4) P_{LcL}, P_{LcR} and P_{Flw} were used from the Random Forest model developed in Sec. 4.8.3. The total number of data available was 160.781 samples. When balancing maneuver classes the number was reduced to 4.097 samples. This means for the training of the LcL and the LcR regression model less than 1% of the data can be basically used for the respective model.The training data for the individual experts was chosen by selecting it according to a winner-takes-all strategy using the probability estimates of the Random Forest Model. The training of the mixture models using the EM algorithm with the structural risk as optimization criterion resulted in $n_{lat}^{Flw} = 20$, $n_{lat}^{LcL} = 19$ and $n_{lat}^{LcR} = 15$ number of components of the GMMs for the respective maneuver class.



(a) *MAE vs. time* **(b)** *MAD vs. time*

Fig. 5.7: *Comparison of the CV, CA, GMR_{lon} and GMR^{κ}_{lon} model vs. predicted time horizon regarding the MAE and MAD.*

5.8.2 Evaluation

In the following an overview on the achieved results of the first experiment on position prediction will be given. The evaluation is separated into three parts. In the first in Sec. 5.8.2.1 the results in the longitudinal domain are presented. The evaluation continues with Sec. 5.8.2.2 which investigates the Mixture of Experts system in the lateral domain. Finally, in Sec. 5.8.2.3 some insights in the properties of the derived model in a test vehicle in field application are shared.

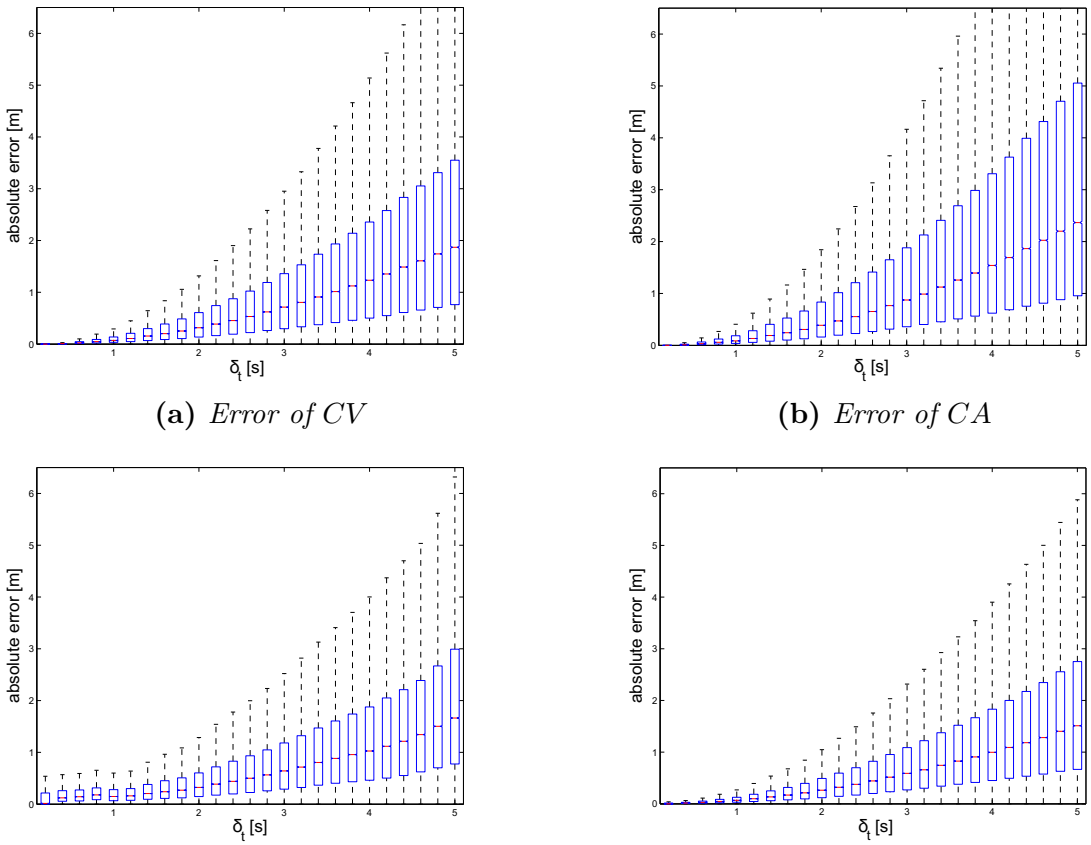
5.8.2.1 Evaluation of Longitudinal Prediction Results

Focusing on lane-following situations when a preceding vehicle is available, the different methods presented in Sec. 5.6.3 were evaluated. The methods which were evaluated are accordingly CA, CV, GMR_{lon} and GMR^{κ}_{lon}. The evaluated properties are mainly the MAE and the MAD. Based on the insights which are derived in this evaluation, the quality of the uncertainty estimates are evaluated for the best performing model. Additionally, a brief look will be taken at the behavior of the proposed method in corner cases. All evaluations were executed on the evaluation set of the data, which consisted of one-third of the total available data.

In a first step, the overall performance was evaluated on the whole evaluation dataset. In Fig. 5.7a the prediction accuracy in terms of the MAE of all models are visualized. As can be seen, both GMR methods clearly outperform the performance of the CV and CA methods for $\delta_t \geq 2.5s$, where the GMR^{κ}_{lon} method shows the best results. The constant acceleration assumption only holds true for very short time horizons. When comparing the GMR_{lon} with the GMR^{κ}_{lon}, the first one is slightly biased when compared to the second model. This especially holds true for short prediction horizons.

Besides the absolute error also the spread of the prediction error is of interest. As can be seen in Fig.5.7b the MAD of the error of the GMR methods are significantly

lower for $\delta_t \geq 2.5s$ compared to the CV and CA model. As in the evaluation of the absolute errors the constant acceleration model shows the worst performance. When looking at the learned models, the GMR^{κ}_{lon} model is superior to the GMR_{lon} model, especially for $\delta_t \leq 0.8s$. The errors for short prediction horizons in the GMR_{lon} model can be interpreted as the bias of the models. The GMR^{κ}_{lon} model combines the precision of a physical model for short prediction time horizons with the situational awareness of the GMR_{lon} model. This is interesting especially due to the fact that it uses only $n^{\kappa}_{lon} = 23$ components compared to $n_{lon} = 90$ in the standard GMR_{lon} model. To give an impression of the overall prediction performance of the models,



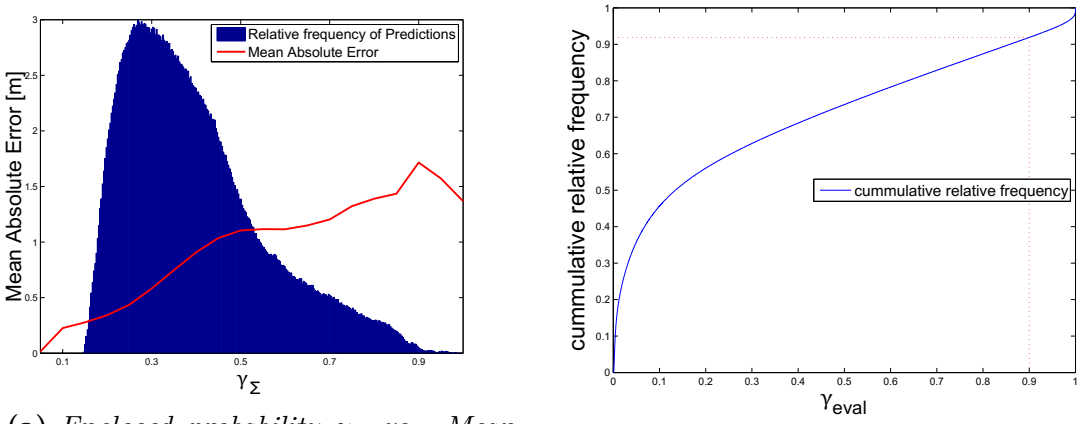
(a) *Error of CV* **(b)** *Error of CA*

(c) *Error of GMR* **(d)** *Error of* GMR^{κ}

Fig. 5.8: *Boxplots of the CV, CA,* GMR_{lon} *and* GMR^{κ}_{lon} *model vs. predicted time horizon. The whisker visualize 99.3% coverage of the data, the red line the median and the boxsize a 50% coverage of the data.*

their statistical properties are visualized in the boxplots in Fig.5.8a-5.8d.

As the GMR^{κ}_{lon} showed the best results in evaluation, its properties regarding the estimation of uncertainties are discussed in the following. In Sec. 5.6.1 γ_{Σ} is proposed as a measure on how certain a prediction is. In Fig. 5.9a the prediction error strongly correlates with γ_{Σ} in the input dimension. This intuitively corresponds to the expected behavior. The more similar situations are to the training data, the less error prone predictions will be. Regarding the proportion of predictions, which



(a) *Enclosed probability γ_Σ vs. Mean absolute error*

(b) *Cumulated histogram of γ_{eval}*

Fig. 5.9: *Evaluation of the goodness of the distribution estimates in the longitudinal dimension.*

are highly uncertain, one can also discuss the blue histogram in the same figure. As expected, the frequency of predictions decreases, the larger γ_Σ representing the 'enclosed probability' gets.

In Fig. 5.9b the comparison of the measured values with the predicted Gaussian Mixture distributions using the confidence intervals γ_{eval} in a cumulated histogram is depicted. The ideal predictor would show a diagonal line starting bottom left raising to the top right. This is obviously not the case. As the cumulated relative frequency rises faster as the values for γ^{κ}_{eval}, this indicates that the variances of the predicted Gaussian Mixture distributions are partially to large. On the other hand, the red lines, which are plotted for a region of 90% confidence show that a similar part of the predicted data is within this interval. This shows the ability of the algorithm to deliver an error range of the prediction error, which is an important property for the envisioned application.

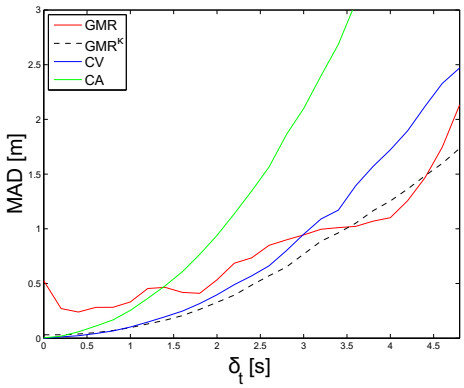


Fig. 5.10: *MAD depicted for error for cases $v^{rel}_{t_0} < 3\frac{m}{s}$.*

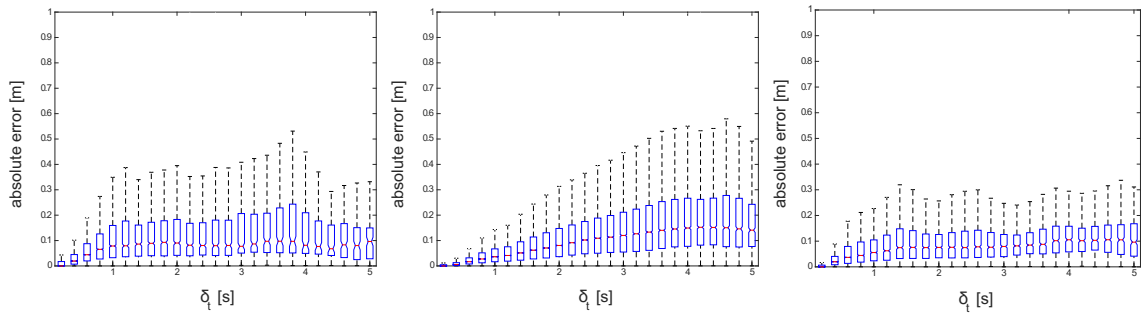
Finally, the properties of the derived model are investigated in situations with conflict potential. To do so, situations were selected, in which a faster vehicle

approaches a slower preceding car. In Fig. 5.10 all four models are evaluated for situations where the relative velocity of the two vehicle exceeds $3\,ms^{-1}$. In the figure, as expected, the difference of the MAD to the constant velocity approach increases significantly compared to the evaluation on all scenarios.

5.8.2.2 Evaluation of Lateral Prediction Results

The evaluation of the lateral prediction is less extensive than the evaluation of the longitudinal model. There are several reasons for this. The primary reason is, that the main properties of the GMR method were already investigated in the evaluation of longitudinal position prediction. The second reason is, that while the constant velocity and acceleration model at least sound feasible for longitudinal behavior of vehicles on highways, their assumptions are pointless for the prediction of the lateral positions. Accordingly, only the properties of the GMR model were evaluated. Again, all evaluations were executed on the evaluation set of the data, which consisted of one-third of the total available data.

For the evaluation of the prediction performance the differences were calculated between predictions and the available pseudo ground-truth from the evaluation data. For the evaluation of the expert nodes, the label information was used to choose the right expert for evaluation. Again, as in the evaluation of the longitudinal models,



(a) GMR_{lat}^{LcL} **(b)** GMR_{lat}^{Flw} **(c)** GMR_{lat}^{LcR}

Fig. 5.11: *Evaluation of the experts nodes.*

the prediction performance vs. the prediction horizon is of interest. Therefore, the errors vs. the time horizon were visualized using a boxplot for every expert. Please see Fig. 5.11a, 5.11b and 5.11c for the results. As can be seen, the results for the lane change classes LcR and LcL are more noisy compared to the Flw class. Reasons for this can be seen in the fact, that only 1% of the data contains lane change trajectories. Additionally, not all trajectories were tracked for the whole time horizon in the dataset, such that the density of the data for higher prediction horizons was relatively sparse compared to short horizons. Even though, it is remarkable in this context, that the median error in none of the experts, at no time-horizon, exceeds $0.2\,m$. In the evaluation of the overall performance, which is evaluated in the same way as for the experts, the error of the Mixture of Experts approach (see Fig. 5.12) is strongly dominated by the results of the Flw class, which contains 99% of the

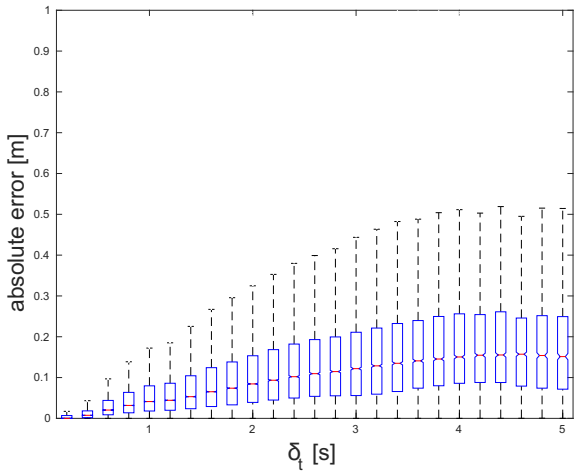


Fig. 5.12: *Absolute error of the predicted mean of the Mixture of Experts approach vs. the ground-truth. The whiskers denote approximately 99.3% coverage of the data, and the box approximately 50% coverage, assuming normal distributed errors.*

data (see Fig. 5.11). However, the overall prediction results are quite precise. At no time-step of the prediction horizon, the median error of the Mixture of Experts model exceeds $0.2\,m$. While the error of the longitidunal model increases, as the prediciton horizon increases, the error in the lateral domain seems to be constrained asymptotically. A possible explanation for this behavior is the width of the road, which is an upper bound for the prediction error in the lateral domain.

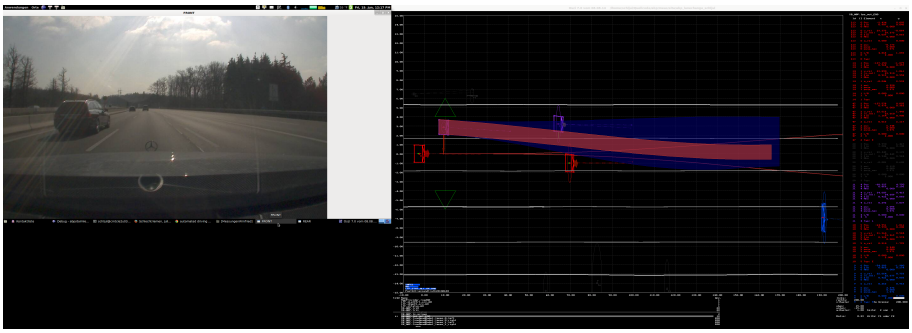
5.8.2.3 Evaluation in Test Vehicle

Besides the offline evaluation of the collected data, the derived models of the first experiment were also evaluated in a test vehicle. The experiments on the algorithms were executed open loop in real traffic on German highways. The prediction results were visualized and inspected visually by the co-driver.

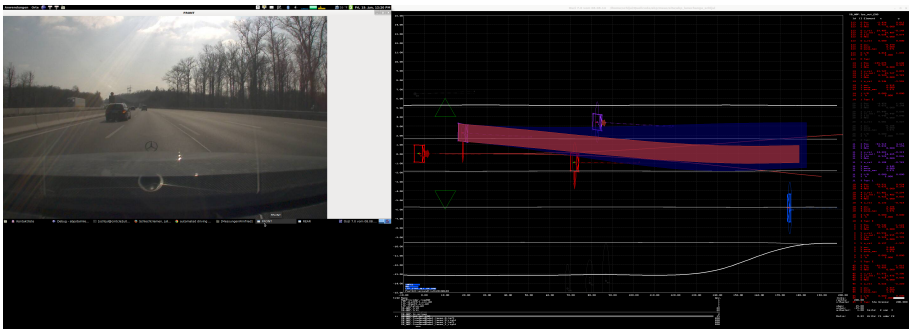
At no time point the results seems to be unfeasible. The strongest issue which was detected is related to the estimation of maneuver probabilities. Even a low, but significant probability of a lane change can result in a situation where the predicted position blocks two lanes. However, these findings are based on the confidence level of $99,73\%$ which was chosen in the test vehicle. Please see Fig. 5.13 for snapshot on the executed tests and especially Fig. 5.13c for a situation in which the maneuver classification is uncertain, resulting in a high uncertain position prediction.

5.8.3 Conclusion

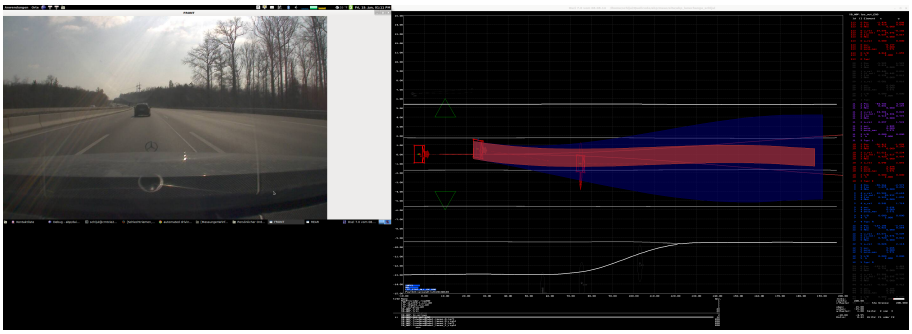
The first experiment illustrates the benefits of a data driven approach for position prediction on highways. The proposed method was demonstrated to be superior to classical techniques for position prediction which assume constant acceleration or constant velocity of the vehicle to be predicted.



(a) *Situation before lane change*



(b) *Situation in the middle of a lane change*



(c) *Situation after lane change*

Fig. 5.13: *Position prediction evaluation closed loop in experimental vehicle. The red area visualizes the mean of the prediction, the blue area the bounds of a 99.73% confidence interval. The size arrows at the front and the right and left side visualize the proportion of the estimated maneuver class probabilities.*

In the investigation of the longitudinal models (Sec. 5.8.2.1), the model which predicts the deviations to the constant velocity hypothesis showed superior results in terms of the mean and the spread of the prediction error. This is especially remarkable as this model has a lower complexity compared to the model which solves the whole problem of position prediction. The benefit which comes hand in hand with this model are substantial reduced run-time and memory requirements when applying it. Thus, it can be concluded, than whenever a feasible and simple analytical model is available to solve parts of a prediction problem it shall be used. The machine learning based model can then focus on learning the errors of this simplified model to improve the overall prediction performance.

The evaluation of the goodness of the PDFs of future positions (Fig. 5.9b) showed that the predicted distributions can be used to derive meaningful confidence bounds of the predicted values. The probability distributions are however not highly precise, and are only an approximation for the exact probabilities especially in the low confidence areas of the distribution. Thus, when setting thresholds for the confidence area, one needs to consider the evaluation of the respective model to determine the correct value. Additionally, the method is able to provide meaningful estimates of the uncertainty of its predictions (Fig. 5.9a).

The evaluation of the Mixture of Experts model used for predicting the lateral positions (Sec. 5.8.2.2) showed also promising results. It was demonstrated, that the estimates of future position provided by the Mixture of Expert approach have a median error of less than $0.2\,m$ for prediction horizons up to $5\,s$. However, the evaluation showed some potential for further experiments. On the one hand the slightly noisy evaluation results indicate, that increasing the number of lateral trajectories available for training and evaluation may be beneficial. Also, an improved maneuver classification algorithm delivering more precise probability estimates for time horizons up to $5, s$ is assumed to improve the prediction results. To get an insight, whether those envisioned modifications really improve the performance of the algorithm, a second experiment was conducted.

5.9 Experiment 2

The research topic of the second experiment is the investigation how an improved classification model which was derived in Sec. 4.9 and an increased number of data can improve the results of the models developed in the first experiment, see Sec. 5.8. For the execution of the second experiment, the findings of the first experiment were considered. The work presented in this section was firstly presented in [76], where the contribution of the author of this thesis is mainly in the research strategy and the conceptual part.

5.9.1 Data Setup

The dataset used in this experiment is based on the data obtained from the second experiment on maneuver recognition in Sec. 4.9. Thus, the results of the developed models of maneuver recognition were used for the Mixture of Experts approach. In contrast to the first experiment on position prediction, the dataset was processed in the same manner for the lateral and longitudinal direction.

The dataset, which is intended for the training and evaluation of the regression models was processed in three steps.

1. In the first step the probability estimates of the derived classifiers in Sec. 4.9 were added to each measured samples in the respective samples in the dataset.

2. Using the data model as described in Sec. 5.5.1 the pseudo ground truth in s and d direction was derived, please see [76] for a more detailed explanation of the pre-processing. This process resulted in a multiplication of the size of the dataset by a factor of 70.

3. The derived dataset was split in two parts, one for training and one for evaluation. The training dataset contained 130 000 and the evaluation set 20 000 trajectories.

The number of samples available in the second experiment highly outnumbers the number of samples in the dataset of the first experiment. Some features however, which were used for the first experiment, were not available in the second experiment. In the lateral dimension this resulted in the following input vector which was defined according to Sec. 5.5.2:

$$I_{t_0}^{lat,Flw} = I_{t_0}^{lat,LcL} = I_{t_0}^{lat,LcR} = \begin{pmatrix} d_t^{cl} & v_{d,t}^0 \end{pmatrix}^T \tag{5.29}$$

In the longitudinal direction, the relative acceleration was removed from the input vector defined in Sec. 5.5.1. This resulted in the adapted input dimensions as follows for situation where a preceding object was available.

$$I_{t_0}^{lon} = \begin{pmatrix} v_{t_0} & a_{t_0} & s_{t_0}^{rel} & v_{t_0}^{rel} \end{pmatrix}^T \tag{5.30}$$

5.9.1.1 Training of Longitudinal Position Prediction Models

Extending the work presented in the first experiment, two models were trained for the longitudinal prediction task. The first model predicts future positions based on the current longitudinal motion state. This model shall be applied in situations, when no preceding vehicle is available in the range of sight of the sensor setup. The second model in accordance with the first experiment predicts the object following behavior. Accordingly, the input dimensions for the learning problem in accordance with Sec. 5.5.1 are defined as follows:

$$I_{t_0}^{lon,follow} = I_{t_0}^{lon} = \begin{pmatrix} v_{t_0} & a_{t_0} & d_{t_0}^{rel} & v_{t_0}^{rel} \end{pmatrix}^T \tag{5.31}$$

$$I_{t_0}^{lon,free} = \begin{pmatrix} v_{t_0} & a_{t_0} \end{pmatrix}^T \tag{5.32}$$

As a result of the first experiment, see also Sec. 5.8.2.1, the approach of predicting deviations to constant velocity was chosen. For the simplicity of notation, the label κ is not used and needed in the second experiment. This is because only the models predicting deviations were evaluated in the longitudinal dimension. For the training process, according to the training for the lateral models, the number of components was limited to 50 and full covariance matrices were used while fitting the Gaussian Mixture models variationally. Regarding the data 2 693 605 samples were used to fit the model where no preceding vehicle was available and 4 345 284 samples with preceding object.

5.9.1.2 Training of Lateral Expert Nodes

For the training of the expert nodes of the Mixture of Experts model, the dataset was split along the maneuver labels, which were defined according to Sec. 4.9.3. Because of the large number of the data, random undersampling was executed on the Flw class, to get a comparable dataset size for all three models. Beside the reduced time needed for training, also the problem of predicting the lateral lane following behavior in the lateral dimension seemed less complex compared to the lane change classes. Thus, the problem shall also be solvable with a comparable number of data compared to the lane change classes. In the undersampling process, the number of data for the Flw class was reduced by 95 %. This resulted in a number of 327 841 samples for the Flw, 201 066 samples for the LcL and 270 151 for the LcR class.

Using the derived three sub datasets, three GMMs were trained as experts for the Mixture of Expert approach, see Sec. 5.6.2. To bound the training complexity and after the findings of the first experiment, the maximum number of components was limited to 50, and full covariance matrices were used. Please see [76] for more details.

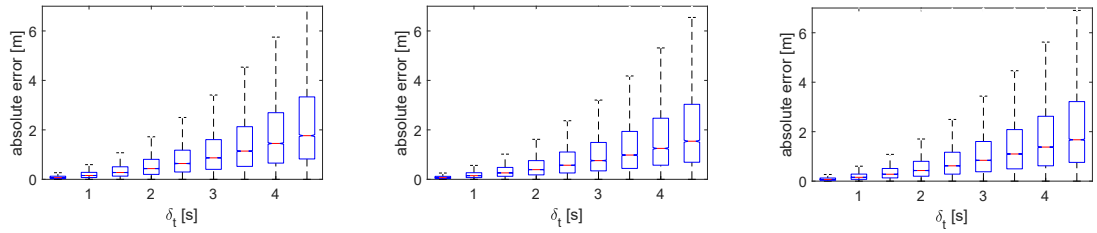
5.9.2 Evaluation

In the following subsection the results of the second experiment on probabilistic position prediction are presented and discussed. The evaluation is splitted into two parts. In the first part in Sec. 5.9.2.1 the results of the longitudinal prediction

models are presented. The second part of the subsection covers the evaluation of the Mixture of Experts model implemented for the prediction of lateral positions in Sec. 5.9.2.1. All evaluations were executed on the evaluation dataset, where 20 000 trajectories were used for the evaluation of the Mixture of Experts approach where the class distribution was equal to the statistical population. The evaluations of the experts were also executed each with 20 000 trajectories of the respective class.

5.9.2.1 Evaluation of Longitudinal Prediction Results

The evaluation of the longitudinal prediction mainly focuses on the predicted mean of future positions. The main difference to the evaluation in the first experiment in Sec. 5.8.2.1 is, that an explicit model GMR_{lon}^{free} handles situations without leading vehicle. Situations in which a preceding vehicle is sensed can be predicted using the GMR_{lon}^{follow} model, which is similar to the model evaluated in the first experiment. The combination of both models is denoted as GMR_{lon}^{all}. The results of the evaluation



(a) GMR_{lon}^{follow} **(b)** GMR_{lon}^{free} **(c)** GMR_{lon}^{all}

Fig. 5.14: *Evaluation of the longitudinal Gaussian Mixture Regression models. Fig. 5.14a displays the error distribution as a function of time for the model which considers a preceding vehicle. In Fig. 5.14b free driving with no sensed preceding vehicle is visualized. The overall results can be seen in Fig. 5.14c.*

are shown in Fig. 5.14. In case a preceding vehicle is present (Fig. 5.14a) the prediction results are slightly less precise when compared to free driving scenarios (Fig. 5.14b). This correlates with an intuitive situation understanding, where a preceding vehicle forces a driver to react, where the way how these reactions are executed depend on the individual driving style.

When comparing the achieved results with the first experiment where only situations with a preceding vehicle where evaluated, see Fig. 5.8c, the results are highly similar to the GMR_{lon}^{follow} model. The overall performance of the combination of both models in Fig. 5.14 shows a median error of round about 1.6 m at 5 s where 99.73 % of all predictions in the longitudinal direction are more precise than 7 m.

5.9.2.2 Evaluation of Lateral Prediction results

To evaluate the lateral prediction model one first has to decide for a model which is used as Gating Node, see Sec. 5.4 for the design of the method. The available models were presented in Sec. 4.9.3. However, in the evaluation of the classification

models in Sec. 4.9.4 the Random Forrest and the MLP model showed comparable good performance in solving the classification problem. For the task of predicting probability distributions of future positions it seems to be feasible to use the classification model, for which the output distributions of the Mixture of Experts model are 'best'. For scoring and comparing the distributions for the respective classification models, the average log-likelihoods $\overline{\mathcal{L}}$ were calculated on the (maneuver-) class-wise unbalanced evaluation dataset. The value of $\overline{\mathcal{L}}$ is derived by computing the average of the evaluations of the PDF at the respective ground-truth sample for every predicted distribution. In Tab. 5.1 the respective values $\overline{\mathcal{L}}$ for the different approaches for the

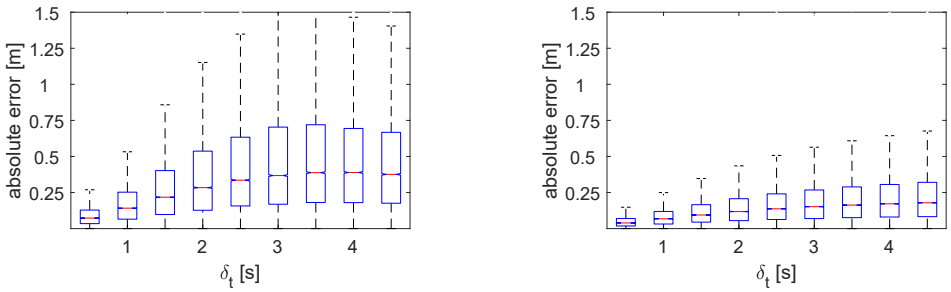
Tab. 5.1: *$\overline{\mathcal{L}}$ derived using the Random Forest and the MLP as Gating Node. The value derived for Labels uses the label information to select the right expert. The bold value denotes the respective best result.*

Classification strategy	$\overline{\mathcal{L}_\mathrm{d}}$
Labels	-7.547
RF	-7.626
MLP	**-7.608**

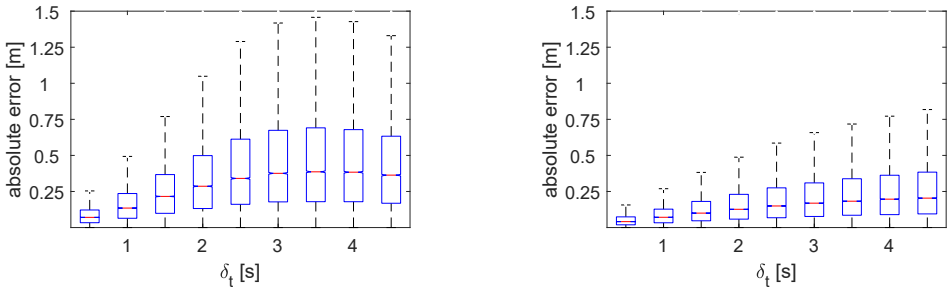
lateral position prediction are documented. For the computations, the results of the Random Forest and the MLP derived in Sec. 4.9.3 were used. As can be seen, the combination of the MLP with the three lateral expert models provides the best results. For a more detailed investigation please also see [76]. Thus, all evaluations presented in the following are based on the use of the MLP.

As in the first experiment, the overall prediction performance was evaluated by calculating the differences between the predicted mean and pseudo ground-truth on the evaluation dataset. In a first step the three experts were evaluated on samples belonging to the respective class of the expert. Please see Fig. 5.15 for the results. As expected the errors for the GMR_{lat}^{Flw} model used for lane following are the lowest. The results achieved for this model are very similar to the results achieved in in the first experiment, see Fig. 5.11 for a comparison. For both lane change classes the median position prediction results show comparable errors up to 0.4 m at 5 s. These errors are higher as the ones which were derived on the insufficient number of data in the first experiment. The spread of the errors on the 99.73 % of the data for both lane change classes is up to round about 1.5 m for every prediction horizon.

In the evaluation of the overall performance of the Mixture of Experts approach in Fig. 5.15d, the *Flw* class clearly dominates the results. This corresponds to the number of samples in the evaluation dataset as presented in Sec. 5.9.2. Regarding the predictive power of the method even for time horizons of 5 s the median error is better than 0.25 m and 99.73 % of all predictions at all evaluated prediction horizons are more precise than 0.8 m when compared with the mean of the predicted distributions. The increased error of the expert models of the lane change classes however result also in a slightly increased error of the overall lateral prediction performance, when compared to the first experiment. The reason, why the model performs slightly



(a) GMR_{lat}^{LcL} **(b)** GMR_{lat}^{Flw}



(c) GMR_{lat}^{LcR} **(d)** GMR_{lat}^{all}

Fig. 5.15: *Evaluation of the lateral position prediction in the second experiment. In contrast to [76], the number of samples for the respective evaluation time points is not constant.*

worse can only be assumed in the adapted feature space. The reduced dimensionality seems to hinder the models to take advantage of the higher number of data.

In Fig. 5.16 the Mixture of Experts approach is compared with the naive CV approach and a Mixture of Experts model which uses the label information as Gating Node. This is done to gain a better understanding of the goodness of the predictions. The Mixture of Experts approach performs significantly better than the constant velocity model. It is also interesting to see, that the predicted positions in the lateral domain are almost comparable to having a perfect classifier as Gating Node. The median prediction error at no time point exceeds 0.21 m within the prediction horizon of 5 s.

5.9.3 Conclusion

The second experiment completes the insights which were already gained in the first experiment. This includes the model for predicting longitudinal future positions in situations without leading vehicle and a more extensive evaluation of the lateral position prediction particularly scenarios with lane changes.

The investigations of the longitudinal models (Sec. 5.9.2.1) shows results which are highly similar to the results of the first experiment. Even though both datasets were collected on German highways, the number of data in the second experiment is significantly higher. The comparability of the results proves that the developed model

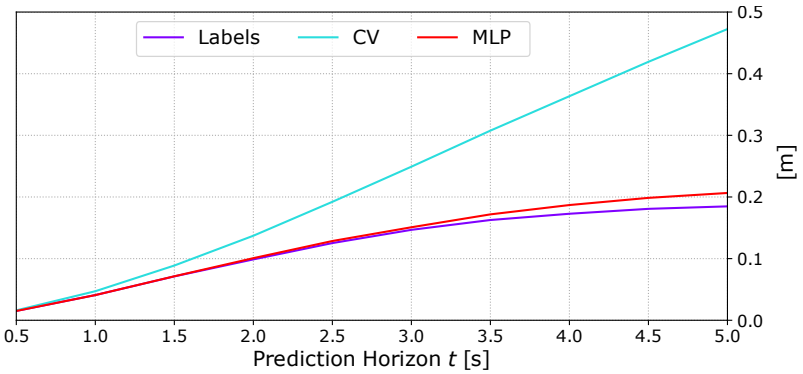


Fig. 5.16: *Comparison of lateral prediction results with reference systems. The constant velocity approach (cyan) and the method using labels as gating node output(blue) are depicted for better comparability of the performance of the Mixture of Experts approach (red).*

for longitudinal prediction generalizes well. The slightly higher error compared to the first experiment is assumed to depend on the missing value of relative acceleration as input dimension of the prediction models.

The evaluation of the Mixture of Experts model (Sec. 5.9.2.2) in the second experiment in the lateral dimension shows also promising results. The tendencies of the first experiment were proven to be right. As in the first experiment, the lane following class dominates the overall results.

The results of the experiment are compared to state-of-the-art approaches in Tab. 5.2. The mean of the predicted distribution of the proposed method are comparable to the results achieved by Yoon et al. in [102]. However, the method of Yoon et al. is lacking the ability to provide meaningful probability estimates. When interpreting the results however one has to consider, that all models were derived and evaluated on different datasets. Overall, the experiment shows that

Tab. 5.2: *Lateral Prediction Performance in Comparison to the first experiment and related work.*

Approach	$\tilde{E}_{y,t}$ [m]	Prediction Horizon t [s]
Experiment 1 (Sec. 4.8)	≈ 0.18	5.0
Yoon et al. [102]	0.23	5.0
Wissing et al. [103]	0.50	3.0
Mixture of Experts using MLP	0.21	5.0

the proposed method is able to provide results which are at least on par with state-of-the-art approaches for the prediction of future mean position of vehicles on highways. Besides this, the method delivers meaningful probability densities of future positions. This information is highly relevant when applying the method for risk assessment or trajectory planning. The evaluation also showed highly similar results to the evaluation of the first experiment. The flaws identified in the first experiment regarding the longitudinal and lateral prediction models were considered in the test

setup and model design. In the experiment the method proved to deliver precise probabilistic predictions which are an enabler for the envisioned use case of planning fallback trajectories. A possible next step to improve the prediction performance of the regression models could be the use of kernel density estimators. Using such kind of non-parametric technique, the probability density of future whereabouts could be derived without any artifacts introduced by a generative model based on a parametric representation. Such kind of approach however would increase memory requirements significantly. Another direction to optimize the prediction performance could be to continue the work on the models of maneuver recognition. A first step could be to add a probability calibration step at the output of the classifiers, see for example [123].

Chapter 6

Trajectory Planning in Structured Dynamic Environments

Humans drive with respect to future positions of other traffic participants. Such kind of prediction information can be derived for example by the method proposed in the previous chapter. When planning trajectories for an automated vehicle, the question arises how such kind of time-dependent information can be incorporated together with a representation of the static environment into the decision making and trajectory planning problem.

Accordingly, in the system concept (Sec. 3.5.2) a function block 'Trajectory Planning' is specified. This block is the last functional block specific to the functional chain of the fallback system introduced in chapter 3. Besides the envisioned use case as a trajectory planner for a fallback system the method presented in this chapter can be applied in different vehicle automation systems, e.g. as trajectory planner for standard automated driving in Level-3 and Level-4 automated systems.

In this chapter, a method for trajectory planning using the bicycle model (Sec. 2.4) is introduced. The method is able to consider future positions of dynamic and static obstacles using a flexible but generic data representation. The output of the method is a composite polynomial which is derived efficiently in real-time.

The chapter starts with the definition of the problem to be solved in Sec. 6.1. Related to the problem definition an overview on related work is presented in Sec. 6.2. Referencing the related methods, the contribution of this chapter is presented in Sec. 6.3. To provide the reader with an high-level overview of the method, the architecture envisioned to solve this problem is detailed in Sec. 6.4. Providing even more background information to better understand the separation between trajectory and behavior planning, a brief sketch on behavior planning is presented in Sec. 6.5. Based on the information which shall be provided by the behavior planner an interface to trajectory planning is proposed in Sec. 6.6. Using the available information of the dynamic freespace and the behavior, the way how the data provided by the interface can be processed and how it can be used to sample trajectories is presented in Sec. 6.7. The main output of this sampling step are safe support point sequences which represent the input of trajectory generation. Accordingly in Sec. 6.8 a flexible method of how trajectories can be generated considering the support points is

described. Subsequently in Sec. 6.9 the capabilities of the method are demonstrated in a simulated scenario and real-world driving. The chapter concludes with Sec. 6.10 in which the findings of the experiments are summarized briefly.

The method presented in this chapter was published beforehand in [124]. The main contribution of the author of this thesis is within the concept of problem design, the interface design, the way how the dynamic freespace is processed and the concept of how trajectories can be sampled within the dynamic freespace.

6.1 Problem Definition

When describing the targeted problem as a black box problem (see Fig. 6.1), planning a trajectory means to find a smooth trajectory from a 'Vehicle State' at a current time point to an unknown state of a vehicle at a future time point. This 'Vehicle State' shall include position, orientation, and its derivatives according to the bicycle model, see Sec. 2.4. To avoid collisions the 'Dynamic Free Space' shall be considered,

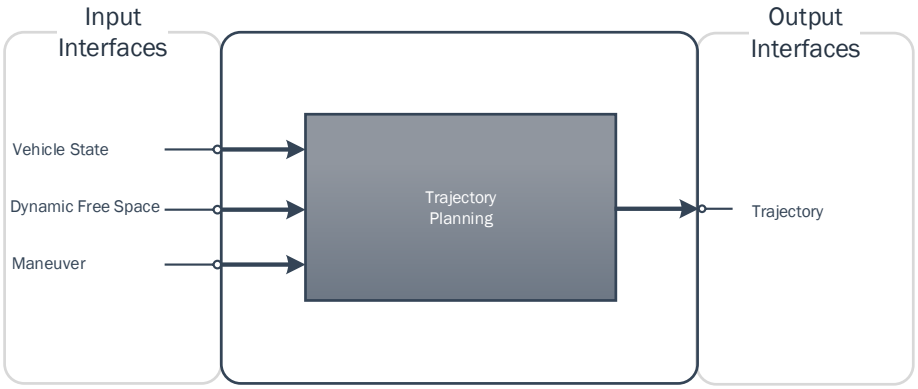


Fig. 6.1: *Architecture level (3) black box view of the functional block Trajectory Planning depicting the consumed information and the output to be computed, see Fig. 3.6.*

which can be interpreted as time-dependent geometric information in which the trajectory shall be planned. The missing information to plan a trajectory is the information of the 'Maneuver' to be executed, i.e. a lane change to the right or lane following. Within the planning black box the capabilities of the vehicle shall be considered, for example the maximum acceleration or the maximum steering angle.

When depicting the planning problem geometrically, Fig. 6.2 provides an overview how a typical situation may look like. Please see, that the curvilinear coordinate system defined in Sec. 2.2.2 is used.

In order to solve the planning problem, it needs also to be defined more formally. In the following it is described as a minimization of a cost function J. Given this cost function J to compute an optimal vehicle state $\boldsymbol{x}^*(t)$, and a control function

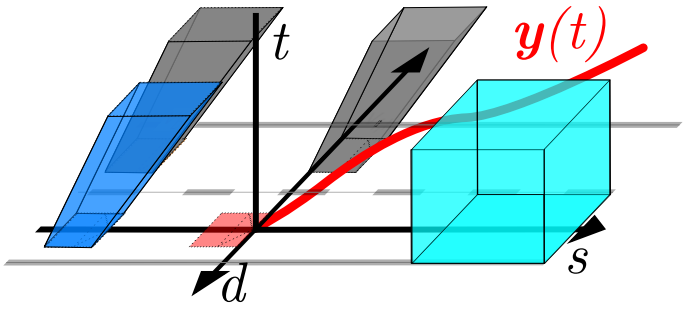


Fig. 6.2: *Schematic visualization of the planning problem from a geometrical perspective. The grey occupancies visualize the predicted positions of two vehicles on the left neighboring lane, the blue prediction shows a vehicle which is following the system vehicle. The cyan box may correspond to a static obstacle, its position is constant at all time points. In red, the system vehicle position coordinates $\boldsymbol{y}(t)$ are depicted.*

$\boldsymbol{u}^*(t)$ both depending on time the optimization problem can be described as follows.

$$\boldsymbol{u}^*(t),\ \boldsymbol{x}^*(t) = \arg\min_{\boldsymbol{u},\boldsymbol{x}} J(T_f, \boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{6.1}$$
$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{6.2}$$
$$\boldsymbol{u}(t) \in \mathcal{U} \tag{6.3}$$
$$\boldsymbol{x}(t) \in \mathcal{X} \tag{6.4}$$
$$\boldsymbol{y}(t) \in \mathcal{Y}(t) \tag{6.5}$$
$$t \in [0, T_f] \tag{6.6}$$

where $\boldsymbol{u}(t)$ is the control input of the vehicle at a time t, $\boldsymbol{x}(t)$ the vehicle state, and $\boldsymbol{y}(t)$ the vehicle position coordinates. Please consider that in the latter the argument t is often neglected for simplification purposes, for example by writing $\boldsymbol{x}(t)$ as $\boldsymbol{x}$.

Formally (6.1) is a kinodynamic trajectory planning problem as described in [125]. It needs to be emphasized that based on prediction information, time-dependent constraints need to be considered. The main challenge therefore is to solve (6.1) in real-time.

6.2 Related Work

One of the most popular methods in the context of automated driving is proposed by Werling et al. in [41]. The method is a sampling based approach working in a curvilinear coordinate system. It distinguishes between a finite set of maneuver options. Using these maneuver options, the author describes a method to plan jerk-optimal trajectories considering multiple traffic participants and lanes. It is important to understand, that the proposed method is not only a pure trajectory planning, but also a method of behavior planning. Most authors however architecture-wise decouple behavior planning from trajectory planning.

An example of a behavior planning method for road vehicles is presented in [126]. Within the paper, a solution for the combinatorial problem which maneuver

variant shall be chosen is presented. A variant herein describes for example in case of an overtaking maneuver which traffic gap shall be chosen. In accordance with the definitions in chapter 3 this discrete selection of maneuver-variants is called behavior-planning, following. When it comes to planning trajectories a solution is described in [127]. Within the proposed method, Sequential Quadratic Programming is used to plan a trajectory in a global coordinate system. The cost function which is minimized is pretty complex and incorporates a weighted combination of the lateral offset to the middle of the lane, the velocity difference to a desired velocity and high values of jerk and acceleration. The robustness of the method was proven on the Bertha Benz Memorial route on an autonomous ride between Mannheim and Pforzheim. One of the main drawbacks of the method is, that depending on the complexity of the constraints, the optimization process runs into a problem that is not solvable in real-time. Both approaches are however closely related to the work presented in this chapter.

Another approach, which is less related but is analyzed in this section is [128]. The method uses a Rapidly-exploring Random Tree (RRT) for motion planning, and proved to be feasible in the 2007 DARPA Urban challenge. In [129] a spatio-temporal-lattice is used for trajectory planning. It is shown, that the proposed method can be easily executed on a GPU in order to achieve real-time capability. While the use of the method on the one hand results in high requirements to the hardware, on the other hand the behavior planning problem is solved implicitly in the planning process. Another approach which explicitly addresses the inflexibility of many sampling based approaches regarding the shape of the trajectories is presented in [130]. To do so, the presented method piecewise concatenates motion primitives, where the approach is demonstrated to be feasible for the application of a small aerobatic helicopter.

6.3 Contribution

The main contribution of this chapter is a trajectory planning method, which is able to plan trajectories using a dynamic freespace representation which can be generated by arbitrary prediction algorithms. The method is inspired by and extends the work presented in [41], [127], [130] and [131]. The contribution can be divided into three parts.

The first part is the definition of a generic interface of dynamic freespace which on the one hand is able to handle arbitrary prediction information and on the other includes the information which kind of maneuver shall be executed. To generate a human-like driving behavior, the idea of [127] to penalize offsets from the center of the lane is modified towards defining the borders of a lane as a hard constraint within the interface. From an architectural point of view the defined interface allows the separation of the functions position prediction, maneuver planning and trajectory planning, see Fig. 3.6 for the context.

The second part of the contribution is the concept of how this dynamic freespace information can be processed to generate support points for the trajectories to be

sampled. Therein, the concept focuses implicitly on sampling along the curvilinear coordinate system which follows the curse of the road. The sampling is executed in the important parts of the solution space which is fostered by the way the freespace is represented. The presented method improves the concepts derived by [41] by avoiding sampling in occupied regions. This in contrast to approaches which are sampling the state space, where large numbers of planned trajectories in dense traffic are discarded due to collisions. It is important to understand, that the construction of the interface provides not only constraints but also a heuristic for trajectory planning which is independent of an underlying trajectory planning method. The approach results in an increased flexibility of trajectory shapes. Many sampling based approaches lack flexibility when it comes to the shape of planned trajectories due to their limitation to a fixed number of motion primitives. Even simple scenes which require a two-step strategy, like catching up to a gap and performing a lane-change afterwards, are often not directly solvable using these approaches.

The third part of the contribution is a method for trajectory generation which is based on composite polynomials of fifth order. The polynomials use the properties of differential flatness and are jerk optimal given the constraints defined explicitly by the vehicle states and the support points. Because jerk optimality is achieved without any iterative optimization, the method is computational inexpensive. The polynomials use the support points which are transformed back to the Cartesian coordinate system for trajectory generation. This guarantees driveability and jerk optimality for the generated trajectories, even when a real road contains discontinuities and curvature, which is the main difference to the method presented in [41]. Besides the envisioned use case of planning fallback trajectories, the introduced method is able to produce dynamic feasible results for different kinds of maneuvers, i.e. turning at intersections, lane-changing or roundabouts. This is achieved by using a longitudinal and lateral coupled model, which differs to known similar methods for the problem domain, where mostly path-velocity decomposition is used for trajectory generation.

The result of the three parts of the contribution is a method which is able to handle the challenging task of trajectory planning in dynamic environments considering arbitrary prediction information in real-time. Major advantages of the method are the architectural clean separation of the planning problem from the methods of object prediction, the high flexibility of the planned trajectory shapes even though using a coupled model and the computational efficiency.

6.4 Solution Design

In this section, a brief overview on the high-level design used to solve the problem as described in Sec. 6.1 is provided. The design complies to the black box architecture visualized in Fig. 6.1 and the L2 system design presented in Sec. 3.5.2. Within this section, four design decisions are presented which are the basis for the development of the presented trajectory planning method.

The problem of planning a trajectory in the dynamic freespace is treated as an optimization problem. In contrast to the presented methods for maneuver recognition

and position prediction in chapter 4 and 5, the uncertainty factor of human behavior is not part of the problem to be solved. Thus, the planning problem shall be solved without use of machine learning techniques. This however requires a lean and carefully designed interface.

Accordingly, the first design decision is, that the trajectory planner shall handle arbitrary prediction information in order to be usable for different applications. Therefore, an interface needs to be designed which provides a sufficient level of abstraction, includes all needed data, but only contains information which every method used for prediction can provide. Because not every planning algorithm is able to provide probabilistic information on future positions, this implies, that all probabilistic information needs to be transformed to non-probabilistic values based on for example the choice of a confidence level.

The second design decision relates to the sampling strategy. Instead of sampling in the state space like for example [41], sampling techniques shall focus explicitly on areas in which the chance of finding collision free trajectories is high. This task however can be executed in a simplified environment representation, in order to reduce the computational effort. This guardrail reduces the effort needed for collision checking after the sampling process which is typically the most computational expensive step. In order to provide higher quality solutions in the important parts of the solution space and to reduce the computational effort, this second design decision introduces more complexity in the sampling process. For further simplification, the sampling process shall be executed in a curvilinear coordinate system, where many geometric calculations can be substituted with simple inequality checks. However, no guarantees can be made on the geometric configuration of the sampled points. Thus, motion primitives cannot be used and challenging requirements regarding the flexibility of trajectories are laid down to the method of trajectory generation.

The third design decision affects the model assumptions made for trajectory generation. When generating trajectories to be executed in a Cartesian coordinate system no model assumptions which are violated obviously shall be made. This especially relates to the use of the lane coordinate system, where many methods use the lane coordinate system for planning and assessing the goodness of trajectories. This introduces physical implausibilities. For example the lateral and longitudinal acceleration of a trajectory in a bad case may have a maximum at a point of the trajectory where the curvature of the road has a maximum, too. Thus the decision is to use the Cartesian coordinate system for trajectory generation. A further popular approach, often called 'Path velocity decomposition' assumes that lateral and longitudinal trajectories can be planned separately. While this approach may work well on almost straight roads, its obviously not valid in every traffic environment and accordingly such kind of simplification may not be used in the trajectory generation process. This design decision however makes the planning problem more complex. Solving the optimization problem in the longitudinal and lateral domain at the same time increases its dimensionality.

The fourth design decision affects the complexity of the model used for trajectory generation. Even though it is possible to generate smooth curves using methods from CAD and Computer Graphics like Hermite Splines, the trajectories shall be

constrained by being compatible to the dynamic bicycle model (Sec. 2.4). In many applications, see for example [132], the non-linear bicycle model has proven to be a reasonable simplification of the dynamical behavior of a real vehicle. This design also allows trajectories to be checked simply using dynamic constraints like maximum acceleration or the steering rate. All four design decisions are reflected in the white

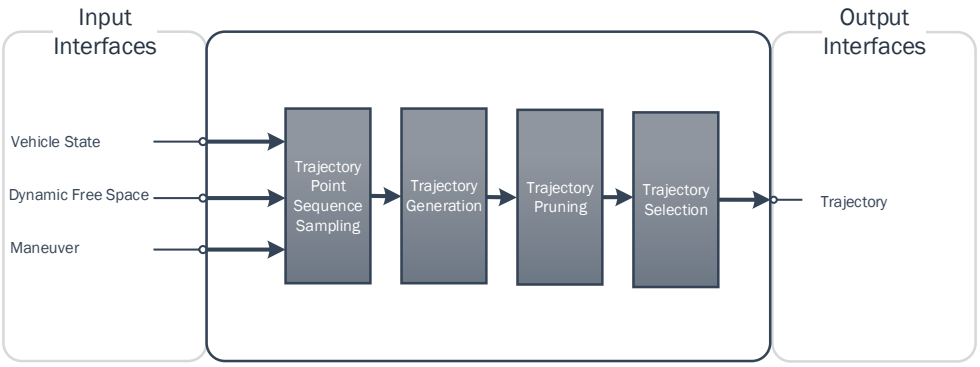
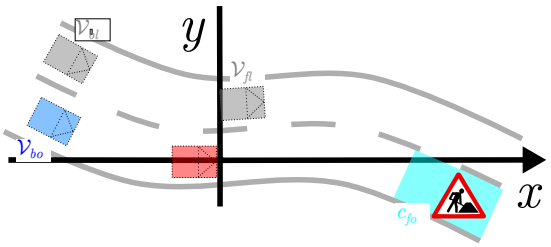


Fig. 6.3: *White box view on architecture level (3) on the trajectory planning method presented in this thesis, see also Fig. 3.6 for the embedding in the system context.*

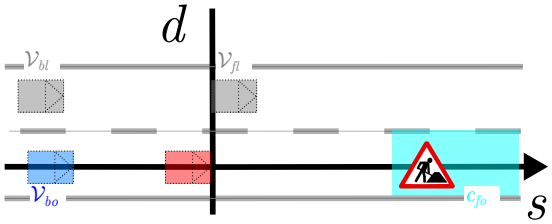
box building block view, depicting the design on architecture level (3) in Fig. 6.3. The method of trajectory planning as proposed in this chapter consists of four subsequent steps. In the step 'Trajectory Point Sequence Sampling' multiple sequences of support points for trajectories are sampled based on the dynamic freespace. These sequences can intuitively be understood as prototypes guiding the trajectory in the three-dimensional space. In the next step 'Trajectory Generation' for every sequence a respective jerk optimal trajectory is generated. The generated trajectories have to intersect all support points within the sequence. Subsequently, in 'Trajectory Pruning' trajectories which are affected by collisions or violate dynamic limits of the vehicle are removed from the set of generated trajectories. In the last step 'Trajectory Selection' the best trajectory with respect to the cost function is selected and send to the output interface.

6.5 Behavior Planning

Even though the problem of behavior planning is not within the scope of the trajectory planner, a sketch of which information shall be provided by such kind of function is needed. The key to behavior planning is scene-understanding. This understanding is hard to achieve in a Cartesian coordinate system. Humans for example determine the relevance of other traffic participants by their current and future lane-assignment. For this reason a curvilinear coordinate system along the curvature of the road is chosen, where the transformations are defined and explained in Sec. 2.2.2. In Fig. 6.4



(a) *Challenging situation for the red ego vehicle. The task is to plan a safe trajectory respecting other traffic participants and static objects.*



(b) *Transformation of traffic scene into curvilinear coordinate system using* Γ *(see Sec. 2.2).*

Fig. 6.4: *For task of planning trajectories, the traffic scene is transformed into a curvilinear coordinate system.*

the transformation is done for an exemplary challenging traffic scene which requires a complex coordination of longitudinal and lateral behavior.

After deriving the prediction information, by using the methods defined in chapter 5, the dynamic occupied space can be illustrated in the longitudinal direction in a path-time diagram (see Fig. 6.5). If the system vehicle, for which a trajectory shall be planned is on the right lane, in which the blue vehicle is relevant (see Fig. 6.4) on can see that at the time point $t = 0$ no collision is present. In the path-time diagram the area between the predicted occupancies of the blue vehicle and the cyan static obstacle visualize this property. The left neighbor lane, which is occupied by two faster grey vehicles shows a gap to merge into, which becomes relevant at $t = 0$ in the origin of the coordinate system. The higher gradient of the both grey predicted occupancies in the figure corresponds to the higher velocity of both vehicles. On the right lane, for a high value of s a non-moving obstacle is present, which behavior wise should result in a decision to brake or to change the lane. The area which is white only, is the longitudinal dynamic freespace for which both lanes are safe, and thus it is the space in which a lane change can be executed.

Based on the insights gained by the analysis of the path-time diagram, a safe driving sequence based on the dynamic freespace can be derived. Thus, the system vehicle can start with moving within the dynamic freespace of the right lane. Then the system vehicle can execute a lane change in the dynamic freespace which is collision free in longitudinal direction for both lanes. This freespace corresponds to the white area in the middle of Fig. 6.5. Subsequently, the vehicle can continue its

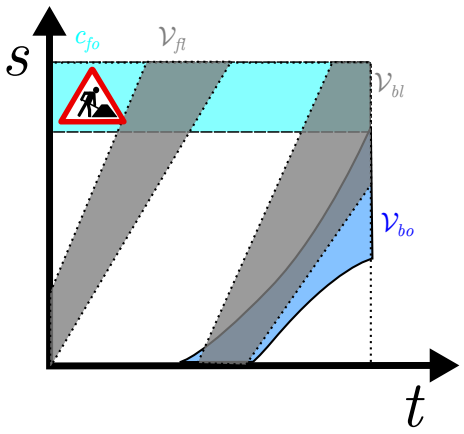


Fig. 6.5: *Using prediction information the occupied space can be visualized as a function of t, see Fig. 6.4b.*

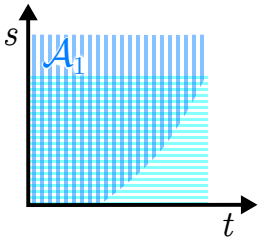
journey in the dynamic freespace between the predicted occupancies of both grey vehicles.

The sequence of dynamic freespace entities therefore allows expressing constraints to the trajectory planning problem while implicitly determining the maneuver to be executed at the same time. A structured method to model the freespace efficiently is presented in the following section.

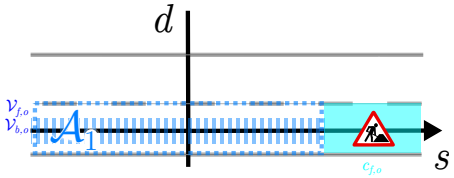
6.6 Interface Definition

When using the dynamic freespace of a traffic scene for trajectory planning as proposed in Sec. 6.5, a model is needed to represent this information. Such kind of model needs to provide sufficient constraints to describe the maneuver on the one hand without losing generality on the other hand. In the former chapter in Fig. 6.5 the future occupancies were depicted in a path-time diagram. This representation of data provides a solution space but does not determine the maneuver to be planned by a trajectory planner. To do so, the sequence of dynamic freespaces to be traversed needs to be determined.

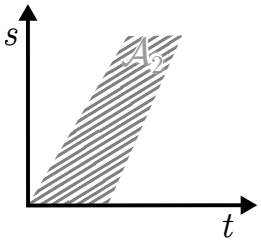
In order to describe such kind of sequence, a representation of a single element of dynamic freespace needs to be available. To derive such kind of representation it is handy to make a lane-wise inversion of the occupancies, for example of the ones depicted in Fig. 6.5. This information is visualized in Fig. 6.6. As can be seen in Fig. 6.6a the dynamic freespace for the right lane depicted blue is named $\mathcal{A}_1$, where the letter corresponds to the name of the dynamic freespace representation 'Action Space'. This 'Action Space' is defined as a connected volume in the t, s and d dimension where each point in this volume is collision free. It can be interpreted as representation of a dynamic safe driveable space. The depicted sectional view of $\mathcal{A}_1$ herein corresponds to the predicted gap on the right lane which can be seen in Fig. 6.5. Accordingly, $\mathcal{A}_2$ corresponds to the gap on the left lane, see Fig. 6.6c. The action space $\tilde{\mathcal{A}}$ corresponds to the white area in the middle of Fig. 6.5, and is the dynamic



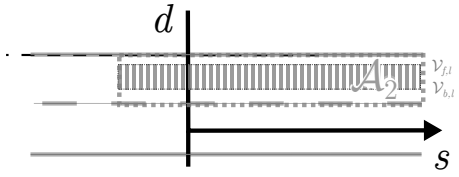
(a) *Dynamic freespace on the right lane, $\mathcal{A}_1$.*



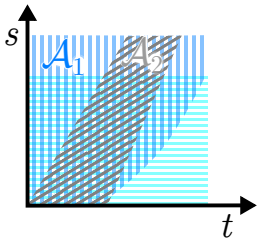
(b) *Projection of Action-Space $\mathcal{A}_1$ in the s-d plane.*



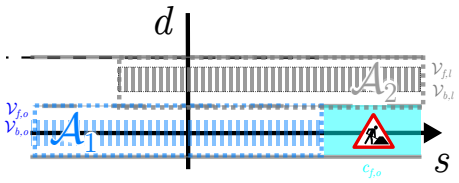
(c) *Dynamic freespace on the left lane, $\mathcal{A}_2$.*



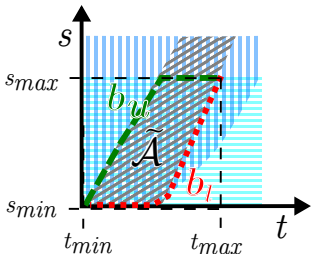
(d) *Projection of Action-Space $\mathcal{A}_2$ in a s-d view.*



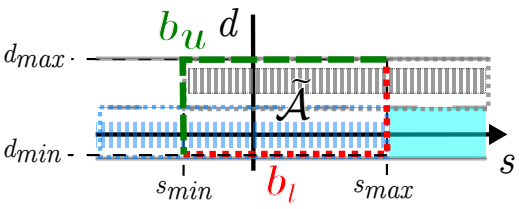
(e) *Overlay of $\mathcal{A}_1$ and $\mathcal{A}_2$ in t-s view.*



(f) *Projection of both $\mathcal{A}_1$ and $\mathcal{A}_2$ in a s-d view.*



(g) *Definition of $\tilde{\mathcal{A}}$ in t-s view*



(h) *Definition of $\tilde{\mathcal{A}}$ in s-d view*

Fig. 6.6: *Derivation of connecting Action Space $\tilde{\mathcal{A}}$ representing a locally and timely limited dynamic freespace. The representation can be derived by inverting the occupancies, see Fig. 6.5.*

freespace where driving is collision free in both lanes in the longitudinal direction, see Fig. 6.6g. Because it connects both lanes it is also named connecting Action Space. The red and green line b_l and b_u depict the lower and upper bound of $\tilde{\mathcal{A}}$.

Up to here only the longitudinal dimension was discussed in the description of the dynamic freespace and the derived Action Spaces. For planning trajectories however, both dimension need to be considered. In Fig. 6.6h the three Action Spaces $\mathcal{A}_1$, $\tilde{\mathcal{A}}$ and $\mathcal{A}_2$ are depicted in a sectional view on the *s-d plane*. The derivation process of this view is visualized in Fig. 6.6b, Fig. 6.6d and Fig. 6.6f. Again each Action Space can be described by an upper and a lower bound, see also the corresponding view in Fig. 6.6g.

While the Action Spaces $\mathcal{A}_1$ and $\mathcal{A}_2$ can be derived straightforwardly using the road geometry and the predicted occupancies, the derivation of the connecting Action Space $\tilde{\mathcal{A}}$ needs a detailed analysis. $\tilde{\mathcal{A}}$ can be defined as the connecting Action-Space of $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ if for all points $\tilde{P} \in \tilde{\mathcal{A}}$ the following holds true:

$$\begin{pmatrix} s_{\tilde{P}} \\ d_{\tilde{P}} \\ t_{\tilde{P}} \end{pmatrix} \in \mathcal{A}_i \qquad \text{or} \qquad \begin{pmatrix} s_{\tilde{P}} \\ d_{\tilde{P}} \\ t_{\tilde{P}} \end{pmatrix} \in \mathcal{A}_{i+1} \qquad \text{and} \tag{6.7}$$

$$d_{\tilde{P}} \in \mathcal{D}_{s,t}^{\tilde{\mathcal{A}}} \qquad \text{with} \qquad \mathcal{D}_{s,t}^{\tilde{\mathcal{A}}} = \mathcal{D}_{s,t}^{\mathcal{A}_i} \cup \mathcal{D}_{s,t}^{\mathcal{A}_{i+1}} \qquad \text{s.t.} \tag{6.8}$$

$$\mathcal{D}_{s,t}^{\mathcal{A}} = \left\{ d_{P_{s,t}} | (P_{s_t} \in \mathcal{A}) \wedge (s_{P_{s,t}} = s) \wedge (t_{P_{s,t}} = t) \right\} \tag{6.9}$$

$$\text{and} \qquad \forall \tilde{P} : \mathcal{D}_{s_{\tilde{P}},t_{\tilde{P}}}^{\mathcal{A}_i} \cap \mathcal{D}_{s_{\tilde{P}},t_{\tilde{P}}}^{\mathcal{A}_{i+1}} \neq \{\} \tag{6.10}$$

Basically this means that all points of a connecting Action Space are either part of an Action Space i or (not exclusive) of the following Action Space $i+1$ within a section of Action Spaces, see (6.7). Additionally, all d values need to be in the set $\mathcal{D}_{s,t}^{\tilde{\mathcal{A}}}$ which includes all d values of the intersected action space for a specific value of s and t, see (6.9). Therein, this set can be derived by the join of the $D_{s,t}$ sets of the Action Spaces i and $i+1$, see (6.8). However, for all points $\tilde{P}$ which are part of the connecting Action Space $\tilde{\mathcal{A}}$ the corresponding lateral sets of the preceding and following Action Space have to intersect (6.10). From a geometrical perspective the connecting Action Space $\tilde{A}$ is the union of $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ at all points where both Action Spaces (at least) touch in *s-d* and intersect in *t-s*.

When describing a single Action Space, see Fig. 6.6, it is defined by multiple properties. The limits of an Action Space $\mathcal{A}$ are defined by the time-point t_{min}, where it is the first time when it can be entered without causing a collision. The latest time-point for leaving it is t_{max}. In the d and s dimension these extreme values are defined accordingly. The shape of an action-space $\mathcal{A}$ is defined by two polylines, the upper-bound $\boldsymbol{b}_u$ and the lower-bound $\boldsymbol{b}_l$, each consisting of points P, see also Fig. 6.7. Please denote, that those two lines are sufficient to describe this three-dimensional shape, as for a defined time the cross section of an Action Space is modeled as rectangle in *s-d*.

When defining an interface $\mathcal{R}$, see Fig. 6.7 a sequence of the previous defined action spaces is needed to traverse in the dynamic freespace. As can be seen in the

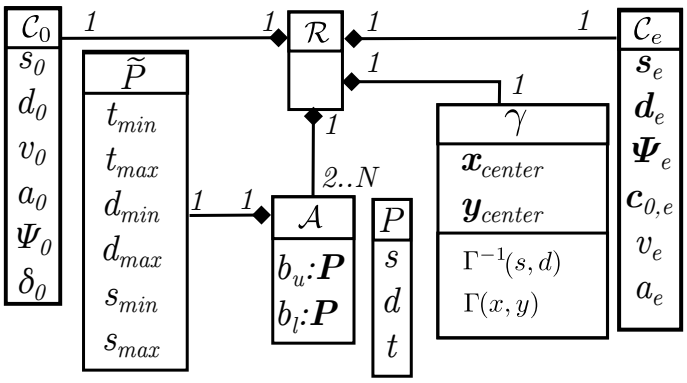


Fig. 6.7: *This slightly simplified class-diagram shows the interface of the trajectory planner.*

figure, the planning interface does not distinguish between connection and 'standard' Action Spaces. Every Action Space is handled in the same fashion. However, the implicit constraint to the sequence of Action Spaces is, that every Action Space $\mathcal{A}_i$ needs to intersect with its following Action Space $\mathcal{A}_{i+1}$.

$$\forall i \in 1..(N-1) \qquad \mathcal{A}_i \cap \mathcal{A}_{i+1} \neq \{\,\} \tag{6.11}$$

Where N denotes the number of Action Spaces in a Action Space sequence. Besides modeling the dynamic freespace the start and end state of a trajectory need to be defined in the interface $\mathcal{R}$. In addition, to solve the planning problem an end state $\mathcal{C}_e$ and a start-state $\mathcal{C}_0$ is defined. Thus, the start-state $\mathcal{C}_0$ consists of the position in the curvilinear coordinate system and the initial values of the bicycle-model. This includes the longitudinal velocity v_0, the longitudinal acceleration a_0, the orientation relative to the road measured by the angle Ψ and the steering angle δ. The end state is defined as a search space along the curvature of the road. It is modeled as polyline represented by $\boldsymbol{s}_e$ and $\boldsymbol{d}_e$ providing curvature $\boldsymbol{c}_{0,e}$ and orientation $\boldsymbol{\Psi}_{0,e}$ at each point of it. Practically this allows to generate a trajectory ending smooth on an arbitrary curved and oriented road geometry. Additionally a target speed v_e and acceleration a_e should be reached at the end of the trajectory. As the last part of the interface one needs a transformation class γ, which implements the transformations between the curvilinear and Cartesian coordinate system and back as described in Sec. 2.2.2.

6.7 Sampling using Action Spaces

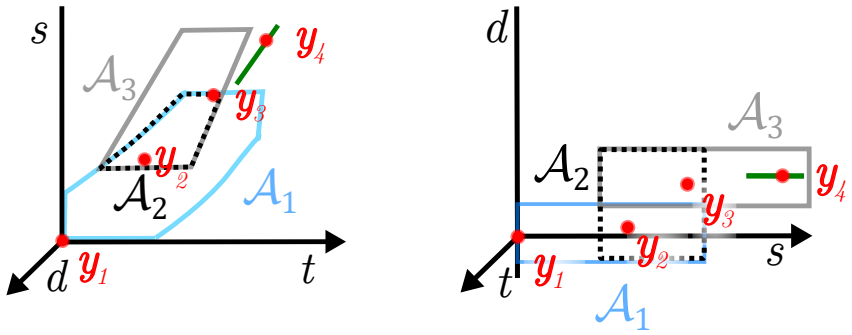
Given the interface definition $\mathcal{R}$ in Sec. 6.6 the task is to plan a trajectory with respect to the constraints defined in (6.1)-(6.5). To do so, multiple samples of trajectories indexed by m shall be generated. The underlying idea of the sampling process it to sample points at different time points in collision free areas along the curvilinear coordinate system. Tuples of combinations of these points can then be used as 'support points' to generate trajectories.

Each trajectory sample can be described uniquely with an output trajectory point tuple $\mathcal{T}^{(m)}$:

$$\mathcal{T}^{(m)} = \{\boldsymbol{y}_1^{(m)}, \boldsymbol{y}_2^{(m)}, .., \boldsymbol{y}_i^{(m)}, .., \boldsymbol{y}_N^{(m)}\}, \tag{6.12}$$

$$\text{where } \boldsymbol{y}_1^{(m)} = h(\boldsymbol{x}_0). \tag{6.13}$$

Using the definition of h in Sec. 2.4, $\boldsymbol{x}_0 = \mathcal{C}_0$ representing the start state as defined in $\mathcal{R}$ can be used to extract the first point of the output trajectory tuple, see (6.13). Please see Fig. 6.8 for the visualization of a tuple consisting of four output trajectory points. The goal of the sampling process is to produce a high rate of trajectory



(a) *Trajectory Point tuple in $t-s$ plane*

(b) *Trajectory Point tuple in $s-d$ plane*

Fig. 6.8: *Output trajectory point tuple in a curvilinear coordinate system for an Action Space Sequence $\mathcal{A}_1..\mathcal{A}_3$.*

point tuples resulting in trajectories which are feasible with respect to the previously defined constraints. To generate an output trajectory point tuple $\mathcal{T}^{(m)}$ an Action Space sequence $\mathcal{A}_i$, $i = 1, 2, .., N$ as visualized in Fig. 6.8 can be used. In this simple example only one connecting Action Space $\tilde{\mathcal{A}} = \mathcal{A}_2$ is present. Additionally, the start state $\mathcal{C}_0$ and the end state $\mathcal{C}_e$ are inputs for the sampling process.

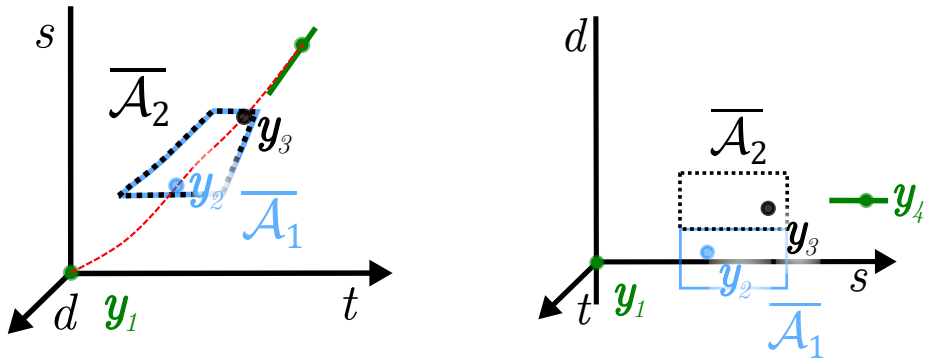
An intelligent sampling approach for output trajectory tuples is needed to reduce the probability of a trajectory to violate the constraints (6.2)-(6.5). Thus, the task is to find the most promising candidates $\mathcal{T}^{(m)}$. To find a strategy a challenging but typical scenarios is depicted in Fig. 6.8. One can see, that sampling output trajectories points from the intersection of two subsequent Action Spaces in most cases ensures that a trajectory stays within the Action Space Sequence. Less formal speaking, the Action Space can be interpreted as 'moving gap'. Then it is obvious, that the transition from one gap to the next is the most narrow part in the situation space. Thus, a search strategy for trajectories best focuses on this narrow part in order to reduce the number of trajectories resulting in collisions.

Accordingly, in Fig. 6.9 intersected Action Spaces $\overline{\mathcal{A}}$ are introduced. Each intersected Action Space is defined using the previous and next Action Space from the

Action Space Sequence.

$$\overline{\mathcal{A}}_i = \mathcal{A}_i \cap \mathcal{A}_{i+1} \quad \text{where} \mathcal{A}_i \cap \mathcal{A}_{i+1} \neq \emptyset \tag{6.14}$$

As visualized in Fig. 6.9 respective one trajectory point is sampled for the start state, each intersected Action space and the end state. The intersected Action Space $\overline{\mathcal{A}_1}$ and $\overline{\mathcal{A}_2}$ overlap in $t-s$ but do not intersect in $s-d$. The final output trajectory



(a) *Trajectory Point tuple in t-s plane*

(b) *Intersected action spaces in $s-d$ plane*

Fig. 6.9: *Output trajectory point tuple in a curvilinear coordinate system w.r.t intersected action spaces $\bar{\mathcal{A}}_1..\bar{\mathcal{A}}_2$.*

which is generated for each $\mathcal{T}^{(m)}$ in the Cartesian coordinate system is defined as:

$$\boldsymbol{y}^{(m)}(t) = [x^{(m)}(t),\ y^{(m)}(t)]^T \tag{6.15}$$

has to pass through all intersected action spaces $\bar{A}_{XY}$, see Fig. 6.9, i.e.

$$\exists t_i,\ i = 1, 2, ..N-1 :\ \boldsymbol{y}^{(m)}(t_i) \cap \bar{\mathcal{A}}_{\mathrm{XY},i} \neq \emptyset \tag{6.16}$$

where $\bar{\mathcal{A}}_{\mathrm{XY},i}$ denotes the i-th intersected action space transformed to the Cartesian coordinate System (see Sec. 2.2.2).

$$\bar{\mathcal{A}}_{\mathrm{XY},i} = \Gamma^{-1}(\bar{\mathcal{A}}_i) \tag{6.17}$$

Pictorially speaking the intersections $\overline{\mathcal{A}}$ are time-dependent gates the trajectory needs to traverse.

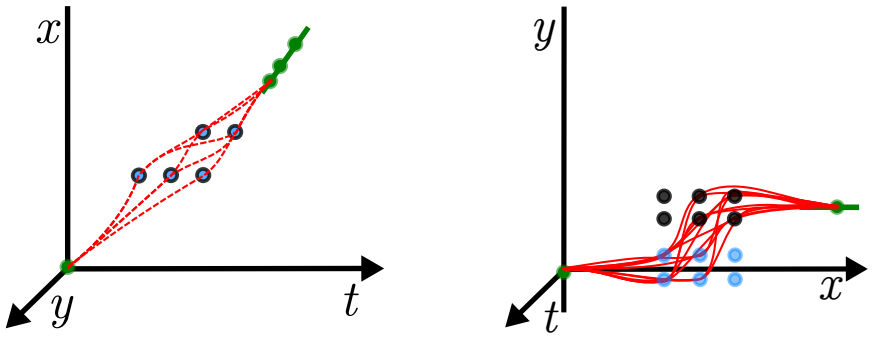
To get candidates of output trajectory points, each intersected Action Space $\overline{\mathcal{A}_n}$ is discretized in all three dimensions. This discretization serves as basis, see Fig. 6.9, to generate output trajectory point candidates.

$$\mathrm{TP}_i^{(m)}(p,q,r) = \begin{bmatrix} t_{i,p}^{(m)} \\ s_{i,q}^{(m)} \\ d_{i,r}^{(m)} \end{bmatrix} = \begin{bmatrix} \mathrm{t_{min}} + p\Delta_t \\ \mathrm{s_{min}} + q\Delta_s \\ \mathrm{d_{min}} + r\Delta_d \end{bmatrix} \tag{6.18}$$

where

$$\Delta_t = (\mathrm{t_{max}} - \mathrm{t_{min}})/N_t, \tag{6.19}$$
$$\Delta_s = (\mathrm{s_{max}} - \mathrm{s_{min}})/N_s, \tag{6.20}$$
$$\Delta_d = (\mathrm{d_{max}} - \mathrm{d_{min}})/N_d \tag{6.21}$$

according to the interface definition depicted in Fig. 6.7. $N_t, N_s and N_d$ define the discretization resolution. In the following $t_i^{(m)}$ is written instead of $t_{i,p}^{(m)}$, same for



(a) *Trajectory Point tuples in $x-t$ plane*

(b) *Trajectory Point tuples in $y-t$ plane*

Fig. 6.10: *Output trajectory point tuples in the Cartesian coordinate system.*

$s_{i,q}^{(m)}$ and $d_{i,r}^{(m)}$. A sample (6.12) is generated by choosing one trajectory point in each intersected action space. By taking every possible combination, all possible tuples $\mathcal{T}^{(m)}$ can be derived, see (6.12).

In Fig. 6.9a in the depicted example the sampled output point $\boldsymbol{y}_3$ sampled from $\overline{\mathcal{A}_2}$ has a greater value for t when compared to $\boldsymbol{y}_2$. A similar observation can be made in Fig. 6.9b where the s value of $\boldsymbol{y}_3$ is larger than the one of $\boldsymbol{y_2}$. Intuitively both properties make intuitively sense. Neither a trajectory shall be able to move backwards in the time dimension, nor backwards along the curvature of the road when driving forward. Such properties need to be considered in a sampling process, in order to reduce the number of infeasible $\mathcal{T}^{(m)}$ prior to trajectory generation. To do so, pruning is done. For example in order to get from $\mathrm{TP}_i^{(m)}$ to $\mathrm{TP}_{i+1}^{(m)}$, the comparison $t_{i,p}^{(m)} \leq t_{i+1}^{(m)}$ and $s_i^{(m)} \leq s_{i+1}^{(m)}$ can be used to remove infeasible combinations. These naive pruning strategies can however be extended using for example approximate maximum dynamic estimates.

6.8 Trajectory Generation based on Differential Flatness

Using the trajectory output point tuples which were derived in Sec. 6.7 jerk optimal output trajectories (see (6.15)) compatible with the dynamic bicycle model shall be generated. In this section a method solving this problem is presented.

One of the fundamental ideas of the proposed method is to neglect dynamic limits and collision checking in the trajectory generation process. Instead, infeasible trajectories shall be filtered out at a later stage of the trajectory planning process. This design is only valid because the sampling strategy presented in Sec. 6.7 minimizes the number of trajectories, which are affected by collisions. Because the number of trajectory output points $\boldsymbol{y}_m$ within a tuple $\mathcal{T}^{(m)}$ which constrain the output trajectory can get high, a method is needed to handle this challenge. Within this section a strategy is presented, which solves the problem using piecewise trajectories. The content of this section is mainly based on the work presented in [124]. The contribution of the author of this thesis is mostly in the formulation and discussion of the problem statement and not in the mathematical foundations of this section. However, for the sake of completeness a brief insight into the method will be given in the following.

The dynamic bicycle model as discussed in Sec. 2.4 is as reasonable choice to model the dynamic behavior of a vehicle within the planning process. For trajectory generation, the property, that the bicycle model is differentially flat with respect to the outputs $x(t)$ and $y(t)$ can be used, see [133] for an introduction into the concept of flat systems. Briefly summarized one can say, that if a system is differentially flat, then there exists a unique transformation between the 'differential flat outputs' $\boldsymbol{z}$, its derivatives, the system states $\boldsymbol{x}$ and the inputs $\boldsymbol{u}$, see Fig. 6.11. When using the

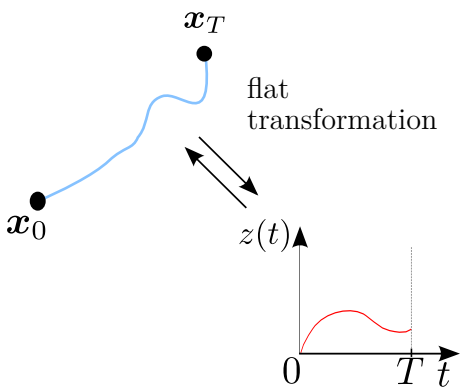


Fig. 6.11: *The flat transformation enables to transform between (flat) output trajectory and state-/input trajectory.*

bicycle model the flat output corresponds to the dimensions of the output trajectory $\boldsymbol{z} = \boldsymbol{y}$. This results in the property, that for an available output trajectory, the system states and inputs can be calculated. Because the principle is also working vice versa, constraints can be put to trajectory output points and its derivatives, in order to ensure they fulfill certain vehicle states and inputs. This property is especially relevant for the start $\boldsymbol{x}_0$ and the end state $\boldsymbol{x}_T$ of the trajectory. To map the inputs $\boldsymbol{u}$ and the states $\boldsymbol{x}$ to the flat outputs $x(t)$ and $y(t)$ using the algebraic transformations Φ_x and Φ_y, the flat outputs need to be differentiated twice as defined

in (6.22) and (6.23), see also [124].

$$\Phi_x : \begin{pmatrix} \boldsymbol{x}(t) \\ \boldsymbol{u}(t) \end{pmatrix} \rightarrow \boldsymbol{z}_x(t) : \tag{6.22}$$
$$\underbrace{\begin{pmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \end{pmatrix}}_{\boldsymbol{z}_x(t)} = \underbrace{\begin{pmatrix} x(t) \\ \cos(\Psi(t))v(t) \\ \cos(\Psi(t))a(t) - \sin(\Psi(t))\frac{1}{L}\tan(\delta(t))v(t)^2 \end{pmatrix}}_{\Phi_x(\boldsymbol{x}(t),\boldsymbol{u}(t))},$$

$$\Phi_y : \begin{pmatrix} \boldsymbol{x}(t) \\ \boldsymbol{u}(t) \end{pmatrix} \rightarrow \boldsymbol{z}_y(t) : \tag{6.23}$$
$$\underbrace{\begin{pmatrix} y(t) \\ \dot{y}(t) \\ \ddot{y}(t) \end{pmatrix}}_{\boldsymbol{z}_y(t)} = \underbrace{\begin{pmatrix} y(t) \\ \sin(\Psi(t))v(t) \\ \sin(\Psi(t))a(t) + \cos(\Psi(t))\frac{1}{L}\tan(\delta(t))v(t)^2 \end{pmatrix}}_{\Phi_y(\boldsymbol{x}(t),\boldsymbol{u}(t))}.$$

As described in Sec. 2.4.2 the letter Ψ denotes the orientation in a global coordinate system. As described in [124], the algebraic transformation (6.22) and (6.23) can be (locally) inverted to

$$\Phi^{-1} : \begin{pmatrix} \boldsymbol{z}_x(t) \\ \boldsymbol{z}_y(t) \end{pmatrix} \rightarrow \begin{pmatrix} \boldsymbol{x}(t) \\ \boldsymbol{u}(t) \end{pmatrix} : \tag{6.24}$$
$$\begin{pmatrix} v(t) \\ x(t) \\ y(t) \\ \Psi(t) \\ a(t) \\ \delta(t) \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{z_{x,2}(t)}{\cos[\tan^{-1}(z_{y,2}(t)/z_{x,2}(t))]} \\ z_{x,1}(t) \\ z_{y,1}(t) \\ \tan^{-1}(z_{y,2}(t)/z_{x,2}(t)) \\ \Phi_a^{-1}(.) \\ \Phi_\delta^{-1}(.) \end{pmatrix}}_{\Phi^{-1}([\boldsymbol{z}_x(t),\boldsymbol{z}_y(t)]^T)}. \tag{6.25}$$

where $\delta(t) = \Phi_\delta^{-1}(.)$ and $a(t) = \Phi_a^{-1}(.)$ are defined according to [124] by

$$\tan\left(\Phi_\delta^{-1}(.)\right) = \frac{z_{y,3}(t) - \tan(\Phi_\Psi^{-1}(.))z_{x,3}(t)}{\Phi_v^{-1}(.)^2(1/L)(\tan(\Phi_\Psi^{-1}(.))\sin(\Phi_\Psi^{-1}(.)) + \cos(\Phi_\Psi^{-1}(.))} \tag{6.26}$$

$$\Phi_a^{-1}(.) = \frac{z_{x,3}(t) + \sin(\Phi_\Psi^{-1}(.))(1/L)\tan(\Phi_\delta^{-1}(.))\Phi_v^{-1}(.)^2}{\cos(\Phi_\Psi^{-1}(.))} \tag{6.27}$$

and Φ_v^{-1} and Φ_Ψ^{-1} are given in rows one and four of (6.25). The inverse transformation (6.25) only holds locally for

$$v(t) \neq 0 \wedge \Psi(t), \delta(t) \in (-\pi/2, \pi/2). \tag{6.28}$$

This is treated as special cases in the implementation. Equations (6.22) and (6.23) allow to verify state and input conditions on the output trajectory. Once an output trajectory is generated, (6.25) can be applied to calculate the corresponding state and input trajectories.
As presented in Sec. 6.7 each sample $\mathcal{T}^{(m)}$ represents a sequence of sampled points $\mathrm{TP}^{(m)}$. Instead of trying to find a global solution which directly connects all points within the sequence, the principle of divide and conquer is used. Thus, for each point and its successor a trajectory piece is generated, which connects $\mathrm{TP}_i^{(m)}$ and $\mathrm{TP}_{i+1}^{(m)}$. The overall trajectory is defined according to (6.15) as:

$$\boldsymbol{y}_i^{(m)}(t) = \begin{cases} \boldsymbol{y}_1^{(m)}(t), & t \in [t_1^{(m)}, t_2^{(m)}] \\ \vdots & \\ \boldsymbol{y}_i^{(m)}(t), & t \in [t_i^{(m)}, t_{i+1}^{(m)}] \\ \vdots & \\ \boldsymbol{y}_{N-1}^{(m)}(t), & t \in [t_{N-1}^{(m)}, t_N^{(m)}]. \end{cases} \tag{6.29}$$

where $\boldsymbol{y}_i(t)^{(m)}$ is a trajectory piece. This trajectory piece again is defined as:

$$\boldsymbol{y}_i^{(m)}(t) = \begin{pmatrix} x_i^{(m)}(t) \\ y_i^{(m)}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{\theta}(t)^T G_i^{(m)} \\ \boldsymbol{\theta}(t)^T H_i^{(m)} \end{pmatrix} \in \mathcal{C}^2 \tag{6.30}$$

based on a yet unknown basis function $\boldsymbol{\theta}(t)$ with coefficients $G_i^{(m)}$ and $H_i^{(m)}$. When slicing the overall trajectory $\boldsymbol{y}_i^{(m)}(t)$ into pieces, the constraints on the start and end state of the overall trajectory, which are the states of the bicycle model, still need to be fulfilled as defined in (6.31) and (6.32).

$$\begin{pmatrix} x_1^{(m)}(0) \\ \dot{x}_1^{(m)}(0) \\ \ddot{x}_1^{(m)}(0) \end{pmatrix} = \Phi_x(\boldsymbol{x}_0, \boldsymbol{u}_0), \begin{pmatrix} x_{N-1}^{(m)}(t_{N-1}^{(m)}) \\ \dot{x}_{N-1}^{(m)}(t_{N-1}^{(m)}) \\ \ddot{x}_{N-1}^{(m)}(t_{N-1}^{(m)}) \end{pmatrix} = \Phi_x(\boldsymbol{x}_T, \boldsymbol{u}_T), \tag{6.31}$$

$$\begin{pmatrix} y_1^{(m)}(0) \\ \dot{y}_1^{(m)}(0) \\ \ddot{y}_1^{(m)}(0) \end{pmatrix} = \Phi_y(\boldsymbol{x}_0, \boldsymbol{u}_0), \begin{pmatrix} y_{N-1}^{(m)}(t_{N-1}^{(m)}) \\ \dot{y}_{N-1}^{(m)}(t_{N-1}^{(m)}) \\ \ddot{y}_{N-1}^{(m)}(t_{N-1}^{(m)}) \end{pmatrix} = \Phi_y(\boldsymbol{x}_T, \boldsymbol{u}_T) \tag{6.32}$$

Besides complying to the start and the end state of the overall trajectory, every connection between two trajectory pieces shall comply to $\mathcal{C}^0$, $\mathcal{C}^1$ and $\mathcal{C}^2$ smoothness conditions, as defined in (6.33) - (6.37). See also Fig. 6.12 for a visualization of the different levels of continuity.

$$\boldsymbol{y}_i^{(m)}(t_i^{(m)}) = \Gamma^{-1}(s_i^{(m)}, d_i^{(m)}), \qquad \forall i = 2, 3, .., N-1, \tag{6.33}$$

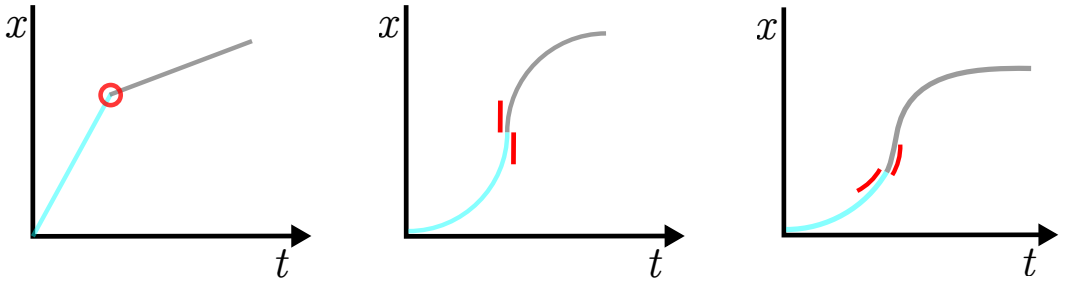
$$\boldsymbol{y}_i^{(m)}(t_{i+1}^{(m)}) = \Gamma^{-1}(s_{i+1}^{(m)}, d_{i+1}^{(m)}), \qquad \forall i = 1, 2, .., N-2, \tag{6.34}$$

$$\dot{\boldsymbol{y}}_i^{(m)}(t_{i+1}^{(m)}) = \dot{\boldsymbol{y}}_{i+1}^{(m)}(t_{i+1}^{(m)}), \qquad \forall i = 1, 2, .., N-2 \tag{6.35}$$

$$\ddot{\boldsymbol{y}}_i^{(m)}(t_{i+1}^{(m)}) = \ddot{\boldsymbol{y}}_{i+1}^{(m)}(t_{i+1}^{(m)}), \qquad \forall i = 1, 2, .., N-2, \tag{6.36}$$

$$\dddot{\boldsymbol{y}}_i^{(m)}(t_{i+1}^{(m)}) = \dddot{\boldsymbol{y}}_{i+1}^{(m)}(t_{i+1}^{(m)}), \qquad \forall i = 1, 2, .., N-2. \tag{6.37}$$

In this definitions (6.33) and (6.34) ensure that the every trajectory piece starts and ends in the corresponding output trajectory points, see Sec. 2.2.2 for the definition of Γ^{-1}. Equation (6.35) aligns the velocity of the vehicle to be the same at the transitions from one trajectory piece to another, while (6.36) does the same for the acceleration. The properties up to here yield the required smoothness conditions on the trajectory for the flat transformation. Finally, (6.37) guarantees smooth jerk at the trajectory output points, which is needed as a consequence of the jerk optimality objective. The property also reduces the degrees of freedom of the resulting optimization problem. As described in the beginning of this section, quadratic jerk-



(a) *C^0 continuity. Trajectories touch at their end or respective beginning.*

(b) *C^1 continuity. Trajectories have the same velocity at the point where they touch.*

(c) *C^2 continuity. Trajectories have the same acceleration at the point where they touch.*

Fig. 6.12: *Different continuity levels for longitudinal trajectories. Lower continuity levels are included in the higher continuity levels.*

optimal trajectories shall be generated. Therefore the (sub)optimization problem is to solve

$$\underset{H_i^{(m)}, G_i^{(m)},\, i=1..N-1}{\operatorname{argmin}} \sum_{i=1}^{N-1} \left(\int_{t_{i,p}^{(m)}}^{t_{i+1}^{(m)}} \left[\dddot{y}_i^{(m)}(t)^2,\ \dddot{x}_i^{(m)}(t)^2 \right] dt \right) \tag{6.38}$$

$$\text{s.t.} \qquad ((6.31)) - \ (6.37)$$

for each sample m. In [41] it is shown that the jerk-optimal output trajectory lies in quintic polynomials. Therefore quintic polynomials for $x_i^{(m)}(t)$ and $y_i^{(m)}(t)$ are chosen in(6.30). Using this knowledge a method of how this optimization problem can be solved analytically is presented in [124], where this solution is not part of the contribution of the author of this thesis.

After deriving all possible trajectories m based on all $\mathcal{T}^m$ each trajectory can be checked, whether dynamic limits regarding the state of the bicycle model are violated. Trajectories violating these limits can be pruned out. The same holds true for trajectories affected by collisions by leaving the sequence of action spaces. From the remaining trajectories, the best (jerk-minimal) that satisfies all constraints is picked subsequently and provided at the output interface, see Fig. 6.1.

6.9 Experimental Results

In this section the proposed method of trajectory planning is evaluated experimentally. To do so the method is evaluated in a synthetic test scenario. Within the scenario, a complex lane change maneuver is designed, which is depicted in Fig. 6.13, in which the system vehicle (blue) is surrounded by four cars. The states of the respective

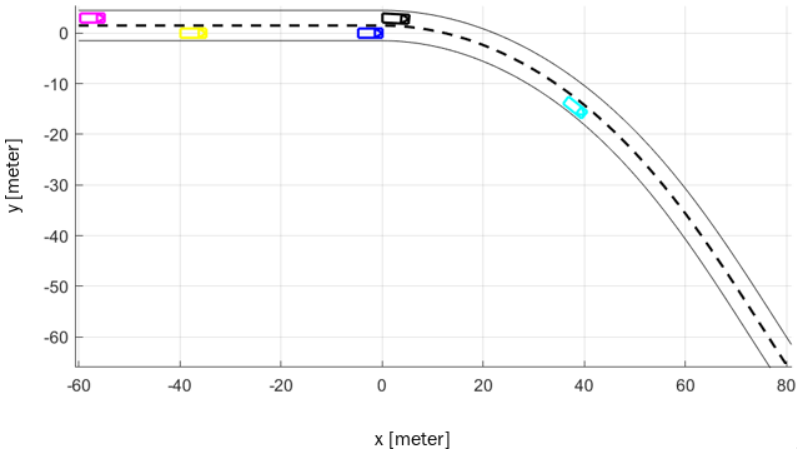


Fig. 6.13: *Traffic scene used for evaluation. System vehicle is depicted in dark blue.*

vehicles are documented in Tab. 6.1. In order to keep the experiment simple, the respective Action Spaces were derived by predicting the vehicles in the scene using the constant acceleration and the lane following assumption. In order to setup a planning

Tab. 6.1: *Object states used to derive the Action Space sequence using the CA model for predictions.*

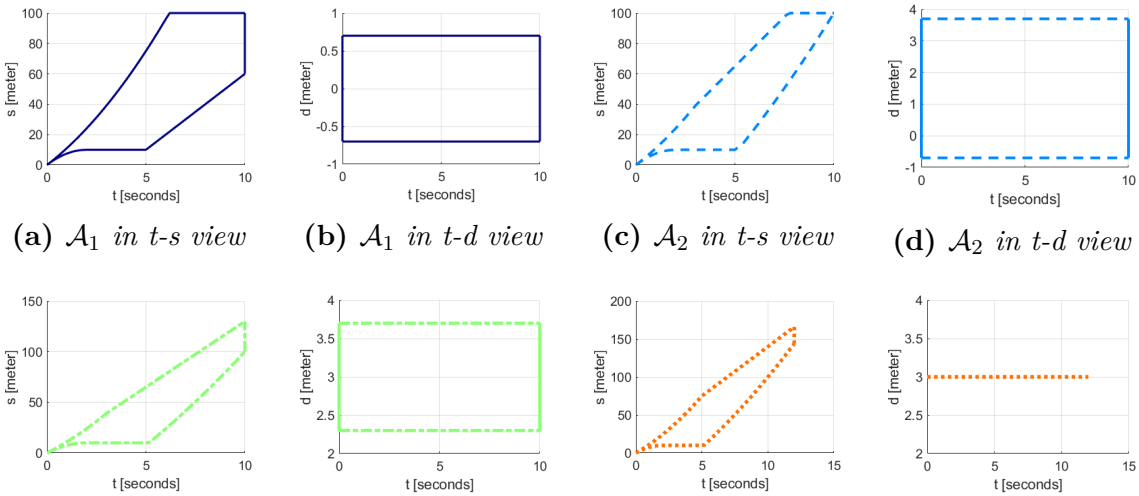
Cars	Pink	Yellow	Black	Cyan
s_0 $[m]$	-60	-40	0	40
v_0 $[m/s]$	11	10	13	10
a_0 $[m/s^2]$	1	0	0	-0.1

request using the interface $\mathcal{R}$, also the states and the constraints which apply to the system vehicle need to be known. These values are documented in Tab. 6.2. The bounds for the $s_{e,min}$ and $s_{e,max}$ for the end state were chosen heuristically using two constant acceleration assumptions.

The trajectory to be planned by the proposed method is a lane change to the left for the blue system vehicle. Thus, the vehicle shall catch the gap defined by the black and pink vehicle. To do so, the system vehicle shall avoid colliding with the black vehicle, while accelerating to the velocity of its future preceding vehicle. This lane change shall be executed before the pink vehicle is able to close the gap. The corresponding Action Space sequence is documented visually in Fig. 6.14a - 6.14h. Using the sequence of action spaces, the intersected action spaces are computed as presented in Sec. 6.7. They are visualized in Fig. 6.15a and 6.15b. Accordingly

Tab. 6.2: *State of the system vehicle and constraints used to setup the planning interface* $\mathcal{R}$.

State	Value	Constraint	Value
s_0 $[m]$	-3	a_{min} $[m/s^2]$	-5
v_0 $[m/s]$	10	a_{max} $[m/s^2]$	2
a_0 $[m/s^2]$	0	v_e $[m/s]$	13
d_0 $[m]$	0	a_e $[m/s^2]$	0
ψ_0 $[°]$	0	d_e $[m]$	3
δ_0 $[°]$	0		

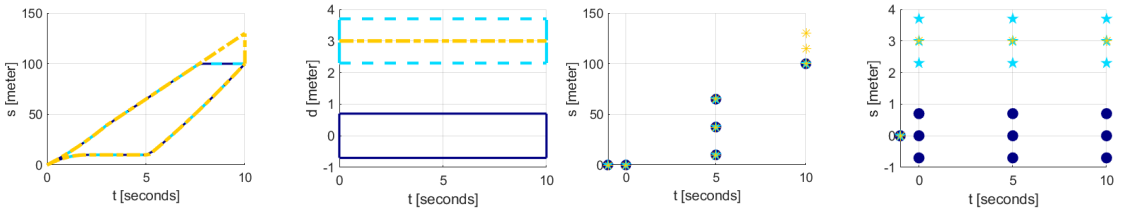


(a) $\mathcal{A}_1$ *in t-s view* **(b)** $\mathcal{A}_1$ *in t-d view* **(c)** $\mathcal{A}_2$ *in t-s view* **(d)** $\mathcal{A}_2$ *in t-d view*

(e) $\mathcal{A}_3$ *in t-s view* **(f)** $\mathcal{A}_3$ *in t-d view* **(g)** $\mathcal{A}_4$ *in t-s view* **(h)** $\mathcal{A}_4$ *in t-d view*

Fig. 6.14: *Action Space sequence used for evaluation.*

Trajectory Output Points TP are sampled from the intersected Action Spaces. These points are visualized as section view in Fig. 6.15c and 6.15d. As one can see many points that would be intuitively pruned out by a human driver are part of the set of Trajectory Output Point tuples. This may result in a large number of maybe driveable, but suboptimal solutions.



(a) $\bar{\mathcal{A}}_1 - \bar{\mathcal{A}}_3$ *in t-s* **(b)** $\bar{\mathcal{A}}_1 - \bar{\mathcal{A}}_3$ *in t-d* **(c)** *TPs in t-s* **(d)** *TPs in t-d*

Fig. 6.15: *Sampling in evaluation scenario using intersected Action Spaces* $\bar{\mathcal{A}}$.

Based on the sampled Points TP trajectories are generated using the method proposed in Sec. 6.8. Infeasible trajectories and trajectories affected by collisions

are pruned out. The resulting best trajectory is visualized in the XY plane in figure Fig. 6.16. Using the property of flatness the in and output states of the dynamic bicycle model are derived, which are visualized in Fig. 6.17. Due to jerk optimality the acceleration profile in Fig. 6.17 is very smooth. Besides the synthetic

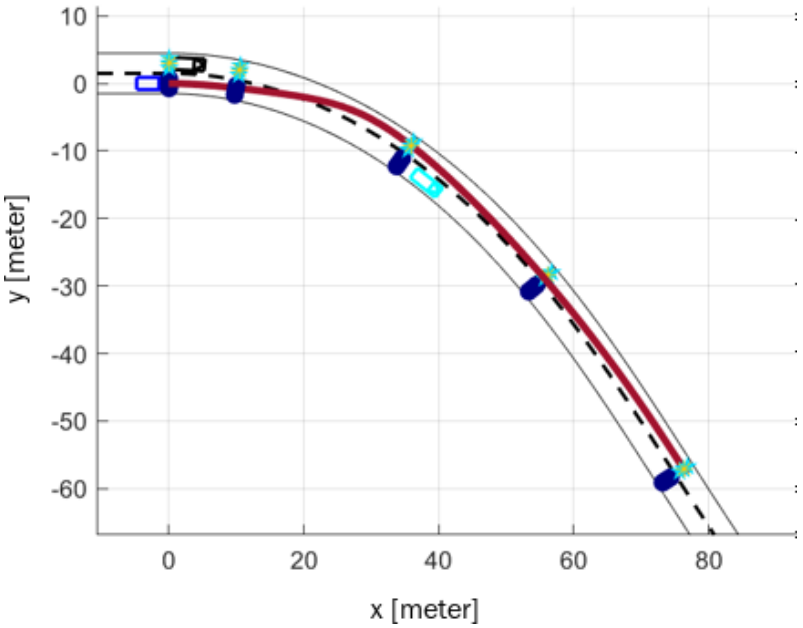


Fig. 6.16: *View on calculated trajectory in experiment in XY plane. The sampled TPs which were used to plan the trajectories are depicted as stars.*

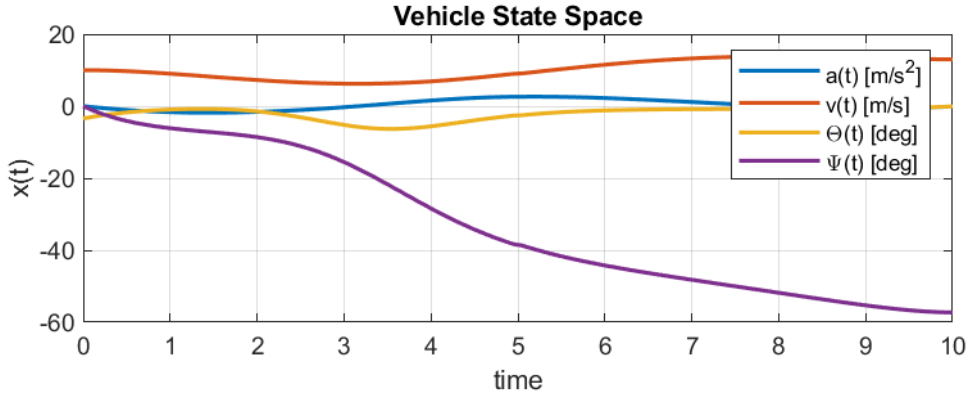


Fig. 6.17: *State space of trajectory calculated for scene in experiment. The scene is depicted in Fig. 6.13.*

test, the algorithm was also implemented in C++. For the implementation Eigen [134] was chosen as library for efficient transformation and trajectory generation. Without any further optimization, around 3000 trajectories can be sampled in the described setup within 50 ms using an off-the-shelf computer, see [124]. This result is promising for the envisioned application on an embedded system with limited computational resources. By using parallelization techniques, these results can be

improved even further. Calculating all samples simultaneously as well as checking their constraints can be parallelized easily, where checking the constraints is the most expensive step. Only 5% of the overall computation time was needed to solve the optimization problem as stated in [124]. In the implementation the constraints were checked discredited at 50 points of a single trajectory and the action spaces were shrinked based on minimum/maximum acceleration of the vehicle. Unrealistic output trajectory points were pruned out. The number of trajectories sampled depends on the density of the grid described in Sec. 6.7 and the pruning based on the current state. For each planning request 2000-3000 trajectories were sampled in the experiment vehicle. The planner was also applied successfully in an closed loop experiment in an urban environment (see Fig. 6.18).

Fig. 6.18: *Experiment vehicle tailored for urban driving driving along inner city road using the proposed method of trajectory planning.*

6.10 Conclusion

In this chapter a method for planning trajectories in structured environments, based on an abstraction of the environment and prediction information was presented. This method allows abstracting traffic gaps and static obstacles as time-dependent geometric bodies (Sec. 6.6). The method allows a flexible but general and simple description of trajectory planning problems. Because main parts of the abstraction are done in curvilinear coordinate system, the method can be easily transferred to different structured dynamic environments, e.g. roundabouts or inner city merge situations. The approach of modeling the constraints as a sequence of action-spaces is an efficient way to encapsulate behavior planning from the problem of planning a driveable trajectory. Using the representation, a sampling method was presented (Sec. 6.7), which uses the sequence of dynamic freespace elements as sampling heuristic. The method proved to be deliver meaningful results in the experiment. The presented method is also highly efficient and real-time capable (Sec. 6.9). This is achieved by an efficient sampling of jerk-optimal trajectories that are piece-wise defined in Cartesian coordinates (Sec. 6.8).

In the application of the method in inner city scenarios it however turned out, that depending on the traffic situation, many sampled trajectories are dynamically infeasible. This points in the direction, that further improvements can be expected by modifying the sampling strategy.

Chapter 7

Epilogue

The technology of automated driving is in full growth. Today's partial automated driving systems available in the market are able to help the human driver in more and more difficult traffic situations. This covers for example lane-keeping, collision avoidance and even traffic light information systems. The growth of driving automation goes hand in hand with a change of the development focus. Coming from a decade, where sensor performance was the limiting factor of system performance, software and data driven methods were identified as the major challenge in the last years. This change goes hand in hand with a change of Electrical / Electronic (EE) architectures. The typical architecture used in Advanced Driver Assistance Systems was a decentralized approach of early information reduction, where every sensor was equipped with intelligence. Modern systems use a centralized architecture. Raw data of different domains is sent to central electronic control units in which all computations are done. This change in architecture rises challenging requirements regarding computational power to such kind of central driving computer. The availability of such hardware allows the execution of powerful methods for situation analysis and trajectory planning.

Despite the technological advances, the step to the next level of automation is difficult. In all partial automated systems in the market, the driver is used as fallback instance in many situations. This approach is not an option for conditional or Level-3 automation. Within its Operational Design Domain (ODD) the Dynamic Driving Task (DDT) needs to be solved by the automation system. Thus, all possible scenarios which require a human driver in a partial automated system, need to be handled by the conditional automated system.

Providing a methodical toolkit for solving these kinds of fallback problems is one of the main topics of this work. Accordingly, a system design was presented in which the developed methods are embedded. This system design implements a safety net by offering a collision-free fallback trajectory which can be used at least for the take-over time. However, the presented methods still leave room for improvements, as will be discussed in the next two sections.

This epilogue is structured as follows. The novel contributions to solve the automated driving task with a focus on the fallback performance are summarized in Sec. 7.1. Achieved results and remaining gaps are discussed within this summary.

Based on this big picture, ideas are presented in Sec. 7.2 which point out possible future research directions in the context of this thesis. Finally, the thesis concludes with Sec. 7.3.

7.1 Summary of Contributions

In this thesis a novel toolkit for implementing a fallback function for conditional automated vehicles has been presented. The contributions to the scientific community are manifold and can be summarized in four clusters:

- A system design implementing a fallback mechanism for a Level-3 automated driving system.
- A holistic approach to the problem of 'Maneuver Estimation' in order to recognize the intent of other traffic participants.
- A novel method of 'Probabilistic Position Prediction' considering the intent and physical state of vehicles to be predicted.
- A 'Trajectory Planning' method capable of handling information of time-dependent freespace.

In the following these four clusters will be explained individually in further details.

The first major contribution of this thesis is the derivation of a functional system design implementing a fallback mechanism for a Level-3 automated driving system. Based on a high level description of the use case, the needed functions and interactions were derived in chapter 3. Function blocks needed in order to perform a meaningful fallback performance were identified. The three identified non trivial function blocks are 'Maneuver Estimation', 'Position Prediction' and 'Trajectory Planning'. With respect to the functionality of providing a fallback trajectory, the basic requirements were analyzed, documented and discussed regarding the technical risks of the proposed system design.

The second contribution to the field of this thesis is a holistic approach to the problem of 'Maneuver Estimation' which was presented in chapter 4. Within the scope of this thesis, the maneuver classes were restricted to the following three classes: lane following, lane change to the left, and lane change to the right. The basic assumption of the novel approach is that decisions of human drivers are based on some (unknown) stimuli. It is assumed to be a priori unknown, which stimuli are triggering a specific decision. Accordingly, an environment model was investigated and derived which systematically defines possible stimuli influencing the decision making of human drivers. This model is assumed to contain all relevant information. Based on this model, data was collected and both machine learning as well as statistical methods were applied to identify the information relevant and measurable. Several models were derived and evaluated in two experiments. The results show that short term maneuver predictions up to 1.5 s can be done close to perfect. For longer horizons up to 5.0 s, maneuver estimation gets more challenging. Still, using

a conservative parameterization, up to 60 % of lane changes can be recognized up to 5.0 s in advance with a false positive rate of close to zero. The approach is accepted as contribution to the field by the scientific community and was published in [73]–[76].

The third contribution of this thesis is a method capable of estimating probability densities of future positions of vehicles. The derived estimates consider the kinematic vehicle state and the situation the vehicle needs to handle. To achieve this, the method of 'Position Prediction' derived in chapter 5 treats the human driver and the way maneuvers are executed as a probabilistic black box. To solve the problem of providing probability distributions of future positions, a Mixture of Experts model was chosen. In this model the methods developed for 'Maneuver Estimation' were used as Gating Nodes. To predict future positions efficiently in real-time, Gaussian Mixture Regression was used for the individual experts. The method was evaluated and compared to classical methods of position prediction in two experiments. The outcome proves that the mean of the prediction is as precise as 0.2 m in the lateral and 0.5 m in the longitudinal dimension at a prediction horizon of 5.0 s, which compares to state-of-the-art methods. In contrast to other methods however, the novel approach is also able to deliver meaningful probability distributions of future whereabouts of other vehicles. This information is extremely valuable, as dynamic freespace is one of the main inputs needed for trajectory planning. The approach for position prediction presented in this thesis is accepted as contribution to the field and was published in [75], [76], [105].

The fourth contribution of this thesis is a 'Trajectory Planning' method capable of considering information of time-dependent freespace. The approach was presented in chapter 6. In contrast to related work, the novelty of the presented method is to handle time dependent constraints using an interface which explicitly models the dynamic freespace. Using this specific representation, the maneuver to be planned can be handed over to the trajectory planner implicitly. Maneuvers in this context can be quite complex sequences, i.e. catching up to a gap followed by an overtaking maneuver. Based on the information provided by this interface, a method of sampling in the curvilinear coordinate system was presented. The method samples trajectory support points using a geometric heuristic. The heuristic hereby minimizes the share of trajectories which are affected by collisions. Subsequently, a method for trajectory generation was presented. The method transforms the support points into a Cartesian coordinate system and generates jerk minimal trajectories using a longitudinally and laterally coupled model. At the same time, the method provides a high level of flexibility regarding the shape of trajectories by using a composite representation. Because the method was designed to operate in the Cartesian coordinate system it is also robust with respect to discontinuities in the road representation and towards high lane curvatures. The method of trajectory planning was successfully evaluated in complex synthetic scenarios and has also proven to be robust in inner city scenarios when evaluated on a small fleet of test vehicles. It is also accepted as contribution to the field by the scientific community in [124].

7.2 Future Research Directions

In this section possible future research directions are outlined. The architecture proposed in chapter 3 includes many possibilities for further improvements. The system was designed according to a pure deliberative architecture paradigm. However, even though a new fallback trajectory can only be planned if a new reliable world model is available, the fallback system could be improved by adding a reactive layer to it. Such kind of reactive layer, knowing about the planned trajectory, could help to avoid collisions when situations do not evolve as predicted. Further research directions regarding the system concept can be found in the documentation of the technical risks in Sec. 3.7.

Regarding maneuver estimation there exist multiple future research directions and topics worth to investigate. The basic assumption which is made in chapter 4 is that the presented environment model contains all data influencing the behavior of human drivers. Obviously this assumption does not hold true and additional features could be incorporated. The possible set of additional features can be divided into three categories. The first category contains features which could be directly measured by a single sensor system. This includes for example the lateral distance to a lane marking. The second category of features are the ones which could be derived by information provided by a global instance, particularly a backend delivering additional map layers via a communication channel. This could include for example the weather and road conditions. The third and most important category of features could be derived by an extension of the feature model in which more surrounding vehicles and the road topology could be included, e.g. the distance to motorway junctions. An orthogonal research direction is the extension of the prediction time horizon. A first step could be to investigate the temporal importance of the different features with respect to the prediction time horizon. Such knowledge could be used to derive the maximum time horizon for which meaningful predictions can be made.

Regarding the methods of position prediction presented in chapter 5 there are several directions to be investigated in future research. Within the presented method, the feature set was derived using expert knowledge. A systematic feature selection process could help to improve prediction results even further. When taking a look at the methods used for position prediction a parametric method resulting in a generative model was chosen. One could also investigate how prediction results change if the parametric model is replaced by non-parametric methods, e.g. multidimensional Kernel Density EstimatorsKDE. As for the methods of Maneuver Recognition, also for position prediction one could investigate the maximum prediction horizon which could be achieved. This could be done by a stepwise extension of the prediction horizon, until the distributions of future positions of different vehicles overlap with each other. As no trajectories can be planned without available freespace, this is a pragmatic upper bound for the maximum feasible prediction horizon. In case one wants to apply the presented method for non-highway scenarios one could investigate whether a lateral and longitudinal coupled model is able to improve the prediction performance.

Furthermore, both machine learning based methods ‘Maneuver Estimation’ and

'Position Prediction' could profit strongly from a fleet based relearning strategy using a common backend, in which relevant events could be collected. These events could be used to improve the models in order to achieve a better generalization and more robust performance in corner cases, where such improvements can only be achieved by massive data. The models could then be generated and deployed to the fleet again. Besides the needed data pipelines in the backend, also methods and tools are needed to handle this relearning task. The challenge of this problem is to improve the prediction performance in the special cases without lowering the predictive performance of the models in the common cases.

Besides these methodical and architectural perspectives, the designed prediction methods could be adapted to inner-city scenarios in the future. The only change needed in architecture would be a modification of the environment model and the maneuver classes of interest.

When discussing future work regarding the presented method of trajectory planning in chapter 6, the experiments hinted in the direction that more powerful sampling strategies and heuristics could be useful to lower the number of trajectories to be sampled. Besides the pruning of sampling points which are not reachable due to the dynamic limitations of a vehicle, other methods could be helpful. A possible solution is the usage of the method of Position Prediction, presented in chapter 5. By sampling support points within the intersected Action Spaces in a frequency according to the derived predicted distribution, the share of 'human driver like' trajectories could be increased. Another research direction worth to investigate is a post-processing of the sampled trajectory. To improve the perceived human comfort an optimizer could smooth and adapt the selected trajectory based on a cost function reflecting the research done in human factors, see for example [135].

7.3 Conclusion

Driving automation is a very active field of research and subject to fast development in both hard- and software. One of the next major milestones is to handover the responsibility from the driver to a Level-3 automated vehicle. Even though a driver is basically available, an additional fallback system is needed. Such kind of system can ensure a safe handover back to the driver within a reasonable take-over time in case of a system failure of the automated driving system.

This thesis addresses this topic by providing a concept and a toolkit of methods to implement such a fallback system. A system concept is presented, which is suitable to solve the outlined problem. Based on this concept, three critical functions are identified which are needed for the implementation of the fallback system. Accordingly, a novel method of intention recognition for vehicles on highways was presented, which was verified and optimized in extensive experiments using real-world data. Significant performance boosts are achieved in evaluation. A method of probabilistic position prediction was developed and evaluated in the same manner achieving state-of-the-art performance regarding the mean of the predicted distributions. In contrast to related work, the derived method is also able to provide meaningful

estimates of the uncertainty of future positions. Based on these estimates, reliable fallback trajectories can be planned. The novel planning method is able to consider prediction information explicitly as time dependent constraints. Using a sampling-based approach, jerk minimal trajectories w.r.t the dynamic bicycle model are sampled efficiently in real-time.

All developed methods in this thesis were not only accepted by the scientific community. They were also implemented and successfully tested in different test vehicles. This confirms the applicability of the presented fallback system as well as its potential for usage in large scale series vehicles.

Publications

This thesis has led to the following publications:

[73] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K. Kuhnert, "A lane change detection approach using feature ranking with maximized predictive power", in *Proceedings of the 2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 108–114.

[74] J. Schlechtriemen, R. Graf, A. Wedel, M. Fritzsche, K.-D. Kuhnert, and K. Dietmayer, "Ein Vergleich von Algorithmen zur Verhaltensprädiktion anderer Verkehrsteilnehmer", in *Tagungsband - VDI Workshop Fahrerassistenzsystem 2014*.

[75] J. Schlechtriemen, F. Wirthmueller, A. Wedel, G. Breuel, and K. Kuhnert, "When will it change the lane? A probabilistic regression approach for rarely occurring events", in *Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1373–1379.

[76] F. Wirthmüller, J. Schlechtriemen, J. Hipp, and M. Reichert, "Teaching Vehicles to Anticipate: A Systematic Study on Probabilistic Behavior Prediction Using Large Data Sets", *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2020.

[105] J. Schlechtriemen, A. Wedel, G. Breuel, and K.-D. Kuhnert, "A probabilistic long term prediction approach for highway scenarios", in *Proceedings of the 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 732–738.

[124] J. Schlechtriemen, K. Wabersich, and K. Kuhnert, "Wiggling through complex traffic: Planning trajectories constrained by predictions", in *Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1293–1300.

[136] V. Gomer, H. Lu, G. Weidl, T. Dang, G. Breuel, J. Schlechtriemen, and W. Rosenstiel, "Freiraumbewertung für Spurwechselmanöver mit Bayes-Netzen", in *Tagungsband - VDI Workshop Fahrerassistenzsystem 2015*.

References

[1] Federal Highway Administration (FHWA), *Next Generation SIMulation Fact Sheet*, https://www.fhwa.dot.gov/publications/research/operations/its/06135/index.cfm, (Accessed on 09/07/2019), Dec. 2006.

[2] T. Winkle, "Safety Benefits of Automated Vehicles: Extended Findings from Accident Research for Development, Validation and Testing", in *Autonomous Driving*, Springer, 2016, pp. 335–364.

[3] T. Gasser, C. Arzt, M. Ayoubi, *et al.*, "Rechtsfolgen zunehmender Fahrzeugautomatisierung", *Berichte der Bundesanstalt für Straßenwesen. Unterreihe Fahrzeugtechnik*, no. 83, 2012.

[4] SAE On-Road Automated Vehicle Standards Committee, *Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems*, (Accessed on 12/10/2016), 2014.

[5] F. Ackermann, "Abstandsregelung mit Radar", *Spektrum der Wissenschaft*, vol. 34, 1980.

[6] Wikipedia, *Autonomous cruise control system — Wikipedia, The Free Encyclopedia*, (Accessed on 12/31/2016), 2016. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Autonomous_cruise_control_system&oldid=757633069.

[7] P. I. Labuhn and W. J. Chundrlik, *Adaptive cruise control*, US Patent 5,454,442, Oct. 1995.

[8] B. Fleming, "New automotive electronics technologies [automotive electronics]", *IEEE Vehicular Technology Magazine*, vol. 7, no. 4, pp. 4–12, 2012.

[9] F. Naujoks, C. Purucker, A. Neukum, S. Wolter, and R. Steiger, "Controllability of Partially Automated Driving functions - Does it matter whether drivers are allowed to take their hands off the steering wheel?", *Transportation research part F: traffic psychology and behaviour*, vol. 35, pp. 185–198, 2015.

[10] S. Lohr, *A Lesson of Tesla Crashes? Computer Vision Can't Do It All Yet - The New York Times*, https://www.nytimes.com/2016/09/20/science/computer-vision-tesla-driverless-cars.html, (Accessed on 01/23/2017), Sep. 2016.

[11] C. Gold, D. Damböck, L. Lorenz, and K. Bengler, "Take over! How long does it take to get the driver back into the loop?", in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 57, 2013, pp. 1938–1942. DOI: 10.1177/1541931213571433. [Online]. Available: http://dx.doi.org/10.1177/1541931213571433.

[12] F. Naujoks, C. Mai, and A. Neukum, "The effect of urgency of take-over requests during highly automated driving under distraction conditions", *Advances in Human Aspects of Transportation*, no. Part I, p. 431, 2014.

[13] K. Zeeb, A. Buchner, and M. Schrauf, "What determines the take-over time? An integrated model approach of driver take-over after automated driving", *Accident Analysis & Prevention*, vol. 78, pp. 212–221, 2015.

[14] K. Zeeb, A. Buchner, and M. Schrauf, "Is take-over time all that matters? The impact of visual-cognitive load on driver take-over quality after conditionally automated driving", *Accident Analysis & Prevention*, vol. 92, pp. 230–239, 2016.

[15] B. Smith and J. Svensson, "Automated and Autonomous Driving: Regulation under Uncertainty", *International Transport Forum Policy Papers*, 2015.

[16] H. Fountain, *A Slow Ride Toward the Future of Public Transportation - The New York Times*, https://www.nytimes.com/2016/11/08/science/finland-public-transportation-driverless-bus.html, (Accessed on 02/08/2017), Nov. 2016.

[17] A. Alessandrini, A. Cattivera, C. Holguin, and D. Stam, "CityMobil2: Challenges and Opportunitieys of Fully Automated Mobility", in *Road Vehicle Automation*, Springer, 2014, pp. 169–184.

[18] M. Isaac, *What It Feels Like to Ride in a Self-Driving Uber - The New York Times*, https://www.nytimes.com/2016/09/15/technology/our-reporter-goes-for-a-spin-in-a-self-driving-uber-car.html, (Accessed on 02/08/2017), Sep. 2016.

[19] J. Radlmayr, C. Gold, L. Lorenz, M. Farid, and K. Bengler, "How traffic situations and non-driving related tasks affect the take-over quality in highly automated driving", in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Sage Publications Sage CA: Los Angeles, CA, vol. 58, 2014, pp. 2063–2067.

[20] F. Wulf, K. Zeeb, M. Rimini-Doring, M. Arnon, and F. Gauterin, "Effects of human-machine interaction mechanisms on situation awareness in partly automated driving", in *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems-(ITSC)*, 2013, pp. 2012–2019.

[21] J. Bredereke and A. Lankenau, "A rigorous view of mode confusion", in *Proceedings of the International Conference on Computer Safety, Reliability, and Security*, Springer, 2002, pp. 19–31.

[22] P. Kumar, M. Perrollaz, S. Lefèvre, and C. Laugier, "Learning-based approach for online lane change intention prediction", in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2013.

[23] D. Kasper, G. Weidl, T. Dang, G. Breuel, A. Tamke, and W. Rosenstiel, "Object-oriented Bayesian networks for detection of lane change maneuvers", in *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 673–678. DOI: 10.1109/IVS.2011.5940468.

[24] H. Graf R.and Deusch, M. Fritzsche, and K. Dietmayer, "A learning concept for behavior prediction in traffic situations", in *Proceedings of the 2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 672–677.

[25] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations", *Physical Review E*, vol. 62, no. 2, p. 1805, 2000.

[26] J. Hillenbrand, A. M. Spieker, and K. Kroschel, "A Multilevel Collision Mitigation Approach; Its Situation Assessment, Decision Making, and Performance Tradeoffs", *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, no. 4, pp. 528–540, Dec. 2006, ISSN: 1524-9050. DOI: 10.1109/TITS.2006.883115.

[27] J. Wiest, M. Hoffken, U. Kressel, and K. Dietmayer, "Probabilistic trajectory prediction with Gaussian mixture models", in *Proceedings of the 2012 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2012, pp. 141–146. DOI: 10.1109/IVS.2012.6232277.

[28] E. Fosler-Lussier, "Markov models and hidden Markov models: A brief tutorial", *International Computer Science Institute*, 1998.

[29] International Electrotechnical Commission (IEC), *Functional safety - Essential to overall safety*, https://www.iec.ch/about/brochures/pdf/technology/functional_safety.pdf, (Accessed on 07/21/2019), 2015.

[30] International Electrotechnical Commission (IEC), *IEC 60050 - International Electrotechnical Vocabulary - Details for IEV number 192-10-06: "fail-safe"*, http://www.electropedia.org/iev/iev.nsf/display?openform&ievref=192-10-06, (Accessed on 07/21/2019).

[31] European Aviation Safety Agency (EASA), *Final Decision on CS-AWO-23 October 2003 _publication ver...* https://www.easa.europa.eu/sites/default/files/dfu/ws_prod-g-doc-Agency_Mesures-Certification_Spec-decision_ED_2003_06_RM.pdf, (Accessed on 07/21/2019).

[32] M. Hillenbrand, *Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen*. KIT Scientific Publishing, 2012, vol. 4.

[33] R. Krüger and B. G. Z. Funktionssicherheit, "Ganzheitliche Sicherheitsbetrachtung am Beispiel von E-Fahrzeugen", in *Tagungsband - Workshop Safety in Transportation-SiT. Braunschweig*, vol. 30, 2011.

[34] S. Becker, "Fahrerassistenzsysteme-Gebrauchssicherheit fuer Jedermann", *Berichte der Bundestanstalt fuer Strassenwesen. Unterreihe Mensch und Sicherheit*, no. 123, 2000.

[35] V. Banks, K. Plant, and N. Stanton, "Driver error or designer error: Using the Perceptual Cycle Model to explore the circumstances surrounding the fatal Tesla crash on 7th May 2016", *Safety Science*, vol. 108, pp. 278–285, 2018, ISSN: 0925-7535. DOI: https://doi.org/10.1016/j.ssci.2017.12.023.

[36] A. Luttenberger, "Legal framework on eSafety communication in road transport", in *Proceedings of the Conference on Tourism and Hospitality Management 2012*, vol. 1, 2012, pp. 126–129.

[37] J. Gurney, "Sue my car not me: Products liability and accidents involving autonomous vehicles", *U. Ill. JL Tech. & Pol'y*, p. 247, 2013.

[38] E. Helming, *Functional Safety in accordance with ISO 26262 and product liability for No Trouble Found events*, http://www.ra-helmig.de/fileadmin/docs/publikationen/2012_ISO_26262_No_Trouble_Found_PHi.pdf, (Accessed on 03/18/2018), Nov. 2012.

[39] J. Anderson, K. Nidhi, K. Stanley, P. Sorensen, C. Samaras, and O. Oluwatola, *Autonomous vehicle technology: A guide for policymakers*. Rand Corporation, 2014.

[40] I. O. for Standardization (ISO), "ISO 8855: 2013-11. Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary", Geneva, CH, Tech. Rep., 2013.

[41] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame", in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 987–993.

[42] A. Ng and M. Jordan, "On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes", *Advances in neural information processing systems*, vol. 14, p. 841, 2002.

[43] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning", *CoRR*, vol. abs/1811.12808, 2018. arXiv: 1811.12808. [Online]. Available: http://arxiv.org/abs/1811.12808.

[44] S. Fortmann-Roe, *Understanding the bias-variance tradeoff*, http://scott.fortmann-roe.com/docs/BiasVariance.html, (Accessed on 01/30/2020), 2012.

[45] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: A comparative", *J Mach Learn Res*, vol. 10, pp. 66–71, 2009.

[46] C. Burges, "A tutorial on support vector machines for pattern recognition", *Data mining and knowledge discovery*, vol. 2, no. 2, p. 123, 1998.

[47] C. Cortes and V. Vapnik, "Support-vector networks", *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[48] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection", in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 14, 1995, pp. 1137–1145.

[49] B. Efron, "Bootstrap methods: Another look at the jackknife", in *Breakthroughs in statistics*, Springer, 1992, pp. 569–593.

[50] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning", in *The elements of statistical learning*, Springer, 2009, pp. 485–585.

[51] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks", *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.

[52] K. H. Brodersen, S. Cheng, K. E. Stephan, and J. M. Buhmann, "The Balanced Accuracy and Its Posterior Distribution", in *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)*, 2010, pp. 3121–3124. DOI: 10.1109/ICPR.2010.764.

[53] T. Fawcett, "An Introduction to ROC Analysis", *Pattern Recogn. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2005.10.010. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2005.10.010.

[54] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance", *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.

[55] T. Chai and R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature", *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[56] I. Myung, "Tutorial on maximum likelihood estimation", *Journal of mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003.

[57] H. Akaike, "Information theory and an extension of the maximum likelihood principle", in *Selected papers of hirotugu akaike*, Springer, 1998, pp. 199–213.

[58] K. Burnham and D. Anderson, "Multimodel inference: understanding AIC and BIC in model selection", *Sociological methods & research*, vol. 33, no. 2, pp. 261–304, 2004.

[59] G. Schwarz *et al.*, "Estimating the dimension of a model", *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[60] J. Kuha, "AIC and BIC: Comparisons of assumptions and performance", *Sociological methods & research*, vol. 33, no. 2, pp. 188–229, 2004.

[61] S. Ruder, "An overview of gradient descent optimization algorithms", *arXiv preprint arXiv:1609.04747*, 2016.

[62] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection", *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003, ISSN: 1532-4435. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944968.

[63] M. Ester, H. Kriegel, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96, Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[64] S. Lloyd, "Least squares quantization in PCM", *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[65] A. Jain, "Data clustering: 50 years beyond K-means", *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[66] K. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[67] I. Filho, *The Chinese Restaurant Process: Bayesian Inference of Mixture Models and Applications in Computational Biology*, http://www.cin.ufpe.br/~igcf/talks/2008-defense.pdf, (Accessed on 12/05/2018), 2008.

[68] R. Das, *Collapsed Gibbs Sampler for Dirichlet Process Gaussian Mixture Models (DPGMM)*, http://rajarshd.github.io/talks/DPGMM_tutorial.pdf, (Accessed on 01/30/2020).

[69] P. Polack, F. Altché, B. D'Andréa-Novel, and A. Dela Fortelle, "The Kinematic Bicycle Model: a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles?", in *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV)*.

[70] G. Starke and P. Hruschka, "Eine Strukturvorlage zur effektiven Dokumentation von Software-und IT Architekturen", in *eOrganisation: Service-, Prozess-, Market-Engineering: 8. Internationale Tagung Wirtschaftsinformatik*, (Accessed on 06/20/2019), vol. 2, Feb. 2007, pp. 77–88. [Online]. Available: http://aisel.aisnet.org/wi2007/61.

[71] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns: Abstraction and reuse of object-oriented design", in *Proceedings of the European Conference on Object-Oriented Programming*, Springer, 1993, pp. 406–431.

[72] S. Fürst, J. Mössinger, S. Bunzel, *et al.*, "AUTOSAR–A Worldwide Standard is on the Road", in *Proceedings of the 14th International VDI Congress Electronic Systems for Vehicles*, vol. 62, 2009, p. 5.

[77] U. Dogan, J. Edelbrunner, and I. Iossifidis, "Autonomous driving: A comparison of machine learning techniques by means of the prediction of lane change behavior", in *Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011, pp. 1837–1843. DOI: 10.1109/ROBIO.2011.6181557.

[78] G. Weidl, A. L. Madsen, S. Wang, D. Kasper, and M. Karlsen, "Early and Accurate Recognition of Highway Traffic Maneuvers Considering Real World Application: A Novel Framework Using Bayesian Networks", *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 3, pp. 146–158, 2018, ISSN: 1939-1390. DOI: 10.1109/MITS.2018.2842049.

[79] C. Wissing, T. Nattermann, K.-H. Glander, C. Hass, and T. Bertram, "Lane Change Prediction by Combining Movement and Situation based Probabilities", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3554–3559, 2017, 20th IFAC World Congress, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.960.

[80] M. Bahram, C. Hubmann, A. Lawitzky, M. Aeberhard, and D. Wollherr, "A combined model-and learning-based framework for interaction-aware maneuver prediction", *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 17, no. 6, pp. 1538–1550, 2016.

[81] D. Broadbent, *Perception and communication*. Elsevier, 2013.

[82] M. Hall, "Correlation-based features selection for machine learning", Ph.D. dissertation, The university of Waikato, 1999.

[83] M. Hall, "Correlation-based feature selection for discrete and numeric class machine learning", in *Proceedings of the Seventeenth International Conference on Machine Learning, 2000*, pp. 359–366.

[84] J. Hauke and T. Kossowski, "Comparison of values of Pearson's and Spearmans's correlation coefficients on the same sets of data", *Quaestiones geographicae*, vol. 30, no. 2, pp. 87–93, 2011.

[85] H. Zhang, "The Optimality of Naive Bayes.", in *Proceedings of the FLAIRS Conference*, V. Barr and Z. Markov, Eds., AAAI Press, 2004.

[86] C. Chang and C. Lin, "LIBSVM: a library for support vector machines", *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[87] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Berlin: Springer-Verlag, 1995, ISBN: 0387945598.

[88] L. Breiman, "Random forests", *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[89] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[90] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[91] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[92] NVIDIA, *Accelerating AI with GPUs: A New Computing Model | NVIDIA Blog*, https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/, (Accessed on 08/21/2019).

[93] A. Maas, A. Hannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models", in *Proceedings of the ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, vol. 30, 2013, p. 3.

[94] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks", *Chemometrics and intelligent laboratory systems*, vol. 39, no. 1, pp. 43–62, 1997.

[95] Q. Le *et al.*, "A tutorial on deep learning part 1: Nonlinear classifiers and the backpropagation algorithm", *Google Inc., Mountain View, CA*, p. 18, 2015.

[96] A. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: A tutorial", *Computer*, no. 3, pp. 31–44, 1996.

[97] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005, ISBN: 0262201623.

[98] A. Barth and U. Franke, "Where Will the Oncoming Vehicle be the Next Second?", in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Eindhoven: Springer, Jun. 2008, pp. 1068–1073.

[99] H. Woo, Y. Ji, H. Kono, *et al.*, "Lane-change detection based on vehicle-trajectory prediction", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1109–1116, 2017.

[100] F. Altché and A. De La Fortelle, "An LSTM network for highway trajectory prediction", in *Proceedings of the 2017 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 353–359.

[101] L. Fahrmeir, C. Heumann, R. Künstler, I. Pigeot, and G. Tutz, *Statistik: Der Weg zur Datenanalyse*. Springer Spektrum, 2016.

[102] S. Yoon and D. Kum, "The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles", in *Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2016, pp. 1307–1312.

[103] C. Wissing, T. Nattermann, K. Glander, and T. Bertram, "Trajectory Prediction for Safety Critical Maneuvers in Automated Highway Driving", in *Proceedings of the 2018 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 131–136.

[104] M. L. Connelly, H. Conaglen, B. Parsonson, and R. Isler, "Child pedestrians' crossing gap thresholds", *Accident; analysis and prevention*, vol. 30, pp. 443–53, Aug. 1998. DOI: 10.1016/S0001-4575(97)00109-7.

[106] M. Liebner, M. Baumann, F. Klanner, and C. Stiller, "Driver intent inference at urban intersections using the intelligent driver model", in *Proceedings of the 2012 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1162–1167.

[107] D. Petrich, T. Dang, D. Kasper, G. Breuel, and C. Stiller, "Map-based long term motion prediction for vehicles in traffic environments", in *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems - (ITSC)*, Oct. 2013, pp. 2166–2172. DOI: 10.1109/ITSC.2013.6728549.

[108] J. Wiest, F. Kunz, U. Kreßel, and K. Dietmayer, "Incorporating categorical information for enhanced probabilistic trajectory prediction", in *Proceedings of the 2013 International Conference on Machine Learning and Applications (ICMLA)*, vol. 1, pp. 402–407.

[109] B. T. Morris and M. M. Trivedi, "Trajectory Learning for Activity Understanding: Unsupervised, Multilevel, and Long-Term Adaptive Approach", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 11, pp. 2287–2301, Nov. 2011, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2011.64.

[110] N. Deo, A. Rangesh, and M. Trivedi, "How would surround vehicles move? A unified framework for maneuver classification and motion prediction", *IEEE Transactions on Intelligent Vehicles (T-ITS)*, vol. 3, no. 2, pp. 129–140, 2018.

[111] N. Deo and M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction", in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pp. 1468–1476.

[112] D. Lenz, F. Diehl, M. Le, and A. Knoll, "Deep neural networks for Markovian interactive scene prediction in highway scenarios", in *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 685–692.

[113] C. Wissing, T. Nattermann, K. Glander, and T. Bertram, "Probabilistic time-to-lane-change prediction on highways", in *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1452–1457.

[114] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles", English, *ROBOMECH Journal*, vol. 1, no. 1, 1, 2014. DOI: 10.1186/s40648-014-0001-z.

[115] G. King and L. Zeng, "Logistic regression in rare events data", *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.

[116] G. M. Weiss, K. McCarthy, and B. Zabar, "Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs?", *Dmin*, vol. 7, no. 35-41, p. 24, 2007.

[117] S. R. Searle, *Linear models for unbalanced data*. John Wiley & Sons, 2006, vol. 639.

[118] S. Calinon, F. Guenter, and A. Billard, "On Learning, Representing and Generalizing a Task in a Humanoid Robot", *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 37, no. 2, pp. 286–298, 2007.

[119] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm", *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[120] W. Chaer, R. Bishop, and J. Ghosh, "A mixture-of-experts framework for adaptive kalman filtering", *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 27, no. 3, pp. 452–464, 1997.

[121] J. Kwok, "Support vector mixture for classification and regression problems", in *Proceedings of the Fourteenth International Conference on Pattern Recognition*, vol. 1, 1998, pp. 255–258.

[122] S. Yuksel, J. Wilson, and P. Gader, "Twenty years of mixture of experts", *IEEE transactions on neural networks and learning systems*, vol. 23, no. 8, pp. 1177–1193, 2012.

[123] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates", in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2002, pp. 694–699.

[125] S. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[126] P. Bender, O. Tas, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments", in *Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1386–1392.

[127] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha—A local, continuous method", in *Proceedings of the 2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 450–457.

[128] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving", *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1105–1118, 2009.

[129] M. McNaughton, C. Urmson, J. Dolan, and J. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice", in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4889–4895.

[130] E. Frazzoli, M. Dahleh, E. Feron, *et al.*, "Maneuver-based motion planning for nonlinear systems with symmetries", *Robotics, IEEE Transactions on*, vol. 21, no. 6, pp. 1077–1091, 2005.

[131] T. Gu, J. Atwood, C. Dong, J. Dolan, and J. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving", in *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 250–256.

[132] N. Kapania, J. Subosits, and C. Gerdes, "A sequential two-step algorithm for fast generation of vehicle racing trajectories", *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 9, p. 091 005, 2016.

[133] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: Introductory theory and examples", *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[134] G. Guennebaud, B. Jacob, *et al.*, *Eigen C++*, http://eigen.tuxfamily.org, (Accessed on 01/18/2016), 2016.

[135] H. Bellem, B. Seitz, M. Schrauf, and J. Krems, "Comfort in automated driving: An analysis of preferences for different automated driving styles and their dependence on personality traits", *Transportation Research Part F Traffic Psychology and Behaviour*, vol. 55, Jun. 2018. DOI: 10.1016/j.trf.2018.02.036.