Perception and Prediction of Urban Traffic Scenarios for Autonomous Driving

Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.) am Fachbereich Mathematik und Informatik der Freien Universität Berlin

vorgelegt von

Andreas Philipp

Berlin, 2021

Erstgutachter:	Prof. Dr. Daniel Göhring
Zweitgutachter:	Prof. Dr. Manfred Hild

Tag der Disputation: 2.12.2021

ABSTRACT

This dissertation describes a novel system for the perception and prediction of urban traffic scenarios for autonomous driving. It is based on the AutoNOMOS self driving car project of the Freie Universität Berlin. The system combines best practices with new methods and findings. The description and evaluation of these new methods and findings form the main contributions of the thesis. These contributions are evaluated using real-world traffic scenarios recorded during test drives with the self driving vehicle MadeInGermany of the FU Berlin. Since the system also handles very complex and potentially dangerous traffic scenarios, a simulation environment has been developed to evaluate the system also under such conditions.

The perception system uses a LIDAR device which readings are processed as range image. The first contribution of this work is a new algorithm to identify pixels originating from the ground which is proven to be more efficient than existing methods.

The tracking of moving objects is often done by Kalman filters and many previous publications propose to use curvilinear motion models for these filters. The evaluation of real-world traffic scenarios proves, that theses motion models are unsuitable for urban traffic, since the filters get unstable at slow velocities.

A new algorithm, which very efficiently calculates the collision probability between two rectangular objects, is the third contribution.

The main contribution of this work is a new rule-based interaction-aware multi-modal prediction method for urban traffic scenarios. The method takes into account all classes of traffic participants as cars, trucks, bicycles and pedestrians and handles all relevant types of motion behaviors, as car following, lane changes and merges, turning and intersection crossing. The system is able to predict very complex urban scenarios with several dozen agents for 10 seconds and more at a frequency of 10 Hz.

The last contribution is a simulation system which allows to evaluate the prediction results also in dangerous scenarios and furthermore to show that the generated prediction can also be useful for the local behavior planning of an autonomous vehicle.

KURZFASSUNG

Diese Dissertation beschreibt ein neuartiges System zur Wahrnehmung und Vorhersage von urbanen Verkehrsszenarien für autonomes Fahren. Es basiert auf dem AutoNOMOS-Projekt für autonomes Fahren der Freien Universität Berlin. Das System kombiniert bewährte Verfahren mit neuen Methoden und Erkenntnissen. Die Beschreibung und Bewertung dieser neuen Methoden und Erkenntnisse bildet die Hauptbeträge der Arbeit. Bewertet werden diese Beiträge anhand von realen Verkehrsszenarien, die bei Testfahrten mit dem autonomen Fahrzeug MadeInGermany der FU Berlin aufgenommen wurden. Da das System auch sehr komplexe und potenziell gefährliche Verkehrsszenarien verarbeitet, wurde eine Simulationsumgebung entwickelt, um das System auch unter solchen Bedingungen zu bewerten.

Das Wahrnehmungssystem verwendet ein LIDAR-Gerät, dessen Messwerte als Entfernungsbild verarbeitet werden. Der erste Beitrag dieser Arbeit ist ein neuer Algorithmus zum Identifizieren von Pixeln, die vom Boden stammen und der sich als effizienter als bestehende Methoden erwiesen hat.

Die Verfolgung sich bewegender Objekte wird oft durch Kalman-Filter durchgeführt und viele frühere Veröffentlichungen schlagen vor, kurvenlineare Bewegungsmodelle für diese Filter zu verwenden. Die Auswertung von realen Verkehrsszenarien zeigt, dass diese Bewegungsmodelle für den Stadtverkehr ungeeignet sind, da die Filter bei langsamen Geschwindigkeiten instabil werden.

Ein neuer Algorithmus, der sehr effizient die Kollisionswahrscheinlichkeit zwischen zwei rechteckigen Objekten berechnet, ist der dritte Beitrag.

Der Hauptbeitrag dieser Arbeit ist eine neue regelbasierte interaktionsbewusste multimodale Vorhersagemethode für urbane Verkehrsszenarien. Die Methode berücksichtigt alle Klassen von Verkehrsteilnehmern wie Autos, Lastwagen, Fahrräder und Fußgänger und verarbeitet alle relevanten Arten von Bewegungsverhalten, wie Kolonnenfahren, Spurwechsel und -zusammenführung, Abbiegen und Kreuzungsüberquerung. Das System ist in der Lage, sehr komplexe urbane Szenarien mit mehreren Dutzend Agenten für 10 Sekunden und mehr bei einer Frequenz von 10 Hz vorherzusagen.

Der letzte Beitrag ist ein Simulationssystem, das es erlaubt, die Vorhersageergebnisse auch in gefährlichen Szenarien auszuwerten und darüber hinaus zu zeigen, dass die generierte Vorhersage auch für die lokale Bewegungsplanung eines autonomen Fahrzeugs nützlich sein kann.

ACKNOWLEDGEMENTS

I would like to thank Prof. Dr. Daniel Göhring for introducing me to the world of autonomous driving and for motivating me to create this thesis. Without his support and encouragement, I would not have made it to this point.

Also I want to thank Prof. Dr. Rául Rojas, who supported me especially in the beginning of my work with his experience and his critical advises.

My further thanks goes to my colleagues and fellow Ph.D students in the AutoNOMOS project of the Freie Universität Berlin: Khaled Alomari, Zahra Boroujeni, Bingyi Cao, Ricardo Carillo, Tobias Langner, Daniel Neumann, Claas Norman Ritter, Nicolai Steinke, Stephan Sundermann, Fritz Ulbrich and Sun Yiteng.

Last but not the least, I would like to thank my sister Karin Philipp for helping me with her profound knowledge of the English language.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Roman Symbols

a	Acceleration in curvilinear motion model (Chapter 3)
a	Desired acceleration for IDM (Chapter 5)
a_c	Centripetal acceleration
a_{free}	Free driving component of IDM acceleration
a_{idm}	IDM acceleration
$\mathbf{a}_k^i$	Action vector of object i at time step k
a_{int}	Interaction component of IDM acceleration
a_{lat}	Lateral acceleration in Frenet frame
a_{lon}	Longitudinal acceleration in Frenet frame
a_{maxlat}	Maximal lateral acceleration during lane change
a_{maxsa}	Lateral acceleration at maximal steering angle
$\operatorname{arctanh}(\cdot)$	Arcus tangens hyperbolicus function
a_x	x-acceleration in free motion model
a_y	y-acceleration in free motion model
b	Comfortable deceleration for IDM
b_{crssng}	Required deceleration for lane crossing
b_{kin}	Kinematic deceleration for IDM

b_{limit} Deceleration required for speed limit

$\mathbf{C}_k$ Collision matrix at time step k

d Lateral position in Frenet frame

$d_{inno,k}$ Innovation distance at time step k

$d_{innoRel,k}$ Relative innovation distance at time step k

$\overline{d}_{innoRel,k}$ Smoothed relative innovation distance at time step k

d_s Driving style

d_{ts} Distance between line of sight and stop line

e_p Position error.

$erf(\cdot)$ Gauss error function

$\mathbf{F}$ System matrix of linear state space system

$\mathbf{f}(\cdot)$ System function of state space system

$\mathbf{F}_k$ Linearized system matrix of non-linear state space system at time step k

$\mathbf{H}$ Measurement matrix for linear measurement function

$\mathbf{h}(\cdot)$ Measurement function

$I_c(\cdot)$ Collision indicator function

$\mathcal{I}$ Set of interaction constraints for scenario prediction

$\mathbb{1}$ Identity matrix

$\mathbf{J}$ Jacoby matrix of system function

$\mathbf{K}_k$ Kalman gain matrix at time step k

l_c Center line of lane

l_k Lane change incentive at time step k

l_{len} Domain limit for lane change incentive

l_{min} Lower limit for lane change incentive

l_p Position likelihood.

l_{veh} Vehicle length

l_{wb} Vehicle wheelbase

$M_k^{i,j}$ Maneuver j of object i at time step k

$\mathcal{M}_k^i$ Set of maneuvers of object i at time step k

$\mathcal{N}(\cdot,\cdot)$ Normal distribution

O_k^i Object i at time step k

$\mathcal{O}_k$ Set of objects at time step k

$p(\cdot)$ Probability density function

p_{act} Actual driving style parameter

p_{aggr} Aggressive driving style parameter

P_B Birth probability

P_C Clutter probability

P_{conf} Confirmation threshold

$\mathbf{P}_{j,k}^M$ Mixin covariance of mode j at time step k

P_D Detection probability

p_{def} Defensive driving style parameter

$\mathbf{P}_{dlat}^k$ Variance of lateral position at time step k

$p_{E,k}$ Existence probability at time step k

$\mathbf{P}_k$ Covariance matrix of state at time step k

$\mathbf{P}_\mathbf{k}^-$ Predicted covariance matrix of state at time step k

$\mathbf{P}_{lat}^k$ Covariance matrix of lateral motion at time step k

$\mathbf{P}_{lon}^k$ Covariance matrix of longitudinal motion at time step k

P_P Persistence probability

P_{unconf} Unconfirmation threshold

$\mathbf{Q}$ Process noise covariance matrix of state space system

q Process noise intensity

$\mathbf{R}$ Measurement noise covariance matrix

$\mathcal{R}$ Set of map based constraints for scenario prediction

r_{min} Minimum curve radius

s Actual distance to preceding vehicle for IDM (Subsection 5.8.2)

s Longitudinal position in Frenet frame (Subsection 5.8.1)

$S(\cdot)$ Set of occupied points of object with given pose

s_0 Minimum distance to static obstacle for IDM

s_{crssng} Distance to lane crossing zone

$\mathbf{S}_{j,k}$ Innovation covariance of mode j at time step k

$\mathbf{S}_k$ Innovation covariance matrix at time step k

S_k Turn signal state at time step k

s_{limit} Distance to begin of speed limit

s_{sight} Distance to line of sight

s^* Minimum gap for IDM

s_{stop} Distance to stop line

T Time gap for IDM

t_{max} Time of maximal lateral acceleration during lane change

$tanh(\cdot)$ Tangens hyperbolicus function

Tc Time constant for Ornstein-Uhlenbeck process

t_{crssng} Time until clearance of lane crossing zone

$T_k^{i,j}$ Predicted trajectory for maneuver j of object i at time step k

$\mathcal{T}_k^i$	Set of predicted trajectories of object i at time step k
t_{yellow}	Duration of yellow phase at traffic light
$\mathcal{U}(\cdot,\cdot)$	Uniform probability distribution
v	Actual velocity for IDM (Chapter 5)
v	Speed in curvilinear motion model (Chapter 3)
v_0	Desired velocity for IDM
$\dot{v}$	Momentary change of velocity for IDM
v_{lat}	Lateral velocity in Frenet frame
v_{limit}	Speed limit
v_{lon}	Longitudinal velocity in Frenet frame
$\mathbf{v}_k$	Measurement noise vector at time step k
v_x	x-velocity in free motion model
v_y	y-velocity in free motion model
$\mathbf{w}_k$	Noise vector of state space system
w_l	Lane width
$w_i^{(c)}$	i-th weight for covariance calculation of UKF
$w_i^{(m)}$	i-th weight for mean calculation of UKF
w_v	Vehicle width
$\mathbf{x}$	State vector
$\tilde{\mathbf{x}}$	Error of state vector
$\mathbf{x}_k$	State vector at time step k
$\mathbf{x}_k^i$	State vector of object i at time step k
$\mathbf{x}_k^{i,j}$	State vector for maneuver j of object i at time step k
$\mathbf{x}_{k:k+T}$	State vectors for time steps k to $k+T$

$\mathbf{x}_k^-$	Predicted state vector at time step k
$\hat{\mathbf{x}}_\mathbf{k}^-$	Mean value of predicted state vector at time step k
$\hat{\mathbf{x}}_{j,k}^M$	Mixin mean state of mode j at time step k
x_n	x-coordinate of object n in local or global coordinate system
$\hat{\mathcal{X}}_k$	Vector of sigma points for UKF at time step k
$\hat{\mathcal{X}}_{i,k}k^-$	i-th predicted sigma point for UKF at time step k
$\hat{\mathcal{X}}_k^-$	Vector of predicted sigma points for UKF at time step k
$\tilde{x}$	Error of x-coordinate of object position
y_n	y-coordinate of object n in local or global coordinate system
$\tilde{y}$	Error of y-coordinate of object position
$\mathbf{z}$	Measurement vector
$\mathbf{z}_D$	Dimension measurement of object.
$\hat{\mathbf{z}}_{j,k}^-$	Mean of predicted measurement vector of mode j at time step k
$\mathbf{z}_k^i$	Measurement vector of object i at time step k
$\hat{\mathbf{z}}_k^-$	Mean vector of predicted measurement at time step k

Greek Symbols

α_i	Decay factor for innovation distance smoothing
α_{LC}	Gradient parameter for lane change incentive
β	Yaw angle of lane
β_k	Steepness factor for lane change trajectory at time step k
δ	Acceleration exponent for IDM
η	Normalization constant
$\mathbf{\Gamma}$	Process noise gain matrix of state space system
κ	Curvature of lane

λ	Scaling parameter for UKF
$\lambda_{j,k}$	Likelihood of mode j at time step k
$\boldsymbol{\mu}_c$	Mean values of object class dimensions.
μ	Mode probability
$\mu_{i,k}$	Probability of mode i at time step k
$\mu^-_{(i\|j),k}$	Mixin probability prior of mode i at time step k
$\mu^-_{i,k}$	Predicted probability of mode i at time step k
ω	Turn rate in curvilinear motion model
ϕ	Heading in curvilinear motion model
φ_n	yaw angle of object n relative to local or global coordinate system
$\boldsymbol{\Pi}$	Transition matrix
$\boldsymbol{\Pi}_{i,j}$	Probability of transition from mode i to mode j
σ_a	Acceleration as process noise for second order state space system
$\boldsymbol{\Sigma}_c$	Covariance matrix of object class dimension.
σ_d	Standard deviation of lateral position in Frenet frame
$\sigma_{\Delta a}$	Acceleration increment as process noise for state space system
$\sigma_{\Delta\omega}$	Turn rate increment as process noise for state space system
$\sigma_{\Delta\phi}$	Heading increment as process noise for state space system
$\sigma_{\Delta v}$	Velocity increment as process noise for state space system
$\sigma_{\Delta x}$	Position increment as process noise for state space system
σ_{dlat}	Standard deviation of lateral position
σ_x	Standard deviation in x-direction
σ_y	Standard deviation in y-direction
σ_s	Standard deviation of longitudinal position in Frenet frame

σ_{vlat} Standard deviation of lateral velocity in Frenet frame

σ_{vlon} Standard deviation of longitudinal velocity in Frenet frame

$\mathbf{\Sigma}_{xy}$ Covariance matrix of object position in the plane.

Acronyms / Abbreviations

ACC Adaptive Cruise System

ADAS Advanced Driver Assistance System

ASLDS Augmented switching linear dynamical system

aSSSM Augmented Switching State-Space Model

BEV Birds Eye View

BFS Breadth First Search

BGMM Bernoulli Gaussian Mixture Model

CA Constant Acceleration

CARLA Car Learning To Act

CBTS Continuous Belief Tree Search

CEP Collision Event Probability

CSP Collision State Probability

CHA Constant Heading and Acceleration

CHV Constant Heading and Velocity

CNN Convolutional Neural Network.

CP Constant Position

CPT Conditional Probability Table

CRF Conditional Random Field

CTRA Constant Turn Rate and Acceleration

CTRV Constant Turn Rate and Velocity

CV Constant Velocity

DARPA Defense Advanced Research Projects Agency

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DCLMR Dahlem Center for Machine Learning and Robotics

DGPS Differential Global Positioning System

EKF Extended Kalman Filter

EM Expectation-Maximization

FDM Forsighted Driver Model

FFT Fast Fourier Transform

FISST Finite Set Statistics

FOV Field of View

FUB Freie Universität Berlin

GAN Generative Adversarial Networks

GCC GNU Compiler Collection

GGIW Gamma Gaussian Inverse Wishart

GMM Gaussian Mixture Model

GNN Global Nearest Neighbor

GNSS Global Navigation Satellite System

GPG Generalized Policy Graph

GPR Gaussian Process Regression

GPS Global Positioning System

GSL GNU Scientific Library

HMM Hidden Markov Model

ICRA International Conference on Robotics and Automation

IDM Intelligent Driver Model

IIDM Improved Intelligent driver model

IMM Interacting Multiple Model

IMU Inertial Measurement Unit

IRL Inverse Reinforcement Learning

JPDA Joint Probabilistic Data Association

KL Keep Lane Maneuver

KL Kullback-Leibler

KNN K-Nearest-Neighbors

LAPJV Linear Assignment Problem Jonker-Volgenant

LCl Lane Change left Maneuver

LCr Lane Change right Maneuver

LIDAR Light Detection And Ranging

LKF Linear Kalman Filter

LMB Labeled Multi-Bernoulli

LNN Local Nearest Neighbor

LSTM Long Short-Term Memory

LTS Long Term Support

MCS Monte Carlo Simulation

MHT Multi Hypothesis Tracking

$MM-UKF$ Multiple Model Uncented Kalman Filter

MOBIL Minimizing Overall Braking Induced by Lane Changes

MPDM Multipolicy Decision-Making

MSE Mean Squared Error

NEES Normalized Estimation Error Squared

NIS Normalized Innovation Squared

OBB Oriented Bounding Box

KL Optimal Control Problem

PDA Probabilistic Data Association

PDF Probability Density Function

PHD Probability Hypothesis Density

POMCP Partially Observable Monte-Carlo Planning

POMDP Partially Observable Markov Decision Process

POSLV Position and Orientation System for Land Vehicles

RANSAC Random Sample Consensus

RFS Random Finite Set

RMSE Rooted Mean Square Error

RNDF Route Network Definition File

ROS Robot Operating System

RTT Rapidly-exploring random tree

SVM Support Vector Machine

TAPIR Toolkit for Approximating and Adapting POMDP Solutions In Real time

TR Trash Maneuver

TTB Time-To-Brake

TTC Time-To-Collision

TTR Time-To-React

TUl Turn left Maneuver

TUr Turn right Maneuver

UKF Uncented Kalman Filter

$VB-EM$ Variational Bayesian Expectation-Maximization

$VGMM$ Variational Gaussian Mixture Model

1 Introduction

1.1 Motivation and Problem Description

Autonomous driving is one of the most watched research topics of the last years. There are hopes that autonomous driving will have various benefits. First of all, it should reduce the number of fatalities and injuries caused by road traffic. According to the most recent official statistic for Germany [165], there were 300,143 accidents with personal injuries in 2019, 82.8 % of them caused by drivers of motor vehicles. The main causes were mistakes in turning, starting, entering the road etc. (15.9 %), failure to yield right of way (14.5 %), insufficient distance (13.9 %) and inappropriate speed (11.6 %). One can assume that the first two causes are mainly based on inattentiveness, while the distance and speed limit violations result mostly from deliberate disregard of traffic rules. From this follows that, while humans are basically able to drive very safely, traffic accidents are unavoidable since humans are not able to stay highly attentive for a long time and there are traffic participants, which are not willing to obey the rules. Automated systems should be able to overcome these problems.

Autonomous driving will also have severe economic impacts on all industry sectors related to transportation of people and goods. There will be sociological benefits by enhancing the mobility of elderly people, by freeing parents from being taxi driver for their children and it may also slow down the depopulation of rural regions, which is partly caused by poor transportation services. Finally, autonomous driving will enhance the comfort of the people by exempting them from the driving task while traveling and freeing them from unpleasant activities like searching for parking places. But there will also be disadvantages, mainly to the job market in transportation industry. Moreover, it is expected that the traffic volume of private cars will further increase and last not least there may be negative impacts on health when more trips are done by car instead of bicycle or by walking.

The Society of Automotive Engineers (SAE) defines the following levels of automated driving in its J3016 standard [40]:

- Level 0 (No automation)

- Level 1 (Driver assistance): The automation system supports in steering or acceleration/deceleration while the driver performs the remaining functions. Usually called Driver Assistance System (DAS)
- Level 2 (Partial automation): The automation system controls steering and acceleration/deceleration under supervision of the driver. Advanced Driver Assistance System (ADAS).
- Level 3 (Conditional automation): The automation system controls the behavior of the car, but still has to be supervised by the driver. Teslas Autopilot is sometimes used as a Level 3 system.
- Level 4 (High automation): The automation system takes full driving responsibility in certain environments, e.g. on highways.
- Level 5 (Full automation): The automation system takes full driving responsibility in all environments.

Most of the above impacts will be fully achieved only, when level 5 is reached. The aim of this work is to support full automation. It considers all types of roads, as highways, rural roads and urban streets, as well as all types of traffic participants, like motor vehicles, bicycles and pedestrians. Moreover, it is not limited to regular actions of agents, but it detects and predicts also exceptional, unlawful behavior. But most contributions of this work are also useful to enhance the functionality of lower level automation systems.

Autonomous cars are a subclass of mobile robots. A robot is an agent, which acts in the real-world. The main components of an autonomous car system are (see Figure 1.1):

- Objective given by the user of the system
- Proprioceptive and exteroceptive perception of the world
- Prediction about how the world will evolve
- Planning of the own future actions
- Control of the robot's actuators to realize the planned actions

The objective of an autonomous car system is to reach the destination given by the user in a timely and comfortable manner under consideration of legal, technical, economic and ecological constraints.

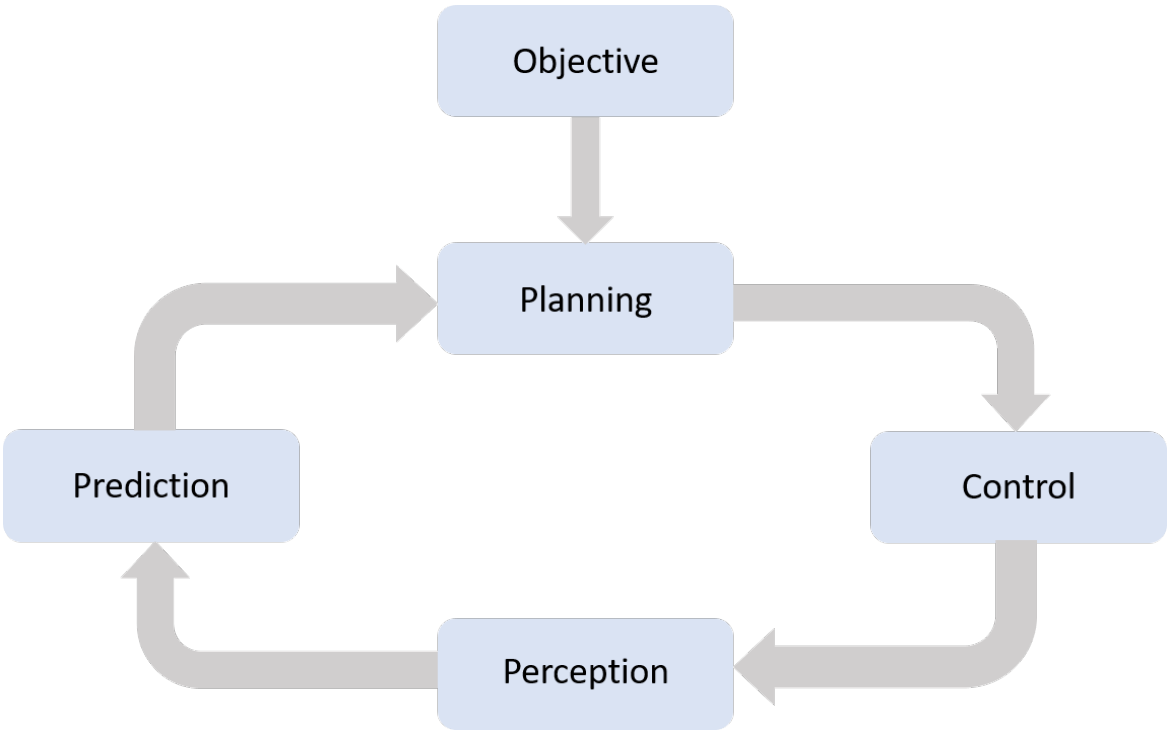


Fig. 1.1 Components of a robotic system.

The proprioceptive perception, i.e. the determination of the autonomous vehicle's own state, is mainly accomplished by the use of a GPS system, Odometry devices and an Inertial Measurement Unit (IMU). Further information may be gained from the data bus of the car electronic. The perception of the ego-state of the car is not further examined in this work and assumed to be given with sufficient accuracy.

The exteroceptive perception, i.e. the determination of the state of the relevant environment of an autonomous car, is mostly accomplished by optical cameras, Radar and LIDAR. In this work, availability of a multi beam LIDAR is assumed and the evaluation of its data is one of the main themes of this work. It has to be noted, however that four out of five theses of this work (see below) are independent of the actual exteroceptive measurement device.

Prediction of the environment is especially important for an autonomous car since the car has to cooperate with many other traffic participants. The main difficulties arise from the high velocities of the agents navigating in limited space, which requires long prediction horizons. The development of a novel method to create a fully interaction-aware traffic scenario prediction is the main contribution of this work.

Planning is usually divided into global and local planning. Global planning is the task of finding a path between the start and the destination of a trip in a low-resolution roadmap. It is today already solved by many excellent route planners and is not further examined in this

work. Local planning deals with the details of the driving path from the current position to the end of the planning horizon. This horizon is either spatially defined. e.g. a few hundred meters, or temporally, e.g. 5 to 10 seconds. Local planning requires a high-resolution roadmap with detailed data about the available lanes and their permitted use. In addition, the interaction with other road users must be taken into account in accordance with the traffic rules. The prediction approach presented in this work allows to create the local planning simultaneously with the prediction.

The control part of an autonomous driving system has to transform the plan into commands to the actuators of the system, e.g. throttle, brake and steering wheel. Control is not part of this work.

It follows a definition of frequently used terms, as they are used in this work:

- Object: An individual entity perceived in the environment of the autonomous car and the autonomous car itself. Sky and ground are not objects.
- Agent: A mobile object.
- Target: The object, which is in the current context under consideration.
- Obstacle: All objects other than the target.
- Ego-Vehicle: The agent, which carries the perception devices and for which a plan is created.
- State: The collection of relevant data about an object at one moment in time.
- Path: The sequence of locations of an agent in the plane, which have been observed or which are predicted over time.
- Trajectory: The sequence of observed or predicted states of an agent. The trajectory is a combination of the path with a speed profile over time.

1.2 Theses

In this work the following theses are presented:

- Thesis 1: To separate ground and obstacle pixels in LIDAR data, an algorithm based on a range image representation can achieve better results than conventional solutions such as RANSAC.

- Thesis 2: For object tracking in urban environments, Kalman filters with curvilinear motion models are not suitable due to their instability at low velocities.
- Thesis 3: The calculation of the future collision risk between two rectangular moving objects based on an analytic algorithm can be efficient enough to check several thousand trajectory pairs per second.
- Thesis 4: A rule based multi-modal interaction-aware prediction system is able to predict urban traffic scenarios of almost any complexity for up to 10 seconds or longer.
- Thesis 5: Local behavior planning for the ego-vehicle can benefit from interaction-aware trajectory predictions.

1.3 Contributions

The contribution of this work is a software framework for perception, prediction and simulation of urban traffic scenarios. Based on this framework, the above theses are proved by the following contributions:

- Contribution 1: A new algorithm for ground separation in range image representations of LIDAR data is presented. Its superiority in accuracy and efficiency is demonstrated by a qualitative and quantitative comparison to a conventional RANSAC implementation for point clouds.
- Contribution 2: Various Kalman filters for object tracking with different motion models are implemented as part of the framework. The performance of these filters is measured and compared in different real-live urban traffic scenarios. It is shown that the curvilinear filters, as CTRV, become unstable for slow and stopped traffic participants, as they are typical for urban environments.
- Contribution 3: Two new algorithms to calculate the collision risk between two rectangular objects are presented. The performances and the suitability for traffic scenarios of the algorithms are evaluated in a real-world scenario and in a simulation. The results are verified by a comparison to a Monte Carlo implementation. Due to the closed form analytic solution, the new algorithms are up to 800 times faster than the MC implementation and therefore capable to check long-time trajectory predictions of many traffic participants for potential conflicts in real-time.

- Contribution 4: An interaction-aware traffic scenario prediction system is presented. It handles most of the relevant urban traffic situations and it efficient enough to cope with many different traffic participants in real-time. The quality and efficiency of the predictions is evaluated and compared to other methods in four real-world scenarios and additionally in two very complex simulated scenarios.

- Contribution 5: A local behavior plan for the ego-vehicle is created based on the global route plan and processed by the controller of fub_roscar. A simulator combines the result with the prediction for the obstacles to forward the scenario by one time step. The evaluation of the simulated scenarios proves that the generated plans are accident free and efficient.

Further minor contributions of the thesis are presented throughout the text of the further chapters and listed in the related summary sections.

1.4 Structure of the Following Chapters

This work is structured into the following chapters:

- Chapter 2 documents the environment used to develop and to evaluate the methods proposed in this work. This is mainly the hardware of the autonomous test vehicle MadeInGermany and the software of the fub_roscar system. Furthermore, Gaussian distributions and covariance matrices, which are frequently used in subsequent chapters, are introduced.

- Chapter 3 describes an approach to perceive the objects surrounding an autonomous vehicle and to track these over time. It is mainly based on the input of a multi-beam laser detector (LIDAR). It includes the whole stack beginning with the raw data input from the device, separation of the ground pixels, clustering and classification of the objects, assignment of objects to tracks and finally filtering of the object state. The resulting set of objects serves as input for the processing described in the subsequent chapters. Theses 1 and 2 are proven in this chapter.

- Chapter 4 presents a novel approach to collision risk calculation. It takes as an input the probabilistic state of two objects and calculates the collision risk density. Most parts of this chapter have been published upfront as a contribution to the ICRA 2019. This chapter proves thesis 3.

- Chapter 5 is about a novel method to predict complex traffic scenarios over 10 or more seconds. It uses the results of Chapter 3 as state estimate, creates based thereon an intention estimate and predicts for each feasible intention the detailed motion behavior. By leveraging the collision risk calculation of Chapter 4, it achieves full interaction-awareness by analyzing the detected risks and propagating them back as additional input of the next scenario prediction. Proof of thesis 4 is the result of this chapter.
- Chapter 6 extends the method of Chapter 5 by showing, how the predicted trajectory can be used as base for local motion planning. The results are fed into a simulation environment, which supports arbitrary road layouts, to demonstrate the suitability of the approach for various traffic situations and to test it under conditions, which are up to now too complex and too dangerous for test drives with real hardware. Thesis 5 is proven in this chapter.
- Chapter 7 summarizes the work and gives an outlook over future extensions to the different approaches presented in the work.

2 Environment and Prerequisites

This chapter will introduce the environment and some prerequisites of the research presented in the following chapters. Section 2.1 gives an overview of the hardware used for testing and evaluating the contributions of this work. Section 2.2 presents the software environment, which was the base for the developed framework. The detailed roadmap used for navigation is the theme of Section 2.3. Finally, Section 2.4 explains, how Gaussian distributions and covariances matrices are used for perception and prediction in the subsequent chapters.

2.1 Hardware Environment of the Autonomous Test Vehicle MadeInGermany

The research of this thesis is mainly based on the hard- and software of the autonomous test vehicle MadeInGermany [13] of the Freie Universität Berlin. This vehicle is the platform for the AutoNOMOS project at the Dahlem Center for Machine Learning and Robotics (DCLMR) [68]. Many successful projects in the field of autonomous driving have been based on this platform during the last years. The projects reach back until the year 2007, when the predecessor vehicle of MadeInGermany, the Spirit of Berlin, took part in the famous DARPA Urban Challenge 2007.

Figure 2.1 shows MadeInGermany with its sensor configuration. It is a series Volkswagen Passat B6 Variant, 1.8 TSI equipped with various sensors for the perception of the ego-state and the environment.

The most important sensor for this work is the Velodyne LIDAR scanner [97] on the roof of the car (see Figure 2.2). It emits 64 Laser beams with a vertical field of view (FOV) of 26.8° and a horizontal FOV of 360°. The lower 32 lasers, which typically hit targets in a short distance, have an angular resolution of 0.5°, while the upper 32 lasers have a higher angular resolution of 0.33°. This uneven resolution in vertical direction has to be taken into account, when representing the LIDAR input in a range image (see Section 3.3). The horizontal FOV of the rotating LIDAR is 360°. The lasers fire at a frequency of 20,833 Hz.

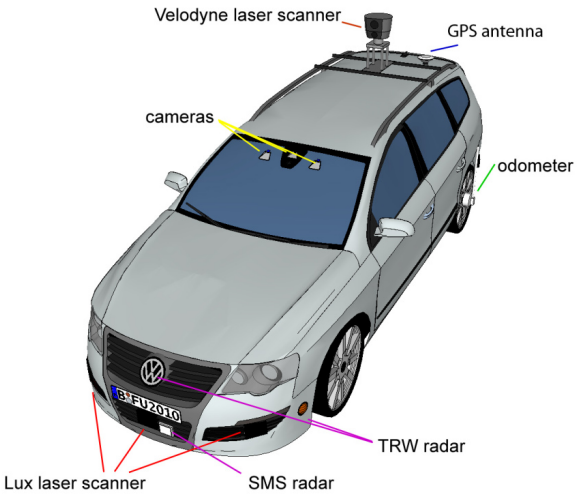


Fig. 2.1 Autonomous test vehicle MadeInGermany of the FU Berlin [13].

For MadeInGermany, a turn rate of 10 Hz of the LIDAR is selected, resulting in a horizontal image size of 2083 pixels ($\approx 0.173^\circ$ angular resolution).

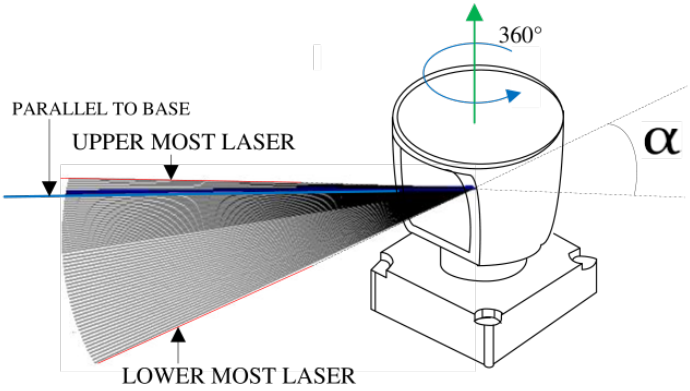


Fig. 2.2 Laser Beam configuration of the Velodyne HDL-64 LIDAR [59].

The second important sensor for this work is the position and navigation system Applanix POS LV 220 [10]. It is the main proprioceptive sensor system of MadeInGermany. It allows highly precise localization of the vehicle in world coordinates. Moreover, it provides accurate information about the orientation and the linear and angular velocity and acceleration. It mainly consists of a Differential Global Positioning System (DGPS) receiver, an inertial measurement unit (IMU), and an odometer. The localization error is under good conditions reduced to a few centimeters, compared to the 5 m of normal consumer electronic devices.

The further sensors of MadeInGermany, as cameras, radar and LUX scanner, are not used in this work.

2.2 Software Environment

The software environment for the research of the work mainly consists of the following components:

- Ubuntu 18.04 (Bionic Beaver) [65], a long term supported (LTS) distribution of the Linux operating system.
- Robot Operating System (ROS Melodic) [66]. ROS is a set of libraries and tools to support the creation and maintenance of robotic applications. It provides a communication infrastructure, which supports the specific needs of real-time operations and standard interfaces for various sensor and actuator devices typically used in robotics. The applications consist of processing units, called nodes. The communication between the nodes is kept as flexible as possible by publishing standardized messages via so called topics, to which the consumer of the data can subscribe. Configuration and operation of the application are supported by some xml-based launch tool. ROS is an open source system mainly implemented in C++.
- fub_roscar, a collection of nodes and libraries for ROS, implementing the autonomous driving functionality of the AutoNOMOS project of the FU Berlin. The software supports the various sensors and actuators of the MadeInGermany test vehicle. Additional nodes implement the required perception, prediction, planning and control functions to be able to operate the vehicle in fully autonomous mode. The software framework used in this work to prove the presented theses is realized as part of the fub_roscar system.
- GNU Compiler Collection (GCC) 7.5 [67], a couple of compilers (mainly C++), and support libraries suitable to develop high performance software for AI applications.

2.3 ATLAS Roadmap

Most parts of this work assume that a detailed roadmap is available and that it is possible to locate the ego vehicle and all other traffic participants in this roadmap.

For the purpose of autonomous driving, a roadmap must be more detailed than the roadmaps used for route planning, like Google maps, Open Street Map or Garmin maps. The exact number of lanes and their location must be defined in the map. The topology of the road network is defined by the lanes and their connections. Further required information are the

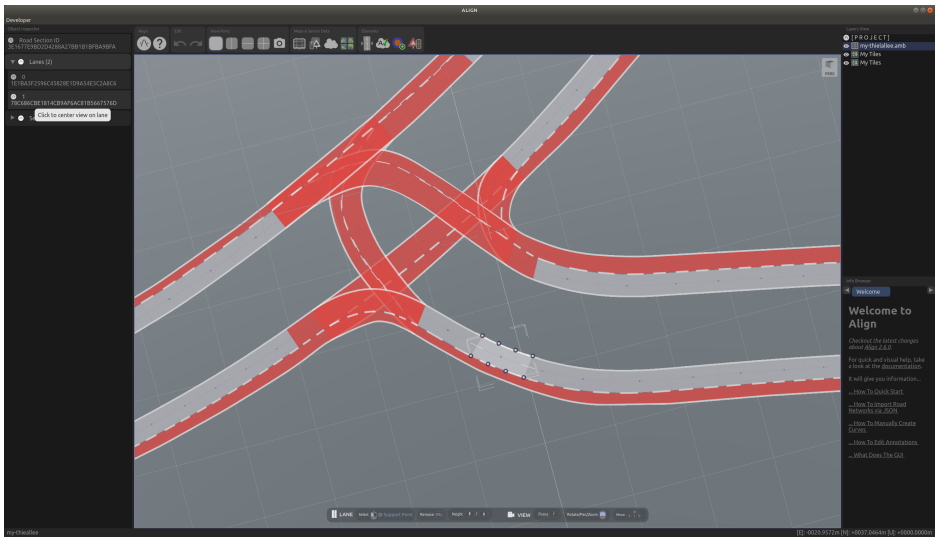


Fig. 2.3 Design of a high precision roadmap by the tool Align v2.6.0 (©2017-2019 TomTom N.V.) .

width of the lane, its driving direction and its type. The roadmap should also provide other information, like speed limits, lane separators, traffic signs and positions of traffic lights. In case of merging and crossing lanes, the priority should be given by the map. In the context of this work, it is assumed that all relevant static environment information comes from the map.

In some approaches, the map data is extracted automatically by the perception system [145] [193].

There are several proposed map formats for autonomous driving. One of the first formats was the Route Network Definition File (RNDF) [37] for the 2007 Darpa Urban Challenge. One of the today's most popular formats is the Lanelet format used for Berta project [27] [144] [8].

The roadmap framework used in this work is the ATLAS Roadmap of TomTom. Figure 2.3 shows a screen shot of the Align tool used to interactively create high precision ATLAS roadmaps.

The Atlas library provides an abstract interface to a underlying roadmap. Similar to the Lanelet format, each roadmap is structured into a number of road sections, which may be connected to each other. Each road sections consists of one or more lanes. The lanes of one road section must be colinear and share the same characteristics, as speed limit and driveability.

In difference to the Lanelet format, each Atlas lane is described by three continuous splines, one center spline, and two boundary splines. Each spline is defined by a couple of support points in 3D coordinates. The advantage of this approach is that the center spline

may be used as reference path for a vehicle. The disadvantage is the higher computational burden when converting from Cartesian coordinates to the lane bound Frenet coordinates and vice versa.

Lanes may be connected in two ways:

- laterally: Two lanes belonging to the same road section may be neighbors. Vehicles switching between neighboring lanes perform a lane change.

- longitudinally: a lane may have incoming and outgoing lanes. If there is more than one incoming lane, vehicles have to perform a lane merge. If there is more than one outgoing lane, vehicles have to decide, whether they go straight or perform a turn.

In the ATLAS library, there are no explicit provisions to model intersections. The presence of an intersection has to be inferred from the geometry of the lanes or from annotations to the map. If the splines of two lanes cross, it has to be inferred from the Cartesian z-value of the intersection point, whether it is an intersection or an overpass/road bridge. This work always assumes intersections.

The only explicit modeling item of the atlas interface are road sections, lane and splines. Any further regulatory traffic information, like speed limits, priority rules, traffic lights must be given in the form of annotations.

2.4 Gaussian Distributions and Covariance Matrices

2.4.1 Gaussian Distribution

In order to quantify the uncertainty about the state of a robot and its environment, a probability distribution must be given. Since most of the uncertain values in robotics are continuous, like position, velocity, orientation etc., the Gaussian distribution is preferably selected to represent the uncertainty. This has several reasons:

- A Gaussian distribution is completely defined by two parameters: the expected value μ and the variance σ^2.
- The distribution of the sum of arbitrarily distributed random variable converges under certain conditions against a Gaussian distribution (Central Limit Theorem [152]).
- Due to the fact being an exponential distribution, exact or approximated inference in probabilistic models is facilitated by use of Gaussian distributions.

The Gaussian distribution is also called Normal distribution. A scalar random variable x is said to be Normal distributed with the expected value μ and the variance σ^2 ($x, \mu, \sigma \in \mathbb{R}$):

$$x \sim \mathcal{N}(\mu, \sigma^2) \tag{2.1}$$

The probability density $p(x)$ of the Gaussian distribution is given by:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.2}$$

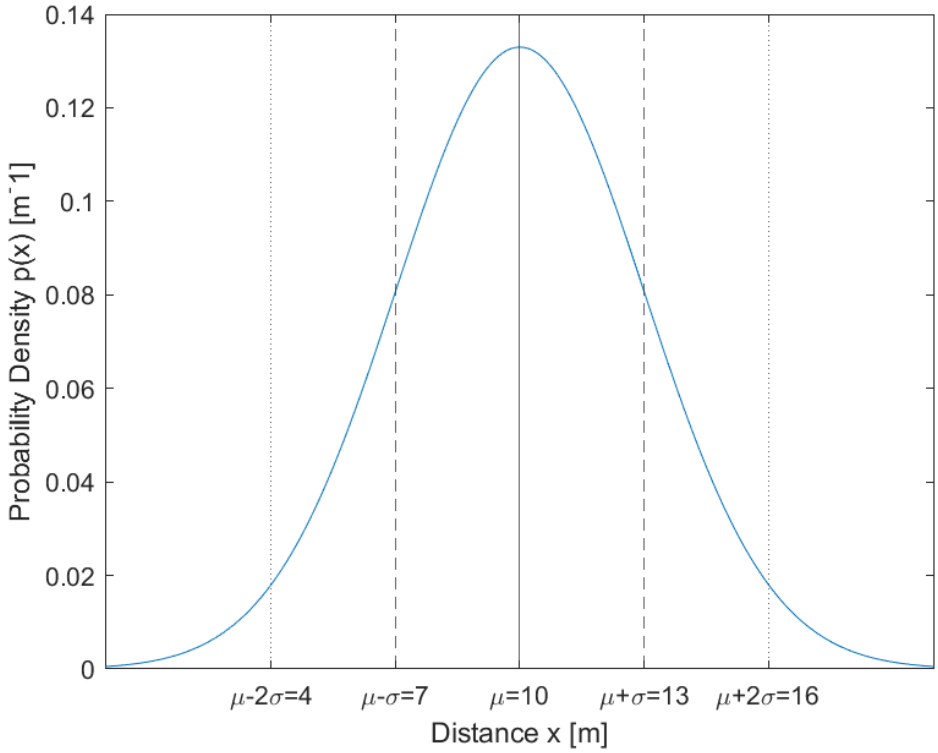


Fig. 2.4 Example for probability density of a Gaussian distribution.

Figure 2.4 shows an example for a Gaussian distribution with $\mu = 10m$ and a standard deviation (the square root of the variance) of $\sigma = 3m$. The example could be the result of the measurement of the distance between a host vehicle and an obstacle using some very noisy sensor. It is assumed that the expected value of the measurement μ meets the real distance x, otherwise the sensor would be biased. The standard deviation σ is usually given by the manufacturer of the measurement device. The area under the curve represents the probability of the received measurement and must be unity for the whole support $\{-\infty \ldots \infty\}$ according to the Kolmogorov axioms [152]. The area in the interval $\mu \pm \sigma$ amounts to $\approx 68,3\%$ probability, the area $\mu \pm 2\sigma$ to $\approx 95.4\%$. Knowledge of this intervals not only gives an intuition of the reliability of a given sensor, but helps also to achieve an ad-hoc estimate if the variance of some distribution is not given. It has to be noted that the Gaussian

distribution always assigns some probability to values < 0, in this example 0.0429%, which is impossible for distance measurements. If that causes problems, another distribution must be selected.

2.4.2 Multivariate Gaussian Distribution

In most realistic cases, the uncertainty of a probabilistic model concerns not only a single isolated random variable, but a combination of values. An example may be the state of a moving obstacle in the plane, consisting of position, velocity and acceleration, all in x- and y-direction. The uncertainty results in a joint distribution of all random variables. If all variables have a Gaussian distribution, the joint distribution is a n-dimensional multivariate Gaussian:

$$\mathbf{x} \sim \mathcal{N}(\mu, \Sigma) \text{ with } \mathbf{x}, \mu \in \mathbb{R}^n \text{ and } \Sigma \in \mathbb{R}^{n \times n} \tag{2.3}$$

The state $\mathbf{x}$ is a column vector consisting of all random variables of the model. The vector μ contains the corresponding expected values. The matrix Σ is the quadratic covariance matrix of the distribution. It contains on its diagonal the variances of the corresponding variables, while the off-diagonal values are the covariances, which quantify the correlation of the variables. If the random variables of the joint distribution are stochastically independent of each other, all covariances are zero, but zero covariances in contrast do not guaranty independence. The probability density of this distribution is given by:

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \, e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \, \Sigma^{-1} \, (\mathbf{x}-\mu)} \tag{2.4}$$

Figure 2.5 shows an example of a 2-dimensional Gaussian distribution. The state consists of the predicted x- and y-position of an obstacle in the plane and the following expected values and covariance matrix:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \quad \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 4 & 0 \\ 0 & 0.25 \end{bmatrix} \tag{2.5}$$

The example could result from the prediction of the position of a moving vehicle on a single lane road with the x-axis in the center of the lane. The dashed curve shows the marginal distribution in x-direction, which has usually a high variance in such a prediction since the longitudinal velocity of a vehicle is uncertain. The dotted curve is the marginal distribution in y-direction, which is much tighter since the lateral deviation from the center

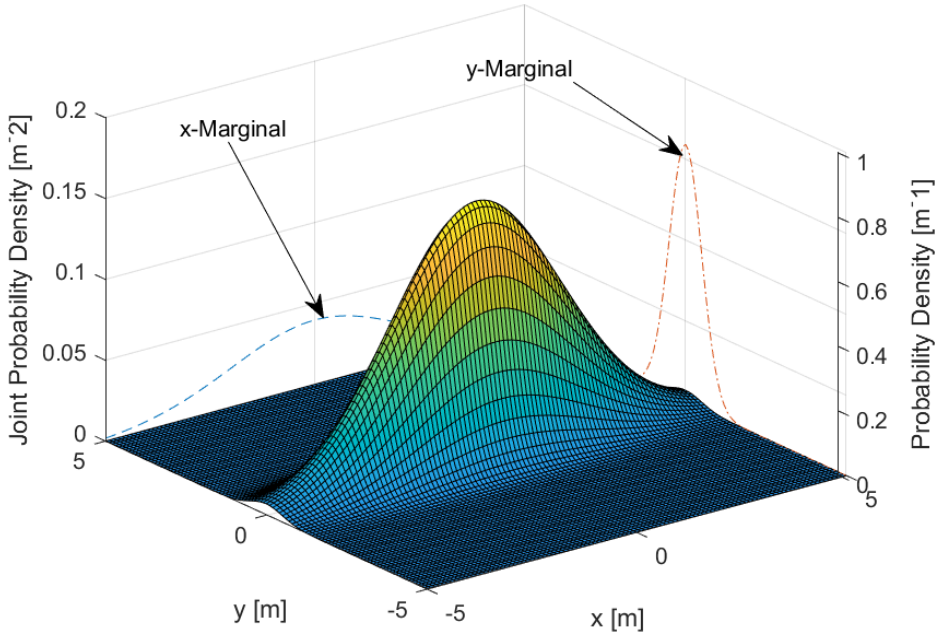
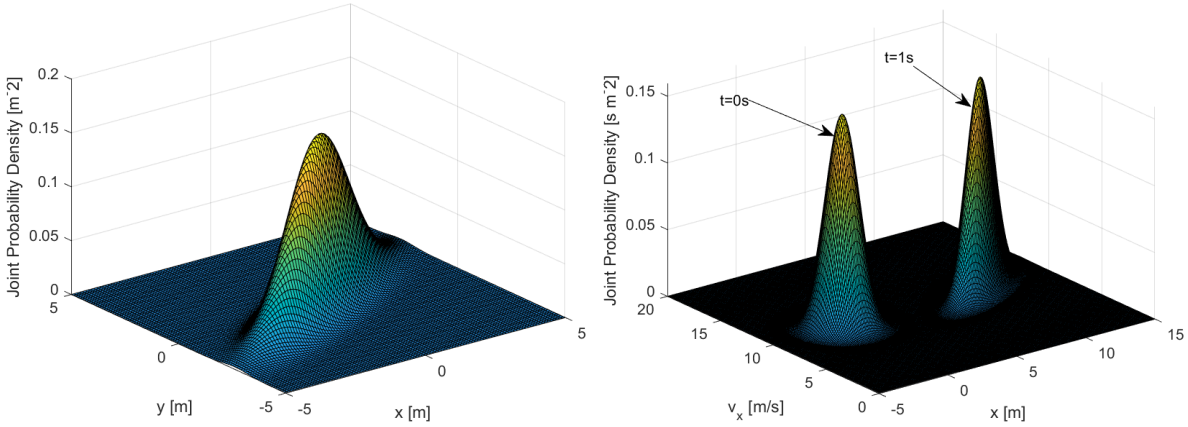


Fig. 2.5 Example for probability density of a multivariate Gaussian distribution and its two marginal distributions.

line is expected to be low. The density of the joint distribution is in m^{-2}, while the probability of marginal distributions is given in m^{-1}.

2.4.3 Linear Transformations of Gaussian Distributions



(a) Gaussian distribution from Figure 2.5 after a rotation of the coordinate system

(b) Gaussian distribution before and after kinematic state transformation

Fig. 2.6 Linear transformations of multivariate Gaussian distributions.

Linear transformations may be applied to Gaussian distributions. One of the most common operations is the transformation to a different coordinate system. Between two Cartesian coordinate systems in the plane, the transformation consists of a translation of the origin T and a rotation of the axes R :

$$\mathbf{T} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} \cos(\alpha) \sin(\alpha) \\ -\sin(\alpha) \cos(\alpha) \end{bmatrix} \tag{2.6}$$

α is the clockwise rotation angle around the origin of the source coordinate system. The transformed expected value and covariance matrix of the Gaussian distribution are:

$$\mu' = \mathbf{R} \times \mu + \mathbf{T} \qquad \Sigma' = \mathbf{R} \times \Sigma \times \mathbf{R}^T \tag{2.7}$$

The covariance matrix is translation independent. Figure 2.6a shows the Gaussian distribution from Figure 2.5 rotated by -30° without any translation. The covariance matrix is now correlated.

Another important linear transformation of a Gaussian distribution is the kinematic state transformation. Assuming a simple one-dimensional system, the uncertain state consists of the position and velocity in x-direction. This state is forwarded one time step Δt by the system matrix $\mathbf{F}$:

$$\mathbf{x} = \begin{bmatrix} x \\ v_x \end{bmatrix} \quad \Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xv_x} \\ \sigma_{xv_x} & \sigma_{v_x}^2 \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{2.8}$$

The system matrix $\mathbf{F}$ describes a constant velocity model. In absence of any system noise, the next state and its covariance matrix is given by:

$$\mathbf{x}^{'} = \mathbf{F} \times \mathbf{x} \qquad \Sigma^{'} = \mathbf{F} \times \Sigma \times \mathbf{F}^{T} \tag{2.9}$$

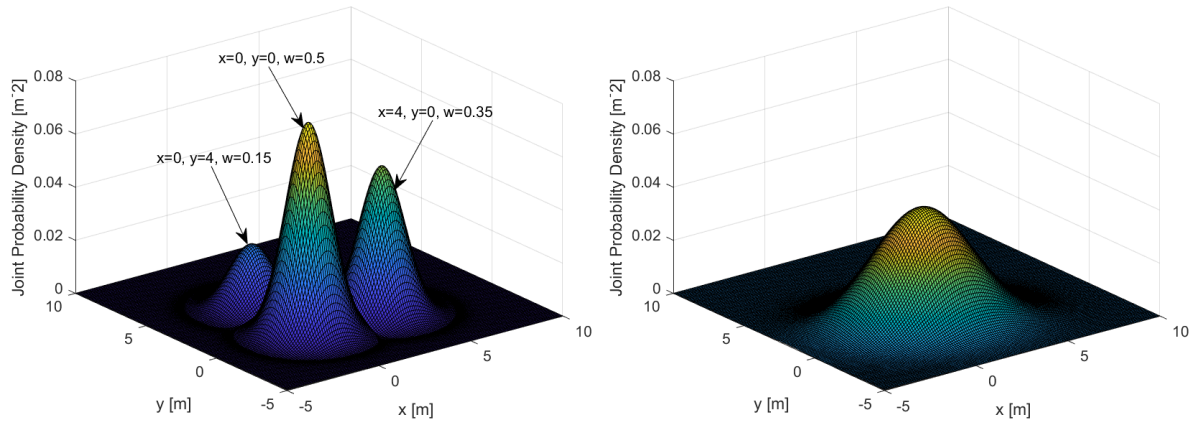
Figure 2.6b shows an example for a kinematic state transformation of a bivariate Gaussian distribution. The initial state is $x = 0m, v_x = 10m/s$ with variances $\sigma_x^2 = 1, \sigma_{v_x}^2 = 1$ and initial covariance 0. After propagating the state for $\Delta t = 1s$ into the future, the position is incremented and the covariance matrix is correlated. The pictures shows that the distribution has rotated and is stretched.

Since both example transformations are deterministic operations, the determinant of the covariance matrix $|\Sigma|$, which is also called the generalized variance, remains unchanged. Therefore, the differential entropy $H(\mathbf{x})$ of a n-dimensional Gaussian distribution [4] given by

$$H(\mathbf{x}) = \frac{1}{2}\ln|\Sigma| + \frac{n}{2}(1 + \ln(2\pi)) \tag{2.10}$$

is constant. From this follows also that the maximal probability density $\max(p(\mathbf{x}))$ does not change. Moreover, after the rotation operation, the covariance matrix has the same Eigenvalues and therefore the shape of the distribution is the same. This does not hold for the kinematic state transformation, which stretches the shape in the presented example.

2.4.4 Gaussian Mixture Distributions



(a) Mixture of three Gaussian distributions with their expected values and weights.

(b) Approximated Gaussian mixture distribution created by moment matching.

Fig. 2.7 Gaussian mixture distribution and its approximation.

There are cases, when the uncertainty cannot be represented by single Gaussian distribution. This can happen, for example, when there is a discrete number of different hypothesis about the state of an object. Figure 2.7a shows an example for the uncertainty of the position in the plane. There are three hypotheses about the state, eventually resulting from different motion models. Each hypothesis has its own expected value, for which reason the distribution is also called multi-modal distribution. Also the covariance matrices may differ. The n-dimensional probability density of such a distribution with m components is given by:

$$p(\mathbf{x}) = \sum_{i=1}^{m} w_i \frac{1}{\sqrt{(2\pi)^n |\Sigma_i|}} e^{-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{i})^T \Sigma_i^{-1} (\mathbf{x}-\mu_\mathbf{i})} \tag{2.11}$$

The weights w_i of the individual components must sum to unity. The resulting probability distribution is not anymore a Gaussian distribution. If required, the Gaussian mixture can be approximated by a uni-modal Gaussian distribution using moment matching:

$$\bar{\mu} = \sum_{i=1}^{m} w_i \mu_\mathbf{i} \tag{2.12}$$

$$\bar{\Sigma} = \sum_{i=1}^{m} w_i (\Sigma_i + (\mu_\mathbf{i} - \bar{\mu})(\mu_\mathbf{i} - \bar{\mu})^T) \tag{2.13}$$

Figure 2.7b shows the approximated Gaussian distribution of the Mixture in Figure 2.7a.

2.4.5 Gaussian Prior and Likelihood

Sometimes, there are two Gaussian distributions for a random variable and they should be combined into a common distribution. One of the distributions may describe the prior knowledge about the variable, resulting in a prior distribution $x \sim \mathcal{N}(\mu_0, \sigma_o^2)$. Additionally, a measurement value z with variance σ_z^2 may be available. If the measurement device is not biased, the random value x is the expected value of the measurement. The distribution for the result of the measurement is then $z \sim \mathcal{N}(x, \sigma_z^2)$, called likelihood. The combination of the two distributions gives the posterior distribution (Bayes rule [152]):

$$p(x|z) = \eta \, \mathcal{N}(z; x, \sigma_z^2) \, \mathcal{N}(x; \mu_0, \sigma_o^2) \tag{2.14}$$

Since the multiplication of two Gaussian functions does not yield a valid probability density function, the normalization constant η is required. Calculation of η requires integration over the multiplication result, for which often no closed solution is available. But since the Gaussian distribution is a conjugate prior to itself [188], it is guaranteed that the posterior distribution is also Gaussian and its parameters can be calculated as follows [152]:

$$\mu_1 = \frac{z\sigma_0^2 + \mu_0\sigma_z^2}{\sigma_0^2 + \sigma_z^2} \tag{2.15}$$

$$\sigma_1^2 = \frac{\sigma_0^2 + \sigma_z^2}{\sigma_0^2 + \sigma_z^2} \tag{2.16}$$

Corresponding formulas are available for the multivariate distributions. These equations are the base for the update step of the Kalman filter (see Chapter 3). Taking the posterior as prior for the next measurement update, a recursion can be build yielding more and more accurate results.

Figure 2.8 shows an example for the recursive application of the Bayes rule. There is an obstacle at position $x = 5m$. The dashed Gaussian is the prior distribution with $\mu_0 = 0m$ and $\sigma_0^2 = 9m^2$. The solid curves are the posteriors after receiving three measurements $z = 2m, z = 6m$ and $z = 4m$ with a variance of $\sigma_z^2 = 4m^2$. The maximum of the posterior density functions moves closer to the real position and the variance is reduced with each step.

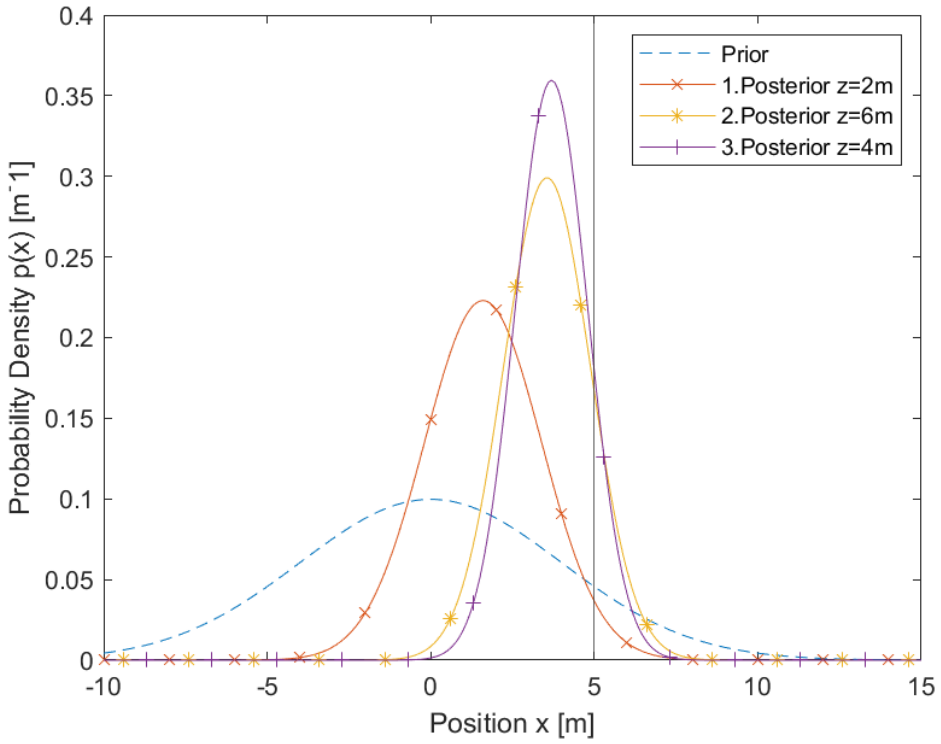


Fig. 2.8 Gaussian prior and posterior distributions after 3 measurements.

3 Perception and Multi Object Tracking

3.1 Motivation and Problem Description

Perception and Multi Object Tracking are prerequisites for each mobile robot to be able to predict the future environment and plan any actions. One important part of the robot perception is the localization and mapping. For the methods presented in this chapter, a detailed roadmap is advantageous, but not mandatory. The kinematic state of the ego-vehicle is measured using a Global Navigation Satellite System (GNSS), Odometry and Inertial Measurement Unit (IMU) devices. The focus of this work lies on perception and tracking of static and dynamic obstacles using a Velodyne HDL 64 LIDAR.

Perception of obstacles requires at first to detect them in the environment and determine their position in the local coordinate system. Next step is to estimate their spatial extent in the form of a bounding box and to determine the orientation of the box. Finally, the object should be at least roughly classified into categories like car, pedestrian, building etc.

The detection of an object from a LIDAR does not allow to determine its kinematic state. The estimation of the velocity and the heading of a moving object requires at least two subsequent detections, acceleration and turn rate at least three. To accomplish this, a track is created for each new object and subsequent measurements are assigned to the same track. The recursive evaluation of the measurements of one track is called filtering and allows to derive the stochastic kinematic state. Moreover, analyzing several measurements of the same object facilitates more precise estimates of the bounding box and the class, especially in presence of temporal partial occlusions. The major problem of tracking lies in the existence of multiple objects and at every time step, the problem of assigning measurements to existing or new objects must be solved.

This chapter is organized as follows: Section 3.2 gives a small overview of the extensive literature about the theme. Section 3.3 presents the approach used to represent the LIDAR data in form of a range image. In Section 3.4, a new algorithm to identify the ground pixels in the LIDAR data is given. A method to cluster the remaining pixels after removing the ground is shown in Section 3.5. Section 3.6 demonstrates a method to create a bounding box around

the pixels of an object. The assignment of the detected objects to the corresponding tracks is handled in Section 3.7. Section 3.8 forms the major part of this chapter and presents different motion models and filters and evaluates their suitability for autonomous driving. An efficient method to detect occlusions in range images and how to enhance the tracking quality based on this knowledge is demonstrated in Section 3.9. A simple, but efficient method for a rough classification of the objects is the theme of Section 3.11. Section 3.12 explains birth, death and existence probability of the objects. Section 3.13 closes this chapter with a summary and conclusion.

3.2 Related Work

A standard work about the perception and tracking of robots has been published by Thrun et.al. [166]. A classical course book about state estimation for the purpose of tracking and navigation has been published by [22]. A more specific survey over recent papers about state estimation of human drivers is given in [35].

In the last years, many publications about analyzing LIDAR data for the purpose of detecting traffic participants appeared. In [51], a set of algorithms for the segmentation of point clouds is presented and compared. In [92], a method for 3D point cloud segmentation based on a connected component algorithm is proposed. Using RANSAC for detecting shapes in point clouds is proposed in [156]. In [12], RANSAC is used for ground detection and a voxel-based representation for the obstacles. The authors of [14] propose to transform the non-ground pixels of the point cloud into a k-d tree representation for clustering. A graph based approach to represent the 3D data and use of a local convexity criterion for segmentation is proposed in [132].

Several papers propose to use a range image representation instead of point clouds. An early comparison of segmentation algorithms for range images is given in [94]. The authors of [149] and [148] compute an unevenness field from the range image to separate ground obstacle pixels. In [38], the scan line segments of the range image are analyzed and classified as ground or obstacle depending on their features. The authors of [29] and [30] also use a range image representation, applying breadth first search (BFS) to cluster ground and obstacle pixels.

Creating an orientated bounding box (OBB) around each pixel cluster is mostly achieved by the rotating calipher algorithm [167]. Based on this algorithm, [136] proposes correction methods to overcome problems caused by incomplete detection of objects by LIDAR. [106] shows, how to compute the uncertainty of the OBB estimate. [34] propose a multivariate Gaussian distribution to achieve a probabilistic estimate of the OBB.

Inspired by the success in object detection and classification, many papers propose to use neural networks for analyzing LIDAR data. In [189], the authors propose SqueezeSeg for pixel wise classification of LIDAR data, a CNN derived from SqueezeNet for image data. For subsequent clustering, a conventional algorithm as DBSCAN is proposed. In [191] SegVoxelNet is proposed, consisting of a voxel feature encoder, semantic context encoder and a depth-aware head to generate 3D bounding boxes. RangeNet++ presented in [130] converts the point cloud into a range image, which does semantic segmentation by a CNN, post processed by the results of a kNN search in the original point cloud. The method proposed in [57] projects the point cloud into birds eye view (BEV) and trains separate detectors for close range and long-range objects to get better results than other CNN based approaches. Also in [60], a BEV representation is used as input for a ResNet-8 architecture. This approach generates additionally to the bounding box estimate a softmax objectness score to quantify the uncertainty of the result. [108] extends the previously mentioned approaches by extending the analyses into the temporal dimension. It does so by aggregating point clouds over time into a 4D tensor to predict OBBs and object classes by a CNN. The authors of [111] combine LIDAR, radar and camera data to detect vehicles. They derive a region of interest from the LIDAR and radar data and feed the corresponding image patch into a CNN for classification. In [23], the authors take the other way around by first applying a CNN for semantic segmentation of the RBG and subsequently doing instance segmentation after projecting the camera pixels into the LIDAR data.

The literature for tracking moving objects is more extensive than for LIDAR processing since object tracking has been applied already for several decades, mostly based on radar to detect aircraft and missiles. The Linear Kalman Filter was already presented in 1960 [101]. Since then, many extensions, as the Extended Kalman Filter [166] [22] or the Uncented Kalman Filter [181] have been published. A extensive survey over target tracking is given in [120], [117], [118], [119] and [121]. The paper [53] presents a method to track an extended object detected by LIDAR using an EKF. The authors of [175] claim that the UKF is suitable for automotive applications with curvilinear motion models. In [44], a direct comparison of EKF and UKF for robot localization is given. The authors of [11] present the Cubature Kalman Filter, an extension of the UKF. Using a Particle Filter for vehicle tracking is presented in [143]. [41] presents an efficient implementation of a computational very demanding Multiple Hypotheses Filter. The Joint Probabilistic Data Association Filter (JPDAF) was first published already 1980 in [64].

Since 2000, many papers about a new generation of tracking filters based on Finite Set Statistics (FISST) have been published. A good introduction to FISST and Random Finite Sets (RFS) is given in [126] and [124]. The first filter based on this theory, the

Probability Hypothesis Density (PHD) Filter, was presented in [125].A Gaussian Mixture (GM) implementation for this filter was given in [178]. A filter using Labeled Random Finite Sets especially suitable for multi object tracking is given in [179]. The authors of [88], [87], [86], [90], [85], [91] and [89] demonstrate various approaches of FISST for extended object tracking, especially in the field of tracking traffic participants using LIDAR.

3.3 Range Image Representation of LIDAR Data

In this work, the 3D-Data from the LIDAR is represented as a Range Image, also called Depth Image. A Range Image is a row by column ($R \times C$) matrix, similar to a camera image. The 3D points of one LIDAR revolution are projected into this Range Image resulting in a cylindrical picture with a horizontal field of view of 360° and a vertical field of view of 26.8°. While the values of the individual pixels of a camera image are gray scale or color indices, the value of a Range Image pixel is the measured distance from the origin of the LIDAR. Optionally, the intensity of the laser return may be evaluated.

A range image can be constructed from a point cloud by simple geometrical transformations. In this work, the raw input of the data packets of the Velodyne HDL-64 are used as input. The advantage is the better computational efficiency by avoiding the intermediate step of generating a point cloud.

The major benefit of the range image representation results from preserving the spatial relationship between points in the 3D space in the 2D representation. By evaluation of the range difference of neighboring points, it can easily be estimated, whether they belong to the same object or not. A point cloud representation in contrast requires computational expensive algorithms as K-Nearest-Neighbors (KNN) [63] or Density-Based Spatial Clustering (DBSCAN) [58] for clustering.

Moreover, the range image representation is beneficial for the detection of partial or complete occlusions between objects, which would require to apply expensive ray tracing algorithms in point cloud representations.

In this work, a 64-beam LIDAR is used with a horizontal resolution of $\frac{1}{6}$ degree at a turn rate of 10 Hz. The resulting range images have therefore a format of 64×2160 pixels.

Figure 3.1 shows a typical urban street scene captured by the LIDAR. The upper picture is the representation of the point cloud by the visualization tool RVIZ. The black points belong to the ground, the red points to obstacles. The lower picture is a clipping of 64×550 pixels of the corresponding range image, resulting in a field of view (FOV) of $\approx 26° \times 90°$. Regions without returns are checkered, ground points are in black and the obstacle points are already clustered and displayed in different colors.

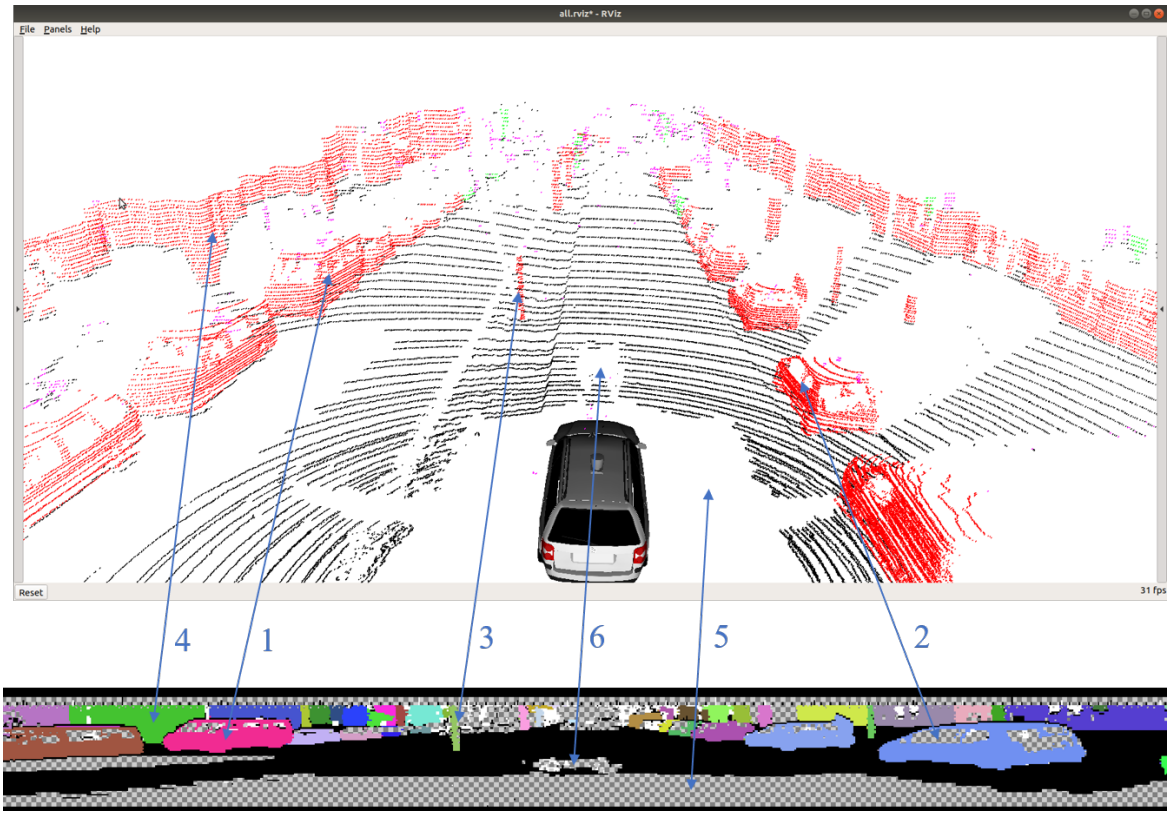


Fig. 3.1 LIDAR street scene in RVIZ (top picture) and represented as range image (bottom picture)

Marker 1 and 2 refer to parked cars. A tree on the central reservation is marked with 3, while 4 is an example of a building in the background. Region 5 has no LIDAR measurements, since the beams hit the roof of the host vehicle. The blind spot 6 results from a camera mounted on the top of the host vehicle in front of the LIDAR.

3.4 Separation of Ground Pixels

First step in analyzing a LIDAR input is identifying and separating points belonging to the ground. The purpose of this separation is to identify individual objects, which actually block the planned trajectory of a vehicle or may move into that trajectory in the near future. Since flying objects do not have to be considered (up to now), all objects of potential interest are based on the ground or are located on the top of a ground based object. Therefore, clustering of points into individual objects requires that beforehand the ground, which is the common base for all other objects, is separated.

It is important to note that curb stones, speed bumps, potholes etc. are not considered as individual objects, but as features of the ground. When clustering the points of a curb stone into an individual object, they appear as flat and narrow obstacles with a length of 20 - 40 m, depending on the resolution of the LIDAR and the height of the curbstone. When tracking such an object, its centroid seems always to be close to the centroid of the host vehicle and the curbstone is therefore considered as a strange obstacle moving permanently parallel to the host. Speed bumps and potholes in turn are hard to detect with LIDAR data only.

From the above follows that it is beneficial to handle curb stones, speed bumps, potholes etc. separately. The ground is not in all cases driveable for a car. The curb stone separates the driveable road from the sidewalk and detecting curb stones may help to locate the car in the map. Speed bumps are considered driveable, but with reduced speed. Their existence should be marked in the map. Potholes form a special challenge to autonomous vehicles since it is hard to decide, whether they are driveable or not, and if so, at which speed they may be traversed. Examination of the ground for driveability is not part of this work.

Many systems based on point cloud representations use the Random Sample Consensus algorithm (RANSAC) [62] for ground separation. The algorithm evaluates the parameters of the most probable plain in the data. Besides being computational very demanding, the algorithm has several disadvantages. At first, due to the unevenness of the ground plane mentioned above, it is necessary to consider also points, which are about 20 - 40 cm above the plain as ground. But in this way, also the lower parts of important obstacles, as cars, are removed. Second, the ground is not always a plain, but there may appear slopes in the field of view of the LIDAR. Examples are ramps before bridges, at highway entrances or in parking garages. RANSAC is not able to detect theses ramps as ground. Finally, the algorithm may completely fail, if the host is located near a wall or a large truck and this surface is accidentally taken as the ground plain.

The following new algorithm aims at a reliable identification of ground points even in the presences of major unevenness and slopes without cutting points from non-ground objects. This is important, because the vertical resolution of the HDL-64 and many other LIDARS is only $\frac{1}{3}$ degree, e.g. in 100 m distance, the vertical resolution is $\approx$ 60 cm. Taking into account, that the upper part of a target vehicle is mostly glass and therefore does not reflect laser beams, there remain often only 2 horizontal lines of points as measurement. If the ground removal algorithm classifies the lower of these two lines as ground, 50 % of the data is lost.

Figure 3.2 shows a narrow fraction of a range image in driving direction of the host vehicle. The yellow rectangle has a size of 5×2 points and represents a preceding car in 100m distance. The points below the car are invalid since the road surface does not reflect the laser beam due to the low impact angle at high distances. The number of valid points varies

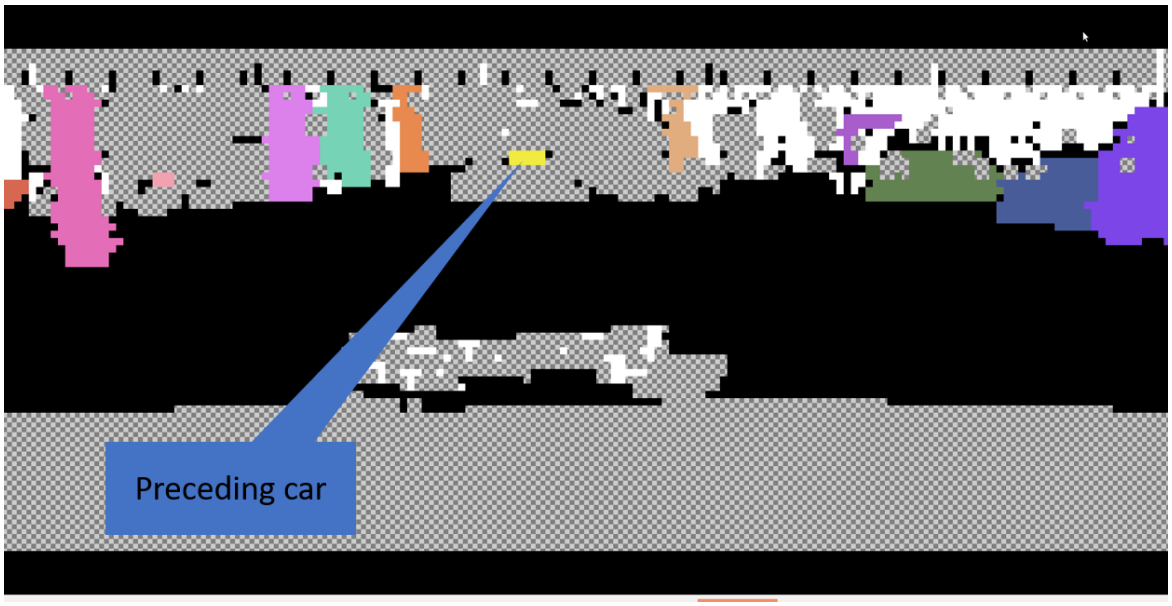


Fig. 3.2 Range image with a preceding car in 100 m distance (in yellow)

from scan to scan, even when following the car in constant distance, due to measurement noise. Therefore, it is unreliable to track objects with too few points.

The proposed Algorithm 1 is very simple and efficient. It takes as input a range image of $C \times R$ points $\mathbb{P} \in \mathbb{R}^3$ and returns a set of ground points $G \in \{\mathbb{P}\}$ and a set of obstacle points $O \in \{\mathbb{P}\}$. Invalid points have a distance of 0.0 m, the minimum distance for a valid point is $\approx 1.4m$. Parameters are the minimum height over ground h_{min} of an obstacle (default = 0.3m), the minimum slope α (default = 30°) and a maximum angle β (default = 30°) to detect the lowest point of an obstacle.

The algorithm scans the range image column by column from bottom to top and tries to classify points as ground or obstacle. In cases of ambiguity, the decision is postponed until one of the next points allows clear classification. Entries without measurement (distance = 0) are skipped. The decision for points, which classify as ground, but which are directly below its successor, is also postponed.

Figure 3.3 shows three examples. The rightmost object is a curbstone delimiting a sidewalk. It is not qualified as obstacle, since it is too low. The middle object is a ramp, also correctly classified as ground although it has a height $> h_{min}$ since the slope between the points is too shallow. The leftmost object is an obstacle with two rows of laser returns, as in Figure 3.2. The lowest point on the obstacle is actually too low and too shallow for an obstacle point, but the subsequent point directly above helps to assign it to the obstacle.

Algorithm 1 Separating LIDAR input into ground and obstacles points

Require:
$\mathbb{P} = \mathbb{R}^3, \mathbf{P} \in \mathbb{P}^{C \times R}$ ▷ Range image as matrix of points
$h_{min}, \alpha, \beta \in \mathbb{R}$ ▷ Parameters
Ensure: $\mathcal{O}, \mathcal{G} \in \{\mathbb{P}\}$ ▷ Sets of obstacle and ground points
for $i \leftarrow 1$ **to** C **do** ▷ For all columns
  $p_{gr} \leftarrow \mathbb{P}(1.0, 1.0, 0.0)$ ▷ Initial pseudo ground point
  $\mathcal{A} \leftarrow \emptyset$ ▷ Set of ambiguous points
  for $j \leftarrow 1$ **to** R **do** ▷ For all rows from bottom to top
    if $|\mathbf{P}_{i,j}| > 0$ **then** ▷ If point is valid
      if $\arcsin(\frac{\mathbf{P}_{i,j}.z - p_{gr}.z}{|\mathbf{P}_{i,j} - p_{gr}|}) > \alpha$ **then** ▷ Check slope
        if $\mathbf{P}_{i,j}.z - p_{gr}.z > h_{min}$ **then** ▷ Check minimum height
          $\mathcal{O} \leftarrow \mathcal{O} \cup \{\mathbf{P}_{i,j}\} \cup \mathcal{A}$ ▷ Add points to obstacle
          $\mathcal{A} \leftarrow \emptyset$
        else
          $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathbf{P}_{i,j}\}$ ▷ add to ambiguous points
        end if
      else
        if $|\mathbf{P}_{i,j+1}| > 0$ **and** $|arccos(\frac{\mathbf{P}_{i,j+1}.z - \mathbf{P}_{i,j}.z}{|\mathbf{P}_{i,j+1} - \mathbf{P}_{i,j}|})| < \beta$ **then** ▷ If valid and
          ▷ point is below successor
          $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathbf{P}_{i,j}\}$ ▷ Add potential lowest point to ambiguous
        else
          $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathbf{P}_{i,j}\} \cup \mathcal{A}$ ▷ Add points to ground
          $\mathcal{A} \leftarrow \emptyset$
          $p_{gr} \leftarrow \mathbf{P}_{i,j}$ ▷ Assign new anchor point
        end if
      end if
    end if
  end for
end for

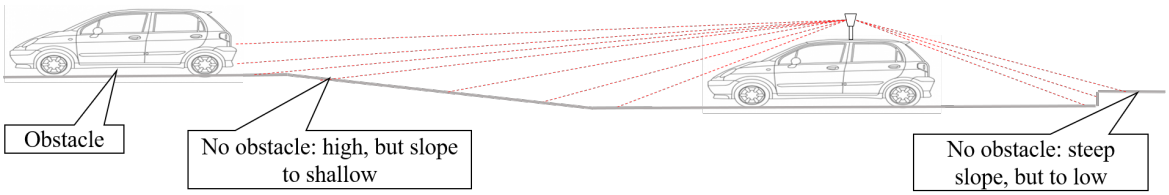


Fig. 3.3 Examples for obstacle and ground detection

3.4.1 Evaluation Ground Removal

The evaluation of the Range Image Ground Detection (RIGD) algorithm is done by comparing it against the RANSAC algorithm (see above). Both algorithms are implemented as part of the fub_roscar perception module and are activated simultaneously for evaluation purposes. Evaluation is done for 4 typical urban traffic scenarios. The data was recorded in ROS bag files during test drives with the FU autonomous vehicle MadeInGermany in Berlin Dahlem and Berlin Reinickendorf.

The four scenarios are:

- Turn and Lane Merge on Thielallee (TU)
- Lane Change on Scharnweberstraße (LC)
- Intersection Crossing Ehrenbergstraße (CR)
- Pedestrian Cross Walk Scharnweberstarße (PED)

These scenarios are also used for evaluations in further parts of this work (see Section 5.10 for more details). The following evaluation is split into two parts: a qualitative and a quantitative evaluation.

Qualitative Evaluation

Figure 3.4 shows one example picture of the ground detection methods. The pixels of the point cloud are colored according to the result of the two algorithms. Pixels, which are classified as ground by both algorithms are colored in red, while the purple pixels are clearly obstacle. The interesting part are the pixels, where both algorithms differ. Yellow pixels are classified as obstacle by RIGD and as ground by RANSAC. Typical examples are marked as "1" in the example figure. RANSAC cuts the lower pixels of the car and the wall because they are below the ground plain calculated by RANSAC. The majority of the yellow pixels is correctly classified by RIGD. There are some rare cases, when the LIDAR hits the ground below some obstacle, which RIGB wrongly classifies as obstacle (see pixels below the car).

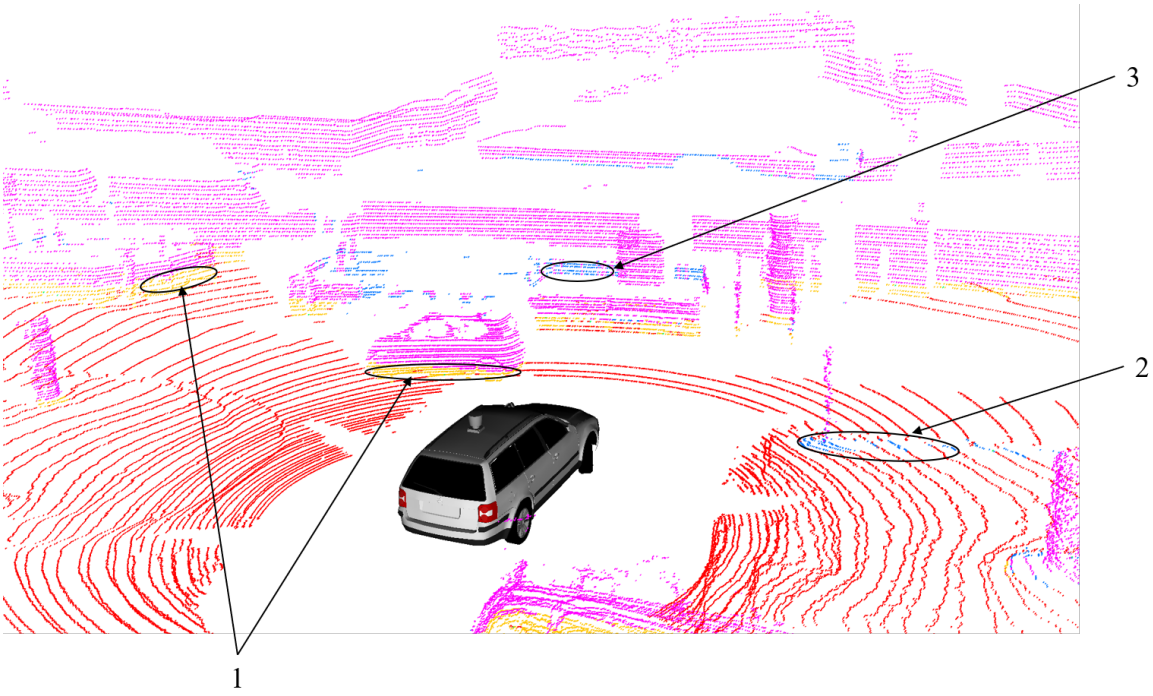


Fig. 3.4 Qualitative evaluation of ground removal by RANSAC and RIGD method

Pixels, which RIGD classifies as ground, but RANSAC as obstacles, are colored in blue. The region on the central reservation marked with "2" shows an example, where RANSAC wrongly classifies ground pixels as obstacles since they are above the RANSAC plain. Region "3" demonstrates a weakness of the RIGB algorithms: when the LIDAR looks over one obstacle, RIGD has difficulties to distinguish the pixels behind the first obstacle between ground and another obstacle.

Quantitative Evaluation

Providing complete ground truth for a quantitative evaluation of LIDAR pixel classification is nearly impossible. LIDAR images, as used in this work, consists of more than 100,000 pixels and manually labeling even one image completely would take days. Therefore, the following approach was taken in this work:

- From each of the four scenarios mentioned above, which are considered representative for urban traffic, five images with $\approx 10s$ temporal distance are selected, resulting in 20 different images for the evaluation.
- From each image, 0.5 % of the pixels are selected randomly, resulting in $\approx$ 10,800 pixels.

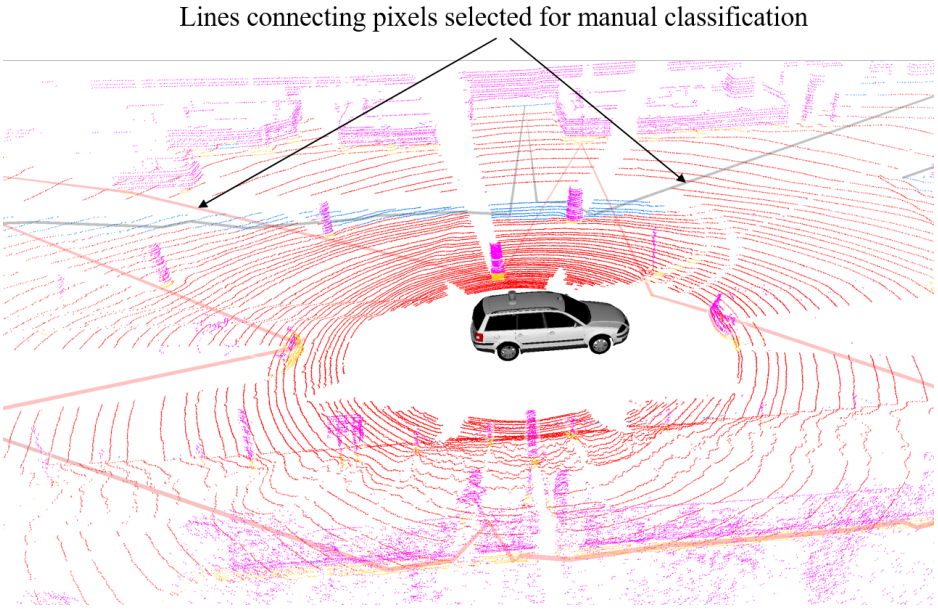


Fig. 3.5 Quantitative evaluation of ground removal by manually inspecting sample pixels (image detail)

- The pixels, for which both algorithms have the same result (ground or obstacle) are considered as correctly classified. There may be cases, in which both algorithms are wrong, but given the fact that they work completely different, these cases should be very rare and are moreover not relevant for the task of comparing the two methods.
- The remaining $\approx$ 840 pixels (42 per image on average) with different classification are marked and linked together visually (see Figure 3.5). These pixels are subsequently inspected manually using the ROS tool RVIZ to classify them as ground, obstacle or ambiguous. The latter ones are pixels, for which no clear classification could be given.

Table 3.1 shows the result of applying the above evaluation method. A summary of the results is given in Figure 3.6. It clearly shows the weakness of the RANSAC algorithm in erroneously qualifying many obstacle pixels (5.3%) as ground, which results in reduced quality of the obstacle detection and may even discard small obstacles completely. This problem could by mediated by lowering the ground threshold of the RANSAC plane. But this would result in qualifying small unevenness in the ground as obstacle with severe disadvantages for the subsequent tracking and prediction modules. 0.8% of the pixels are qualified as ground instead of obstacle by the RIGD algorithm. Since the majority of these pixels belong to objects, which are located behind other correctly labeled obstacles (see above), the disadvantages for tracking and prediction are considered acceptable. Moreover, the RANSAC algorithm takes on average 44.8*ms* per image, which is not acceptable for

Scenario	Time [s]	RIGD				RANSAC			
		Obstacle		Ground		Obstacle		Ground	
		Correct	Error	Correct	Error	Correct	Error	Correct	Error
PED	2.4	252	2	234	3	234	0	236	21
PED	12.9	207	1	333	3	191	5	329	19
PED	23.8	188	0	356	5	178	1	355	15
PED	34.5	248	4	283	5	218	4	283	35
PED	45.1	255	4	156	6	231	2	158	30
LC	2.5	234	4	312	2	205	1	315	31
LC	13.0	263	3	286	2	234	0	289	31
LC	23.7	234	0	198	4	209	1	197	29
LC	34.5	312	6	211	4	269	0	217	47
LC	45.3	317	2	194	4	294	1	195	27
TU	2.3	235	1	345	4	204	15	331	35
TU	12.9	208	2	358	2	188	6	354	22
TU	23.7	282	1	281	4	246	4	278	40
TU	34.5	214	3	343	7	193	3	343	28
TU	45.1	177	0	393	4	142	48	345	39
CR	2.3	308	1	217	11	292	1	217	27
CR	12.9	235	0	323	3	211	6	317	27
CR	23.7	226	1	337	7	216	3	335	17
CR	34.4	188	1	390	6	172	0	391	22
CR	45.0	325	2	225	4	295	0	227	34
Sum		4908	38	5773	90	4422	101	5710	576

Table 3.1 Evaluation of Ground Detection Algorithms.

real-time operation of the whole perception and tracking module in 10 Hz. The RGIB algorithm takes on average 9.4*ms*.

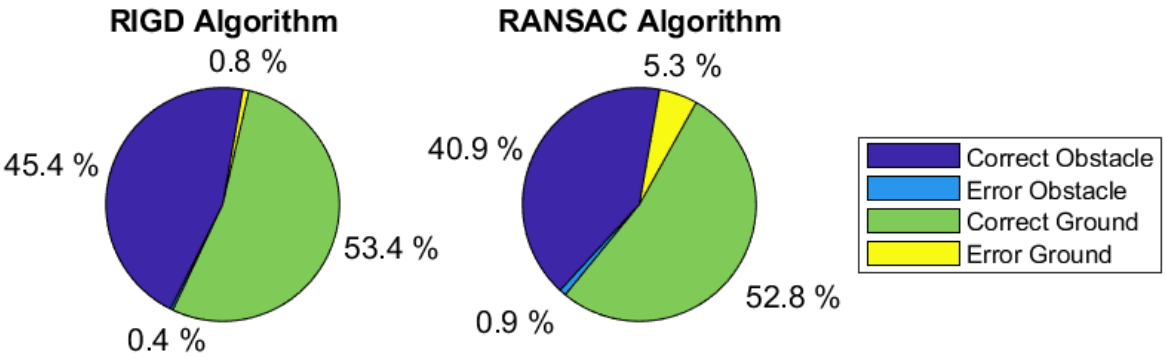


Fig. 3.6 Comparison of the results of the quantitative evaluation.

3.5 Clustering of Range Image into Objects

After having separated the ground pixels, all distinguishable objects should be isolated from each other. In the easiest case, a simple region growing algorithm in the range image could cluster all pixels into distinct objects. Unfortunately, there are difficulties:

- Overlapping objects: the objects in the range image may overlap and the nearer object may partially occlude the farther one. Therefore, two neighboring pixels in the range image may belong to different objects. See Figure 3.1 for examples.
- No returns: Some laser beams do not return a measurement. This may be caused by a very acute impact angle or by surfaces like glass (also seen in Figure 3.1). In some cases, there remains no pixel connection between the chassis and the roof of a car and a simple region growing algorithm would detect two separate objects.
- Image cut: The LIDAR produces a 360° range image, but there must be some cut to project it into a R $\times$ C matrix. Assuming a new image starts, when the laser points forward and then turns clockwise, pixels of an object direct in front of the host vehicle are split into the leftmost and rightmost columns of the image. These pixels must be clustered together.

In [30], an efficient clustering algorithm has been published, but it does not take care of the No Returns and the Image Cut.

Algorithm 2 clusters the range image into individual objects. It uses an extended point structure compared to the Algorithm 1. The main procedure in line 1-10 searches sequentially for unassigned pixels as seed points for the next object and invokes the recursive procedure ClusterObject. Its main loop cycles through all neighbor pixels. Function *neighbors* (not shown) collects all pixels of the 8-neighborhood of the current pixel, taking into consideration

Algorithm 2 Cluster range image into object

Require: $\mathbb{P} = (p : \mathbb{R}^3,\ l, v : \mathbb{N}), P \in \mathbb{P}^{C \times R}$ ▷ Point as tupel of vector and two labels

Label $\leftarrow 1$ ▷ Intialize Label
for $i = 1$ **to** C **do** ▷ For all columns
  for $j = 1$ **to** R **do** ▷ For all rows
    if $|P_{i,j}.p| > 0$ **and** $P_{i,j}.l = 0$ **then** ▷ Point is valid and unlabeled
      $P_{i,j}.l \leftarrow Label$
      $Label \leftarrow Label + 1$
      CLUSTEROBJECT((P, i, j, i, j)) ▷ Start clustering with current point
    end if
  end for
end for

procedure CLUSTEROBJECT($P, col, row, lastCol, lastRow$) ▷ Range image,
▷ current column and row, last valid column and row
  $P_{col,row}.v \leftarrow P_{lastCol,lastRow}.l$ ▷ Mark point as visited during this search
  for $(i, j) \in neighbors(col, row)$ **do** ▷ Check all neighbor points
    if $P_{i,j}.v <> P_{col,row}.v$ **then** ▷ If unvisited during current search
      if $|P_{i,j}.p| = 0$ **then** ▷ If invalid
        if $|lastCol - i| + 2|lastRow - j| < MaxManhatten$ **then** ▷ Check
        ▷ for Manhatten distance
          CLUSTEROBJECT($i, j, lastCol, lastRow, P$) ▷ Continue clustering
        end if
      else
        $X \Leftarrow \{P_{i,j}.p, P_{lastCol,lastRow}.p\}$ ▷ Current and last valid point
        $p_1 \leftarrow \arg\max_{x \in X}(|x|)$ ▷ Farther point
        $p_2 \leftarrow p_1 - \arg\min_{x \in X}(|x|)$ ▷ Vector from farther to closer point
        if $arccos(\frac{p_1 \cdot p_2}{|p_1||p_2|}) > MinAngle$ **then** ▷ Check inclination
          $P_{i,j}.l \leftarrow P_{lastCol,lastRow}.l$ ▷ Add point to cluster
          CLUSTEROBJECT(i, j, i, j, P) ▷ Continue clustering
        end if
      end if
    end if
  end for
end procedure

the cut boundary between the first and last column of the image. A valid and unvisited neighbor pixel belongs to the same object when the inclination angle ϕ to the previous point is above a certain threshold *MinAngle* (see Figure 3.7). The search continues then from this pixel (lines 22-26). The search is also continued through regions of invalid pixels, if the Manhattan distance to last assigned pixel is $<$ *MaxManhattan* (lines 18-19).

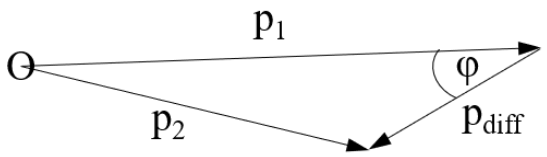


Fig. 3.7 Clustering based on slope. Points p_1 and p_2 belong to the same object, if the angle ϕ is above a threshold

3.6 Oriented Bounding Box Estimation

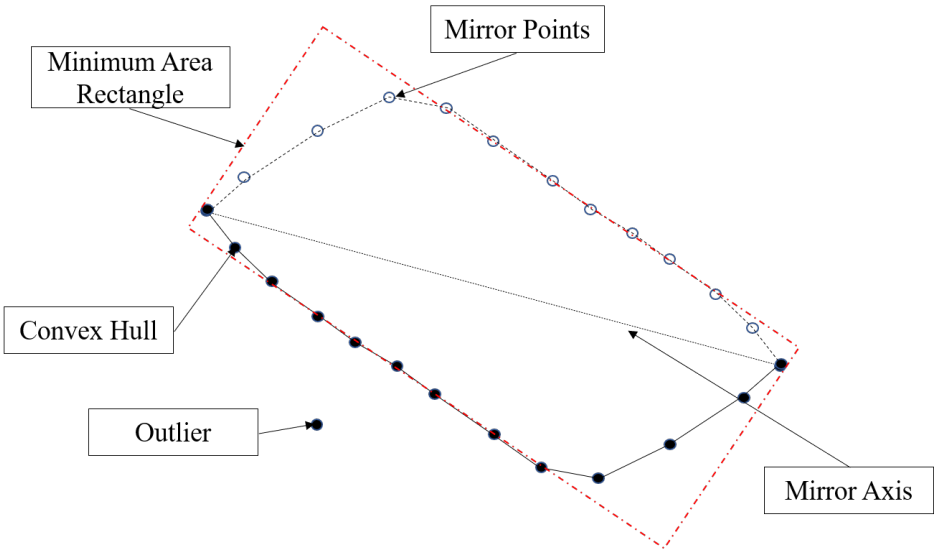


Fig. 3.8 Bounding box computation for the convex hull of the laser returns and its mirror points

After having clustered the range image into objects, the oriented bounding box (OBB) of all objects is computed. For this purpose, all points are projected into the x-y plane and the convex hull of the points is calculated (see Figure 3.8). Outliers are removed during this stage. Since the laser beams always hit only one side of the object, the contour on the other object side of the assumed symmetry axis has to be completed to get a rectangle. This is done

by drawing an axis between the left most and right most point of the convex hull and then mirroring all points across this axis. Afterwards, the rotating calipher algorithm [167] is used to compute the minimum area rectangle enclosing all points. The height of the bounding box results from the difference between the highest and lowest point before projecting them into the ground plane. The centroid and the orientation of the bounding box are taken furthermore as pseudo measurement of the pose of the object.

Some authors, as [158], propose to use the orientation of the OBB as measurement input of the heading for the filter. This is problematic for several reasons:

- It works only for vehicles, which have usually a clearly identifiable orientation axis in direction of the movement. This is not true for pedestrians, for which the orientation of the OBB is often unrelated to the heading.
- For cars or trucks driving directly in front of the host, there are normally only laser returns of the vehicles rear side. The measurement of their length is truncated, and they appear therefore wider than long. The orientation of the OBB is then rotated 90° to the heading (see Figure 3.9.)
- Even if the OBB is measured correctly, its orientation deviates from the heading, when turning with a high steering angles due to the kinematic properties of the Ackermann drive.

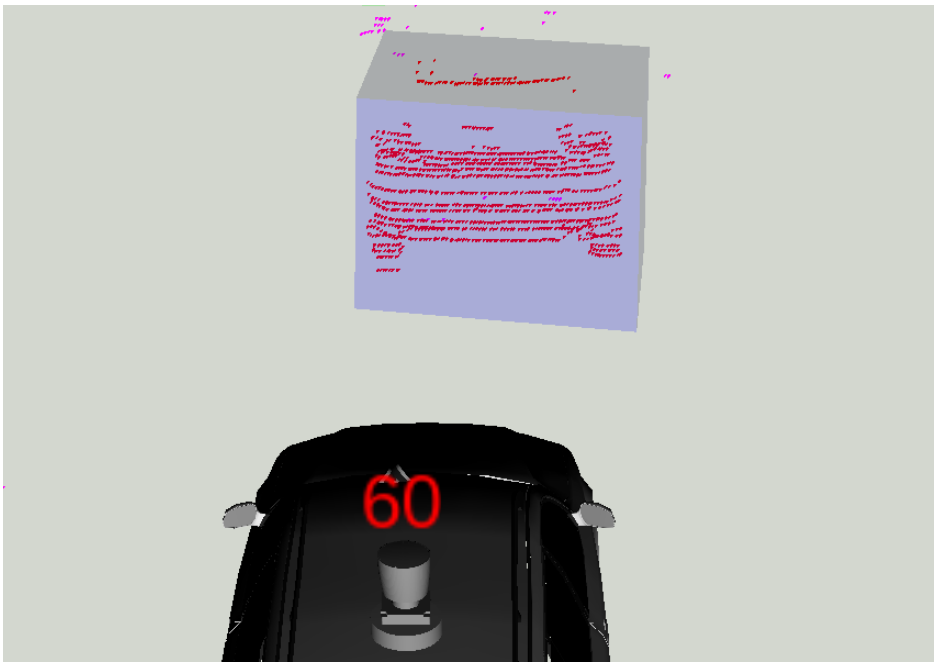


Fig. 3.9 OBB of the car driving in front of the host. It seems to be wider than long.

3.7 Measurement Assignment

The result of the clustering and the OBB estimation is the measurement vector $\mathbf{z} = [x, y, z, \phi, l, w, h]^T$. This vector has to be transformed from the moving host vehicle coordinate system to the inertial coordinate system of the roadmap (see Subsection 2.4.3). The next step in the perception pipeline is to qualify the measurement and eventually assign it to a tracked object. In this stage, several problems must be solved:

- Object assignment: If there are several measurements and several objects, there must be some matching algorithm.
- Clutter: Some measurements may be clutter, e.g. are invalid or result from irrelevant objects.
- Missed detection: Some tracked objects may not be observed in every time step, for example due to temporary occlusion.
- Object birth: New objects may appear in the field of view of the LIDAR and a new track has to be initialized.
- Object death: Objects may disappear permanently from the field of view and their track should be deleted.

- Object split: An existing object may be split up into several objects during life time.

Much research has been carried out over the past few decades to solve the above problems. Many approaches combine the assignment task with the filtering. The majority of the research has been done however in the field of aircraft and missile tracking, where objects are typically tracked by radar. These approaches assume that there is at most one measurement per object, that the objects navigate in considerable distance to each other and that there are a lot of clutter measurements (especially when trying to detect stealth aircraft). The problems of tracking traffic participants with LIDAR, especially in urban environments, are quite different. The LIDAR produces many individual pixels for each object and errors may occur during the clustering stage, resulting in missed detections or multiple pseudo measurements per object. The objects may be located very close to each other, especially if they are stopped. On the other hand, the clutter problem is usually solved by using a threshold for the minimum number of pixels per object. The split object problem is relevant only in the military field to detect missiles with multiple warheads, even if some similar phenomena may appear in ground traffic, for example when observing passengers leaving a bus and walking away independently.

There are several methods, which try to solve some or all of the above problems simultaneously. Some of the methods are deeply interwoven with the filtering (see Section 3.8). All methods make use of the predicted object state, which is also handled in Section 3.8.

- Local Nearest Neighbor (LNN) [21]: Each track is updated with the measurement having the lowest Mahalanobis distance to its predicted measurement. Consequently, the same measurement may be assigned to several tracks. Clutter must be filtered in advance. There must be some upper limit for the Mahalanobis distance to avoid assignment of very far away measurements. Unassigned measurements may be used for object births. Object deletion is handled separately. The method is only suitable for scenarios with well separated objects and low clutter rate.

- Global Nearest Neighbor (GNN) [36]: Each measurement is assigned to maximal one track. Each possible assignment is associated with some cost value, usually the Mahalanobis distance between measurement and prediction. Some optimization algorithm is then used to calculate the assignment of measurements to tracks with the minimal cost sum. Clutter must be removed in advance, unassigned measurements result in new tracks. Object deletion is handled separately. The method is suitable for environments with low clutter rate and good measurement and prediction quality. The method yields good results for objects close side by side.

- Probabilistic Data Association (PDA) [20]: All measurements within the validation gate of the predicted object state are used for updating the track. The validation gate is a limited sub region of the measurement space around the predicted measurement. It is assumed that at most one measurement results from a real object, all others are clutter. As with the LNN, the same measurement may update several tracks. The track update in the filter is done probabilistically, using the measurement likelihoods and the missed detection probability to calculate the weights for the update. Object birth and death is handled separately. The method is well suited for environments with high clutter rate and clearly separated objects, like on highways.

- Joint Probabilistic Data Association (JPDA) [81]: This method is a combination of PDA and GNN. It uses a soft assignment, like the PDA, but also considers all possible combinations of measurement assignments to tracks, when calculating the update of the weights. By this means, it is guaranteed that the weights of each measurement sum up to unity. This method also handles the object creation probabilistically. The disadvantage of this method are the high computational costs.

- Multi Hypothesis Tracking (MHT) [41]: The MHT approach creates a track hypothesis for every possible sequence of measurement assignment. Since this becomes intractable after a few measurements, approximation methods, like track pruning, are used in practice. It is suitable for aircraft tracking, where the update rate from radar is comparably slow and the number of tracked objects within a certain region is also low.

- Labeled Multi-Bernoulli (LMB) [150]: This approach is based on the Random Finite Set (RFS) theory. It handles measurement assignment, clutter, missed detection, birth and death of tracks simultaneously in a strictly probabilistic manor. Unfortunately, there are up to date no real-time capable implementations for autonomous driving in realistic urban scenarios.

- Gamma Gaussian Inverse Wishart LMB (GGIW-LMB) [25]: Extension of the LMB approach for extended targets, which generate multiple measurements (like LIDARS). In this case, even the clustering stage (see Section 3.5) is integrated into the data association. Example implementations of this approach are able to handle a handful of objects with a few measurements, but with unacceptable running times.

In this work, the GNN approach is utilized for several reasons:

- In urban environments, objects are not well separated from each other, especially when stopped. Therefore, the LNN and PDA approach are not suitable.

- Clutter can be filtered in advance by setting a threshold for the number of pixels forming a measurement. The predictions of moving objects are quite accurate due to the high frequency (10 Hz) of the LIDAR. The measurement noise of the LIDAR is low compared to radar. Therefore, the computational effort for the JPDA, MHT and LMB approaches is not justified.
- The major problem of LIDAR tracking is the uncertainty in clustering. This could be solved using a GGIW-LMB or a similar RFS based extended object approach, but the computational effort is currently nearly intractable for realistic setups with dozens of objects and more than 10000 non-ground pixels.
- To compute the cost optimal assignment of measurements to tracks for the GNN approach, the very efficient Linear Assignment Problem Jonker-Volgenant (LAPJV) algorithm [99] is utilized and has been tested successfully in scenarios with more than 100 objects.

3.8 Filtering for Multi Object Tracking

In robotics, filtering is the task of inferring the state of a dynamic object from noisy measurements. The state of the object is assumed to change over time and is not directly observable. The temporal process is modeled as special form of temporal Bayesian network, a Hidden Markov Model (HMM), see Figure 3.10:

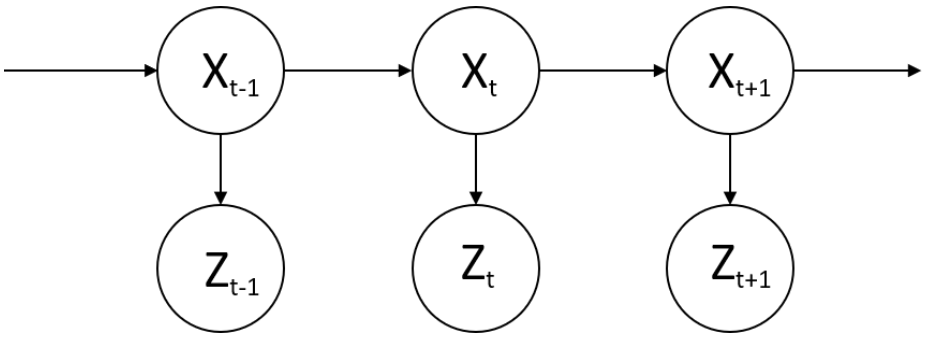


Fig. 3.10 Hidden Markov Model. The temporal sequence of the hidden states **X** are estimated using the observations **Z**.

The state evolves over time and generates measurements z_k. The goal is to infer recursively the state from the measurements under consideration of the process and measurement noise. The formula for the Bayesian filter is given by equation:

$$p(\mathbf{x_k}|\mathbf{z_{1:k}}) = \frac{p(\mathbf{z_k}|\mathbf{x_k})}{p(\mathbf{z_k}|\mathbf{z_{1:k-1}})} \int_{\mathbf{x_{k-1}}} p(\mathbf{x_k}|\mathbf{x_{k-1}}) p(\mathbf{x_{k-1}}|\mathbf{z_{1:k-1}}) d\mathbf{x_{k-1}} \tag{3.1}$$

The right part of the Equation 3.1 is the transition model, also called Chapmann-Kolmogorov equation [103]. It implements the prediction of the state to the next time step. In this way, the prior knowledge about the physical properties of the system can be incorporated into the estimation process. For moving traffic participants, the assumption may be that their speed and heading stays almost constant during the near future (constant velocity model). Other assumptions are possible. Part of the motion model is the specification of the transition probability $p(x_k|x_{k-1})$.

The result of the prediction is the prior for the left part of the equation, the measurement update. It is an application of the Bayes formula and its purpose is to combine the information in the measurement with the prediction (see Subsection 2.4.5). The result is the probability distribution of the next state.

For a derivation of the Bayes filter and its prerequisites, like the Markov assumption, see [166].

In its general form, the Bayes filter is intractable, but there are various approximations. The most prominent ones are:

- Linear Kalman Filter (LKF)
- Extended Kalman Filter (EKF)
- Uncented Kalman Filter (UKF)
- Particle Filter
- RFS based Filters

The Kalman filters are exact or approximated filters for discrete time Gaussian systems with closed form solutions. Kalman Filters will be examined more deeply below. Particle filters in contrast can model any type of distribution and also highly non-linear systems, but they suffer from high calculation times due to their sampling based implementation. Both, Kalman and Particle Filters, are used to track the state of individual targets. Measurement assignment, clutter removal and track initialization/deletion are not handled by the filters.

RFS based filters do not consider individual targets separately but try to infer a set-valued random variable, containing the states and existence probabilities of all targets. All measurements of one time step are part of another set-valued variable. The benefit of the approach lies in a joint Bayesian inference not only for the kinematic state, but also for

measurement assignment, clutter, missed detection as well as birth and death of targets. But this comes with a high computational complexity and is not suitable for current and near future hardware equipment of autonomous vehicles. RFS based filters are therefore not further considered in this work.

Other Bayesian filters like Information Filter, Cubature Kalman Filter, Histogram Filter, Rao-Blackwelized Filter, etc. are also not considered by this work.

Kalman Filters consists, like any Bayesian Filter, of two parts: Prediction and update of a stochastic state. The state is represented by a Gaussian distribution, which is given by its mean and variance. For the tracking of traffic participants, the state consists of several values for position, velocity, etc. The distribution is therefore multivariate with a mean vector and a covariance matrix.

The difference between the Kalman filters lies in the type of the motion and measurement models. The motion model handles the prediction of the state to the next time step. This prediction is assumed to be disturbed by zero mean, white Gaussian noise. The measurement model handles the prediction of the next measurement values based on the predicted state from the motion model [166]. The measurements are also assumed to be disturbed by zero mean, white Gaussian noise.

The LKF assumes that both models are linear functions. The EKF can handle non-linear motion and measurement models by linearization of the motion and measurement function using a first or higher order Taylor expansion [166]. The UKF enhances the EKF for non-linear models by usage of the uncented transform [181].

Strictly speaking, the terms LKF, EKF and UKF are misleading since they indicate, that motion and measurement model of a filter must be handled in the same way. This is by no means necessary since the functions of both steps are independent of each other. It is perfectly fine to combine a linear motion prediction with an uncented measurement prediction or vice versa. Therefore, a designation like e.g. ETUM-KF for extend transition, unscented measurement Kalman filter would be more suitable.

The rest of this section is divided into the following subsection: State prediction based of motion models is described in Subsection 3.8.1. Subsection 3.8.2 handles the measurement models and the filter update. The Interaction Multiple Model (IMM) Filter is presented in Subsection 3.8.3. The final Subsection 3.8.4 contains the evaluation of the different filters.

3.8.1 Motion Models for State Prediction

The discrete time prediction of a stochastic state space system with unknown control input is specified by:

$$\mathbf{x}_\mathbf{k}^- = \mathbf{f}(\mathbf{x}_{\mathbf{k}-\mathbf{1}}) + \mathbf{w}_\mathbf{k} \tag{3.2}$$

with $\mathbf{x}_{\mathbf{k}-\mathbf{1}} \in \mathbb{R}^n$ as n-dimensional Gaussian variable for the previous state, $\mathbf{x}_\mathbf{k}^- \in \mathbb{R}^n$ the predicted state, $\mathbf{w}_\mathbf{k} \in \mathbb{R}^n$ as n-dimensional, zero mean, white Gaussian noise and $\mathbf{f}(\cdot)$ being the system function.

If the system function $\mathbf{f}(\cdot)$ is linear in all members of the state variable and the system is time-invariant, the mean $\hat{\mathbf{x}}_\mathbf{k}^-$ and the covariance $\mathbf{P}_\mathbf{k}^- \in \mathbb{R}^{n\times n}$ of the predicted state can be calculated from previous values as (see Subsection 2.4.3):

$$\hat{\mathbf{x}}_\mathbf{k}^- = \mathbf{F}\hat{\mathbf{x}}_{\mathbf{k}-\mathbf{1}} \tag{3.3}$$

$$\mathbf{P}_\mathbf{k}^- = \mathbf{F}\mathbf{P}_{\mathbf{k}-\mathbf{1}}\mathbf{F}^T + \mathbf{\Gamma}\mathbf{Q}\mathbf{\Gamma}^T \tag{3.4}$$

with the system matrix $\mathbf{F} \in \mathbb{R}^{n\times n}$, the process noise covariance matrix $\mathbf{Q} \in \mathbb{R}^{q\times q}$ and the noise gain matrix $\mathbf{\Gamma} \in \mathbb{R}^{n\times q}$.

The EKF equations are similar to the above with the difference that the system function is not any more linear and that the covariances of the state and the process noise are propagated by time dependent Taylor expansions of the system function [184]:

$$\hat{\mathbf{x}}_\mathbf{k}^- = \mathbf{f}(\hat{\mathbf{x}}_{\mathbf{k}-\mathbf{1}}) \tag{3.5}$$

$$\mathbf{P}_\mathbf{k}^- = \mathbf{F}_\mathbf{k}\mathbf{P}_{\mathbf{k}-\mathbf{1}}\mathbf{F}_\mathbf{k}{}^T + \mathbf{\Gamma}_\mathbf{k}\mathbf{Q}\mathbf{\Gamma}_\mathbf{k}^T \tag{3.6}$$

The system matrix $\mathbf{F}_\mathbf{k}$ is the Jacobian of partial derivatives of the system function $\mathbf{f}(\cdot)$ with respect to $\mathbf{x}$, e.g. to all variables in the state.

$$\mathbf{F}_\mathbf{k} = \left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right]_{\mathbf{x}=\mathbf{x}_{\mathbf{k}-\mathbf{1}}} \tag{3.7}$$

The noise gain matrix $\mathbf{\Gamma}_\mathbf{k}$ is the Jacobian of partial derivatives of the system function $\mathbf{f}(\cdot)$ with respect to $\mathbf{w}$, e.g. to all variables in the state, which are disturbed by noise:

$$\mathbf{F}_\mathbf{k} = \left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{w}}\right]_{\mathbf{w}=\mathbf{w}_{\mathbf{k}-\mathbf{1}}} \tag{3.8}$$

The above formulation of the noise gain assumes a formulation of the noise in terms of the state variables. Otherwise, the state must be augmented (see below).

The UKF tries to get better results than the EKF by avoiding the linearization. Instead, it calculates an array of sigma points in the vicinity of the current state (two points for each dimension of the state plus one point for the state itself) [181]:

$$\hat{\mathcal{X}}_{\mathbf{k-1}} = \left[\hat{\mathbf{x}}_{\mathbf{k-1}} \;\; \hat{\mathbf{x}}_{\mathbf{k-1}} \pm \sqrt{(n+\lambda)\mathbf{P}_{\mathbf{k-1}}}\right] \tag{3.9}$$

For an explanation of the parameter λ see [181].

These sigma points are propagated through the system function, resulting in a list of predicted sigma points.

$$\hat{\mathcal{X}}_{\mathbf{k}}^{-} = \mathbf{f}(\hat{\mathcal{X}}_{\mathbf{k-1}}) \tag{3.10}$$

The predicted mean and covariance are calculated as weighted averages from the sigma points:

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2n} w_i^{(m)} \hat{\mathcal{X}}_{\mathbf{i,k}}^- \tag{3.11}$$

$$\mathbf{P}_{\mathbf{k}}^- = \sum_{i=0}^{2n} w_i^{(c)} (\hat{\mathcal{X}}_{\mathbf{i,k}}^- - \hat{\mathbf{x}}_k^-)(\hat{\mathcal{X}}_{\mathbf{i,k}}^- - \hat{\mathbf{x}}_k^-)^T + \mathbf{Q}_{\mathbf{k}} \tag{3.12}$$

For the calculation of the weights $w_i^{(m)}$ and $w_i^{(c)}$ see [181]. In [181] is also proposed to use an augmented state $\hat{\mathbf{x}}_{\mathbf{k}}^{\mathbf{a}}$ including the process noise variables and propagate them together with the state variables through the uncented transformation. This is an equivalent approach to the EKF noise handling in Equation 3.6. In this case, the addition of the noise covariance $\mathbf{Q}_{\mathbf{k}}$ in Equation 3.12 is omitted. But in practice, most implementations select the easier and more efficient way to add the noise directly as in Equation 3.12.

The black art of Kalman filter design is the specification of the process noise. While the measurement noise (see below) is defined by the corresponding devices and is often given by the manufacturer of the equipment, the process noise is usually tuned intuitively. Even the objective of the tuning is not clearly defined: achieving an optimal Mean Squared Error (MSE) by usage of a low process noise is not really helpful, if tracks are sometimes lost due to failed measurement assignments (see Section 3.7). Moreover, since the process noise models the unknown influence of the controller, e.g. the driver, it is not at all white, but heavily auto-correlated.

In literature, there are different ways to model the influence of the process noise to the state. All have in common that the noise disturbs the highest order variable(s) of the state space . As example, a simple one-dimensional second order system is assumed, i.e. position

and velocity $\mathbf{x} = [x, v]^T$. The noise enters the system by disturbing the velocity. There are four approaches for the process noise of the above second order system:

a) Continuous time process noise with noise intensity $q \left[\frac{m^2}{s^3}\right]$. Discretizing the noise for time period Δt yields [22, p. 270]:

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} & \Delta t \end{bmatrix} q \tag{3.13}$$

b) Direct discrete time model with piece-wise constant noise. The noise is specified as acceleration $\sigma_a \left[\frac{m}{s^2}\right]$ (the next higher order after velocity) and noise gain $\mathbf{\Gamma} = \left[\frac{1}{2}\Delta t^2 \; \Delta t\right]^T$, resulting in [22, p. 273]:

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^2}{2} & \Delta t^2 \end{bmatrix} \sigma_a^2 \tag{3.14}$$

c) Direct discrete time model, where the noise increments the highest order variable (velocity) at the beginning of the time period. The noise is specified as velocity increment $\sigma_{\Delta v} \left[\frac{m}{s}\right]$ and noise gain $\mathbf{\Gamma} = [\Delta t \; 1]^T$, resulting in [22, p. 274]:

$$\mathbf{Q} = \begin{bmatrix} \Delta t^2 & \Delta t \\ \Delta t & 1 \end{bmatrix} \sigma_{\Delta v}^2 \tag{3.15}$$

d) Direct discrete time model, where the noise increments the highest order variable (velocity) at the end of the time period. The noise is specified as velocity increment $\sigma_{\Delta v} \left[\frac{m}{s}\right]$ and noise gain $\mathbf{\Gamma} = [0 \; 1]^T$, resulting in [109, p. 236]:

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \sigma_{\Delta v}^2 \tag{3.16}$$

Approaches a) and b) are more accurate for long intervals Δt, but since the interval in traffic tracking is usually quite short ($0.1s$ for LIDAR, $\approx 0.025s$ for cameras), there is no big difference to approaches c) and d), when selecting the noise magnitude appropriately. The advantage of the latter methods is that they can be applied in the same way for multi-dimensional, non-linear systems. This is important for the task of comparing different motion models and filter configurations concerning their suitability for a specific task, like autonomous driving. Otherwise, the results are influenced by the intuitively selected noise model.

When specifying the noise completely in terms of the highest order state variables and formatting it as diagonal matrix of the same dimension as the state, the system matrix F can be taken as noise gain matrix $\mathbf{\Gamma}$ in case c) and Equation 3.6 becomes:

$$\mathbf{P_k^-} = \mathbf{F}\mathbf{P_{k-1}}\mathbf{F}^T + \mathbf{F}\mathbf{Q}\mathbf{F}^T = \mathbf{F}\left(\mathbf{P_{k-1}} + \mathbf{Q}\right)\mathbf{F}^T \tag{3.17}$$

and in case d), $\mathbf{\Gamma}$ is the identity matrix $\mathbb{1}$ and Equation 3.6 becomes:

$$\mathbf{P_k^-} = \mathbf{F}\mathbf{P_{k-1}}\mathbf{F}^T + \mathbb{1}\mathbf{Q}\mathbb{1}^T = \mathbf{F}\mathbf{P_{k-1}}\mathbf{F}^T + \mathbf{Q} \tag{3.18}$$

In other words: the diagonal noise matrix is added to the state covariance either before or after the time prediction. This works for all motion models and all Kalman filters and is therefore appropriate for comparing them.

For tracking moving objects in a x-y plane, there are two major classes of stochastic motion models [22] [166] [158]]:

- Free Motion Model: The motion of the object in x and y direction is independent and therefore uncorrelated. These models are linear and can be handled by a LKF.
- Curvilinear Motion Model: The motion of the object underlies some non-holonomic constraints and the expected motion in x and y direction is therefore correlated. These models are non-linear and require an EKF or UKF.

In the rest of this subsection, the following motion models are described:

- Constant Position (CP) Model
- Constant Velocity (CV) Model
- Constant Acceleration (CA) Model
- Constant Heading and Velocity (CHV) Model
- Constant Turn Rate and Velocity (CTRV) Model
- Constant Heading and Acceleration (CHA) Model
- Constant Turn Rate and Acceleration (CTRA) Model

Some of the motion models described in the following are never used in isolation for the tracking of traffic participants, but are useful as fallbacks for other models or in combination with other models in an Interacting Multiple Model (IMM) filter.

Constant Position Model

The Constant Position Model is the most basic Free Motion Model and assumes that the object remains at its position. Any change in position is caused by noise. The state consists of the Cartesian coordinates of the object in x and y direction:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.19}$$

The system function is linear and is given by the following matrix:

$$\mathbf{F_{CP}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.20}$$

The process noise matrix $\mathbf{Q}$ considers the possible position increments during one sampling period in x and y direction $\sigma_{\Delta x}[m] = \sigma_{\Delta y}[m]$:

$$\mathbf{Q_{CP}} = \begin{bmatrix} \sigma_{\Delta x}^2 & 0 \\ 0 & \sigma_{\Delta y}^2 \end{bmatrix} \tag{3.21}$$

Constant Velocity Model

The Constant Velocity Model is a Free Motion Model and assumes that the object keeps its velocity in x and y direction. It describes the behavior of a rigid body with inertial mass in absence of any external forces. Any change in speed or heading caused by braking or steering is considered as noise. The state consists of the Cartesian position and velocity of the object in x and y direction:

$$\mathbf{x} = \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix} \tag{3.22}$$

The system function is linear and is given by the following matrix:

$$\mathbf{F_{CV}} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.23}$$

The process noise matrix $\mathbf{Q}$ considers the possible velocity increments during one sampling period in x and y direction $\sigma_{\Delta vx}[\frac{m}{s}] = \sigma_{\Delta vy}[\frac{m}{s}]$:

$$\mathbf{Q_{CV}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{\Delta vx} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\Delta vy} \end{bmatrix} \tag{3.24}$$

Constant Acceleration Model

The Constant Acceleration Model is a Free Motion Model and assumes that the object keeps its acceleration in x and y direction. It describes the motion of a rigid body under control of external forces, which are kept constant. Any change in acceleration, as caused by turning the steering wheel or changing the position of the brake or throttle pedal, is considered as noise. The state consists of the Cartesian position, velocity and acceleration of the object in x and y direction:

$$\mathbf{x} = \begin{bmatrix} x \\ v_x \\ a_x \\ y \\ v_y \\ a_y \end{bmatrix} \tag{3.25}$$

The system function is linear and is given by the following matrix:

$$\mathbf{F_{CA}} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.26}$$

The process noise matrix $\mathbf{Q}$ considers the possible acceleration increments during one sampling period in x and y direction $\sigma_{\Delta ax}[\frac{m}{s^2}] = \sigma_{\Delta ay}[\frac{m}{s^2}]$:

$$\mathbf{Q_{CA}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\Delta ax} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{\Delta ay} \end{bmatrix} \tag{3.27}$$

Constant Heading and Velocity Model

The Constant Heading and Velocity (CHV) Model is a Curvilinear motion model. It assumes essentially the same behavior as the CV Model. The difference is that the process noise in longitudinal direction may be specified different to the noise in lateral direction. The longitudinal noise disturbs the velocity, the lateral noise disturbs the heading. This model is more appropriate for vehicles with non-holonomic kinematics. The state consists of the Cartesian position in 2D (x, y), the heading and the velocity (speed):

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \phi \\ v \end{bmatrix} \tag{3.28}$$

The motion model is non-linear and is given by the system function:

$$\Delta f_{CHV}(t) = \begin{bmatrix} v\Delta t \cos(\phi) \\ v\Delta t \sin(\phi) \\ 0 \\ 0 \end{bmatrix} \tag{3.29}$$

To predict the covariance matrix of the CHV filter, the Jacobian matrix of the system function is needed:

$$\frac{\partial f_{CHV}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & -v\Delta t \sin(\phi) & \Delta t \cos(\phi) \\ 0 & 1 & v\Delta t \cos(\phi) & \Delta t \sin(\phi) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

The process noise matrix $\mathbf{Q}_{CHV}$ is a 4×4 diagonal matrix with heading increment $\sigma_{\Delta\phi}[rad]$ and velocity increment $\sigma_{\Delta v}[\frac{m}{s}]$:

$$\mathbf{Q_{CHV}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\Delta\phi} & 0 \\ 0 & 0 & 0 & \sigma^2_{\Delta v} \end{bmatrix} \tag{3.31}$$

Constant Turn Rate and Velocity Model

The Constant Turn Rate and Velocity Model is the most often used curvilinear motion model. In aviation application, it is also called Coordinated Turn Model. It extends the CHV model by assuming a constant turn rate. The longitudinal and the radial acceleration are considered as process noise. The state consists of the Cartesian position in 2D, the heading, the velocity (speed) and the turn rate:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \phi \\ v \\ \omega \end{bmatrix} \tag{3.32}$$

The motion model is non-linear and is given by the following system function. Since the system function has the turn rate in the denominator, it cannot be used for objects going straight. The CTRV motion model needs therefore the CHV motion model as fallback in case of zero or very small turn rates. Moreover, the filter becomes very unstable for combinations of low turn rate and velocity.

$$\Delta f_{CTRV}(t) = \begin{bmatrix} \frac{v}{\omega}(\sin(\phi + \Delta t\,\omega) - \sin(\phi)) \\ -\frac{v}{\omega}(\cos(\phi + \Delta t\,\omega) - \cos(\phi)) \\ \Delta t\,\omega \\ 0 \\ 0 \end{bmatrix} \tag{3.33}$$

To predict the covariance matrix of the CTRV filter, the Jacobian matrix of the system function is needed:

$$\frac{\partial f_{CTRV}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & \frac{v(\cos(\phi+\Delta t\omega)-\cos(\phi))}{\omega} & \frac{\sin(\phi+\Delta t\omega)-\sin(\phi)}{\omega} & \frac{-\Delta t v\cos(\phi+\Delta t\omega)}{\omega} + \frac{v(\sin(\phi+\Delta t\omega)-\sin(\phi))}{\omega^2} \\ 0 & 1 & \frac{v(\sin(\phi+\Delta t\omega)-\sin(\phi))}{\omega} & \frac{-(\cos(\phi+\Delta t\omega)-\cos(\phi))}{\omega} & \frac{\Delta t v\sin(\phi+\Delta t\omega)}{\omega} + \frac{v(\cos(\phi+\Delta t\omega)-\cos(\phi))}{\omega^2} \\ 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.34}$$

The process noise matrix $\mathbf{Q}_{CTRV}$ is a 5×5 diagonal matrix with turn rate increment $\sigma_{\Delta\omega}[\frac{rad}{s}]$ and velocity increment $\sigma_{\Delta v}[\frac{m}{s}]$:

$$\mathbf{Q_{CHV}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\Delta v}^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\Delta\omega}^2 \end{bmatrix} \tag{3.35}$$

Constant Heading and Acceleration Model

The constant Heading and Acceleration Model (CHA) is a curvilinear motion model. It is rarely mentioned in literature, but its usage is mandatory as fallback motion model for the more popular CTRA model in case of zero or small turn rates. It assumes external forces which accelerate the object in longitudinal direction, while keeping its heading. The noise enters as acceleration and heading increment. The state consists of the Cartesian position in 2D, the heading, the velocity and the acceleration:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \phi \\ v \\ a \end{bmatrix} \tag{3.36}$$

The motion model is non-linear and is given by the following system function:

$$\Delta f_{CHA}(t) = \begin{bmatrix} (v\Delta t + a\frac{\Delta t^2}{2})\cos(\phi) \\ (v\Delta t + a\frac{\Delta t^2}{2})\sin(\phi) \\ 0 \\ a\Delta t \\ 0 \end{bmatrix} \tag{3.37}$$

To predict the covariance matrix of the CHA filter, the Jacobian matrix of the system function is needed:

$$\frac{\partial f_{CHA}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & -(v\Delta t\sin(\phi)+\frac{1}{2}a\Delta t^2\sin(\phi)) & \Delta t\cos(\phi) & \frac{1}{2}\Delta t^2\cos(\phi) \\ 0 & 1 & v\Delta t\cos(\phi)+\frac{1}{2}a\Delta t^2\cos(\phi) & \Delta t\sin(\phi) & \frac{1}{2}\Delta t^2\sin(\phi) \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.38}$$

The process noise matrix $\mathbf{Q}_{CHA}$ is a 5×5 diagonal matrix with heading increment $\sigma_{\Delta\phi}[rad]$ and acceleration increment $\sigma_{\Delta a}[\frac{m}{s^2}]$:

$$\mathbf{Q_{CHV}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\Delta\phi} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\Delta a} \end{bmatrix} \tag{3.39}$$

Constant Turn Rate and Acceleration Model

The Constant Turn Rate and Acceleration Model (CTRA) is a curvilinear motion model. It is an extension of the CTRV model and is the equivalent of the CA model in curvilinear motion. It assumes that both longitudinal and lateral motion are controlled by some external forces. The noise enters the system as acceleration and turn rate increment. The state consists of the Cartesian position in 2D, the heading, the longitudinal velocity and acceleration and the turn rate:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \phi \\ v \\ \omega \\ a \end{bmatrix} \tag{3.40}$$

The motion model is non-linear and is given by the following system function. As the CTRV, the system function has the turn rate in the denominator and therefore cannot be used for objects going straight. The CHA model is then used as fallback [190].

$$\Delta f_{CTRA}(t) = \begin{bmatrix} \frac{v}{\omega}(\sin(\phi+\Delta t\omega)-\sin(\phi)) + \frac{a\Delta t}{\omega}\sin(\phi+\Delta t\omega) + \frac{a}{\omega^2}(\cos(\phi+\Delta t\omega)-\cos(\phi)) \\ -\frac{v}{\omega}(\cos(\phi+\Delta t\omega)-\cos(\phi)) - \frac{a\Delta t}{\omega}\cos(\phi+\Delta t\omega) + \frac{a}{\omega^2}(\sin(\phi+\Delta t\omega)-\sin(\phi)) \\ \Delta t\omega \\ \Delta t a \\ 0 \\ 0 \end{bmatrix} \tag{3.41}$$

The Jacobi matrix for the CTRA becomes a little bit unhandy:

$$\frac{\partial f_{CTRA}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & f_1 & \frac{\sin(\phi+\Delta t\omega)-\sin(\phi)}{\omega} & f_2 & \frac{\Delta t\sin(\phi+\Delta t\omega)}{\omega} + \frac{\cos(\phi+\Delta t\omega)-\cos(\phi)}{\omega^2}) \\ 0 & 1 & f_3 & \frac{-(\cos(\phi+\Delta t\omega)-\cos(\phi))}{\omega} & f_4 & \frac{-\Delta t\cos(\phi+\Delta t\omega)}{\omega} + \frac{\sin(\phi+\Delta t\omega)-\sin(\phi)}{\omega^2}) \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.42}$$

$$f_1 = \frac{(a\Delta t+v)\cos(\phi+\Delta t\omega)-v\cos(\phi)}{\omega} - \frac{a(\sin(\phi+\Delta t\omega)-\sin(\phi))}{\omega^2} \tag{3.43}$$

$$f_2 = \frac{-(a\Delta t^2+v\Delta t)\cos(\phi+\Delta t\omega)}{\omega} + \frac{2a\Delta t\sin(\phi+\Delta t\omega)+v(\sin(\phi+\Delta t\omega)-\sin(\phi))}{\omega^2} + \frac{2a(\cos(\phi+\Delta t\omega)-\cos(\phi))}{\omega^3} \tag{3.44}$$

$$f_3 = \frac{(a\Delta t+v)\sin(\phi+\Delta t\omega)-v\sin(\phi)}{\omega} + \frac{a(\cos(\phi+\Delta t\omega)-\cos(\phi))}{\omega^2} \tag{3.45}$$

$$f_4 = \frac{(a\Delta t^2+v\Delta t)\sin(\phi+\Delta t\omega)}{\omega} + \frac{2a\Delta t\cos(\phi+\Delta t\omega)+v(\cos(\phi+\Delta t\omega)-\cos(\phi))}{\omega^2} - \frac{2a(\sin(\phi+\Delta t\omega)-\sin(\phi))}{\omega^3} \tag{3.46}$$

The process noise matrix $\mathbf{Q}_{CRTA}$ is a 6×6 diagonal matrix with turn rate increment $\sigma_{\Delta\omega}[\frac{rad}{s}]$ and acceleration increment $\sigma_{\Delta a}[\frac{m}{s^2}]$:

$$\mathbf{Q_{CTRA}} = \Delta t^2 \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\Delta\omega}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\Delta a}^2 \end{bmatrix} \tag{3.47}$$

3.8.2 Measurement Models and Filter Update

The first part of the Kalman filters, the prediction, is often also used in isolation to do long term predictions of the state in several steps without any measurement updates. In literature is then often claimed that a Kalman Filter is used for prediction, but this formulation is misleading since there are no measurements.

The second part of all Kalman filters is the update step, the left part of Equation 3.1. It requires the computation of the measurement likelihood $p(\mathbf{z_k}|\mathbf{x_k})$ in the nominator. The calculation of the evidence $p(\mathbf{z_k}|\mathbf{z_{1:k-1}})$ in the denominator can be avoided due to the property of the Gaussian distribution being the conjugate prior to itself.

The measurement vector $\mathbf{z_k} \in \mathbb{R}^p$ of the discrete time stochastic state space system is generated from the state $\mathbf{x_k}$ by the measurement function $\mathbf{h}(\cdot)$:

$$\mathbf{z_k} = \mathbf{h}(\mathbf{x_k}) + \mathbf{v_k} \tag{3.48}$$

The measurement is disturbed by the zero mean, white Gaussian noise vector $\mathbf{v_k} \in \mathbb{R}^p$.

When tracking traffic participants with LIDAR, the measurements of the LIDAR are usually not processed directly. Instead, several LIDAR measurements, which are assumed to originate from the same object, are clustered together (see Section 3.5) and a bounding box is computed (see Section 3.6). The tracking algorithm uses then the centroid or some corner of the box as reference point to be tracked. The Cartesian coordinates of this reference point is the pseudo measurement, which is input to the Kalman filter. From this follows that the measurement model is always linear, independent from the process model and independent from the filter type used for the prediction. The predicted measurement $\mathbf{z_k^-}$ and its innovation covariance $\mathbf{S_k}$ is given by:

$$\hat{\mathbf{z}}_\mathbf{k}^- = \mathbf{H}\hat{\mathbf{x}}_\mathbf{k}^- \tag{3.49}$$

$$\mathbf{S_k} = \mathbf{H}\mathbf{P_k^-}\mathbf{H^T} + \mathbf{R} \tag{3.50}$$

with constant measurement matrix $\mathbf{H} \in \mathbb{R}^{p \times n}$ and measurement noise covariance $\mathbf{R} \in \mathbb{R}^{p \times p}$. As already mentioned above in this section, there is no problem in combining a linear measurement model from the LKF with non-linear motion models of the EKF or UKF. For the EKF and UKF formulas for non-linear measurement models see [184] and [181].

The Kalman gain matrix $\mathbf{K_k} \in \mathbb{R}^{n \times p}$ is given by the combination of the predicted state covariance and the innovation covariance:

$$\mathbf{K_k} = \mathbf{P_k^-}\mathbf{H^T}\mathbf{S_k^{-1}} \tag{3.51}$$

In the final correction step, the mean and the covariance of the next step are calculated as:

$$\hat{\mathbf{x}}_\mathbf{k} = \hat{\mathbf{x}_\mathbf{k}}^- + \mathbf{K_k}(\mathbf{z_k} - \hat{\mathbf{z}}_\mathbf{k}^-) \tag{3.52}$$

$$\mathbf{P_k} = \mathbf{P_k^-} - \mathbf{K_k}\mathbf{S_k}\mathbf{K_k^T} \tag{3.53}$$

The measurement matrix for the Constant Position Model from Subsection 3.8.1 is given by:

$$\mathbf{H_{CP}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.54}$$

The measurement matrices for all other motion models are constructed by padding $\mathbf{H_{CP}}$ to the right with $\mathbf{0}$-columns according to the dimension of the state.

3.8.3 Interacting Multiple Model (IMM) Filter

The above motion models and their corresponding filters have the characteristic that the higher the order of the model is, the more sensitive they are to measurement errors. This is due to the fact that normally only the lowest order of the state space, the position, is measured and the higher orders, as velocity or acceleration, are derived by the filter. On the other hand, the lower order filters behave poorly, if their assumptions, as constant velocity or constant heading, are violated by a maneuvering target. Therefore, it may be beneficial to use a mixture of the above filters for tracking. This idea follows the IMM filter [158] [22].

The Hidden Markov model of the IMM filter (see Figure 3.11) is an extension of the HMM of the Kalman filter (see Figure 3.10). It is extended by an unobservable mode M_k, which is inferred by the IMM filter together with the state $\mathbf{x_k}$ from the measurements $\mathbf{z_k}$.

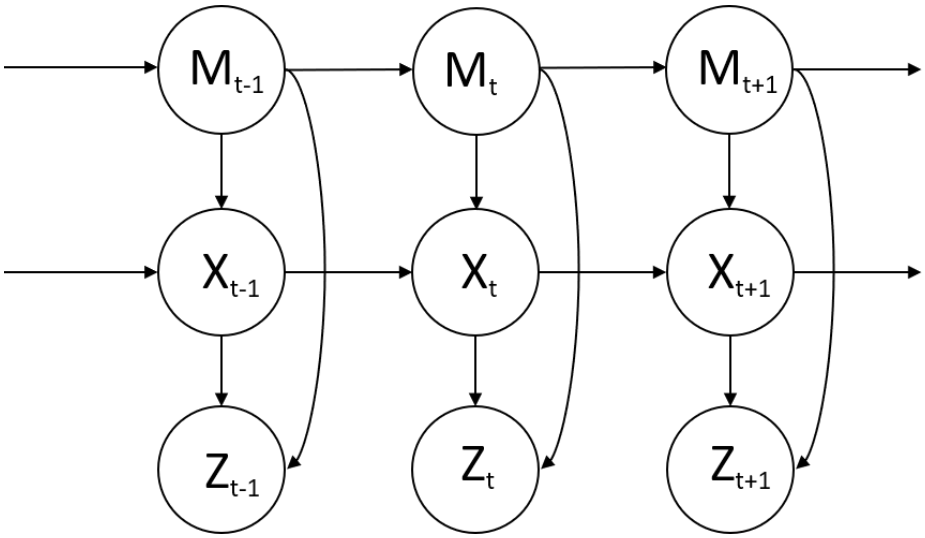


Fig. 3.11 Hidden Markov Model of IMM Filter. The temporal sequence of the hidden states **X** and modes **M** are estimated using the observations **Z**.

The IMM filter represents the state as a Gaussian mixture by running two or more Kalman filters in parallel. LKF, EKF and UKF filters may be combined, but it must be possible to mix the system states of all filters. Therefore, it is not recommendable to combine free motion and curvilinear filters, since this would require elaborate conversions of the state and covariance before each filtering step. Filters of different orders, as CV and CA, may be combined, but the state of the lower order filters must be augmented by some default values, usually 0.

Each filter represents a mode $i \in \{1..r\}$ with a mode probability $\mu_{i,k}$ at time step k. The mode probabilities are the weights of the Gaussian mixture components (see Subsection 2.4.4). Before the prediction step, the mixin probability priors of each mode $\mu_{(i|j),k-1}$ are calculated from the previous mode probability $\mu_{i,k-1}$ and the mode transition matrix $\mathbf{\Pi} \in \mathbb{R}^{r \times r}$:

$$\mu_{(i|j),k-1} = \frac{\mathbf{\Pi}_{i,j}\mu_{i,k-1}}{\bar{\mu}_{j,k}} \tag{3.55}$$

with the predicted probability of mode j:

$$\bar{\mu}_{j,k} = \sum_{i=1}^{r} \mathbf{\Pi}_{i,j}\mu_{i,k-1} \tag{3.56}$$

These mixin probabilities are the weights for the moment matching from the previous Gaussian mixture:

$$\hat{\mathbf{x}}^{M}_{j,k-1} = \sum_{i=1}^{r} \mu_{(i|j),k-1} \hat{\mathbf{x}}_{\mathbf{i},\mathbf{k-1}} \tag{3.57}$$

$$\mathbf{P}^{\mathbf{M}}_{\mathbf{j},\mathbf{k-1}} = \sum_{i=1}^{r} \mu_{(i|j),k-1} (\mathbf{P}_{\mathbf{i},\mathbf{k-1}} + (\hat{\mathbf{x}}_{\mathbf{i},\mathbf{k-1}} - \hat{\mathbf{x}}^{M}_{j,k-1})(\hat{\mathbf{x}}_{\mathbf{i},\mathbf{k-1}} - \hat{\mathbf{x}}^{M}_{j,k-1})^{T}) \tag{3.58}$$

Subsequently, the means and covariances of the filters are predicted and updated independently by the specific Kalman filters,

The update of the mode probabilities is based on their likelihoods $\lambda_{j,k}$:

$$\lambda_{j,k} = \frac{1}{\sqrt{|2\pi S_{j,k}|}} e^{-\frac{1}{2}(\mathbf{z}_k - \hat{\mathbf{z}}^{-}_{j,k})^{T} S_{j,k} (\mathbf{z}_k - \hat{\mathbf{z}}^{-}_{j,k})} \tag{3.59}$$

Normalization of the likelihoods results in the posterior of the mode probability $\mu_{j,k}$:

$$\mu_{j,k} = \frac{\lambda_{j,k} \mu^{-}_{j,k}}{\sum_{i=1}^{r} \lambda_{j,k} \mu^{-}_{j,k}} \tag{3.60}$$

The different posteriors of the individual filters are subsequently input for the next recursion step. If required, moment matching can be used to calculate a combined output of the IMM filter:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{r} \mu_{j,k} \hat{\mathbf{x}}_{\mathbf{j},\mathbf{k}} \tag{3.61}$$

$$\mathbf{P}_{\mathbf{j},\mathbf{k}} = \sum_{i=1}^{r} \mu_{j,k} (\mathbf{P}_{\mathbf{j},\mathbf{k}} + (\hat{\mathbf{x}}_{\mathbf{j},\mathbf{k}} - \hat{\mathbf{x}}_{\mathbf{j}})(\hat{\mathbf{x}}_{\mathbf{j},\mathbf{k}} - \hat{\mathbf{x}}_{j})^{T}) \tag{3.62}$$

3.8.4 Evaluation Filtering

To evaluate the suitability of different popular Kalman filters for state estimation in urban environments, the different filter types have been implemented as part of the fub_roscar system and have been tested in various real-world situations using the MadeInGermany test vehicle. The following filter configurations have been examined:

- LKF with Constant Velocity Model
- LKF with Constant Acceleration Model.
- IMM filter with a combination of the previous two.
- EKF with CTRV Model.

- EKF with CTRA Model.
- IMM filter with a combination of the previous two.

The curvilinear motion models have also been evaluated with a UKF filter instead of the EKF, but since there haven't been any major differences in the result, these test runs are not presented in detail here. In all cases, the measurement model of the filters is linear. The filters are run with a frequency of $\approx 10Hz$.

The main purpose of the filter is to reduce the influence of the measurement noise to the state estimate . Therefore, a single test run cannot be taken as base for the evaluation since the noise sequence is different on every run. The evaluation approach taken in this work is as follows:

- Test drives of MadeInGermany in different real-world environments are registered in ROS-bag format.
- The highly exact position measurements of the Applanix POS LV (see Section 2.1) are taken as ground truth for the position of the ego-vehicle.
- On replay of the registered drives, these position measurements are disturbed by random noise and fed into the filter under test.
- For each filter configuration, several replay runs are executed to get averaged results.

The comparison of the filters is done based on the following criteria:

- Absolute filter performance. This is given by the RMSE of the position error. The x- and y-position estimates are the only states common for all filters under consideration. Lower RMSE means better performance. The RMSE over N runs is calculated by [22, p. 243]:

$$RMS(\tilde{\mathbf{x}}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\tilde{x}_i^2 + \tilde{y}_i^2)} \tag{3.63}$$

- Normalized Innovation Squared (NIS). For the consistency of the filter, it is important, that the innovations, e.g. the difference between the measurements and its expected values, are commensurate with the innovation covariance (see Equation 3.50). The mean of the NIS is then calculated as [22, p.236]:

$$NIS(\mathbf{z}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{z}_i - \hat{\mathbf{z}}_i^-)^T\, \mathbf{S}_i^{-1}\, (\mathbf{z}_i - \hat{\mathbf{z}}_i^-)} \tag{3.64}$$

For 2-dimensional measurements, the NIS value should be ≈ 2.

- Normalized Estimation Error Squared (NEES). For a consistent filter, the state estimation errors should be commensurate with the covariance matrix of the state. The evaluated state error for the free motion models CV and CA is the position and velocity error in x- and y-dimension. For the curvilinear motion models CTRV and CTRA, the speed and the heading error is evaluated instead of the velocities. Ground truth for velocity/speed and heading is taken from Odometry. The mean of the NEES is calculated as [22, p.234]:

$$NEES(\tilde{\mathbf{x}}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \tilde{\mathbf{x}}_i^T \, \mathbf{P}_i^{-1} \, \tilde{\mathbf{x}}_i} \tag{3.65}$$

For 4-dimensional state estimates, the NEES value should be ≈ 4.

The measurements used in the system are not the individual pixel values returned by the LIDAR, but the centroids of the clustered objects. So the measurement noise must include the effects of clustering errors and also consider partial occlusions. The magnitude of the measurement noise has been estimated from the variance of repeated measurements of stationary objects while passing them with the ego-vehicle. Based on these results, a zero-mean measurement noise with a standard deviation of $0.3m$ in x- and y-direction is assumed, resulting in $\approx 0.42m$ standard deviation for the position.

The process noise parameters have been fitted manually to the following values:

- CV model (see Equation 3.24): $\sigma_{\Delta vx} = \sigma_{\Delta vy} = 10\frac{m}{s}$
- CA model (see Equation 3.27): $\sigma_{\Delta ax} = \sigma_{\Delta ay} = 10\frac{m}{s^2}$
- CTRV model (see Equation 3.39): $\sigma_{\Delta v} = 14.14\frac{m}{s}, \sigma_{\Delta\omega} = 45\frac{deg}{s}$
- CTRA model (see Equation 3.47): $\sigma_{\Delta a} = 10\frac{m}{s^2}, \sigma_{\Delta\omega} = 45\frac{deg}{s}$

The following real-world scenarios have been evaluated:

- Lane Change Scenario
- Turn and Merge Scenario
- Intersection Crossing Scenario
- Acceleration / Brake Scenario

- Turn Right / Turn Left Scenario

The first three scenarios are also used for evaluations in further parts of this work (see Section 5.10 for more details). For each scenario, the evolution of the RMSE, NIS and NEES values over time is depicted. Additionally, the heading RMSE is shown, which is only for the curvilinear filters part of the state. For the other filters, it is calculated from the x- and y-velocities. The heading errors are important to understand the inconsistency of some of the filters. All values are averaged over 20 runs.

Lane Change Scenario

The Lane Change Scenario (see Figure 3.12)is the most simple scenario for all filters. The vehicle drives at a speed of $7 - 10 m/s$ on a straight road, performing two smooth lane changes. The position errors fluctuate around $\approx 0.25\ m$. The heading errors are constantly around $4\ deg$. The NIS values are mostly below the critical threshold of 2, even if there are a few outlier, which is acceptable. The NEES values are good for the lower order filters (CV and CTRV), while the higher order filters (CA and CTRA) show too many high values. This result is typical for higher order filters, which are more prone to measurement noise in simple scenarios. Since there are no significant maneuvers during the scenario, the IMM filters are dominated by the lower order filters and show therefore good results.

Intersection Crossing Scenario

In the Intersection Crossing Scenario(see Figure 3.13, the test vehicle drives on a straight road and crosses a main road with a central reservation between the two lanes. The vehicle has to wait before it can cross the first lane of the main road, advances to the central reservation, where it has to stop again before it can finally accelerate again. The position errors and the NIS of all filters are acceptable for the whole scenario. When moving slow and during the stops, the heading error gets very high for all filters. The reason for that is that the moved distance between two measurements becomes lower than the standard deviation of the measurements. The measured positions start to jump forwards and backwards, to the left and right. The value for the heading becomes erratic. This is no problem for CV filter since the heading is not part of the estimate and has no influence on future predictions, but as can be seen from the NEES values, the curvilinear filters become unstable. The CA filter also shows some deterioration of the estimates over time when moving slow or being stopped, caused by small accelerations induced by the noise.

Turn and Merge Scenario

In the Turn and Merge scenario (see Figure 3.14) the vehicle drives first with constant speed on a two lane road with central reservation and then slows down to take a U-turn into the opposite direction. During the U-turn, it comes almost to a stop since it has to give way to an upcoming vehicle, before it may complete the turn and accelerate again. The position errors and the NIS are again acceptable for all filters for the whole scenario. But the heading error becomes again very high for all filters, when the vehicle gets slow, for the same reason as in the Intersection Crossing Scenario. The NEES values for the curvilinear filters are unacceptable, even so they should perform better during a turn maneuver. But the real change of the heading during the U-turn is negligible compared to the measurement noise.

Acceleration/Brake Scenario

The Acceleration/Brake Scenario (see Figure 3.15) has been registered on a parking place. The vehicle is initially stopped and accelerates then with up to $4m/s^2$ until it reaches $\approx 50km/h$. After a few seconds with constant velocity, it decelerates with up to $-5m/s^2$ until it comes to a stop again. As in the previous scenarios, there is a high heading error, when stopped, resulting in high NEES values for the curvilinear filters. The turn rate of the curvilinear filters induced by the heading errors increases to $\approx 300deg/s$ on average while being stopped. When accelerating, these filters need about 5 s before they start to stabilize. During this phase, also the NIS values become unacceptable high and the position RMSE increases to almost 1 m. This is caused by the erroneous turn rate, which is propagated forward by the CTRV and CTRA filters.

Turn Right/ Turn Left Scenario

The Turn Right/ Turn Left Scenario (see Figure 3.16) is the most challenging scenario for all filters. The vehicle accelerates from a stopped state and then starts turning right with a turn rate of almost $50deg/s$. After a short stop, it starts turning left with a turn rate of up to $53deg/s$. For the curvilinear filters it can be seen again that they have difficulties to stabilize after the initial stop, resulting temporarily in bad NIS and position error values. In this scenario, also the NEES values for the CV filter become temporarily too high, but are still better than the ones of the other filters.

Summary of Evaluation

In almost all urban scenarios, the filters based on curvilinear motion models show an unacceptable performance. This is caused by high heading and turn rate errors caused by

measurement noise while being stopped or driving slowly. If the distance covered between two measurements becomes significantly smaller than the measurement noise, the filters will sooner or later become unstable and will have difficulties getting back on track after the target has accelerated again. This proves thesis 2 of this work that Kalman filters with curvilinear motion models are not suitable for tracking in urban traffic scenarios.

Why are they nevertheless so popular? Kalman filters were originally designed to track planes or missiles. These objects seldom go slow and never stop, while they are being tracked. Moreover, typical flight surveillance radar stations take measurements at a rate of $\approx 0.1 - 0.2Hz$. This poses completely different demands on the filter than vehicle tracking by LIDAR.

Some authors report nevertheless good results with curvilinear filters for vehicle tracking. In [157], the measured orientation of the targets bounding box is taken as input data for the heading to stabilize the filter. For the problems with this approach, see Section 3.6. The frequently cited paper [159] evaluates the filters only for the ego-vehicle and uses the yaw rate, obtained from the internal CAN bus, as additional input data to stabilize the filters. In [175], the evaluation is only done for simulated scenarios, in which the ego-vehicle and the target both are permanently in motion. Neither of these approaches is suitable for general obstacle tracking in an urban setting.

Therefore, usage of free motion model filters, as CV, CA or combined as IMM is recommended. The evaluations above show that the position RMSE for the CA is mostly slightly better than for the CV filter, while the NEES for the CA filter sometimes is unsatisfactory. This may result from sub-optimal tuning of the process noise. But altogether, the differences are so small that the simplest approach, the CV, should do well in all situations, mainly because of the high update rate.

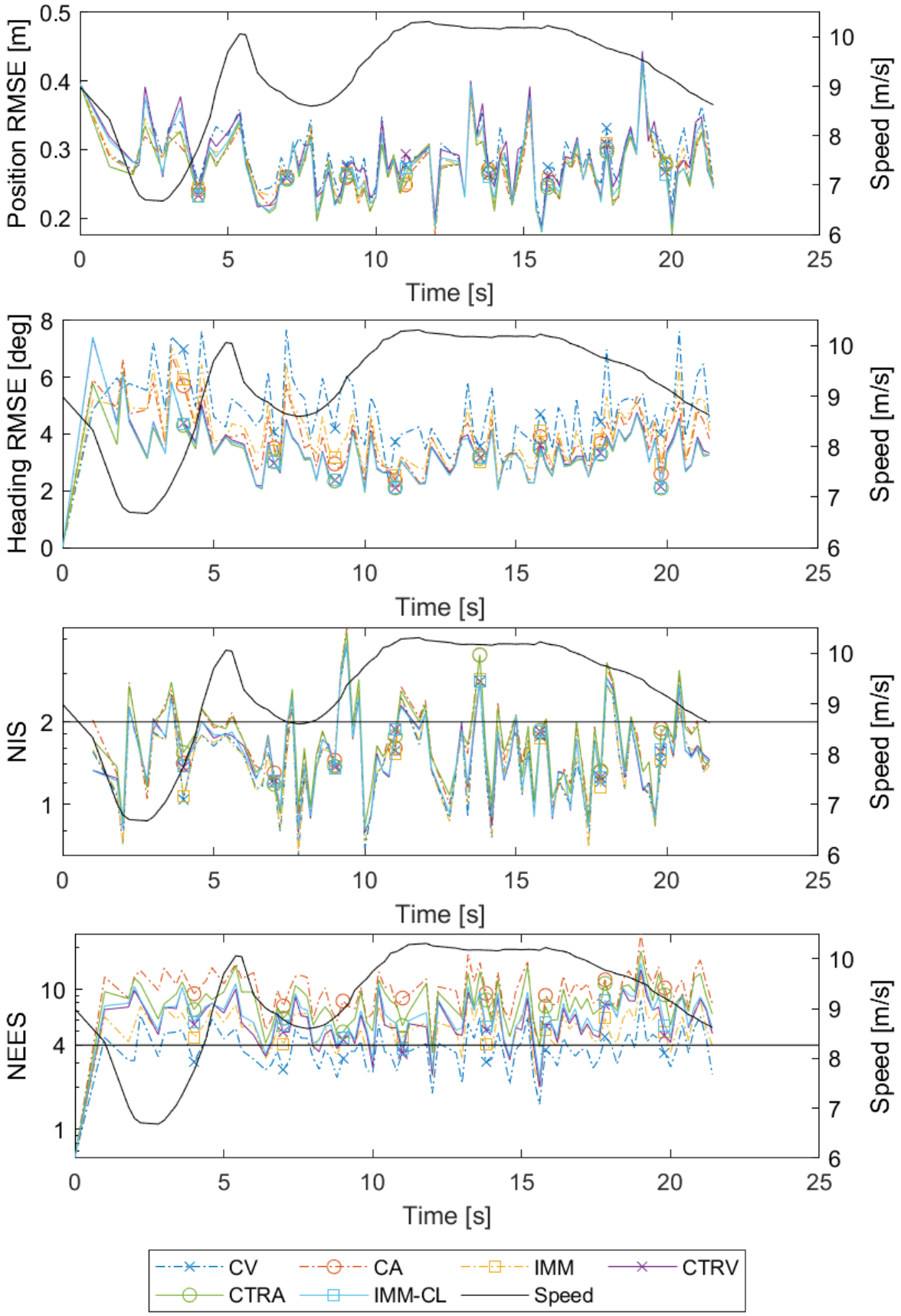


Fig. 3.12 Filter evaluation for the lane change scenario: Position and heading RMSE are low for all filters. The NIS values are acceptable, even if there are some outliers. NEES values are too high for the higher order filters (CA and CTRA).

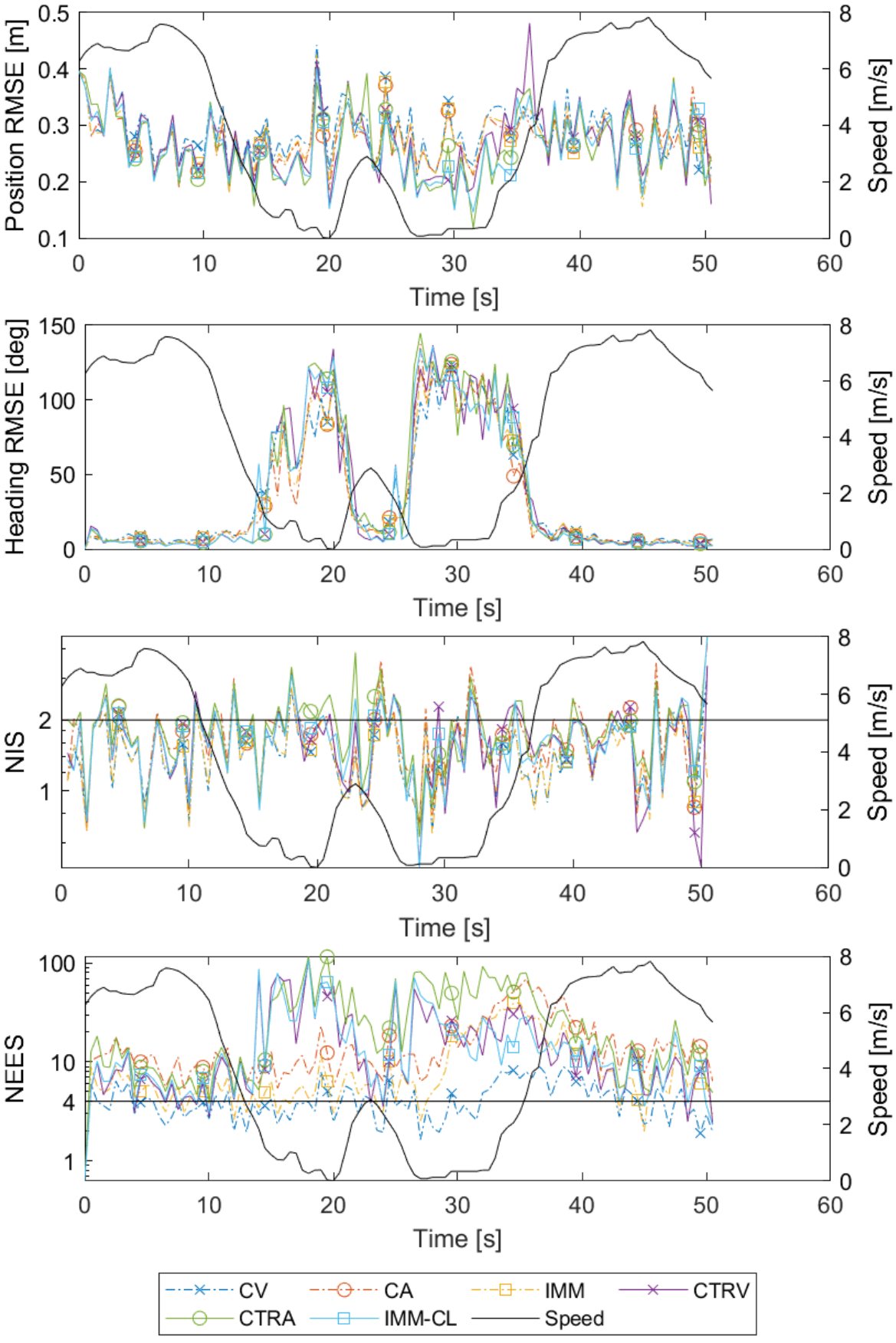


Fig. 3.13 Filter evaluation for the intersection crossing scenario: Position RMSE and NIS values are acceptable for the whole scenario. During stops, the heading RMSE of all filters becomes very high. The NEES values for the curvilinear filters are unacceptable during stops and have difficulties to stabilize thereafter.

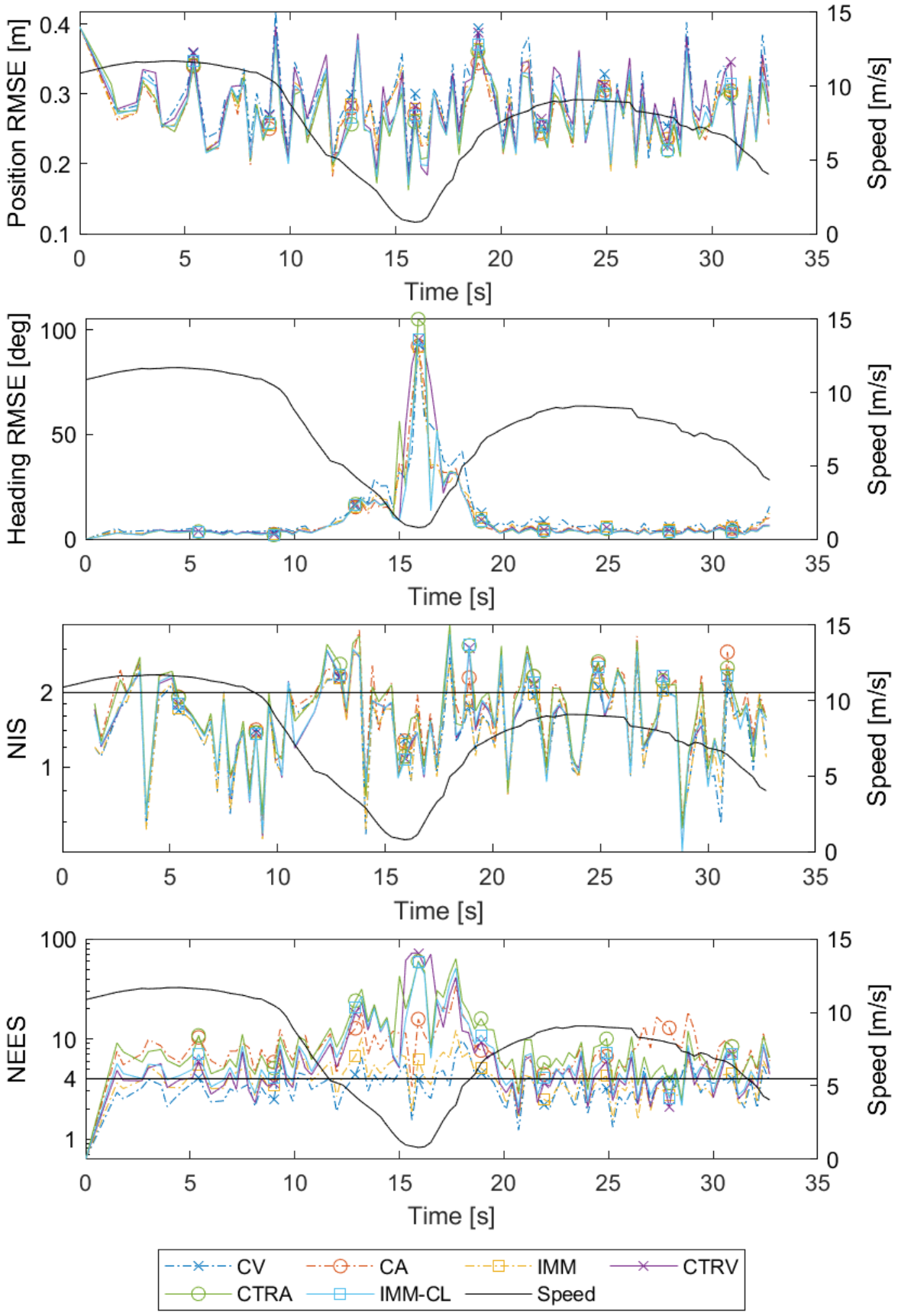


Fig. 3.14 Filter evaluation for the turn and merge scenario: As in Figure 3.13, position RMSE and NIS values are good, while the heading RMSE is very high during slow motion. Again, the NEES values for CTRV and CTRA are unacceptable.

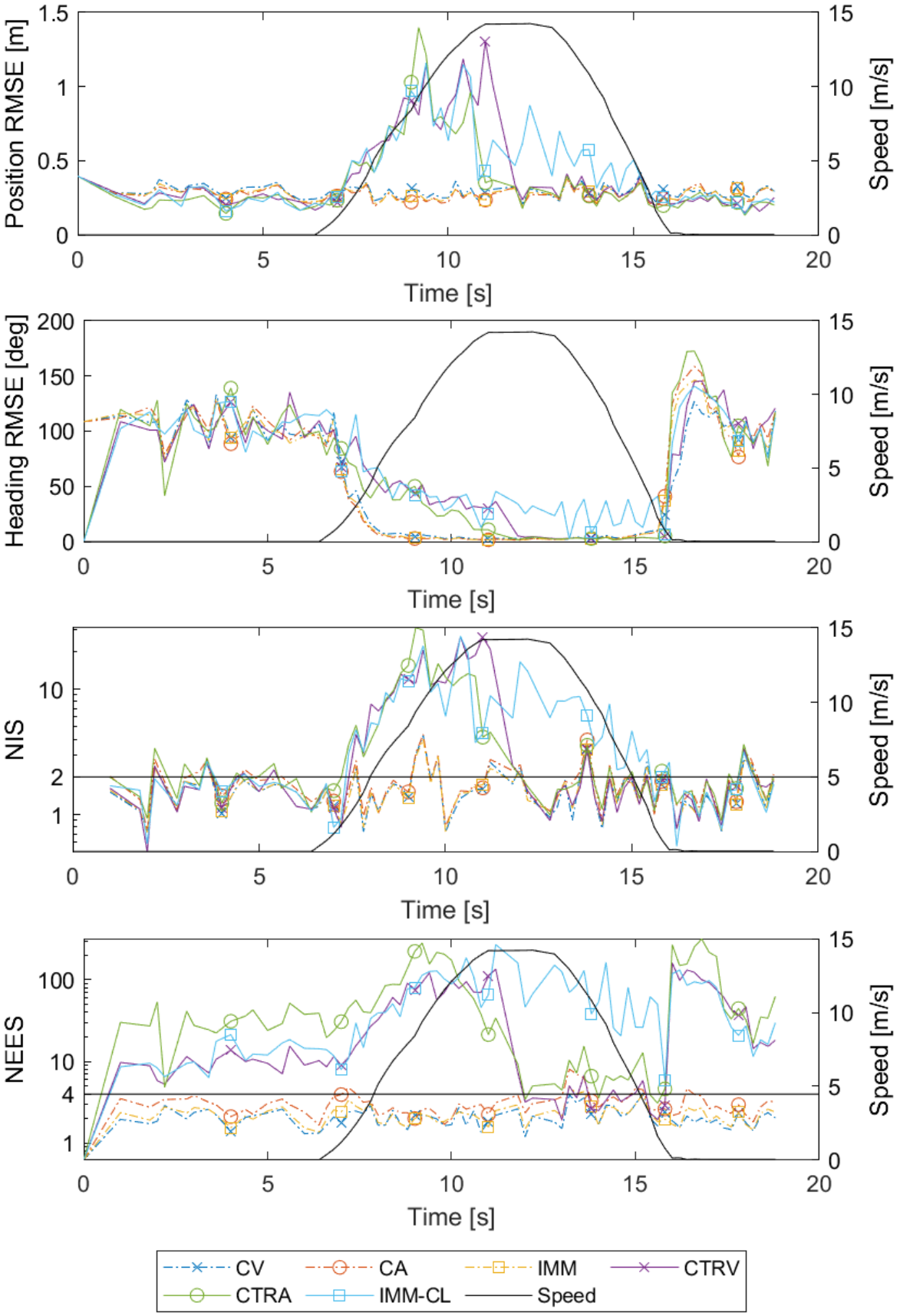


Fig. 3.15 Filter evaluation for the acceleration-brake scenario: Since this scenario starts with a long stopped period, heading RMSE and NEES values of the curvilinear filters are bad from beginning. When starting to move, also position RMSE and NIS values get unacceptable, since have difficulties to stabilize.

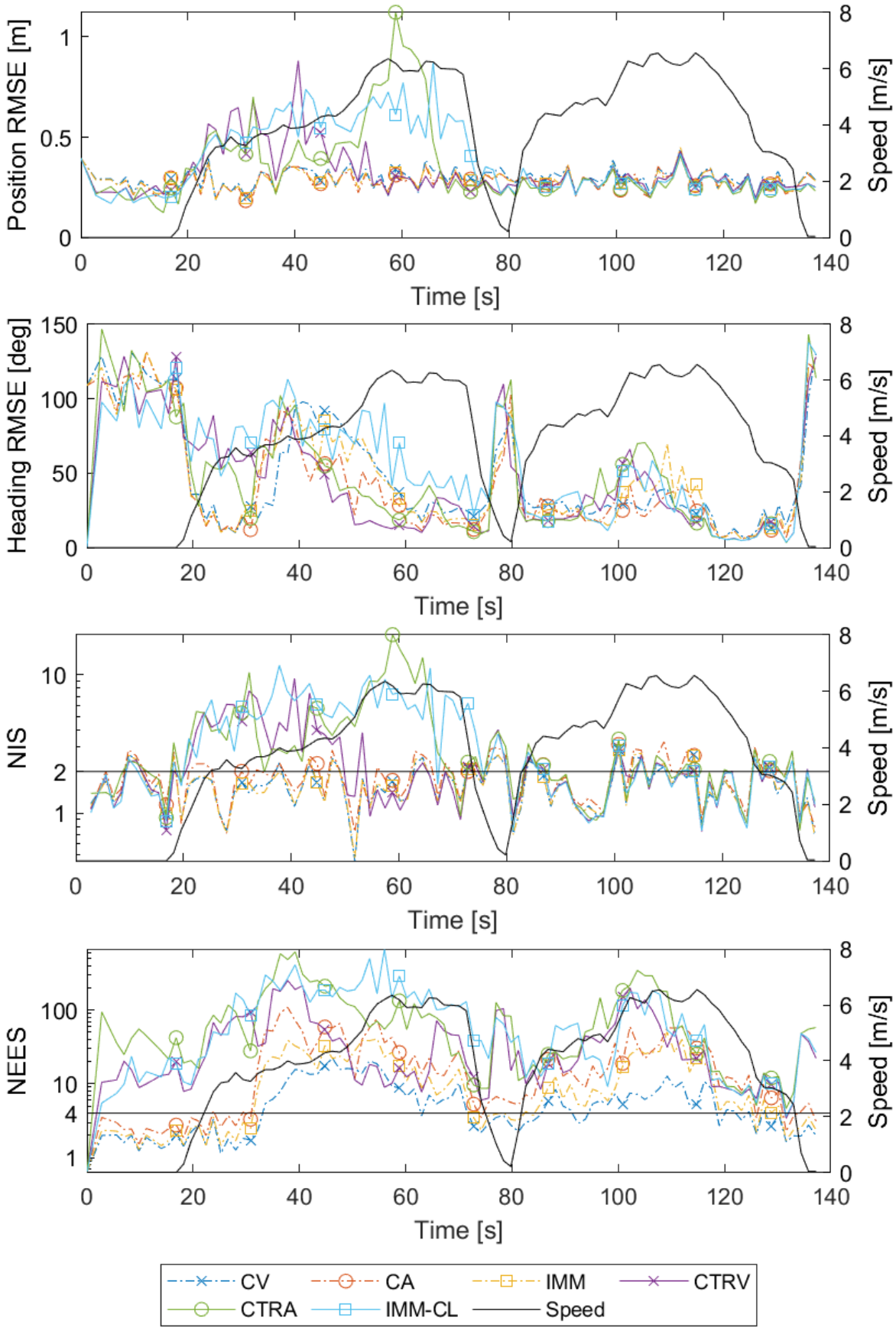


Fig. 3.16 Filter evaluation for the turn right / turn left scenario: Position RMSE and NIS values are only acceptable for CV and CA filters. The NEES values are even for these filters partially too high.

3.9 Occlusion Handling

Occlusion occurs frequently in traffic scenarios, especially in urban environments. Depending on the type of occlusion, it has to be distinguished between:

- Object occlusion: Static or dynamic objects in the street scene are partly or completely occluded. This results in problems for the detection and tracking of such objects.
- Free space occlusion: Some space in the environment is occluded by some occluder. This is a problem for prediction and planning because it is uncertain, whether the occluded space is free of obstacles and is driveable.

For the purpose of multi object tracking, only the first occlusion type is relevant. The effect of free space occlusion must be handled by the prediction and planning system (see Subsection 5.6.2).

With object occlusion, a distinction must be made between:

- Intrinsic occlusion: The real dimensions of an object cannot be detected, since parts of the object are occluded by itself. Intrinsic occlusion depends on the pose of the occluded object relative to the observer.
- Extrinsic occlusion: Another object, the occluder, is located between the observer and the occludee. Extrinsic occlusion yields to partial or complete occlusion of the tracked object.

Insufficient detection and handling of occlusion may cause various failures:

- Wrong estimate of object appearance (shape, length, width and height)
- Wrong estimate of object state (pose and velocity)
- Losing track of objects
- Failures in classification of objects
- Disregard of risks arising from occluded regions

Figure 3.17 shows two typical examples of occlusion. The back of the right car is occluded by the front of the left car in foreground. This may lead to a wrong estimate of the length of the car. Since the length influences the centroid position (see Section 3.7), the pose and the velocity of this object may also be slightly disturbed. More severe is the splitting

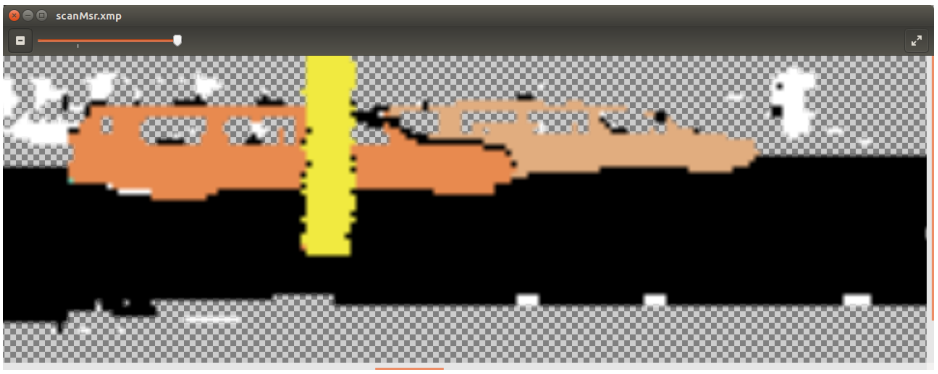


Fig. 3.17 Occluded cars in range image. The right car in the background is partially occluded by the left car in the foreground. The left car is split by a pole in the front.

occlusion of the left car by the tree. This causes the clustering algorithm (see Section 3.5) to detect two independent measurements, which could result in creation of a new track.

In the following subsections it is shown how the neighborhood preserving property of the range image representation may be used to avoid the most severe effects of object occlusion with comparable modest computational effort.

3.9.1 Pose Correction for Partial Occlusion

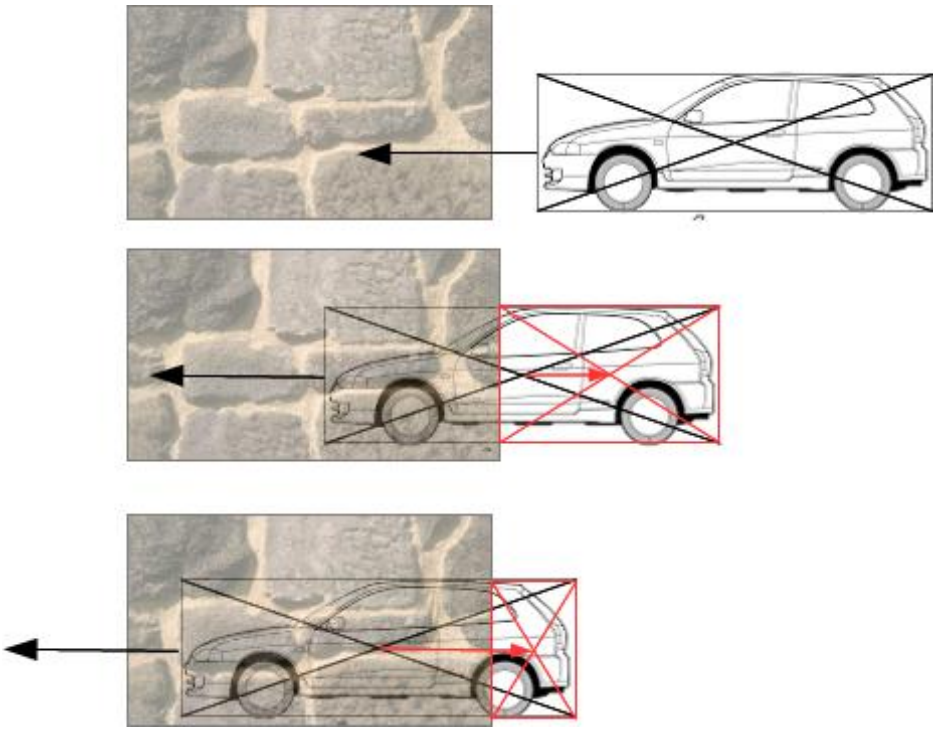


Fig. 3.18 A moving car passes behind some object in the foreground and becomes more and more occluded by the object. The black cross marks the real centroid of the car, the red one the visible centroid.

Figure 3.18 shows the effect of partial occlusion. The initially correctly measured object becomes shorter and since the centroid moves to the right, seems to decelerate. Detection of this situation in the range image is quite easy: For each laser row, the leftmost boundary pixel of a row assigned to this object is checked, whether its left neighbor is closer than the boundary pixel and belongs therefore to some occluding object. If the percentage of left-side occlusion pixels is higher than a certain percentage, the object is considered as left-occluded. The same test is done for the right side.

If a left- or right-occluded measurement is assigned to an existing object, the update of the object dimension is suppressed and centroid position is corrected before it is applied as measurement input for the filter.

The same effect as above may occur in case of a stopped target when the observer is moving to the left. In this case, the target seems to start moving to the right, if the occlusion effect is not handled.

3.9.2 Merging of Split Objects

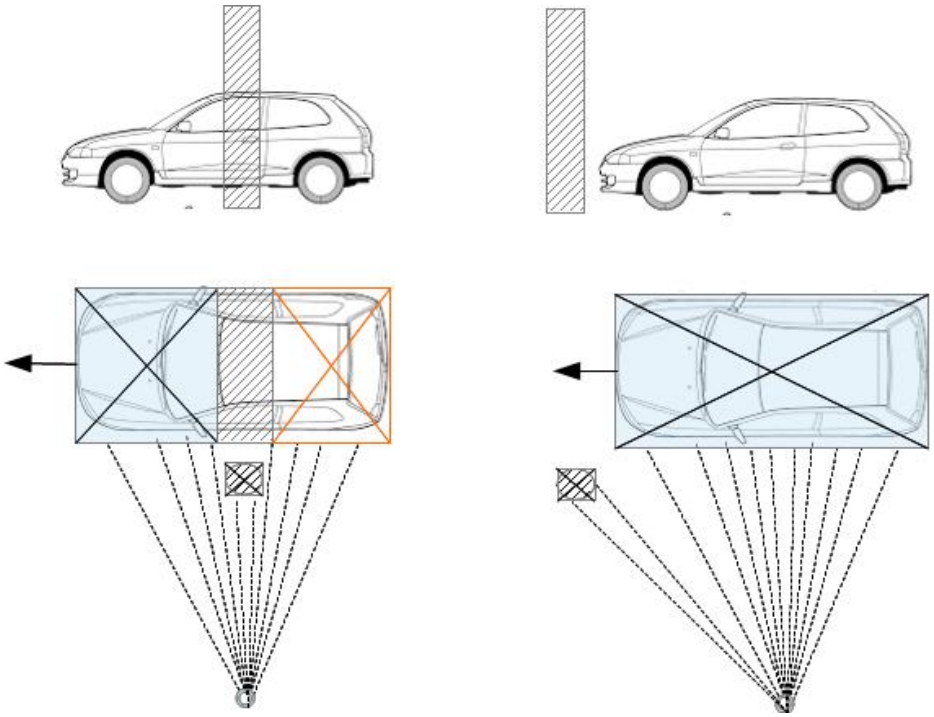


Fig. 3.19 A moving car passes behind some pole. The rightmost laser pixels of the front part of the car are to remote from the left most pixels of the rear part and therefore two separate objects are detected.

Figure 3.19 shows the splitting of a moving car by central occlusions caused by some pole. Without occlusion handling, one of the parts will be assigned to an existing track, causing errors in dimension, pose and velocity estimates, while the other part will cause creation of a new track. Moreover, when the object has completely passed the splitting pole, there will be left only one measurement for two tracks, causing one of the tracks to be orphaned.

The above problems may be avoided by the following procedure: After clustering, but before measurement assignment, for all pairs of left- and right-occluded measurements it is checked, whether their centroids are inside the predicted bounding box of an existing track. If so, the pixels of these two measurements are merged into a new joined measurement.

3.9.3 Existence Probability of Fully Occluded Objects

Figure 3.20 shows a situation, where one object temporarily occludes another one completely. To detect such a case, all tracks to which no measurements were assigned, are examined.

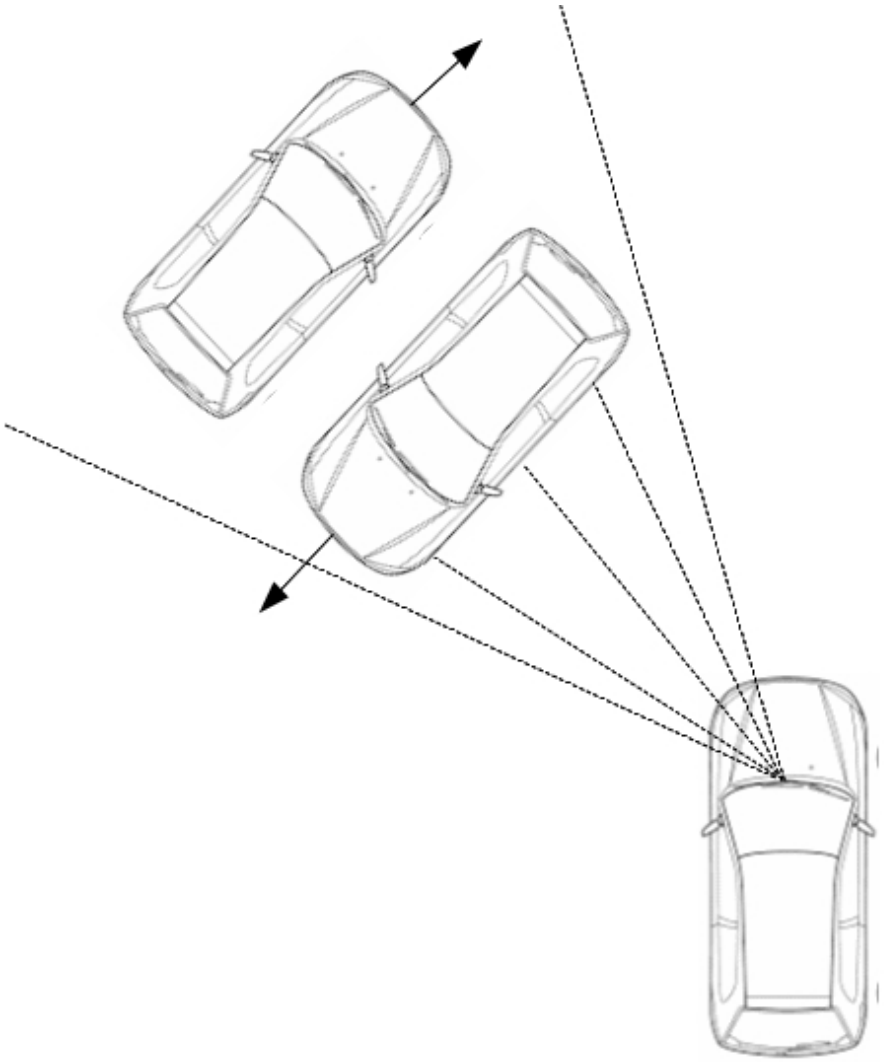


Fig. 3.20 Complete occlusion of a moving object. The upper left car is temporarily completely occluded by the car in the middle.

Their contour is projected to the predicted position in the actual range image and it is checked, whether the corresponding pixels belong to objects closer to the observer. If so, the track is marked as probably fully occluded. This evidence is valuable for the existence estimate (see Section 3.12).

3.10 Motion Type Estimate

Due to the measurement noise or noise in the localization of the ego-vehicle, static objects occasionally seem to reach velocities of $1m/s$ or even more. This may result in predicting, that a tree on the sidewalk will cross the street within the next 10 seconds. But simply setting

low velocities to 0 will not work since pedestrians may really walk at those low velocities. Therefore, a more reliable estimate of the real motion type of a track is needed. The system distinguishes 4 motion types:

- Unknown: For new tracks since a couple of observations are needed to verify the motion type.
- Moving: Object moves consistently.
- Stationary: Object doesn't move consistently and has never been observed moving.
- Stopped: Object doesn't move consistently, but was moving before.

The detection of movements is based on the filter innovation of the position, e.g. the distance between the predicted position and the measured position at time step k:

$$d_{inno,k} = |\mathbf{z_k} - \hat{\mathbf{z}}_\mathbf{k}^-| \tag{3.66}$$

These innovations are set in relation to the predicted motion during the time step, resulting in the relative innovation distance:

$$d_{innoRel,k} = \frac{d_{inno,k}}{|\mathbf{v_k}|\,\Delta t} \tag{3.67}$$

The relative innovations are smoothed using an exponential decay factor α_i resulting in the smoothed relative innovation distance:

$$\overline{d}_{innoRel,k} = \alpha_i\, d_{innoRel,k} + (1-\alpha_m)\,\overline{d}_{innoRel,k-1} \tag{3.68}$$

The following thresholds for $\overline{d}_{innoRel,k}$ have been found empirically:

- < 3: Object is moving consistently.
- > 4: Object is stationary or stopped.
- $3-4$: Previous motion type remains unchanged.

3.11 Classification of Obstacles

Classification of objects based on LIDAR data only is challenging due to the low resolution of the image. Some approaches based on Convolutional Neural Networks (CNN), which

integrate the task of segmentation and classification, have proven to be promising, but still suffer from high computational demands.

In this work, a simple multi-variate Gaussian classifier is proposed, which yields acceptable results due to the fact that the distinction of only a few classes is required for the purpose of traffic scenario prediction. The classes considered in this work are:

$$C \in \{Car, Truck, Bicycle, Pedestrian, OtherSmall, OtherBig, Elevated\} \tag{3.69}$$

The prior probability distribution is assumed to be:

$$p(C) = [0.3,\ 0.05,\ 0.04,\ 0.04,\ 0.1,\ 0.45\ 0.02]^T \tag{3.70}$$

The dimension measurements $\mathbf{z_D} \in \mathbb{R}^3$ are the length, width and height of the objects. Applying Bayes rule results in the conditional class probability:

$$p(C|\mathbf{z}_D) = \eta\ p(C)\ p(\mathbf{z}_D|C) \tag{3.71}$$

with normalization constant η. The likelihoods for the object classes c are given by:

$$p(\mathbf{z}_D|C=c) = \frac{1}{\sqrt{(2\pi)^3|\mathbf{\Sigma}_c|}}\ e^{-\frac{1}{2}(\mathbf{z}_D-\boldsymbol{\mu}_c)^T\ \mathbf{\Sigma}_c\ (\mathbf{z}_D-\boldsymbol{\mu}_c)} \tag{3.72}$$

Mean values $\boldsymbol{\mu}_c$ and covariances $\mathbf{\Sigma}_c$ of the object classes are given by the following table:

Dimension	Length		Width		Height	
Value	μ	σ	μ	σ	μ	σ
Car	5.0 m	1.0 m	2.0 m	0.2 m	1.6 m	0.2 m
Truck	12.0 m	3.0 m	2.5 m	0.3 m	3.0 m	1.0 m
Bicycle	2.0 m	0.3 m	0.8 m	0.2 m	1.7 m	0.2 m
Pedestrian	0.8 m	0.1 m	0.8 m	0.1 m	1.7 m	0.2 m
Other Small (Poles)	0.5 m	0.5 m	0.5 m	0.5 m	2.5 m	3.0 m
Other Big (Buildings)	8.0 m	8.0 m	5.0 m	5.0 m	3.0 m	3.0 m
Elevated (Curbs, Guards)	5.0 m	5.0 m	0.5 m	0.2 m	0.3 m	0.1 m

Table 3.2 Mean values and standard deviations for obstacles classes

The values in the above table are the authors own estimates.

3.12 Existence Estimate and Track Management

The number and the identity of the objects observable from the host vehicle change permanently, especially when the host itself is moving. If measurements are detected, which cannot reasonably be assigned to one of the existing tracks, a new track is created and initialized. These new tracks are considered as unconfirmed until their existence probability raises above a certain threshold. The existence probability is calculated using a Bayesian filter [1]. It depends on the following constants:

- Birth probability P_B: Probability that a detected object did not exist before (e.g. was unobservable).
- Persistence probability P_P: Probability that an existing track will survive until the next time step (e.g. continue to be observable).
- Detection probability P_D: Probability that an observable object is detected (e.g. has a measurement assigned).
- Clutter probability P_C: Probability that a measurement results from clutter.

For a new track, the existence probability p_E is initialized with the birth probability P_B. At subsequent time steps, the new predicted existence probability is

$$p_{E,k}^{-} = p_{E,k-1} P_P + (1 - p_{E,k-1}) P_B \tag{3.73}$$

The Bayesian update of the existence probability depends on the fact, whether the track was detected among the new measurements [1]:

$$p_{E,k} = \begin{cases} \frac{(1-P_D)p_{k-1}^{-}}{(1-P_D)p_{k-1}^{-}+(1-P_C)(1-p_{k-1}^{-})} & \text{if track was not detected} \\ \frac{P_D p_{k-1}^{-}}{P_D p_{k-1}^{-}+P_C(1-p_{k-1}^{-})} & \text{if track was detected} \end{cases} \tag{3.74}$$

When the existence probability raises over the confirmation level P_{conf}, it is confirmed. Only confirmed tracks are reported to subsequent modules of the system. When the existence probability falls below the unconfirm level P_{unconf}, it is unconfirmed. Tracks are deleted, when they are unconfirmed for several time steps.

The parameter values used in the present system are listed in Table 3.3.

The detection probability is corrected to $P_D = 0.2$, if the track has no measurement assigned, but is probably occluded (see Subsection 3.9.3).

Birth Probability	P_B	0.1
Persistence Probability	P_P	0.95
Detection Probability	P_D	0.9
Clutter Probability	P_C	0.2
Confirm Threshold	P_{conf}	0.6
Unconfirm Threshold	P_{unconf}	0.1

Table 3.3 Parameters for Existence Probability Calculation.

3.13 Summary and Conclusion

This chapter has presented a complete system for the perception and multi-object tracking of traffic participants in urban environments based on LIDAR input. The output of the system is a probabilistic state estimate of all relevant objects surrounding the ego-vehicle and provides thereby the base for the prediction and planning components of an autonomous driving system. The output is generated at a frequency of 10 Hz and therefore supports real-time operation. It is intended to support fully automated driving (level 5), but will be useful also for application in lower automation levels.

The major contributions achieved during this research work are the proves of two theses:

- To separate ground and obstacle pixels in LIDAR data, an algorithm based on a range image representation can achieve better results than conventional solutions such as RANSAC. (Section 3.4).
- For object tracking in urban environments, Kalman filters with curvilinear motion models are not suitable due to their instability at low velocities. (Section 3.8).

Both theses have been proven by detailed evaluations of real-world scenarios. Further contributions of this chapter are:

- An enhanced method for range image clustering, which takes into account Non Returners and the Image Cut (Section 3.5).
- An innovative procedure for occlusion detection in range images and methods to mitigate the effect of occlusions on the tracking performance (Section 3.9).
- A new method to distinguish moving and stationary/stopped objects in the presence of measurements noise (Section 3.10).
- An innovative way to consider object occlusion in the existence estimate of objects (Section 3.12).

The result of this chapter about perception and multi-object tracking forms the basis for the subsequent chapters about collision risk calculation (Chapter 4), traffic scenario prediction (Chapter 5) and planning and simulation (Chapter 6).

4 Collision Risk Calculation

The content of this chapter has been published by the author in advance at the International Conference on Robotics and Automation (ICRA) 2019 together with Prof. Dr. Daniel Göhring as advisor.

4.1 Motivation and Related Work

Estimating the risk of a future collision is essentially for autonomous driving [9] as well as for driver assistance systems [95]. In both cases, systems in the car have to perceive the environment, identify and track the relevant objects and to avoid collisions with them. Collision risk includes strictly speaking not only the probability of collision, but also the expected costs. This chapter will concentrate on the probability estimation.

Collisions occur, when the ego vehicle and an obstacle are at the same position at some time in the future. In classical tracking applications, as air traffic control, objects are usually considered as points or circles. This is reasonable since the regular distances between objects are very large compared to their extension due to the high velocities. In ground traffic situations, especially in urban environments, the distances between objects are much shorter and they have to be handled more realistically as extended objects. A common approach is to model traffic participants as oriented rectangles. Two rectangles are in a collision state if they at least partially overlap.

The relevant objects in the environment may be static or dynamic. Avoiding collisions with static objects is straight forward, despite the case of very noisy sensors. Dynamic objects are much harder to handle since their movement in the future has to be predicted. Usually, the future states of the objects are predicted iteratively in steps of 0.1 or 0.2 seconds. With each step, the uncertainty of the state increases, depending on the selected motion model and the corresponding process noise. Since it is impossible to predict the point in time with the highest collision risk, the risk must be computed for every time step.

In many systems, instead of the collision risk, the time to collision (TTC) is estimated deterministically. Some similar measures as time to react (TTR) or time to brake (TTB) are

also used [115] [95]. None of these measures consider the variances of the calculated time, nor the probability of colliding at all.

There are two different approaches for the probabilistic collision risk calculation:

- Collision state probability (CSP): probability of spatial overlap of two objects at a certain point in time [110] [52] [95]. All existing approaches use Monte Carlo Simulation (MCS) to calculate the probability.

- Collision event probability (CEP) density: probability density of a collision event at a certain point in time [6] [139]. While [139] also uses MCS, [6] presents a analytic solution, but does not consider extended obstacles.

The CEP, which is the probability of a collision during a period of time, is calculated by integrating the CEP density over time.

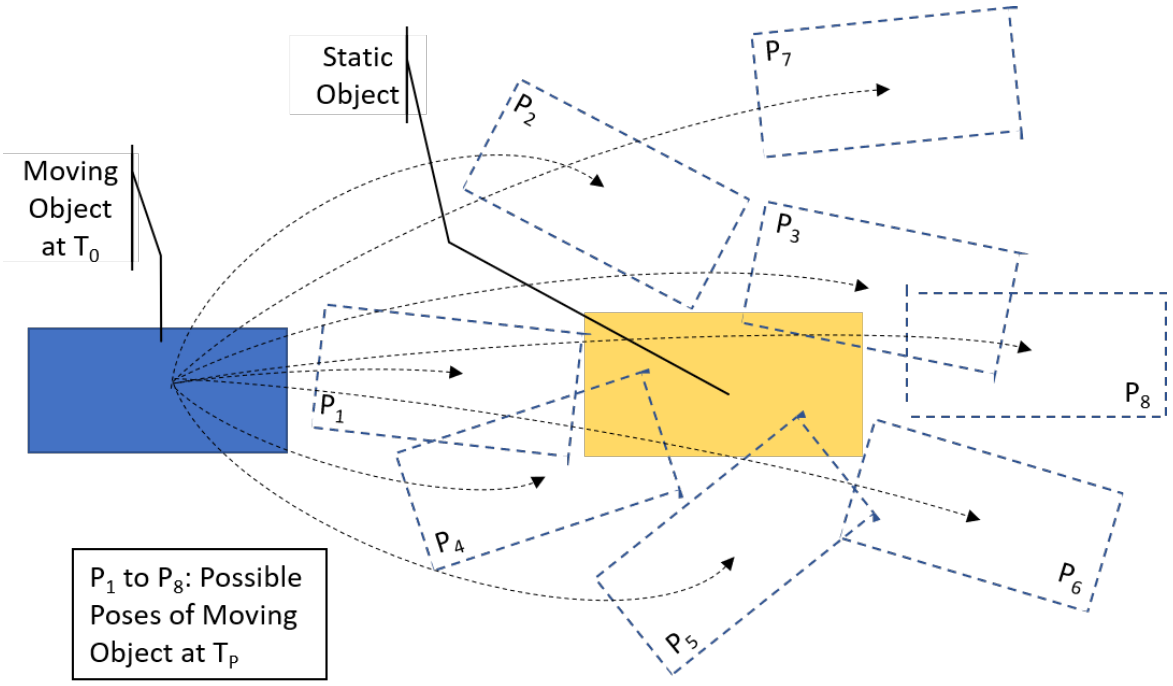


Fig. 4.1 Samples of predicted poses of moving object. P1 to P6 are examples for collision states, while only P1 and P2 represent collision events.

Fig.4.1 shows examples of collision states and events. In this case, a moving object approaches a static object. The probabilistic state of the moving object at time T_0 is provided by some tracking module. The prediction of future poses is further disturbed by process noise. The figure shows 8 samples of possible poses at prediction time T_p. Poses 1 and 2 represent collision events: The boundary of the moving object is just penetrating the boundary of the static object. Pose 6 is not a collision event since it is not a boundary crossing from outside

to inside. Poses 1-6 represent collision states: the object rectangles overlap at least partially. Poses 7 and 8 are neither collision events nor states.

Section 4.2 presents the general approach for CSP and CEP calculation. In Section 4.3, possible solutions using Monte Carlo Simulations are shown. Analytic methods to compute CSP and CEP in real-time are presented in Section 4.4. Evaluation results are shown in Section 4.5 and a summary and conclusion Section 4.6 closes the chapter.

4.2 General Solution

In the general case, the future 2D poses of the two objects are uncertain (see Fig. 4.2).

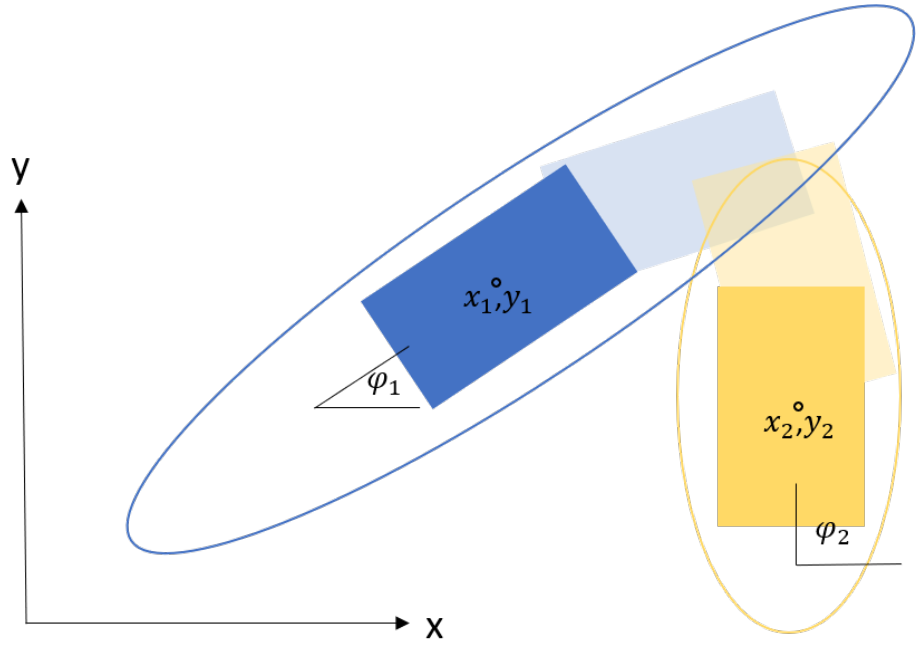


Fig. 4.2 Overlap uncertainty of 2 oriented rectangles at point in time. The ovals symbolize the uncertainty about the poses of the objects.

To compute the collision risk, one has to solve the integral over their common state distribution (4.1). It is assumed that the states of the two objects are independent of each other.

$$\int_{x_1}\int_{y_1}\int_{\varphi_1}\int_{x_2}\int_{y_2}\int_{\varphi_2} I_C(x_1,y_1,\varphi_1,x_2,y_2,\varphi_2) \; p(x_1,y_1,\varphi_1)\;\; p(x_2,y_2,\varphi_2)\;\; \mathrm{d}x_1\mathrm{d}y_1\mathrm{d}\varphi_1\mathrm{d}x_2\mathrm{d}y_2 d\varphi_2 \quad (4.1)$$

The indicator function (4.2) yields 1, if the two rectangles overlap at least partially. This indicator function is the reason, why the integral in the general case cannot be solved analytically.

$$I_C(x_1, y_1, \varphi_1, x_2, y_2, \varphi_2) = 1 \quad if \; S(x_1, y_1, \varphi_1) \cap S(x_2, y_2, \varphi_2) \neq 0 \tag{4.2}$$

Since the predicted states of an object at different points in time are not independent of each other, the collision risk cannot be accumulated over time to get the total collision risk over a time span [157]. Instead, the common distribution of the two objects over all time steps of the prediction horizon would have to be evaluated, which is of course intractable.

4.3 Solution Using Monte Carlo Simulation

CSP and CEP calculation can be achieved by Monte Carlo Simulation. These simulations usually not fulfill the time requirements of real driving solutions, but they are useful to provide ground truth for the evaluation of more efficient analytic solutions.

For the CSP calculation, the trajectories of both objects are predicted using some suitable process model. Both, the state and the covariances have to be forwarded. At each time step a number of samples are taken from the state distributions of both objects and checked, whether the rectangles overlap. The proportion of overlapping cases of all samples is the CSP at that point in time.

Calculation of the CEP requires drawing a number of complete trajectory samples. At simulation start, samples from the initial distribution are drawn to initialize each trajectory. At each time step, these trajectories are predicted forward using a suitable process model, and afterwards samples from the process noise distributions are drawn and added to the state. At each time step, the part of newly collided trajectories represents the CEP density. These trajectories are removed before applying the next prediction step. The proportion of collided trajectories over a time span is the CEP for that time span.

As the number of MC samples increases, the result converges to the true probability value due to the strong law of large numbers [9]. The number of samples required for realistic results depends mainly on the number of probabilistic state variables. But it has also to be observed that small risk values require more samples than higher ones [110].

4.4 Analytic Solution

To achieve an analytic solution for autonomous driving, it is assumed that the state vectors and the covariance matrices of two vehicles are given in the same coordinate system. The relative state is then given as the difference of the states, while the covariance matrices may simply be added in first approximation. The common state and covariance is subsequently transformed into a coordinate system with origin in the centroid of one of the two vehicles.

The proposed solution requires that the relative orientation of both vehicles is deterministic. Given the fact that cars normally follow the drive spline of the road, the uncertainty of the relative orientation is usually small compared to the uncertainty of the position and the velocity.

4.4.1 Collision octagon

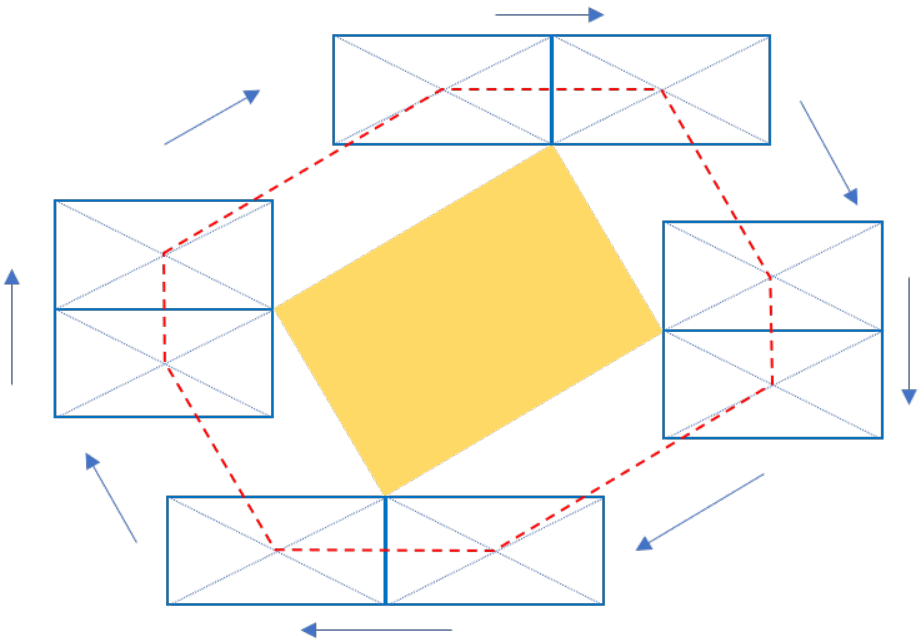


Fig. 4.3 Obstacle rectangle moving around the ego vehicle. In red the resulting collision octagon.

Both types of risk calculation can be simplified by transforming the problem of collision between two oriented rectangles into the collision between a point and a collision octagon. The collision octagon is the result of the convolution of the two oriented rectangles (see Fig. 4.3).

The rectangle of the ego vehicle is replaced by the collision octagon, which depends on the centroid position of the ego vehicle and the relative orientation angle as well as the length and width of both objects. The collision octagon is the trace of the obstacle centroid when the obstacle rectangle is moved around the border of the ego rectangle. The corners of the collision octagon can be computed by simple vector arithmetic.

The CSP at a certain point in time is then the probability that the centroid of the obstacle is somewhere inside the octagon.

The CEP density is the probability density that the obstacle centroid crosses one of the eight edges of the octagon from outside to inside.

4.4.2 Collision State Probability Calculation

The obstacle is placed at the center of the coordinate system and its length is aligned with the x axis. Therefore, the expected x and y - values are zero. Using a Free Motion Model (see Subsection 3.8.1) the prediction of the lateral and longitudinal motion of the vehicles is independent of each other, which results in a correlation coefficient of zero between x and y position in the common covariance matrix. A Gaussian distribution of the uncertain relative position is assumed. The integral (4.1) simplifies to (4.3):

$$P(C,T_p) = \int_y \int_x I_C(x,y) \ \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})} \ \mathrm{d}x\mathrm{d}y \tag{4.3}$$

This integral is the probability mass of the bivariate Gaussian integrated over the area of the octagon at time T_p. This is split into three integrals each for the upper and lower bounding edges of the octagon, the integrals for the two vertical edges are zero. Other solutions are possible. See Fig. 4.4 for the integration boundaries of the six integrals.

The integrals have a variable integration limit for y, given by the straight line equation $y = mx + b$ of the three upper and three lower octagon edges (4.4).

$$P(C_i,T_p) = \int_{y=0}^{y=m_i x+b_i} \int_{x=x_{l,i}}^{x=x_{u,i}} \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})} \mathrm{d}x\mathrm{d}y \tag{4.4}$$

Due to the variable boundaries of the outer integral, only the first part of the integration can be solved analytically.

$$P(C_i,T_p) = \frac{1}{\sqrt{8\pi}\sigma_x} \int_{x=x_{l,i}}^{x=x_{u,i}} \mathrm{erf}(\frac{m_i x + b_i}{\sqrt{2}\sigma_x}) e^{-\frac{1}{2}\frac{x^2}{\sigma_x^2}} \ \mathrm{d}x \tag{4.5}$$

The remaining integral (4.5) over a smooth continuous function with constant integration limits can be evaluated very efficiently using numerical integration (Example:

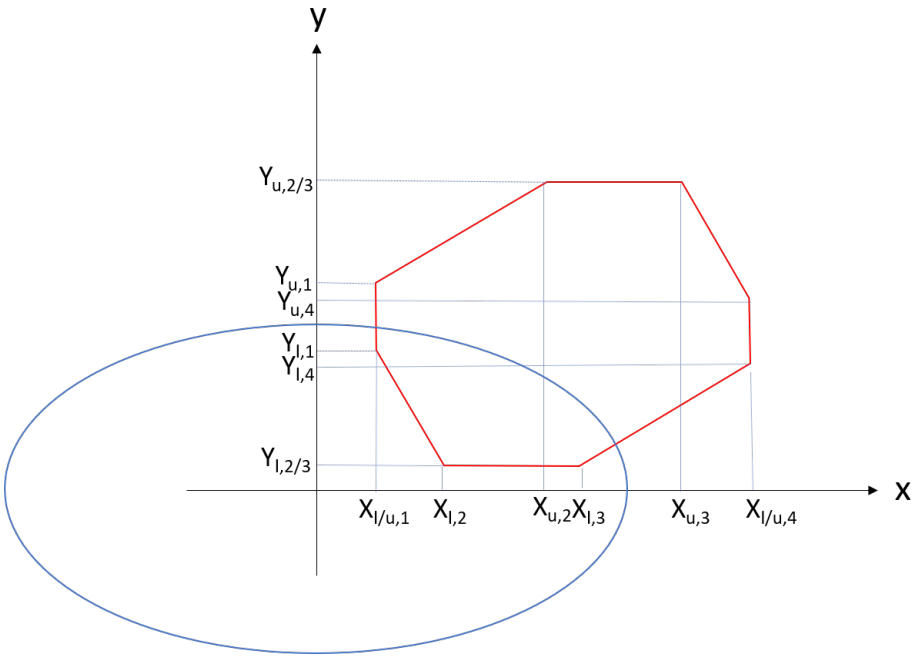


Fig. 4.4 Upper and lower integration boundaries. Collision octagon is shown in red, 1-Sigma ellipsis of Gaussian state distribution of obstacle in blue.

$gsl\_integration(\cdot)$ from the GNU Scientific Library (GSL) [72]) . The sum of the six integrals gives the CSP at time T_p (4.6).

$$P(C,T_p) = \sum_{i=0}^{i<3} P(C_i,T_p) - \sum_{i=3}^{i<6} P(C_i,T_p) \tag{4.6}$$

4.4.3 Collision Event Probability Calculation

This calculation method has recently been presented in [6], where it is used to calculate the CEP density between a rectangle representing the ego vehicle and a point obstacle. A short summary of the approach is presented here and it is extended to the CEP between two rectangles using the collision octagon.

The approach is based on the probability rate for boundary crossings of stochastic vector processes [26] . The centroid of the ego vehicle is placed in the origin of the coordinate system. Each edge of the octagon is treated as a boundary and the probability rate of the obstacle centroid penetrating this boundary is computed (see Fig. 4.5).

The integral (4.7) is derived in detail in [6]. The result is the probability density of crossing one edge of the collision octagon at time T_p.

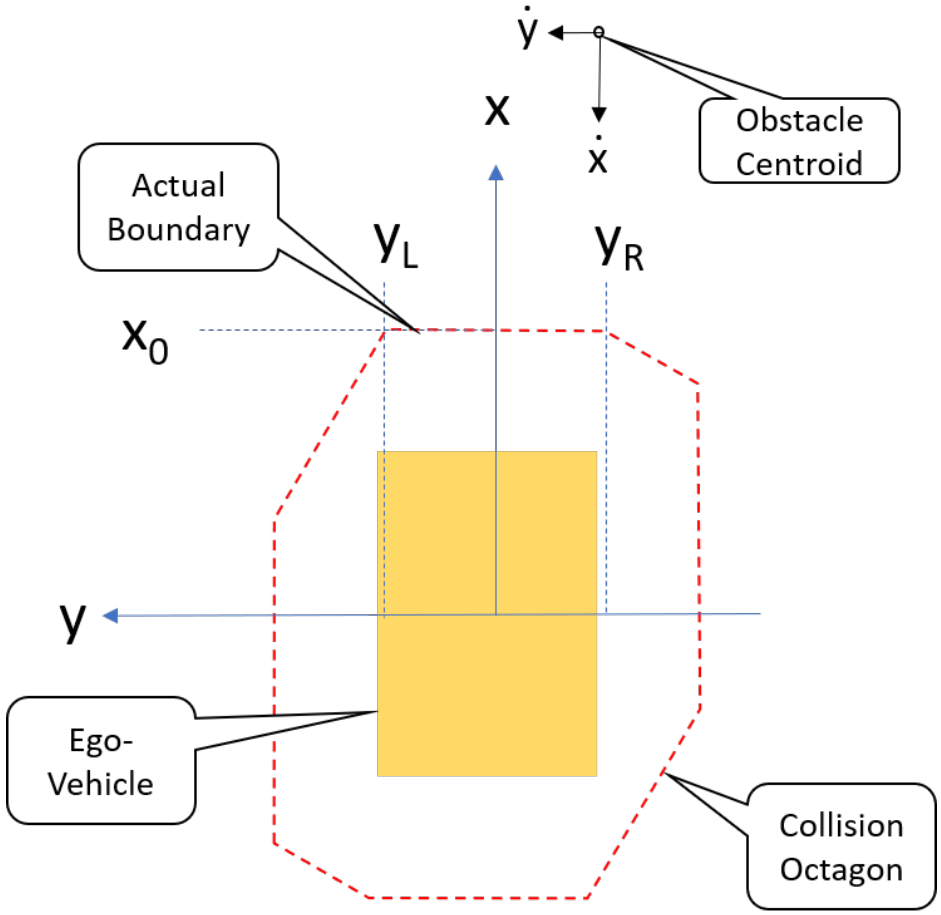


Fig. 4.5 Boundary crossing probability for one edge of the collision octagon. The moving centroid of the obstacle may cross the upper edge of the collision octagon.

$$\frac{dP(C_i,T_p)}{dt} = -p_{T_p}(x_0)\int_{\dot{x}\leq 0}\int_{y\in I_y} \dot{x}\ \ p_{T_p}(\dot{x},y|x_0)\ \ \mathrm{d}\dot{x}\mathrm{d}y \tag{4.7}$$

The integral depends on the probability of the obstacle being at the edge of the octagon $p(x_0)$ at time T_p, the relative x-velocity between the obstacles and the ego vehicle $\dot{x}$, and the common distribution of this velocity and the y-position of the obstacle conditioned on the x-position of the boundary $p(\dot{x},y|x_0)$. Integration limits are all negative velocities (directed to the inside of the octagon) and the y-limits of the octagon edge.

This integral can be computed approximately by usage of a Taylor expansion for the off-diagonal element of the inverse covariance matrix. Evaluated to zeros order this yields (4.8)

$$\frac{dP(C_i, Tp_)}{dt} =$$
$$-\mathcal{N}(x_0;\mu_x,\sigma_x)(\mu_{\dot{x}|x_0}\Phi(\frac{-\mu_{\dot{x}|x_0}}{\tilde{\sigma}_{\dot{x}|x_0}}) - \tilde{\sigma}^2_{\dot{x}|x_0}\mathcal{N}(0;\mu_{\dot{x}|x_0},\tilde{\sigma}_{\dot{x}|x_0}))(\Phi(\frac{y_R-\mu_{y|x_0}}{\tilde{\sigma}_{y|x_0}}) - \Phi(\frac{y_L-\mu_{y|x_0}}{\tilde{\sigma}_{y|x_0}})) \tag{4.8}$$

$\mu_{\dot{x}|x_0}$ and $\mu_{y|x_0}$ are the expected relative x-velocity and y-position of the obstacle conditioned on the x-position of the boundary. $\tilde{\sigma}_{\dot{x}|x_0}$ and $\tilde{\sigma}_{y|x_0}$ are the corresponding standard deviations. $\Phi(\cdot)$ denotes the standard normal cumulative distribution function.

This calculation may be refined by higher order terms of the Taylor expansion.

To get the total CEP density, the rates of all eight edges of the collision octagon (4.9) must be added. For this purpose, the coordinate system has to be rotated, so that the respective edges of the octagon become orthogonal to the x-axis.

$$\frac{\mathrm{d}P(C,T_p)}{\mathrm{d}t} = \sum_{i=0}^{i<8} \frac{\mathrm{d}P(C_i,T_p)}{\mathrm{d}t} \tag{4.9}$$

The accumulation of the event rates over a time span T_1 to T_2 amounts to the CEP over that time span (4.10)

$$P(C,T_1,T_2) = \sum_{t=T_1}^{t<T_2} \Delta t \frac{\mathrm{d}P(C,t)}{\mathrm{d}t} \tag{4.10}$$

4.5 Evaluation

Evaluation of the collision risk algorithms is done in two parts. In the first part, a hypothetical scenario is used. The two analytic algorithms are compared with the corresponding MC implementation and it is shown that only the analytic solutions are real-time capable. Furthermore, the CSP is contrasted with the CEP and the reasons for their deviations are given.

In the second part, a real-world example recorded with the MadeInGermany test vehicle is presented, where the algorithms are embedded into a complete tracking and prediction pipeline and the collision risks are computed for the planned trajectory of the ego vehicle.

4.5.1 Simulated Scenarios

The following scenario is considered: The position of the ego vehicle is in the coordinate origin, while the obstacle approaches the ego vehicle from diagonally in front. The prediction is done using a simple constant acceleration model.

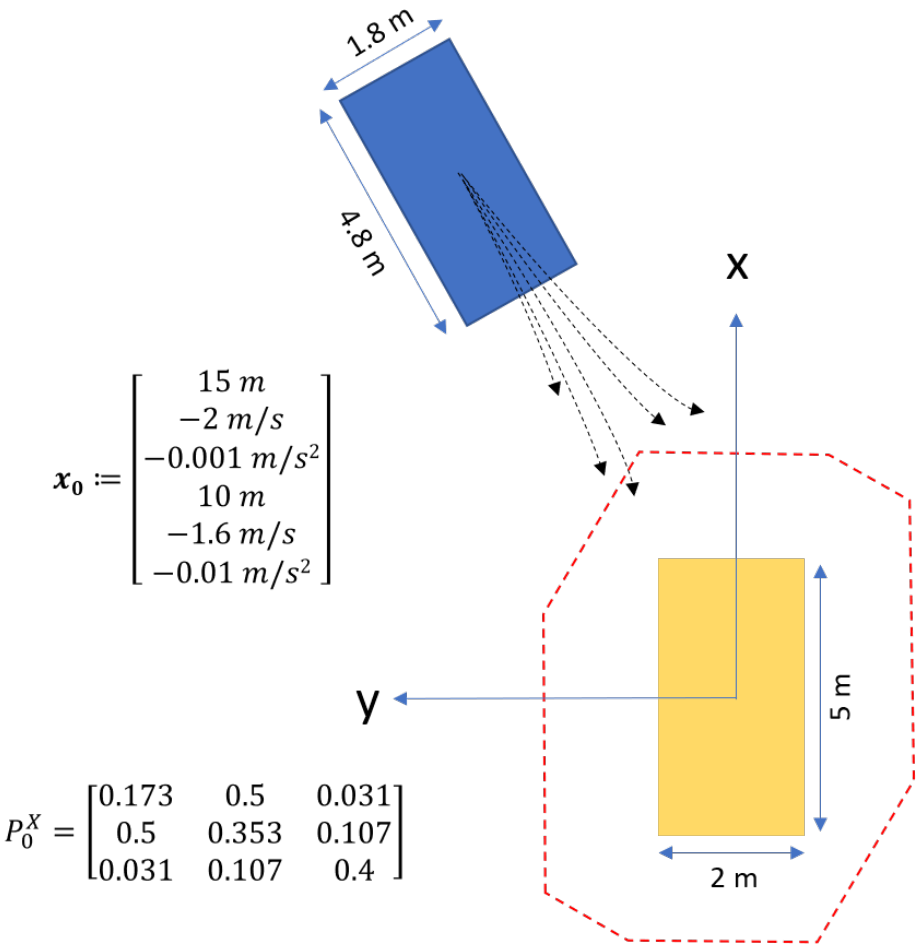


Fig. 4.6 Simulated collision scenario: initial obstacle state and random trajectories. Assumed process noise $q_x = q_y = 0.1\frac{m^2}{s^5}$. Initial state covariance matrix is shown for X dimension, Y variances are identically.

Fig. 4.7 shows the CEP density calculated by the analytic solution and the MC simulations with $10^3, 10^4, 10^5$ and 10^6 runs. It has its peek at 3.5 seconds with a density of $\approx 0.1/s$. While the Monte Carlo simulations with 10^3 and 10^4 samples show significant deviations from the analytic solution, the MCS results for 10^5 and 10^6 samples converge strongly. The MC simulation with 10^6 samples can be taken as ground truth and the good coincidence with

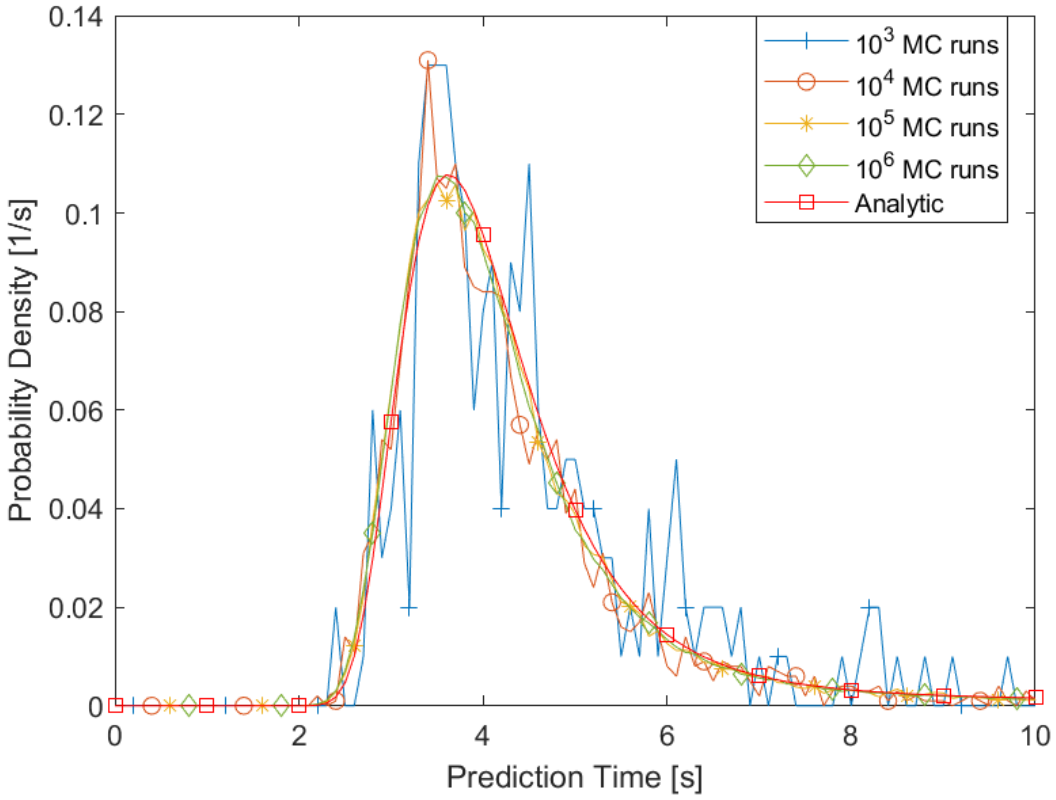


Fig. 4.7 CEP density over time: Comparison of analytical solution with Monte Carlo Simulations.

the analytic results shows that the Taylor expansion of the CEP calculation is sufficiently accurate.

Fig. 4.8 shows the CEP, which is the accumulated event rate, over time. After 6 seconds, it has almost reached its saturation level of ≈ 22 %. This relatively low risk results from the high variances in all directions, which makes it probable that the obstacle passes the ego-vehicle on the left or right side. Again, MCS results for 100,000 and 1,000,000 runs are very close.

Fig. 4.9 shows the CSP over time. It reaches its peak at ≈ 4.4 seconds and then decreases again. The reason for this decrease of the probability is that states of the obstacle behind the ego vehicle (at $x \ll 0$) are not collision states, even though the obstacle must have passed through the ego vehicle before. Again, MCS results for 10^5 and 10^6 runs are satisfactory. The good coincidence between the results of 10^6 MC runs and the analytic solution proves the sufficient accuracy of the numerical integration part in the CSP calculation.

Fig. 4.10 (solid lines) shows the analytic state probability, event probability and event rate in one diagram. The state probability reaches only about the half of the event probability before it decreases again. This is due to the high variances in the initial state and the added

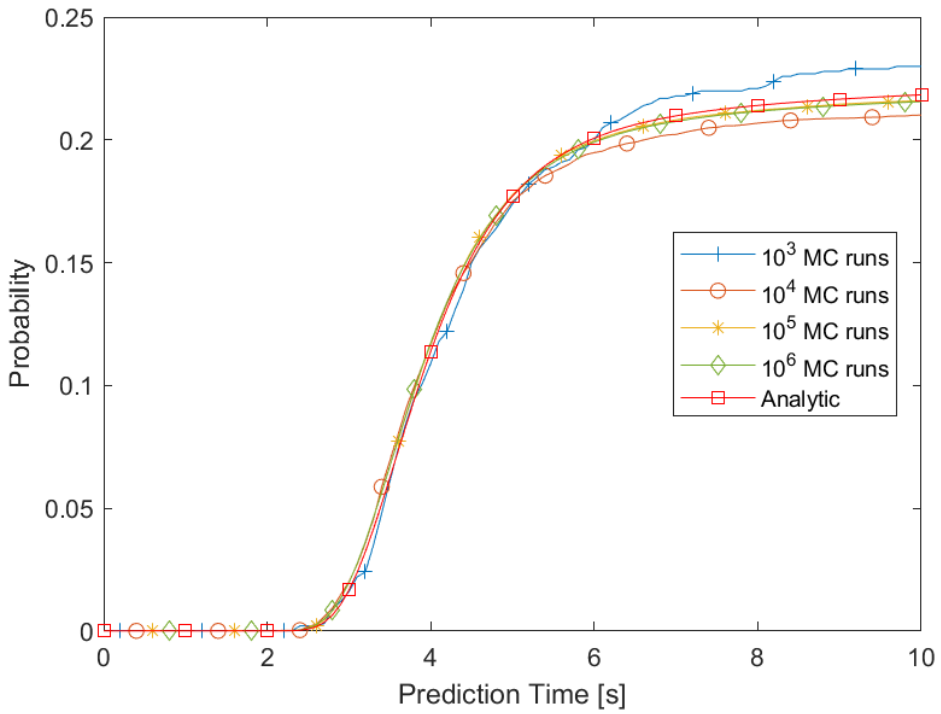


Fig. 4.8 CEP over time: Comparison of analytical solution with Monte Carlo Simulations.

process noise. The dotted lines show the results for the same scenario, but with reduced initial variances and process noise. In this case, the resulting collision probability is much higher and the deviation between the maximal CSP and CEP is strongly reduced.

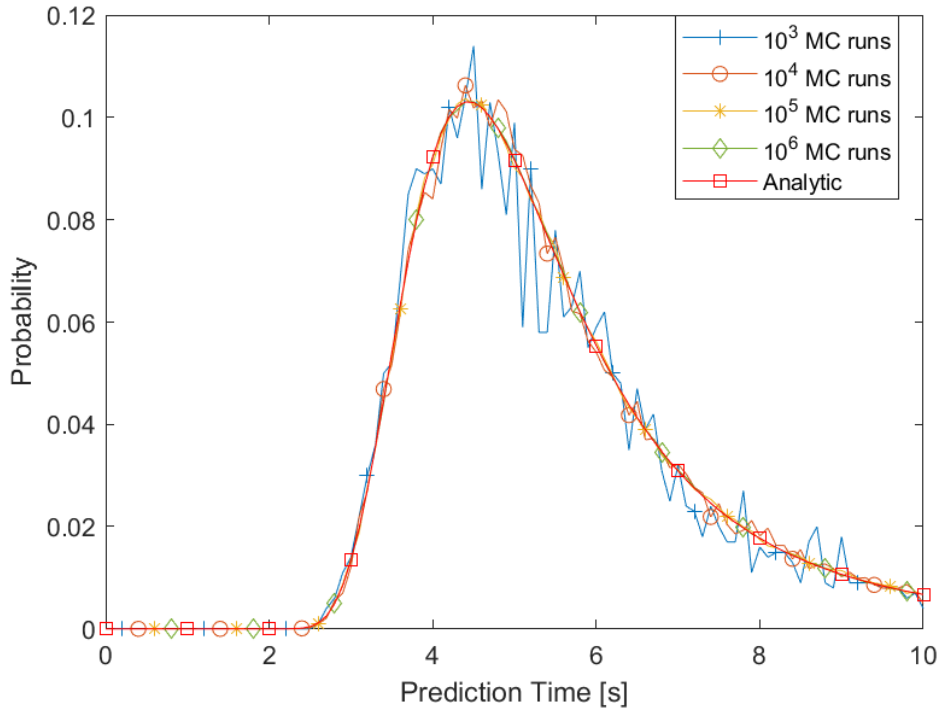


Fig. 4.9 CSP over time: Comparison of analytical solution with Monte Carlo Simulations.

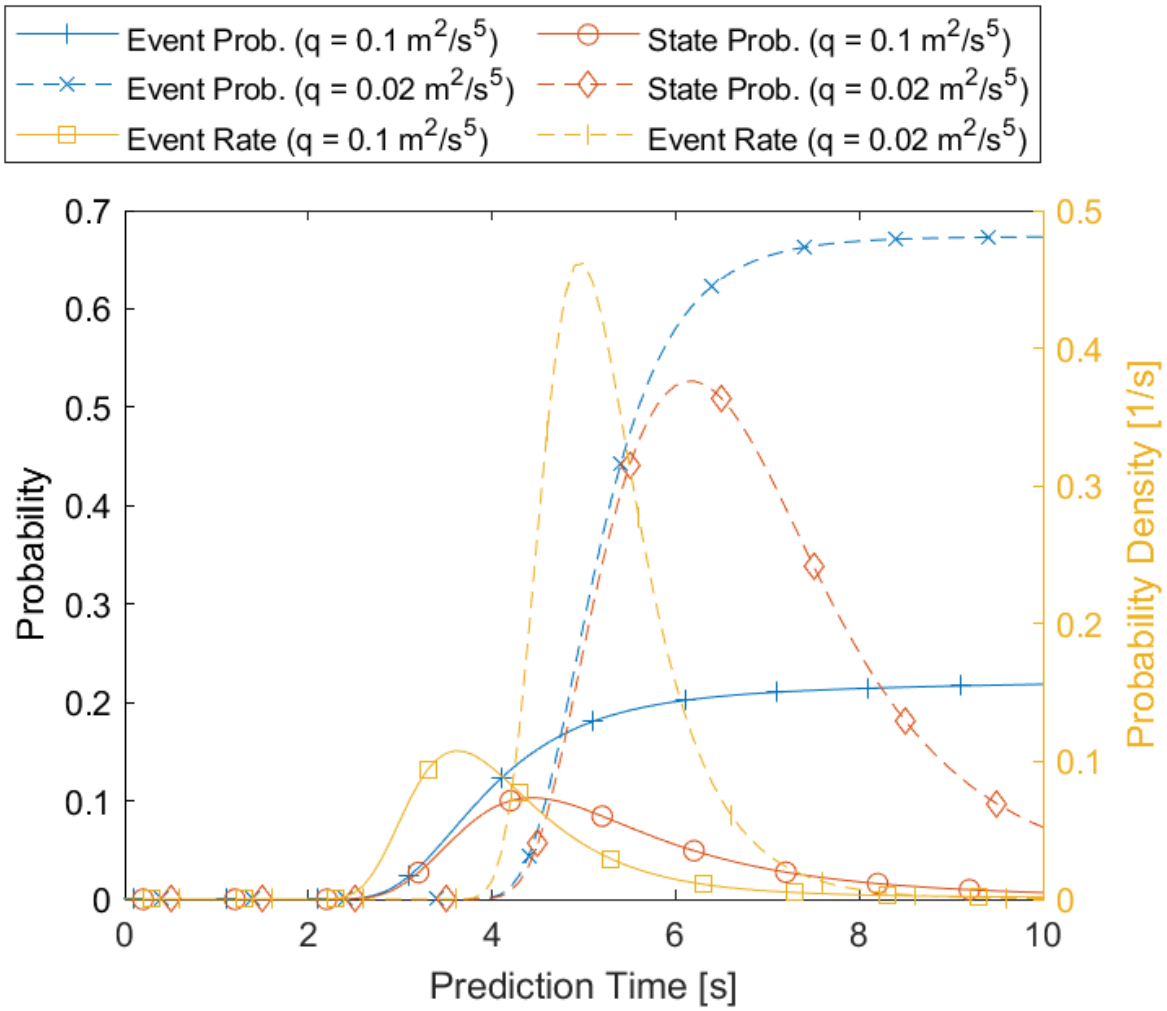


Fig. 4.10 CSP, CEP and CEP density over time for the same scenario with different process noise rates.

4.5.2 Real-World Scenario

Fig. 4.11 shows a real-world scenario recorded during a test drive with the MadeInGermany self-driving car in front of the Freie Universität in the Thielallee in Berlin-Dahlem. The planner of ego-vehicle prepares a left turn, while from the right side approaches an obstacle (no. 38), which has priority. The Gaussian trajectory prediction module assumes that vehicle no. 38 tries to keep the middle of its lane with low variance and will continue in this case with nearly constant velocity.

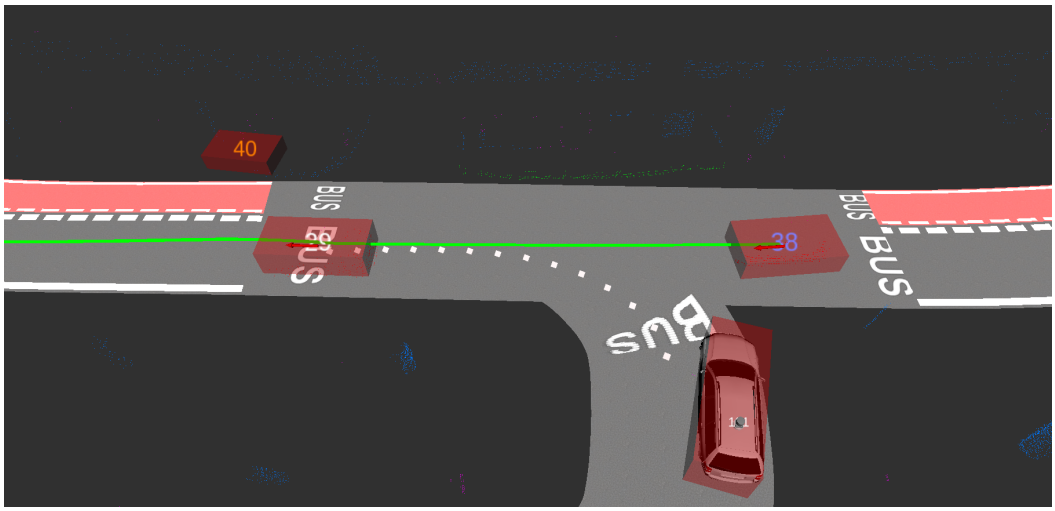


Fig. 4.11 Real-world collision scenario: planned trajectory of ego-vehicle (white dotted line) and predicted trajectory of obstacle (green line).

Fig. 4.12 shows the same scenario as seen by the front mounted fish-eye camera of MadeInGermany.

Fig. 4.13 show the resulting CEP density, CEP and CSP. The event rate has a sharp peak of 2.5/s at about 1.7 seconds. The CEP increases very fast to more than 0.9, as does the CSP. This shows that a collision is unavoidable, if the ego-vehicle would continue with constant velocity.

Fig. 4.12 Front camera fish eye view of potential collision scenario. The oncoming cars have the priority.

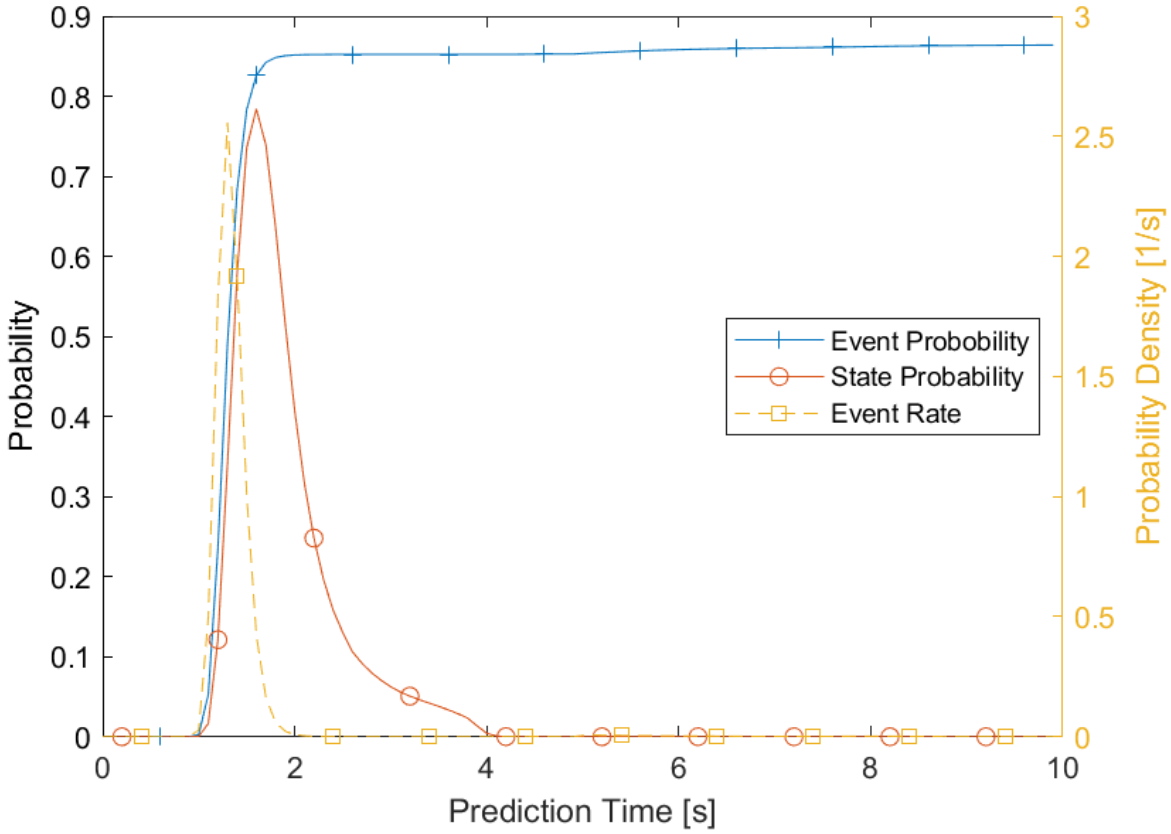


Fig. 4.13 CSP, CEP and CEP density over time (real-world scenario).

4.5.3 Timing Evaluation

Using the simulated scenario, the following performance values have been measured using an Intel i7-8750H (see Table 4.1). All figures are for the calculation of a 10 seconds predicted trajectory (100 positions). The Monte Carlo simulations include 10^4 runs.

Evaluation results (ms)			
	MCS	Analytic	Factor
CSP	95,6 ms	0,9 ms	106
CEP	266,3 ms	0,3 ms	887

Table 4.1 Result of timing evaluation.

4.6 Summary and Conclusion

In this chapter, two analytic solutions for computation of the CSP and CEP have been presented. Both solutions have shown to provide numerically accurate results by comparing them to large scale Monte Carlo simulations. Their efficiency is about 100 - 800 times better than the MC simulations. The analytic calculation for oriented rectangles is supported by the concept of the collision octagon.

Comparing both approaches yields to the following results:

- CSP calculation needs only position data and variance as input, while CEP needs position, velocity and the corresponding covariance matrix.

- CSP may be calculated for a single trajectory point, while CEP is only meaningful, when evaluated for whole trajectories.

- CEP calculation results in a probability density function and a cumulative distribution function. CSP calculates probability masses at discrete points in time, but these form no valid probability mass function since they are not independent of each other.

- If variances are low, the maximal CSP probability is close to the accumulated event probability.

- CEP calculation is about 3 times faster than CSP.

Both solutions have successfully been integrated into the self-driving car system MadeInGermany of the Freie Universität Berlin. The differences between the two probability approaches are low when applied to a real-world driving situation due to the lower variances.

This chapter has proven thesis 3: The calculation of the future collision risk between two rectangular moving objects based on an analytic algorithm can be efficient enough to

check several thousand trajectory pairs per second. These results form the basis for the interaction-aware traffic scenario prediction presented in the next chapter.

5 Traffic Scenario Prediction

5.1 Motivation and Problem Description

In order to plan a safe and comfortable trajectory for an autonomous vehicle, it is necessary to predict the future evolution of the current traffic scenario for a sufficiently long time period. A normal lane change maneuver takes 5-7 seconds. It takes about the same time to pass a multi-lane intersection after having stopped. Overtaking a truck may take 20 seconds or more. All these maneuvers can be completed successfully only, if there are no conflicts with other traffic participants during the execution of the maneuver.

Maneuvers in an urban environment are more variant and complex than those on highways, where car-following and lane changes predominate. There are different types of intersections with a variant of priority rules. Deadlocks are possible, even if all participants obey the priority rules. Lane merges and diverges are more common and often there are not even lane markings on the road. In narrow roads, there is only one common lane for both driving directions. Curves, especially when turning right, are often so tight that longer vehicles, like trucks and buses, cannot take the curve without swinging into adjacent lanes. Special care has to be taken for vulnerable traffic participants, like pedestrians and bicycles.

A traffic scenario and its elements are defined in [177] as follows:

- Scenario
  - Situation at t_0
  - Situation at t_1
  - ...
  - Situation at t_n
    - State of ego vehicle
    - Goal of ego vehicle
    - Scene

- Dynamic Obstacles
- Scenery

  Static Obstacles

  Road Infrastructure

The scenery combines the road infrastructure and any static obstacles. A scene includes the scenery and all relevant dynamic obstacles. A situation consists of the scene, the state of the ego-vehicle and its goal. A scenario is the evolution of a situation over time.

The following core and auxiliary tasks of driver behavior modeling are defined in [35]:

Driver Modeling Tasks	
Core Tasks	Auxiliary Tasks
State Estimation	Risk Estimation
Intention Estimation	Anomaly Detection
Trait Estimation	Behavior Imitation
Motion Prediction	Traffic Simulation

Table 5.1 List of Driver Modeling Tasks.

- The state estimate has to infer the physical state of the objects in the environment from the measurements received. The result of the state estimate is usually a list reporting the position, velocity, size etc. of the obstacles. In a stochastic system, the covariances of the estimated values are also reported.

- The intention estimate reasons about the intentions of the drivers in the environment. It should answer the question what the drivers intend to do. The intention defines the spatial route. Since the intention is uncertain, the result of the estimate is a probability distribution over a discrete number of possible intentions.

- The trait estimate tries to evaluate the driving style of a driver. These are driving parameters, like preferred acceleration and braking rates, minimum headway, desired velocity etc. Evaluating these parameters online for each individual driver is difficult due to the typical short observation periods and is therefore seldom accomplished.

- The result of the motion prediction is the predicted trajectory. It answers the question how the driver intends to do it. The trajectory is an enumeration of future positions in discrete time steps, spanning the maximal prediction horizon. In a stochastic system, the covariances of the predicted states must also be given. The simplest trajectory prediction is based on the current state and the physical laws. Maneuver based motion

prediction is conditioned on the intention and usually also on the driving style. In this case, a separate trajectory has to be predicted for each maneuver. Interaction-aware motion prediction also considers the influences of the dynamic objects to each other,

- The risk estimation tries to predict potential collisions based on the motion prediction for each vehicle. Risk estimation is a prerequisite for interaction-aware motion planning. Risk estimation can be based on heuristics or calculated as collision probability. In the latter case, the motion prediction must include the covariances of the predicted states.

- Anomaly detection tries to identify potentially dangerous situations. Anomalies are usually recognized, if the observed motion of some vehicle deviates significantly from the predicted motion. Anomaly detection can be performed as replacement for risk estimation or as additional task.

- Behavior imitation tries to learn the motion behavior from examples of human drivers. It's mainly related to motion planning, which is a variant of the motion prediction.

- Microscopic traffic simulation is often used to test the whole prediction and planning framework of autonomous vehicles. It can be achieved for example by taking the most probable predicted motion behavior of each dynamic object as a motion plan and evaluate the resulting configurations.

The ego vehicle may play different roles in a scenario prediction. We have to distinguish three use cases:

- ADAS (Advanced Driver Assistance System) - The intention and the planned motion of the ego-vehicle are uncertain and must be predicted. The ego vehicle is just one of the dynamic objects, but for which more precise data is available. If the risk estimation or anomaly detection recognizes a dangerous situation, the ADAS should warn the driver or intervene in the vehicle control.

- Autonomous vehicle with separate motion planning - In case of an autonomous vehicle with separate planning and prediction systems, an interaction-aware prediction system must consider the planned trajectory of the ego vehicle to be able to predict the reactions of the other agents. The motion planning system must then evaluate the predicted risks and eventually adapt the planned trajectory.

- Integrated prediction and planning - The prediction system may take the global path intention of the ego vehicle's route planner as input and create a predicted trajectory for

the ego-vehicle, which is compatible with the predicted maneuvers of the other traffic participants. In this case, the task of maneuver prediction and high-level behavior planning for the ego-vehicle coincide and the planning stage concentrates on finding an optimal control policy.

The rest of this chapter is organized as follows: Section 5.2 gives an overview of the literature about the theme. Section 5.3 contains an outline of the proposed system. The Sections 5.4 to 5.9 present the details of the system . The evaluations in Section 5.10 and the summary and conclusion Section 5.11 complete this chapter.

5.2 Related Work

There is a lot of literature about the prediction of traffic scenarios. A definition of scenery, scene, situation and scenario is given in [177] and [142]. An overview over physics based, maneuver based and interaction-aware approaches is given in [115]. A very recent survey over 200 papers about traffic prediction is in [35].

An early work concerning prediction of driver behavior for an ADDS is [43]. The authors propose a Dynamic Belief Network (DBN) to infer the motivation and intention of drivers on a highway. The network is based on car following and lane change models and consists of nodes describing the context and the state of the target vehicle. All continuous state variables are discretized. In [75], this approach is extended by introducing continuous state variables to predict trajectories. They use a particle filter as inference engine. The authors of [2] and [3] build on the former works but reduce the computation time drastically by applying a deterministic inference method called augmented switching linear dynamical system (ASLDS). The noise parameters of the system are learned using the Expectation-Maximization (EM) algorithm.

In [114] and [112], the authors propose to use a Bayesian network to predict the driver's maneuver intention at road intersections. They combine information from the roadmap about the intersection layout with dynamic state data of the observed vehicles. In [113] they extend their work to access the risk of a traffic situation by comparing the driver's probable intention with the expected behavior.

Many prediction approaches are based on the Intelligent Driver Model (IDM) [169] presented by Treiber and Kesting. It calculates the acceleration of a vehicle in presence of a preceding car in single lane traffic and was initially developed to simulate and analyze traffic flow. The heuristic formula for car-following behavior turned out so realistic that it was adopted by many other authors. Subsequent publications enhanced the model for delays,

inaccuracies and anticipation [174], lane changes [105], traffic light approaching [172] and stochasticity [173].

Liebner et al. [122] where among the first to utilize the IDM to predict the driver's intention in an urban traffic scenario. They create one predicted trajectory for each possible driver-intention using the IDM formula and use a Bayesian network to infer the probability distribution of the driver-intentions.

Damerow, Eggert et al. extended the IDM to the Forsighted Driver Model (FDM) [55]. It uses predictive risk maps [54] to quantify the risk of the predicted trajectories and to make the prediction interaction-aware. They extend their method by a RTT* based behavior planning approach [45], lane change planning [100] and overtakes [46]. In [56] they present FDM++ including a driver model based on survival theory.

Schulz et al. present in [160] an interaction-aware approach to predict the driver-behavior at an urban intersection. They propose a Bayesian network for the intention estimation and an extended IDM version to generate the corresponding trajectories. By evaluating the possible crossing sequences of conflict zones as pass and yield maneuvers, they consider the interaction among the vehicles. The probabilistic inference is implemented using a particle filter. In [161], they propose the usage of a Multiple Model Uncented Kalman Filter (MM-UKF) to overcome the performance problems of the particle filter. In [162], they replace the IDM based trajectory generation by a learning based approach using a deep neural network.

Schreier presents in [158] and [157] a system to predict the traffic scenario and to access the criticality of the current situation. The system is part of an ADAS and its purpose is to generate realistic warnings in potentially dangerous situations. The scenario is structured defining some reasonable standard maneuvers for the dynamic obstacles and the ego vehicle, like car following, lane changing or turning. All other observed behaviors are abstracted into a so-called trash maneuver. A Bayesian network calculates the probability distribution of the maneuvers. Gaussian processes are used to create a long-term prediction for each maneuver. Finally, the evolution of the probability distribution for a collision of the ego vehicle with a static or dynamic obstacle is evaluated using a particle filter.

While most systems rely on a detailed roadmap and some robust localization inside the roadmap using GPS, the authors of [73] infer the whole scene from short video sequences. They create a probabilistic model for the geometry of the road layout and the positions of the obstacles using scene flow, occupancy grid maps, vanishing points, semantic labels and vehicle tracklets.

The authors of [168] base their work on Gaussian Process Regression (GPR). They train the hyper parameters of the GPR using example trajectories at an intersection, which have previously been normalized temporally and spatially. They use the learned GP models to

calculate the probability of the maneuver intentions and to create a multi-modal trajectory prediction using a particle filter.

In [16], the authors use a game theoretic approach to infer an interactive scene prediction of highway traffic. They state that interactive planning and prediction are equivalent problems. They distinguish intention-based maneuver probability, which models the hidden intent of the drivers, and observation-based maneuver probability, which is derived from the incoming data of the perception system. The interaction-awareness is achieved by evaluating the collision risk between the traffic participants in form of the Time-To-Collision (TTC) and by taking account of the risk in the re-planning of the maneuver. In [15], they present a modified approach, suggesting a combination of model-based and learning-based prediction. They utilize a spatio-temporal cost map based on the multivariate normal-distributed states of the vehicles and some context features and assume that the future plans of the drivers are orientated on the negative gradient of the cost maps.

The authors of [186] have the focus on trajectory prediction at an intersection. They use short trajectory pieces, called snippets, as input. They propose a Gaussian Mixture Model (GMM) and based on the GMM a Variational Gaussian Mixture Model (VGMM) to predict the future trajectories. The VGMM is trained using a real-world dataset by applying Variational Bayesian Expectation-Maximization (VB-EM). In [187], they extend the input vector by categorial and binary features, as traffic light state, presence of a leading vehicle and lane configuration. For these extensions, they use a Bernoulli-Gaussian Mixture Model (BGMM), which is trained using EM. To cope with non-stationary data, they present an extension to the BGMM for online parameter estimation.

In [78], the authors develop a realistic driver model to predict the long-term evolution of highway traffic scenes. They propose Inverse Reinforcement Learning (IRL) to learn the cost function of the model, which is based on static and dynamic features of the traffic scene. In [80] they extend the approach by introducing a augmented Switching State-Space Model (aSSSM), which uses the IDM calculated acceleration as input. Based on this prediction framework, they propose in [79] to apply Partially Observable Monte-Carlo Planning (POMCP), an online variant of Partially Observable Markov Decision Process (POMDP), to create a decision-making system for autonomous driving on highways.

While the above approaches are mainly rule-based, more and more authors propose deep learning methods based on neural networks. [131] presents a comparison between rule-based and machine learning methods for lane change detection. In [127], the authors propose deep recurrent CNNs for lane change predictions on highways. They convert the camera input from the environment into a generic visual representation, which is fed into the network. The output of the network is a dynamic flow filter used to generate the visual representation of the

next time step. The authors of [24] use a CNN with a symmetric encoder-decoder architecture to predict the path of the ego-vehicle only based on sensor data in absence of a road-map. In [48] and [141], encoder-decoder architectures based on LSTM are proposed to predict lane changes. [135] proposes an extended LSTM, the convolutional-LSTM. The approach in [116] uses feature sets of the ego-vehicle, the surrounding vehicles and the road geometry to create an interactive scene prediction using deep neural networks. Different network architectures are compared to each other. [151] and [107] propose Generative Adversarial Networks (GAN) for the prediction of traffic scenes.

A very obvious feature for the prediction of lane changes and turn maneuvers is investigated by none of the above researchers: the turn indicators of the surrounding vehicles. In [70], features are extracted from camera images and a Fast Fourier Transform (FFT) is applied to the signal. An AdaBoost classificator is subsequently used to identify the turn signal in the frequency domain.

All of the above publications are limited to the prediction of vehicles. Several other authors aim at the prediction of pedestrian behavior. In [147], the influence factors of pedestrian behavior, as crossing patterns, non-verbal communication, environmental factors, age and temporal gap acceptance are examined without proposing a certain prediction method. [32] proposes an ADAS based on Bayesian networks for behavior prediction to prevent collisions with pedestrians. In [180], a data driven approach for pedestrian prediction is proposed and several neural network architectures are compared for their suitability for this purpose. The authors of [98] specifically analyze the situation at pedestrian crosswalks and use a Support Vector Machine (SVM) to predict pedestrian gap acceptance behavior.

5.3 System Overview

This and the following sections of this chapter describe the scenario prediction system, as it is implemented as part of the fub_roscar autonomous driving system. The rule-based multi-modal interaction-aware system models the various objects of a scenario and their hypothetical motion behavior.

Figure 5.1 provides an overview of the major elements of the system. The environment perception provides the input for the estimate of the current state of all obstacles. Based on the roadmap and the localization of the obstacles in the roadmap, the intention estimate for each obstacle is created and updated. It consists of the lane sequences of all currently feasible maneuvers of each obstacle. The center lines of the lanes are the bases of the predicted paths. In the next step, the motion prediction for each maneuver is created in form of a sequence of trajectory steps. The predicted states of the trajectory steps depend on map based

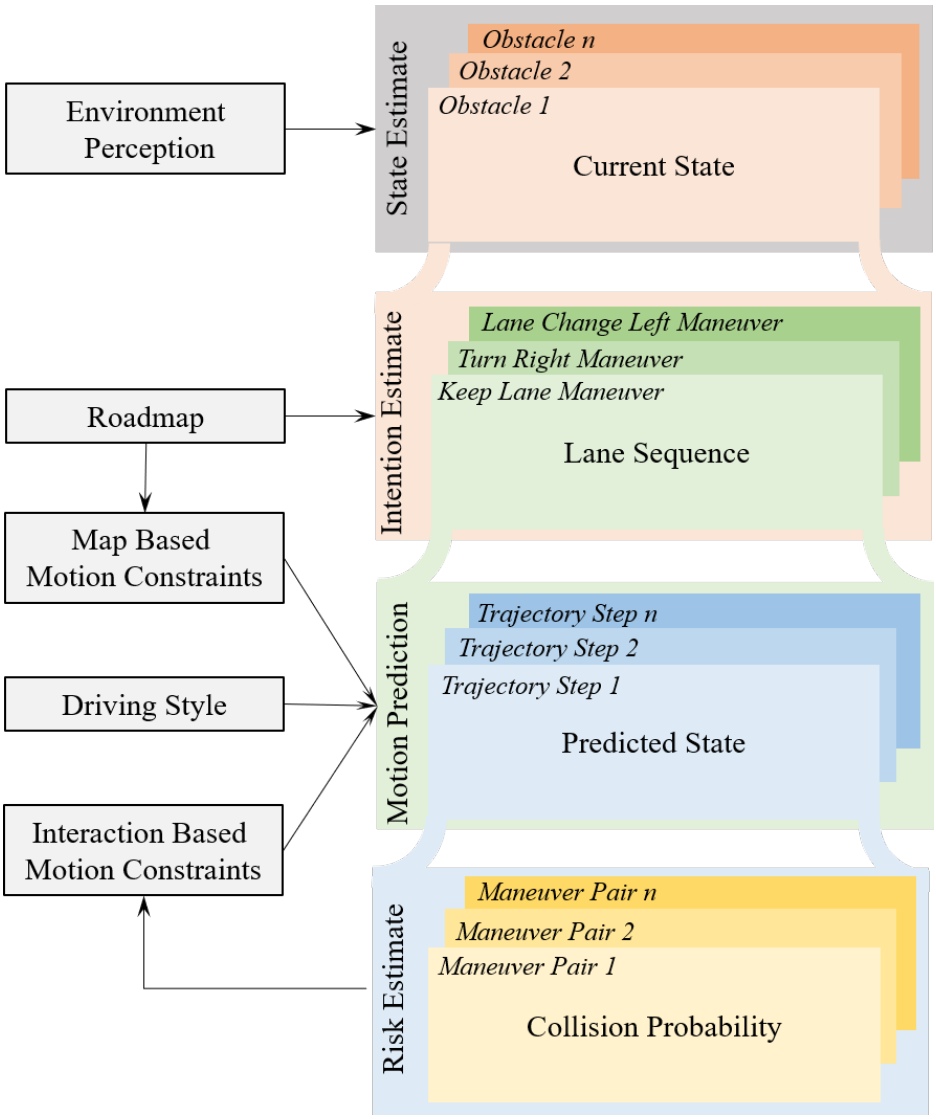


Fig. 5.1 Major elements of the traffic scenario prediction system.

constraints dictated by the infrastructure, as curvature, speed limits, stop signs, etc. Further influence results from the driving style, which is initially assumed as neutral. Subsequent updates of the motion prediction, which are performed in receding window fashion, take the interaction based constraints as additional input. These constraints result from the risk estimate, which calculates the collision probabilities for all trajectory steps of a maneuver related to the trajectory steps of all maneuvers of the other agents. The subordinate agent, which is identified by evaluation of the relevant traffic rules, is expected to adapt its motion plan to avoid a collision.

The system is innovative in the following ways:

- It provides a unified model for all types of traffic participants, as passenger cars, trucks, bicycles and pedestrians.
- All typical motion behaviors, as car following, lane changing, lane merging, intersection crossing etc. are handled by the unified model.
- The potential conflicts between all agents are detected and analyzed. The required measures to avoid these conflicts have decisive influence on the predicted motion.
- The traffic rules are hard coded in the system and any violation of them is made explicitly. Traffic rules are not learned from examples and their enforcement is not subject to some optimization. This facilitates the general approval for autonomous driving by the traffic authorities.
- The predicted behavior of the agents is fully explainable by the underlying model and this model may iteratively be refined.

The system consists mainly of the following random variables. All variables are indexed by the time step k:

- O_k^i: an object, detected by the perception system.
- $\mathbf{z}_k^i$: the measurement vector originating from the object i
- $\mathbf{x}_k^i$: the state vector estimated for object i.
- M_k^i: the maneuver intention of object i
- $\mathbf{a}_k^i$: the action vector of object i, given state and intention. The actions in the present system are the continuous accelerations in longitudinal and lateral direction.
- T_k^i: the predicted trajectory of object i. This is an abbreviation for $[\mathbf{a},\mathbf{x}]_{k+1:k+1+T}^{(T),i}$

The dependencies between the random variables are shown in Figure 5.2. The layout of this figures is kept intentionally similar to the figures in [160] and [161], but there are some fundamental differences in the dependencies.

Causal relationships between random variables of an object appear as solid arrows, temporal relationships between subsequent time steps as dashed arrows and interactive relationships as dotted arrows. Not shown are the dependencies on the roadmap, which is assumed to be static and deterministic.

The measurement $\mathbf{z}$ is generated by the dynamic state $\mathbf{x}$ and by the maneuver intention M. The action $\mathbf{a}$ of an agent depends on the current state and the intention. The predicted

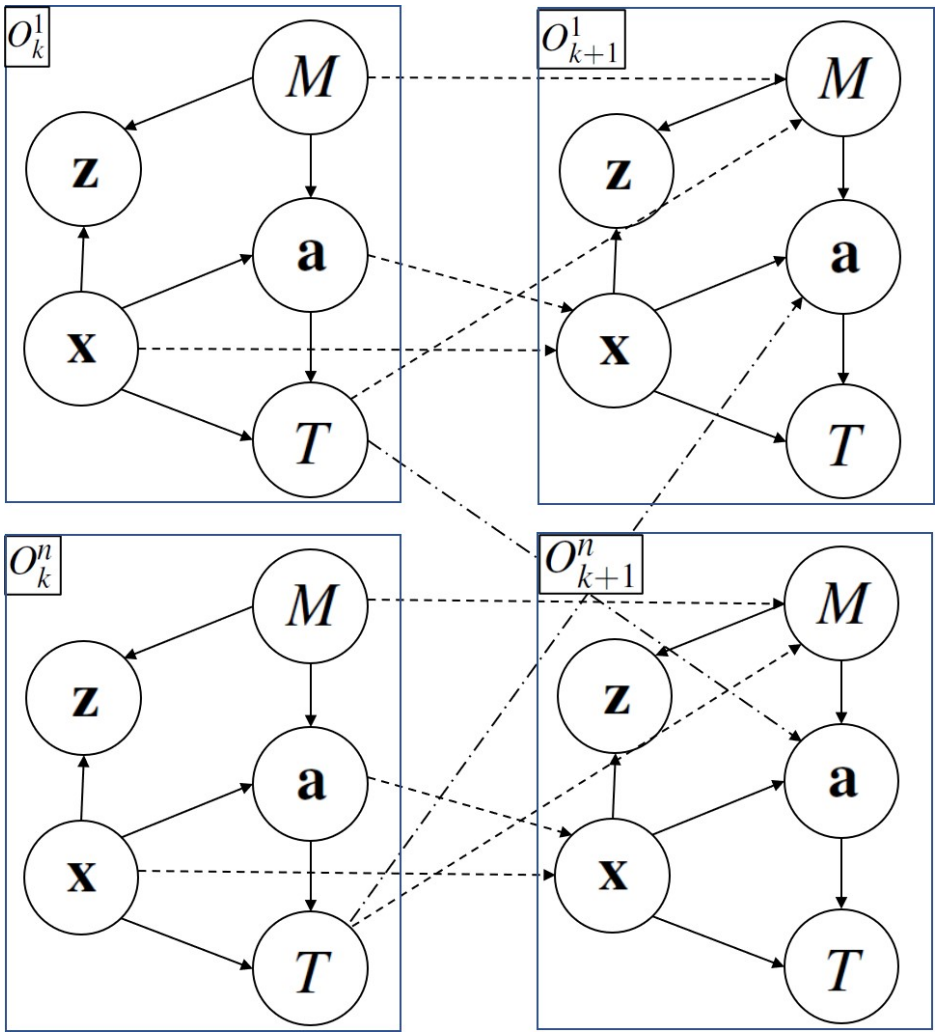


Fig. 5.2 Interdependencies between two agents (1 and n) at two subsequent time steps (k and $k+1$). The diagram shows causal relationships (solid arrows), temporal relationships (dashed arrows) and interactive relationships (dotted arrows).

trajectory T at one time step depends on the current state as starting point and the selected actions. The state of one time step depends on the state of the previous time step and the previous action. The maneuver intention is influenced by the previous maneuver intention and the previous predicted trajectory of the same agent.

The action of an agent depends additionally on the predicted future trajectories of the other agents, as far as the target has to yield to one or more other agents. This is the major difference to other approaches. Most authors regard only the obstacles in the near vicinity of the target and try to infer the next action of the target from the configuration of the current situation and the current states of the agents. In this approach, the predicted actions of the target take the predicted evolution of the entire scenario during the whole prediction horizon

into account, as far as it is relevant. Relevant are trajectories, which constitute a collision risk for the target.

Algorithm **??** gives a high-level overview over the calculation of a scenario prediction. The algorithm Create Scenario Prediction is triggered on arrival of a new set of measurements $\mathcal{Z}$. In the present system, this happens with a frequency of 10 Hz. Further input to the prediction are the sets of objects $\mathcal{O}$, maneuvers $\mathcal{M}$ and trajectories $\mathcal{T}$ as well as the collision matrix $\mathbf{C}$ from the previous prediction. The procedure UPDATEOBJECTS adds newly detected objects, removes old objects and updates the state of preexisting objects. The procedure UPDATEINTENSIONS evaluates the set of currently feasible maneuver intentions and calculates their probabilities. Procedure MAPCONSTRAINTS examines the infrastructure data and establishes constraints resulting from speed limits and intersection properties to be obeyed for a specific maneuver. Procedure INTERACTIONS examines the collision matrix of the previous prediction and infers the type of situation, the applicable traffic rules and the resulting responsibilities. Based on this, the interaction based motion constraints for the next prediction are calculated. The procedures TOFRENET and TOCARTESIAN convert states and trajectories between Cartesian and Frenet coordinates. The procedures ACTION and TRAJECTORYSTEP calculate the predicted trajectory for the next T steps (default 100 steps). Procedure COLLISIONMATRIX finally calculates the collisions risks between any pair of trajectories.

The actual implementation of the system differs in the sequence slightly from the shown algorithm to allow a parallelization of the CollisionMatrix procedure, which is the computational bottleneck of the system. This parallelization allows to the keep the frequency of 10 Hz in situations with several dozen agents.

Require:

$\mathcal{O}_{k-1}, \mathcal{M}_{k-1}, \mathcal{T}_{k-1}$ ▷ Sets of previous obstacles, maneuvers and trajectories

$\mathbf{C}_{k-1}, \mathcal{Z}_k$ ▷ Previous collision matrix and actual measurements

Ensure:

$\mathcal{O}_k, \mathcal{M}_k$ ▷ Sets of updated obstacles and maneuvers

$\mathcal{T}_k, \mathbf{C}_k$ ▷ Set of new trajectories and new collision matrix

$\mathcal{O}_k \leftarrow$ UPDATEOBJECTS$(\mathcal{O}_{k-1}, \mathcal{Z}_k)$ ▷ Update obstacle set

for $i \leftarrow 1$ **to** $|\mathcal{O}_k|$ **do** ▷ For all obstacles

  $\mathcal{M}_k^i \leftarrow$ UPDATEINTENTIONS$(O_k^i, \mathcal{M}_{k-1}^i, \mathbf{z}_k^i)$ ▷ Update maneuvers of obstacle

  for $j \leftarrow 1$ **to** $|\mathcal{M}_k^i|$ **do** ▷ For all maneuvers

    $\mathcal{R} \leftarrow$ MAPCONSTRAINTS$(M_k^{i,j})$ ▷ Evaluate map based constraints

    $\mathcal{I} \leftarrow$ INTERACTIONS$(M_k^{i,j}, \mathcal{T}_{k-1}, \mathbf{C}_{k-1})$ ▷ Evaluate interaction constraints

12: $\mathbf{x}_k^{i,j} \leftarrow \text{ToFrenet}(\mathbf{z}_k^i)$ ▷ Convert state to Frenet
13: **for** $t \leftarrow 1$ **to** T **do** ▷ For all time steps
14: $\mathbf{a} \leftarrow \text{Action}(\mathbf{x}_{k+t-1}^{i,j}, \mathcal{R}, \mathcal{I})$ ▷ Evaluate next action
15: $\mathbf{x}_{k+t}^{i,j} \leftarrow \text{TrajectoryStep}(\mathbf{x}_{k+t-1}^{i,j}, \mathbf{a})$ ▷ Next trajectory step
16: **end for**
17: $\mathcal{T}_k \leftarrow \mathcal{T}_k \cup \text{ToCartesian}(\mathbf{x}_{k:k+T}^{i,j})$ ▷ Convert to Cartesian and add to trajectories
18: **end for**
19: **end for**
20: $\mathbf{C}_k \leftarrow \text{CollisionMatrix}(\mathcal{O}_k, \mathcal{M}_k, \mathcal{T}_k)$ ▷ Calculate new collision matrix

The following section presents the details of the system. Section 5.4 describes, how the state of the objects is extracted from the result of the perception system from Chapter 3. Section 5.5 documents, how the feasible maneuver intentions of a target are detected and how their probability if computed. Details about the map based motion constraints are given in Section 5.6, followed by the interaction based motion constraints in Section 5.7. The motion prediction itself resulting in the predicted trajectory of a maneuver is the theme of Section 5.8. The calculation of the collision matrix for the next prediction is presented in Section 5.9. The evaluation of the presented system is analyzed in Section 5.10. The summary and conclusion Section 5.11 closes this chapter.

5.4 State Estimate

The purpose of the state estimate is to infer the physical state and some additional information about the object from the measurements. The state estimation procedure of this system is described in Chapter 3. The input from the perception system includes the following data:

- Unique Object Id
- Kinematic State (Section 3.8)
  - X and Y position in roadmap coordinates
  - X and Y velocity
  - X and Y acceleration
  - Covariances for all above states
- Bounding Box (Section 3.6)
  - Length

  - Width
  - Height
  - Orientation
  - Centroid Offset to position

- Motion Type (Section 3.10)
- Classification (Section 3.11)
- Existence Probability 3.12)
- Turn Signal State (if available)

The state estimation system tracks the object over time and provides an object identity (Object Id). Only confirmed objects are included in the list. Objects temporarily without measurement are kept in the list until their existence probability falls below a certain threshold and they are unconfirmed.

The ego vehicle must be included in the list as regular object with a predefined Object Id.

Since the above data includes already a filtered dynamic state, this state is directly taken as the state $\mathbf{x}_k^i$ of the prediction system. Acceleration values reported by the perception system are considered as measurements of the actions of the previous step. If only raw, unfiltered measurement data is available, a Kalman filter has to be implemented for each maneuver and the common state of the object has to be calculated by moment matching, similar to the IMM filter in Subsection 3.8.3.

5.5 Intention Estimate

The purpose of the intention estimate is to infer the decisions of traffic participants during the next prediction period. The result of the intention estimate is a set of possible intentions and its probability distribution.

Lane Bound Intention Classes					Trash
Turn Left	Lane Change Left	Keep Lane (Forward)	Lane Change Right	Turn Right	(Physical Laws)

Table 5.2 Maneuver Intention Classes

In the case of vehicles, it makes sense to consider the lanes that can be reached in the near future as a spatial alternatives. These are the lane bound intention classes. The lane

bound intention classes are defined by the sequence of lanes to be used during the prediction horizon. This work considers five classes: keep lane, lane change left and right, turn left and right.

But sometimes, vehicle drivers do not adhere to these standard intentions. Other traffic participants, as pedestrians, move generally unrelated to lanes. To capture the intention in these cases, [158] has invented the so-called Trash maneuver class. It allows to predict the motion of a traffic participants based on physical laws.

In [158], also Follow Vehicle and Target Brake are handled as lane bound maneuvers. In this work, vehicle following and target braking are not considered as separate maneuvers, but as consequences of interactive constraints, which may apply to all lane bound maneuvers classes.

This section is divided into the following subsections: Subsection 5.5.1 explains the trash maneuver class. The lane bound maneuver classes are presented in Subsection 5.5.2. Subsection 5.5.3 gives details about the maneuver life cycle. An example of maneuver intentions is the theme of Subsection 5.5.4. The final Subsection 5.5.5 documents the calculation of the maneuver probabilities.

5.5.1 Trash Intention Class

Every agent in the system may execute a trash maneuver. For vehicles, it serves as fallback, if the observed motion cannot be explained by some of the standard intention classes. Examples are taking a U turn or leaving the road for a parking lot. For pedestrians and other non-vehicle objects on the road, the trash maneuver is the only alternative. The same holds for any objects moving off-road. The purpose of the trash maneuver is also to cover cases, where drivers do not adhere to traffic laws during lane bound maneuvers, e.g. running a red light or a stop sign.

The trash maneuver is used to allow some basic motion prediction for all these cases. Typical implementations of the trash maneuver are constant velocity model (CV), constant acceleration model (CA) or constant turn rate and velocity (CTRV).

The prediction quality of the physical motion models is good for short-term predictions (1-3 seconds). Especially motor vehicles, due to their high inertial mass, are not able to change their kinematic state quickly. In Chapter 6 it will be shown in a simulation that it is possible to drive safely using only predictions based on the trash maneuver. But the drive becomes very uncomfortable and slow due to many unnecessary and abrupt braking maneuvers.

5.5.2 Lane Bound Intention Classes

The standard lane bound intentions in this system are defined by spatial decisions.

The considered spatial decisions in this work are keep lane, turn left or right and lane change left or right. Only the next upcoming spatial decision of each class is taken into account. This system could be extended including additional behavior, as parking or U turns, or by defining sequences of decisions during the prediction horizon, like double lane changes.

The intended path of a vehicle is defined by a lane sequence. A lane sequence is a sequence of consecutive lanes from the roadmap. At the start of the prediction, the current lane of the vehicle becomes the first lane of the lane sequence. Additional lanes are added to the lane sequence, until the total length of the lane sequence covers the maximal driving distance during the prediction time-span. The maximal driving distance is calculated using the current speed limit for the lane plus some surcharge for traffic rule violators.

The simplest form of a lane bound intention is the keep lane intention. A vehicle is predicted to stay on its lane. At the end of the lane, it continues its path on the successor lane. If the lane has several successors, the keep lane maneuver uses the forward lane. The forward successor lane is the lane with smallest deviation in the yaw angle of its ancestor. The keep lane maneuver is defined for every vehicle on the road.

In case that the lane sequence contains possible diverges, e.g. there are lanes with several successors, turn intentions are feasible for this vehicle. The lane sequence of the turn intention starts with a copy of the lane sequence of the keep lane maneuver until the first possible diverge lane and is completed after the turn with forward lanes until the prediction horizon is reached. When constructing the lane sequence, only one left or right turn is considered, so there are maximal two turn maneuvers in the prediction.

In the case that the current lane of the vehicle has neighbor lanes, to which a lane change is allowed, a lane change intention is feasible. A lane change maneuver has two lane sequences. The current lane sequence is the same as the keep lane maneuver. The LC lane sequence starts with the left or right neighbor of the current lane sequence and is completed using consecutive forward lanes. The two lane sequences are required since at the start of the prediction, it is not known, when the lane change can start and if the lane change is successful at all. Therefore, it is possible that the vehicle, even though the driver has a lane change intention, is forced to stay on the forward lane sequence. The lane change, e.g. the actual crossing of the lane boundary, must be completed as long as the two lane sequences are in parallel. The implemented system considers only one lane change to the left and one to the right for one scenario prediction.

5.5.3 Maneuver Life Cycle

A maneuver is the combination of a path intention, its predicted trajectory and the estimated probability of the maneuver. A maneuver has during its life time a unique id, which serves as reference to the maneuver. The predicted trajectory and the probability of the maneuver are updated on each subsequent prediction. Occasionally, the path intention of a maneuver may change.

Whenever a new obstacle is detected, all feasible maneuvers for this obstacle are evaluated. For each obstacle, there is always a Trash maneuver. A vehicle on a lane has also at least a keep lane maneuver. Turn and lane change maneuvers are only added, if the required lanes appear on the roadmap in front of the vehicle within the maximal prediction horizon (given in meters).

In subsequent predictions for an obstacle, the path intentions of the existing maneuvers are checked for validity based on the new position of the obstacle. If that position is no longer on the lane sequence of the maneuver, e.g. the diverge region has been passed and the turn is no longer feasible, the maneuver is discarded. The lane sequences may be updated by discarding lanes, which are in between behind the current vehicle position. Since the horizon of the maneuver moves forward, new lanes may be added at the end of the lane sequence. When extending the keep lane maneuver, new occasions for lane changes or turns may became visible and new turn or LC maneuvers are created, if not already existing.

A LC or turn maneuver is occasionally executed. In case of a lane change, this happens, when the centroid of the vehicle has crossed the lane boundary. In case of a turn, this happens, when the centroid is only on the turn lane and not any more on the KL lane sequence, e.g. the keep lane maneuver becomes infeasible. When a LC or turn maneuver has been executed, this maneuver gets the role of the keep lane intention. All other maneuvers are discarded and new LC or turn maneuvers are initialized, if feasible.

Prerequisite for a lane bound maneuver is a valid lane assignment. A vehicle is assigned to a lane, if its centroid is located between the left and right boundary of a lane and between its start and end. In practice, it has proven useful to allow for some tolerance outside this region, because neither the roadmap nor the localization of an obstacle inside the roadmap are exact.

A lane assignment may be ambiguous, when the regions of two or more lanes overlap, as on intersections, lane merges and diverges. If one of the possible lanes is part of the lane sequences of a preexisting KL, LC or turn maneuver, this lane becomes the new current lane. Otherwise, the lane which fits best to the orientation of the vehicle is selected.

5.5.4 Intention Estimate Example

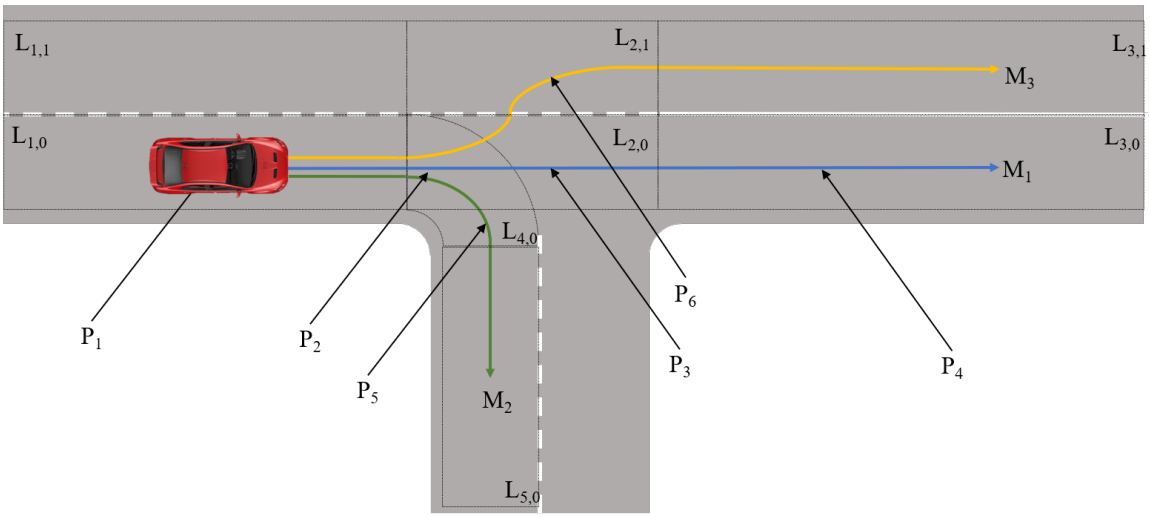


Fig. 5.3 Examples for lane bound intentions (keep lane, turn right, lane change left).

In Figure 5.3, an example of Lane Bound Intentions is given. In the beginning, the car is in position P_1 and is assigned to lane $L_{1,0}$. Besides the trash maneuver (not shown), the three following maneuvers are feasible based on the given roadmap:

M_1: Keep lane. The lane sequence is $L_{1,0} - L_{2,0} - L_{3,0}$

M_2: Turn right. The lane sequence is $L_{1,0} - L_{4,0} - L_{5,0}$

M_3: Lane change left. The KL lane sequence $L_{1,0} - L_{2,0} - L_{3,0}$ equals to M_1, the LC lane sequence is $L_{1,1} - L_{2,1} - L_{3,1}$

During the next time steps, the vehicle may have advanced to the following positions:

P_2: Vehicle is assigned to lane $L_{2,0}$. It is also on lane $L_{4,0}$, but the forward lane is the default. All three maneuvers are still valid. The lane $L_{1,0}$ is discarded from the lane sequences, the new lane sequences are $L_{2,0} - L_{3,0}$ for M_1, $L_{4,0} - L_{5,0}$ for M_2 and $L_{2,0} - L_{3,0}$ / $L_{2,1}$ - $L_{3,1}$ for M_3.

P_3: Vehicle is still assigned to lane $L_{2,0}$, but the turn maneuver M_2 has become unfeasible and is discarded. The lane sequences for M_1 and M_3 remain the same.

P_4: Vehicle is assigned to lane $L_{3,0}$. Since a lane change is now forbidden, the maneuver M_3 is discarded. The only remaining maneuver M_1 has the lane sequence $L_{3,0}$. Executing nevertheless a lane change from this position would be considered as a trash maneuver.

P_5: If the vehicle moves from P_2 to this position, it is assigned to lane $L_{4,0}$ and has executed the turn maneuver M_2. Maneuver M_2 becomes therefore the new keep lane maneuver, while its lane sequence remains unchanged. Maneuvers M_1 and M_2 have become infeasible and are therefore discarded.

P_6: If the vehicle moves from P_2 to this position, it is assigned to lane $L_{2,1}$ and has executed the lane change left maneuver M_3. Maneuver M_3 becomes therefore the new keep lane maneuver, its lane sequence is $L_{2,1} - L_{3,1}$. Maneuvers M_1 and M_2 have become infeasible and are therefore discarded. A new lane change right maneuver M_4 (not shown) would be initialized with lane sequences $L_{2,1} - L_{3,1}$ / $L_{2,0}$ - $L_{3,0}$.

In all positions it may be necessary to extend the lane sequences of the maneuvers by the corresponding forward lane(s), if the maximal prediction horizon, which depends on the current speed limit, is not covered.

5.5.5 Probability Calculation

This section describes the calculation of the probability distribution of the maneuver intention at time step k M_k:

$$M_k \in \{LCl,\ LCr,\ TUl,\ TUr,\ TR,\ KL\} \tag{5.1}$$

The calculation is based on a Hidden Markov Model (HMM) (see [74] for an introduction to HMMs). The maneuver intention probability is modeled as a hidden state, which emits 3 observables:

- Dynamic state $\mathbf{x}_k \in \mathbb{R}^6$: lateral and longitudinal position, velocity and acceleration. The dynamic state is taken as input data from the perception system. The state is used in Cartesian coordinates for the trash maneuver and in Frenet coordinates (see Subsection 5.8.1) for all other maneuvers.
- Turn signal state $S_k \in \{left, right, none, both\}$ [1] It is the main indicator for an imminent turn or lane change.
- Lane change incentive $l_k \in \mathbb{R}$. The lane change incentive measures the potential preference for a discretionary lane change based on comparing the trajectory length of the LC maneuver with that of the keep lane maneuver. The lane change incentive for left and right is asymmetric to support the German Rechtsfahrgebot.

The three observables are assumed to be independent of each other given the maneuver intention. The prior for the probability distribution is assumed to be:

$$p(M_0) = [0.045,\ 0.045,\ 0.045,\ 0.045,\ 0.015,\ 0.805]^T \tag{5.2}$$

[1] Currently, the system does not detect the turn signal state from sensor data, but turn signal status is used by the simulator.

The values are hand crafted and should be verified by real data. If one of the maneuvers is unfeasible, its probability is set to 0 and the distribution has to be renormalized. The same applies, if a former unfeasible maneuver is added later and its probability is initialized with the appropriate value from $p(M_0)$. On subsequent predictions, the maneuver probability is forwarded by the following recursion:

$$p(M_k|\mathbf{x}_k, s_k, l_k) = \eta\ \mathbf{\Pi}_{i,j} \times p(M_{k-1})\ p(\mathbf{x}_k|M_k)\ p(S_k|M_k)\ p(l_k|M_k) \tag{5.3}$$

with η as a normalization constant, transition matrix $\mathbf{\Pi}_{i,j}$, dynamic state likelihood $p(\mathbf{x}_k|M_k)$, turn signal probability $p(S_k|M_k)$ and lane change incentive density $p(l_k|M_k)$.

Transition Model

$\mathbf{\Pi}_{i,j}$ is the transition matrix, which propagates the previous maneuver probability $p(M_{k-1})$ one time step forward:

$$\mathbf{\Pi}_{ij} = \begin{bmatrix} 0.739 & 0.02 & 0.02 & 0.02 & 0.001 & 0.2 \\ 0.02 & 0.739 & 0.2 & 0.2 & 0.001 & 0.2 \\ 0.02 & 0.02 & 0.889 & 0.02 & 0.001 & 0.05 \\ 0.02 & 0.02 & 0.02 & 0.889 & 0.001 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0.1 & 0.6 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.001 & 0.959 \end{bmatrix} \tag{5.4}$$

These values are currently hand tuned and should in the future be learned by some machine learning algorithm, as the Baum-Welch algorithm [146]. The steady state distribution of this transition matrix is calculated by the Markov chain partitioning algorithm [163]:

$$p(M_\infty) = [0.047,\ 0.047,\ 0.1,\ 0.1,\ 0.001,\ 0.706]^T \tag{5.5}$$

Probability Density of Dynamic State

The probability of the dynamic state conditioned on the maneuver is distributed as a multivariate Gaussian and is given by Equation 5.6:

$$p(\mathbf{x}_k|M_k) = \frac{1}{\sqrt{(2\pi)^6 det(\Sigma_k^{(M)})}}\ exp\{-\frac{1}{2}(\mathbf{x}_k - \mathbf{x}_k^{(M)})^T\ \Sigma_k^{(M)^{-1}}\ (\mathbf{x}_k - \mathbf{x}_k^{(M)})\} \tag{5.6}$$

$\mathbf{x}_k^{(M)}$ is the predicted state for maneuver M at time step k, $\Sigma_k^{(M)}$ its covariance matrix. For the maneuver state prediction see Section 5.8.

Probability of Turn Signal

The conditional probability table (CPT) for the turn signal state is given in Table 5.3. The table reflects the fact that typically maneuvers to the left are more often announced than those to the right and that especially the lane change right after an over-take is not signalized. A verification of the figures based on real-world data is pending.

	Maneuver Type					
Turn Signal	LCl	LCr	TUl	TUr	TR	KL
Left	0.9	0.001	0.9	0.001	0.005	0.01
Right	0.001	0.7	0.001	0.8	0.005	0.01
None	0.098	0.298	0.098	0.198	0.98	0.97
Both	0.001	0.001	0.001	0.001	0.01	0.01

Table 5.3 CPT for turn signal conditioned on maneuver type.

Probability Density of Lane Change Incentive

The lane change incentive compares the expected length of the predicted trajectories of a lane change maneuver $|\mathbf{x}_{k:k+T}^{LC}|$ with that of the keep lane maneuver $|\mathbf{x}_{k:k+T}^{KL}|$. Since the prediction time span $k:k+T$ is equal for both, the longer trajectory represents the higher average speed and is therefore preferable.

$$l_k = \begin{cases} \max(l_{min}, \min(l_{min}+l_{len}, \frac{|\mathbf{x}_{k:k+T}^{(M)}|}{|\mathbf{x}_{k:k+T}^{KL}|})) & M \in \{LCl, LCr\} \\ 1 & \text{otherwise} \end{cases} \tag{5.7}$$

The domain of the l_k quotient is limited to $\{l_{min}...l_{min}+l_{len}\}$ to restrict the influence of the LC incentive. The probability distribution of l_k conditioned on the maneuver type is then:

$$p(l_k|M_k) = \frac{1+\alpha_{LC}(l_k-(l_{min}+\frac{l_{len}}{2}))}{l_{len}} \tag{5.8}$$

Since this function integrates to 1, it is a valid PDF. The parameter α_{LC} controls the gradient of the LC incentive and must be chosen to be $< \frac{1}{l_{min}+\frac{l_{len}}{2}}$ to avoid negative densities. In the present system, α_{LC} and l_{len} are set to 1. For the lane change left, $l_{min} = 0.5$, which makes the density function centered around 1, so that there is no preference for any maneuver

on equal trajectory length. For the lane change right, $l_{min} = 0.4$ is chosen, which moves the center to 0.9, e.g. a lane change to the right is preferred, even if the resulting trajectory is slightly shorter. This enforces the German Rechtsfahrgebot (see Figure 5.4).

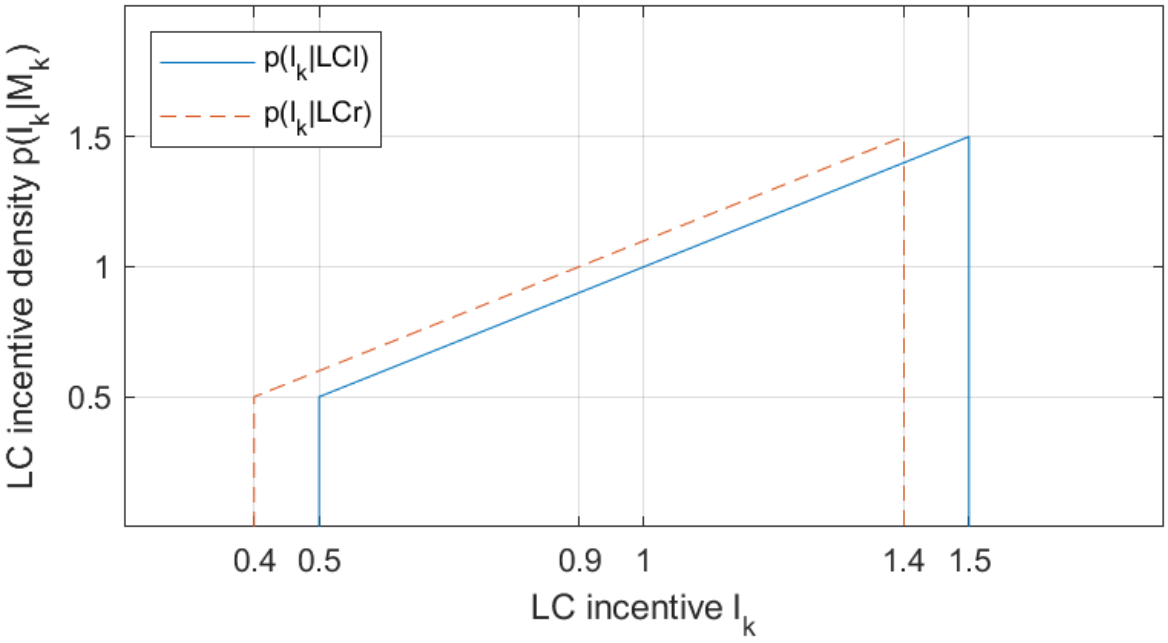


Fig. 5.4 PDF of lane change incentive for LC left and LC right. The difference results from the Rechtsfahrgebot.

5.6 Map Based Motion Constraints

Map based motion constraints are any restriction on an agent's motion, which arise from the infrastructure and that are assumed to be annotated in the roadmap. The constraints only apply to lane-bound maneuvers. These constraints may force the agent to decelerate and they are independent of the presence of other traffic participants.

The parameters of the map based constraints for each maneuver are evaluated before the motion prediction by examining the roadmap data of the maneuver lane sequence. The set of these parameters must be obeyed on each time step of the motion prediction.

There are two classes of constraints: Speed limits and intersection properties.

5.6.1 Speed Limits

For all types of speed limits, three parameters are calculated:

- Start of speed limit, measured in $[m]$ from the start of the first lane of the maneuver lane sequence. The distance is measured in Frenet coordinates.
- Optional length of the speed limit in $[m]$.
- Maximal allowed speed in $[m/s]$.

There are three types of speed limits: Legal speed limits, curvature speed limits and visibility speed limits:

- Legal speed limits are indicated by traffic signs or are valid for certain road types (highway, urban, ...). The parameters for legal speed limits are taken directly from the map. Legal speed limits are valid, until a new limit is given.
- Curvature speed limits. The curvature speed limit is calculated from the curvatures of the drive splines of the maneuver lanes. All sub sections of a lane having a curve radius below a certain threshold (example: $100m$) are considered as a curve. The speed limit for the curve is calculated as $v = \sqrt{a_c \; r_{min}}$ with a_c the maximal desired centripetal acceleration of the driver and r_{min} the minimum radius of the curve. The curvature speed limit is only checked, if it is lower than the legal speed limit. The curvature speed limit must be reached before the start of the curve and is valid until the end of the curve.
- Visibility speed limits apply due to scare visibility when approaching a pedestrian crossing or an unsignalized intersection without stop sign on a subordinate lane. Even if no other object is currently visible, the speed may have to be reduced. The visibility speed limit starts at an imaginary line of sight, which is either given by the roadmap or which is assumed to be at a fixed position, for example $10m$ before the pedestrian crossing or intersection. The maximal speed at the line of sight is calculated as $v = \sqrt{2 \; b \; d_{ls}}$, with the desired deceleration of the driver b and the distance d_{ls} between the line of sight and the stop line before the pedestrian crossing or intersection. The length of the speed limit is set to $1cm$ and the target may re-accelerate immediately, if no obstacle has become visible.

5.6.2 Intersection Properties

During a lane bound maneuver, an object may have to approach one or more intersections. The properties of these intersections may force the target to decelerate, independent of the presence of any other object. To avoid a re-examination of the roadmap on each prediction

step, the parameters of these properties are extracted once before the start of the prediction. These parameters are also used to avoid collisions with other agents, as described in Section 5.7.

- Conflict type: Crossing one or more other lanes or only merging with one other lane (on turn right).
- Priority: Yes or no. If two lanes intersect having the same priority, the right-before-left rule is applied. This holds also for lanes turning left and crossing the oncoming lane.
- Stop Type: GiveWay, Stop or FourWayStop.
- Traffic light: Yes or no. If a traffic light applies to the lane and it is operational, it supersedes all other rules.
- Distance to stop line. The position, where the target has to stop, if it has to stop.
- Distance to line of sight. The position before the stop line, from where the incoming lanes are visible (see Subsection 5.6.1).
- Distance to save line. The save line is a line behind the intersection, which should be reached by a target within the prediction horizon. Otherwise, it should not enter the intersection because there is the danger of stopping on the intersection and blocking it.

The distances of the lines are measured in $[m]$ from the beginning of the lane entering the intersection. The line of sight may be before this lane, having a negative distance. The save line may be located on a successor lane.

If no intersection parameters are given in the roadmap, the following default rules apply:

- The road having more parallel lanes has priority.
- If two roads with the same number of lanes cross, right-before-left is used.
- The stop line is $1m$ before the overlapping area of two crossing lanes.
- The line of sight is $10m$ before the stop line.
- The save line is directly behind the overlapping area.

Remark: Traffic lights and four-way-stop are not yet implemented in the current version of the system. The same is true for intersections, which are temporarily regulated by a police officer.

5.7 Interaction Based Motion Constraints

The interaction based motion constraints have to be determined, if the maneuver is involved in one or more collision risks. The collision risks are always calculated for the trajectories of the previous prediction. Therefore, no risks are available, when the first trajectory of a new maneuver is rolled out. The following steps are required to determine the interaction based motion constraints:

- Select risks to be considered
- Establish type of the risk

  Single Lane Car Following

  Multi Lane Traffic with Lane Changes

  Lane Merge

  Intersection Crossing

  Pedestrian Crossing

  Other Risks
- Evaluate responsibility for risk mitigation
- Provide parameters for risk mitigation

The potential risks are all risks from the collision risk analysis, which belong to a maneuver with a certain probability and which exceed a certain accumulated collision risk until the end of the prediction horizon. Risks below this threshold are ignored for the time being.

The type of the risk depends on the current traffic situation, especially on the relationship between the two traffic participants. The type of risk determines the traffic rules to apply. The traffic rules normally decide, who of the two agents has to mitigate the risk during the prediction horizon. In some cases, the responsibility may change during the execution of a maneuver, so both have to take care of the risk. Example: Before the start of a lane change, the lane changer has to assure not to endanger the new follower. Later, after crossing the lane boundary, it is up to the new follower to avoid a rear end collision.

The detected risk has to be obeyed by the responsible agent during many or all prediction steps. Example 1: During car following, the preceding car has to be taken care of until the end of the prediction horizon or until it has left the lane of the follower. Example 2: A crossing pedestrian has to be taken care of, until it has left the conflict zone. For efficiency

reason, certain parameters of the risk, like position and size of conflict zones or time of a predicted turn maneuver of the preceding car, are evaluated once before the trajectory prediction starts.

In this system, only two methods for risk mitigation are considered: braking or postponing a planned lane change. Accelerating is not considered for risk mitigation, since it is assumed, that the agents already drive as fast as possible and therefore further acceleration would violate traffic rules or be uncomfortable. But it is true that certain risks at intersections or during overtaking could be mitigated using increased acceleration values. Moreover, this system does not yet consider swerving to evade an obstacle. Instead, in cases of blocked lanes, it will always predict a full stop or a lane change.

This section is divided into the following subsections: In 5.7.1, the car following risk in single lane traffic is described. The more complex multi lane situation, including lane changes, is presented in Subsection 5.7.2. Subsection 5.7.3 handles pedestrian crossing, followed by the intersection crossing risk in Subsection 5.7.4. Lane merges are the theme of Subsection 5.7.5. Other risks and pseudo risks are presented in the Subsections 5.7.6 and 5.7.7. Finally, the forwarding of analyzed risks to subsequent predictions is handled in Subsection 5.7.8.

5.7.1 Single Lane Car Following

A single lane car following risk may be present, if the obstacle is initially on a lane belonging to the lane sequence of the target and the obstacle is ahead of the target. The target intention may be lane keeping or turning, the obstacles may have any lane bound intention. Potential rear end collision risks between two agents after lane changes or lane merges are handled separately in the subsequent sections.

Special attention is required, if a lane has several successor lanes (lane diverge). Directly after the diverge, the two or more successor lanes overlap and therefore, two objects may collide in this diverge zone, even though they are driving on different lanes.

The target may have to obey several car following risks during one prediction. Example:

Figure 5.5 shows the red target car following the silver car, which is turning right, and the yellow car, which blocks the lane. The lane sequence of the target is $L_{1,0} - L_{2,0} - L_{3,0}$, for the silver obstacle $L_{1,0} - L_{4,0} - L_{5,0}$ and $L_{3,0}$ for the yellow obstacle. The shown arrows show the velocity at the begin of the prediction, but these may not be constant. The silver car may have to brake for some pedestrian or the yellow car may accelerate after a traffic light turned green.

The required deceleration for the red car is calculated during motion prediction (see below) based on the previous trajectories of the silver and yellow car. The calculation is done

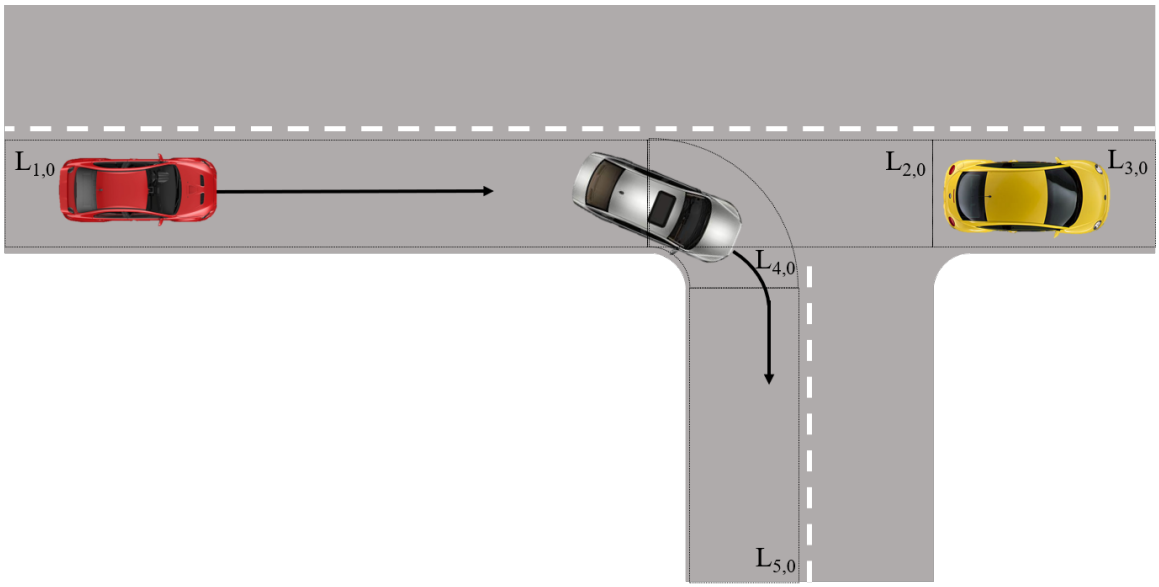


Fig. 5.5 Car following risks: The red car may have to brake for the slower silver car or for the stopped yellow car.

in the Frenet frame. The distance between the Frenet frames of the cars is given as parameter to the motion prediction. In this case, it is $0m$ for the silver car and the length of the lanes $L_{1,0}$ and $L_{2,0}$ for the yellow car.

5.7.2 Multi Lane Traffic with Lane Changes

A lane change risk may be present, if the target plans a lane change (active lane change) or an obstacle plans to change to the lane of the target (passive lane change). Both may appear in combination, which results in the very dangerous situation of simultaneous lane changes, especially on 3-lane highways or during on ramp maneuvers.

Special care has to be taken for lane changes in urban traffic. Vehicles frequently try to change lanes shortly before an intersection or in traffic jams, while driving at low speed. The boundary crossing phase may be quite long in these cases and the lane changer may even come to a full stop, blocking both lanes. The following cars have to be prepared for this kind of situation.

The system in this work not only tries to detect a lane change as early as possible, like many other systems, but also tries to predict future lane changes.

In the situation in Figure 5.6, the red car (the target) approaches with constant velocity the yellow obstacle, which blocks the lane. Moreover, the target has activated the left turn signal. The probability that the target has a lane change intention is near 100 %. On the other hand, it's highly unlikely that it will start the lane change immediately, causing a collision

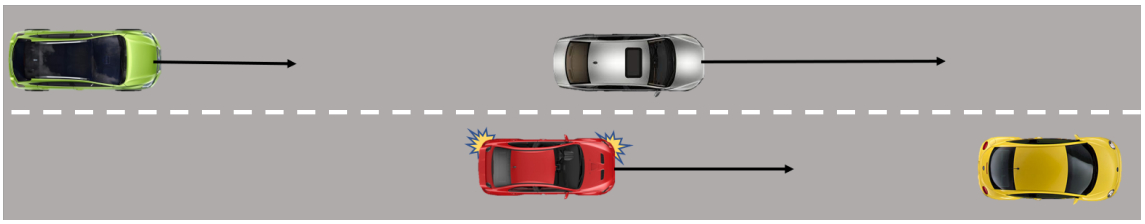


Fig. 5.6 Lane change intention: Typical situation for impending lane change of the red car. The yellow car has stopped and blocks the right lane. The length of the arrows is proportional to the speed.

with the silver car. It will therefore postpone the start of the lane change, until a suitable gap is available. In contrast, if the red car would decelerate and activate the right turn signal, a lane change intention would be very unlikely. Depending on the likelihood of the lane change intention, small deviations of the car from the center of the lane to the left may be an indicator for the start of a lane change or not.

A target with lane change intention may be in different stages of the LC (see Table 5.4):

- Phase 1 - LC lane ahead: The start of the destination lane is still ahead of the target position. Currently LC not yet possible.
- Phase 2 - LC lane available: The destination lane is available, but the LC has not yet started. The motion prediction checks on every time step, whether the gaps to objects on the destination lane are sufficiently large to start the LC.
- Phase 3 - LC in progress: The target is changing to the destination lane. The lateral motion for the LC is calculated by the motion prediction.
- Phase 4 - LC completed: The target has crossed the lane boundary completely. Lateral motion to reach the center line of the destination lane has to be continued.

The target may be in any of the four phases on start of the new prediction. At the end of the prediction time span, the LC may not have completed. Related to an obstacle, the following risks for the target may appear:

- Risk A - Obstacle is ahead on forward lane or changes to forward lane ahead of the target.
- Risk B - Obstacle is ahead on LC lane or changes to LC lane ahead of the target.
- Risk C - Obstacle is behind on LC lane or changes to LC lane behind the target.

	Obstacle Risk Type		
Phase	A	B	C
1 LC lane ahead	Brake	./.	./.
2 LC lane available	Brake	Check Front Gap	Check Rear Gap
3 LC in progress	Brake	Brake	./.
4 LC completed	./.	Brake	./.

Table 5.4 Risk handling during different LC phases

Table 5.4 shows required reactions of the target depending on LC phase and risk type.

During the situation analysis, the following parameters are evaluated for each obstacle constituting a collision risk:

- Time and position when the obstacle enters and leaves the forward lane sequence of the target.
- Time and position when the obstacle enters and leaves a lane diverging from the forward lane sequence of the target.
- Time and position when the obstacle enters and leaves the LC lane sequence of the target.
- Time and position when the obstacle enters and leaves a lane diverging from the LC lane sequence of the target.

Knowledge of the above parameters speeds up the motion prediction considerably since the number of checks on each prediction steps is reduced.

5.7.3 Pedestrian Crossing

Pedestrians are always assumed to execute a trash maneuver since their motion is not bound to any lanes. The trash maneuver is predicted with a physical motion model like constant velocity or constant acceleration. This means that an agent executing a trash maneuver does not react on any risks. The consequence is that the other agents, e.g. vehicles, have to take care of the collision risk with a pedestrian (collisions between pedestrians are not predicted by the system).

In Figure 5.7, the car approaches a pedestrian. The predicted trajectory of the pedestrian across the vehicle's current lane defines the conflict zone (red rectangle). The car has to decelerate to enter the conflict zone only, after the pedestrian has left it. It is important to note that the point in time when the pedestrian enters the conflict zone is irrelevant: If the car

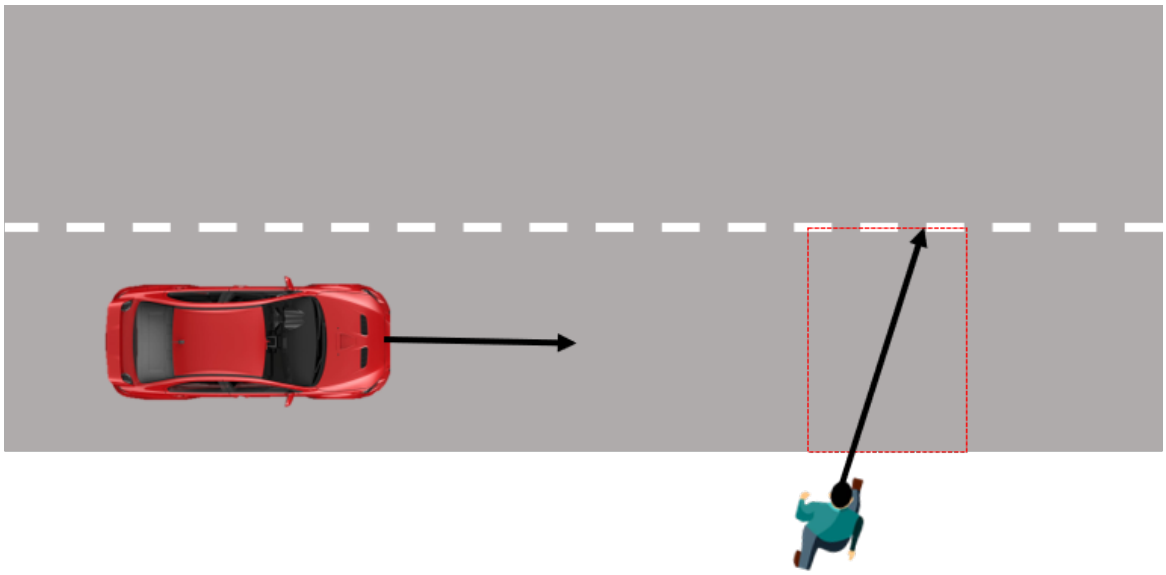


Fig. 5.7 Pedestrian crossing the road. The red rectangle marks the conflict zone.

would be able to cross the conflict zone before the pedestrian, no collision risk would have been reported.

Pedestrian cross walks need special handling. In Figure 5.8, the stop line for the car is not the border of one of the individual conflict zones, but the complete overlapping area between the vehicle lane and the cross walk is considered as conflict zone. The cross walk is handled by the system as special type of lane. The car is allowed to enter the cross walk zone only, when all pedestrians have left it and when it is able to pass the zone completely, before any other pedestrian will enter it. The last part of the condition becomes relevant in traffic jams. When approaching a pedestrian cross walk, a line of sight may be given to enforce reduced speed (see Subsection 5.6.1).

The parameters for the pedestrian crossing:

- Point in time in $[s]$ when the pedestrian will have left the conflict zone.
- Start position of conflict zone in $[m]$.
- Stop line position $[m]$ before the pedestrian cross walk (optional)
- Line of sight position $[m]$ for pedestrian cross walk (optional)
- Length of pedestrian cross walk $[m]$ (optional)

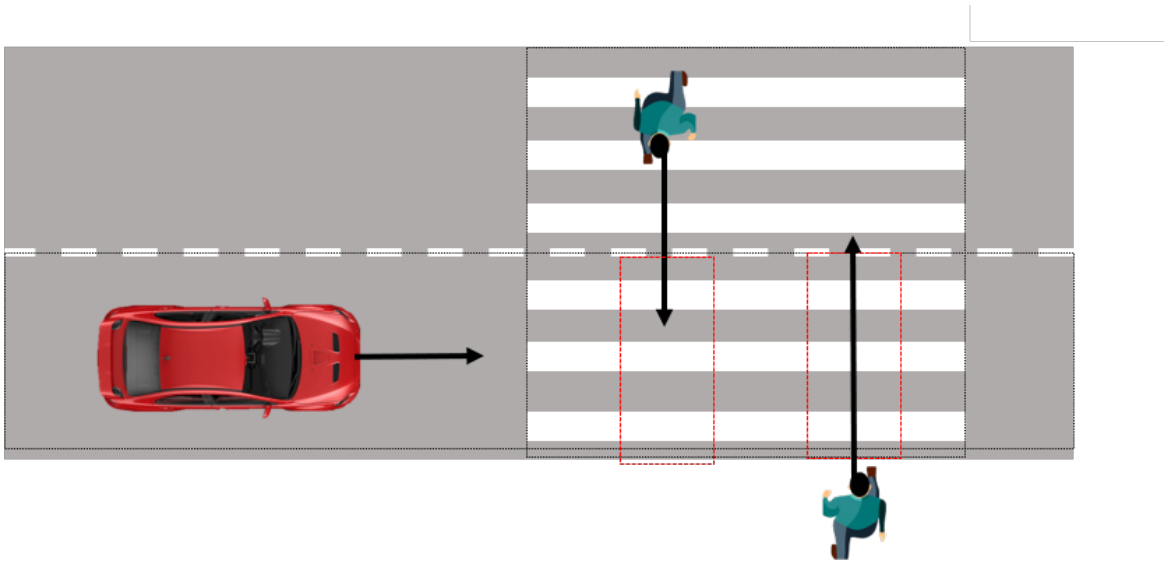


Fig. 5.8 Two pedestrians at a cross walk with individual conflict zones.

5.7.4 Intersection Crossing

Intersections are overlapping areas of two or more otherwise unrelated lanes. Unrelated means that the lanes do not have a common ancestor or successor.

A risk is a lane crossing risk, if the lane sequence of the target maneuver crosses the lane sequence of the obstacle maneuver. Since both maneuvers can be lane changes, the LC lane sequences have to be considered as well when identifying lane crossing risks.

The method to approach an intersection independent of any risk has been described in Subsection 5.6.2. The parameters for the priorities in case of collision risks are also given in Subsection 5.6.2.

If the target is on a priority lane, a lane with a green traffic light or if the target approaches from right at a right-before-left intersection, the risk can be ignored. In all other cases, the target has to give way.

Figure 5.9 shows an example of a situation at a simple give-way intersection. The target (red car) is approaching a give-way intersection on lane $L_{1,0}$ and has to yield for the yellow and green car, if a collision risk has been detected. If the target had been able to pass the conflict zone completely before the yellow car, the previous scenario prediction would not have reported any risk. Yielding means that it may not enter the red conflict zone before any obstacle has left it completely. The target has to reduce its velocity before it reaches the line of sight s_2 to be able to halt at the stop line s_1. Since there is not a stop sign, it has to halt at the stop line only, if the collision risk with one of the obstacles on lane $L_{2,0}$ cannot be mitigated by slowing down.

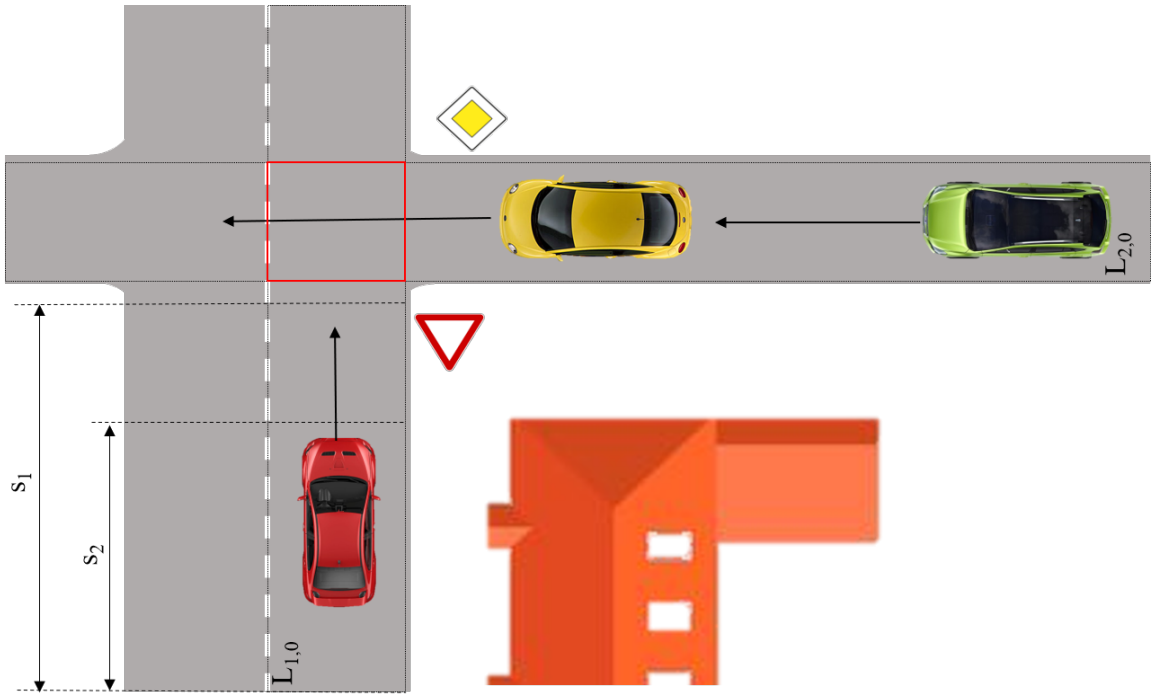


Fig. 5.9 Simple give-way intersection. The red rectangle marks the overlapping are of the lanes (conflict zone).

Figure 5.10 shows the possible maneuvers of two oncoming vehicles at an intersection. The possible combinations are listed in Table 5.5. Only the cases 2 and 4 are lane crossing risks, while the cases 3 and 7 are lane merging risks, which are handled in more detail in Subsection 5.7.5. The upper red conflict area is for case 3, the lower one for case 4, the conflict areas for cases 2 and 7 are not shown.

In contrast to the approach presented in [160], the present proposal does not search actively for conflict zones and possible passing sequences of agents through the conflict zone. It relies moreover on the collision risk calculation and examines the related trajectories only if a collision risk is detected.

In Figure 5.11, the red target tries to cross a multi lane intersection. It has to wait at the stop line until it is able to reach the save line within the prediction horizon. Assuming a main road width of $14m$, a vehicle length of $5m$ and an acceleration of $1m/s^2$, the crossing will take $\approx 6.2s$ without any intermediate delays.

Assuming, the red target has just reached the stop line, it will be prevented from restarting immediately by the collision risk with the yellow car. The prediction algorithm will therefore plan to start accelerating only after the yellow car has left the first conflict zone. Even if the resulting trajectory is long enough to pass the save line within the prediction horizon, it will probably generate another collision risk with the silver or green car. For this reason,

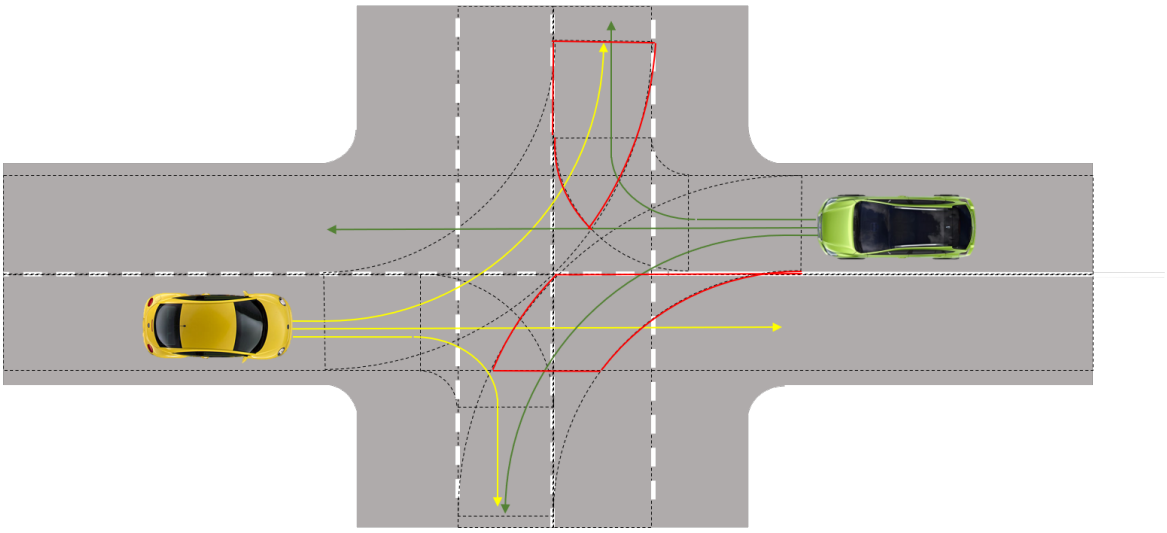


Fig. 5.10 Oncoming obstacle at intersection. The red areas are two examples for conflict zones. The yellow and green arrows symbolize the feasible maneuvers of the cars.

the subsequent predictions will have to plan another stop before the second conflict zone, which will make it very unlikely to reach the save line within the required time limit. The whole process is repeated at each new scenario prediction until the priority traffic allows to cross the intersection fast enough. In a multi lane environment it is moreover possible that the obstacles or the target start a lane change shortly before the intersection or even while crossing it. In these cases, the conflict area is doubled or even quadrupled in size to cover all relevant lanes.

Figure 5.12 shows an intersection with a priority road with central reservation. This scene is modeled by splitting it into two independent intersections. The red target may start accelerating after the yellow obstacle has left the conflict zone, provided it fits into the zone between the 1.save line and the 2.stop line. In this situation, the prediction algorithm has to obey that approaching a 2.stop line will prevent the car from full acceleration. This may in turn result in a new collision risk with the green car in the first conflict zone.

All the above scenarios assumed that the subordinate road user obeys the traffic rules. If this is not the case, it will not decelerate, but keep in contrast to the prediction its previous velocity or even accelerate. The maneuver probability calculation will therefore result in an increase of the probability of the trash maneuver and raise it over the criticality threshold. Since a traffic participant executing a trash maneuver (like all pedestrians) has always priority, the original priority vehicle will be predicted to yield. The same mechanism is applied, if the subordinate road user comes to a significantly lower collision risk estimate as the prediction algorithm and proceeds therefore.

Case	Yellow Maneuver	Green Maneuver	Risk Type	Priority
1	turn left	turn left	—	—
2	turn left	keep lane	lane crossing	green
3	turn left	turn right	lane merge	green
4	keep lane	turn left	lane crossing	yellow
5	keep lane	keep lane	—	—
6	keep lane	turn right	—	—
7	turn right	turn left	lane merge	yellow
8	turn right	keep lane	—	—
9	turn right	turn right	—	—

Table 5.5 Maneuver combinations with oncoming car

A second exception from the standard priority rules concerns vehicles blocking an intersection due to a traffic jam. A road user approaching an intersection on a priority lane will have to yield, if there is a collision risk with a crossing vehicle, which is not able to clear the intersection in time.

Besides the intersection parameters already given in Subsection 5.6.2, the following parameters are evaluated:

- Actual distance to conflict zone in $[m]$.
- Time, when the obstacle will have left the conflict zone in $[s]$.

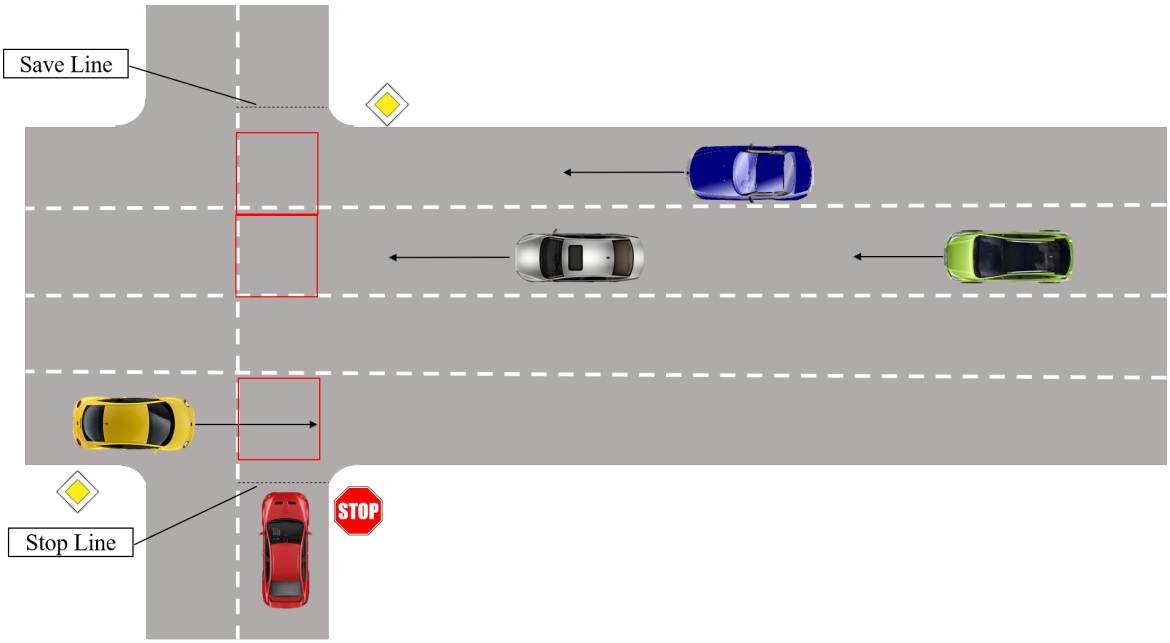


Fig. 5.11 Crossing a multi lane intersection. The red target vehicle has to path several conflict zones. It must be able to reach the save line within the prediction horizon to avoid blocking the intersection.

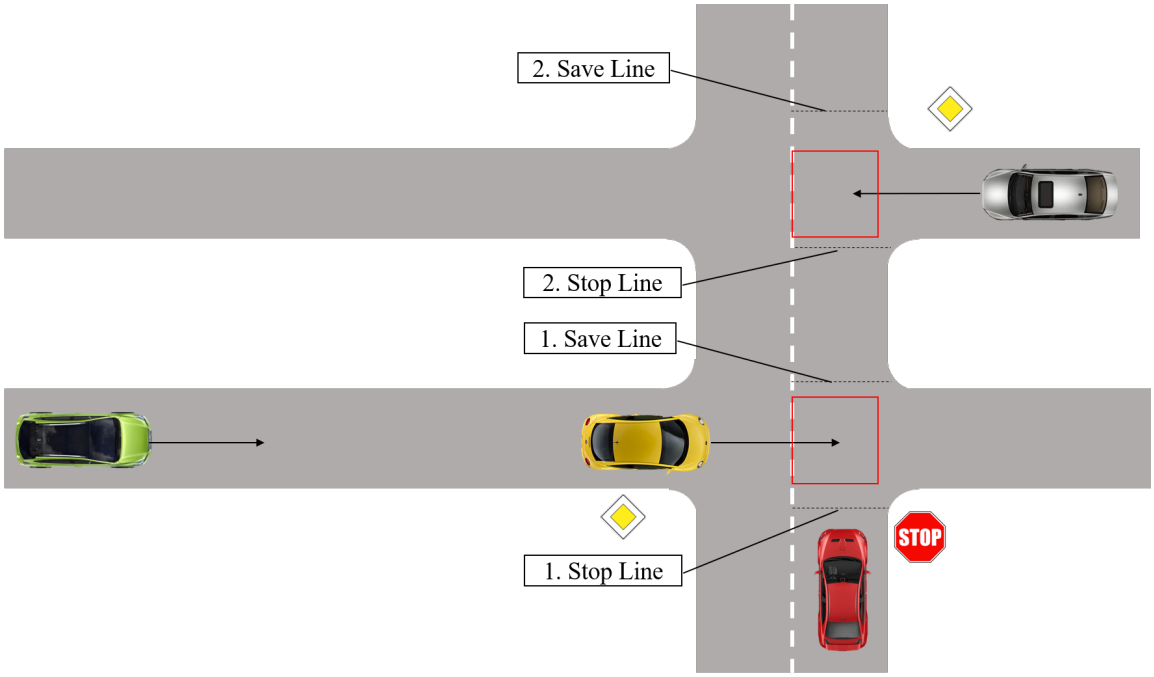


Fig. 5.12 Crossing a priority road with central reservation. The distance of the conflict zones allow pathing in two steps.

5.7.5 Lane Merge

A lane merge risk occurs, if two agents drive on different lanes, which have a common successor lane. Lane merges have to be handled differently from lane crossings since both agents are after the merge on the same lane. The risk of a side impact in the overlapping zone of the merging lanes is immediately followed by the risk of a rear end collision.

There are two types of lane merges, the forced lane merge and the avoidable lane merge.

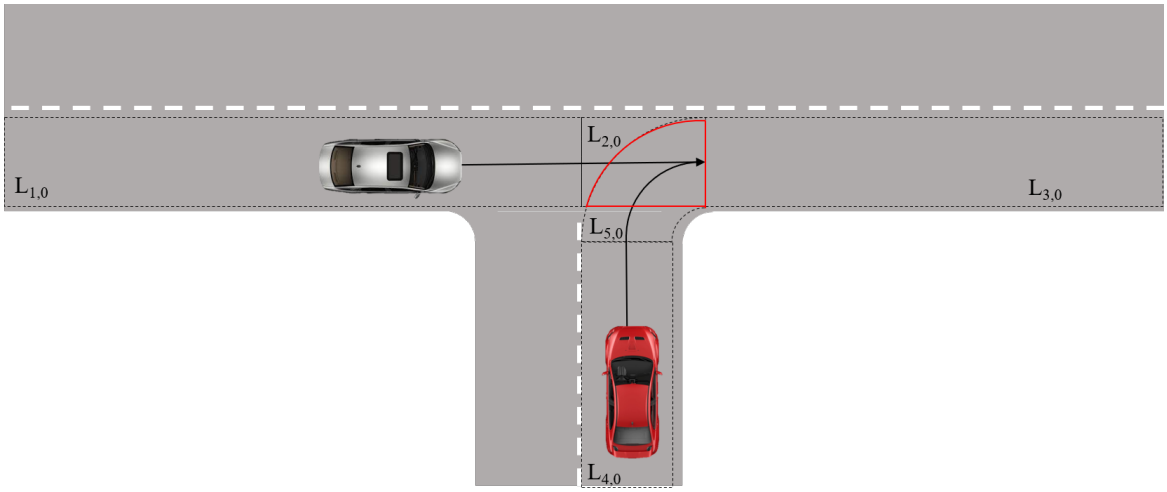


Fig. 5.13 Forced lane merge at intersection. The red conflict zone must be swept through one after the other.

Figure 5.13 shows a forced lane merge. The red target must go through the conflict zone of the lanes $L_{5,0}$ and $L_{2,0}$ to reach its destination lane $L_{3,0}$. The priority can be derived in different ways:

- roadmap annotation: An annotation in the map assigns the priority to one of the merging lanes.
- Intersection rules: The merging lanes inherit the priority of their predecessor lanes $L_{1,0}$ and $L_{4,0}$. Due to the right-before-left rule, the red car has priority in the shown example, but this could be regulated otherwise by traffic signs.
- Road geometry and lane markings: One of the lanes is marked as ending and the other has therefore priority. This rule is typical for optional lane merge (see below).
- Yaw angle difference: In absence of other applicable rules, it is assumed that the merging lane with the lower difference in the yaw angle compared to the common successor lane has the priority.

Figure 5.14 shows an example for an avoidable lane merge as it is typical for highway on-ramp situations. The red car may continue with its keep lane maneuver ($L_{1,2} - -L_{4,0} - -L_{3,1}$), which forces him to pass the conflict zone between the lanes $L_{4,0}$ and $L_{2,1}$. Since he has not the priority, the situation may force him to stop before entering the conflict zone. Passing the short conflict zone after a stop before another vehicle arrives may not be a problem, but due to the typically high velocities on highways, the new follower may be forced to a dangerous braking maneuver. To avoid such a situation, most drivers will prefer to execute a lane change before the merge zone. The difference to a normal lane change is that the target may not continue with constant velocity, if the lane change is not feasible, but has to decelerate before the end of the lane, making the lane change more and more difficult. The example situation gets even more critical since the green car plans a lane change to the right into the conflict zone. The present system is able to handle this type of situation.

In case of traffic rule violations during a lane merge, the system behaves as documented in Subsection 5.7.4 by raising the probability of a trash maneuver.

The present system does not currently predict courtesy behavior or alternate merging traffic.

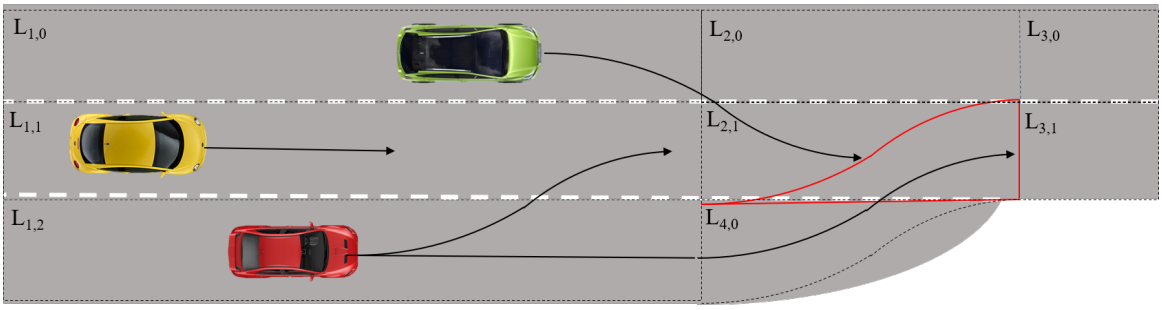


Fig. 5.14 Avoidable lane merge at highway on-ramp. The red car may avoid to pass the conflict zone by performing a lane change beforehand.

The following parameters are evaluated for a lane merge:

- Distance to conflict zone in $[m]$.
- Time, when the obstacle will have left the conflict zone in $[s]$.

If the lane merge occurs on an intersection, the intersection parameters already given in Subsection 5.6.2 will be used.

5.7.6 Other Risks

Since the risk calculation algorithm works in Cartesian space and not in Frenet coordinates, it is able to detect arbitrary collision risks, even those, which are not directly explainable by the

maneuver model. Collision risks are detected independent of the roadmap and independent of a specific traffic situation.

An example is given in Figure 5.15. The red target approaches the rear end of the green car, which protrudes into lane $L_{1,0}$. The situation results probably from an incomplete lane change of the green car, but the lane change maneuver is considered as completed, as soon as the centroid of the car has passed the lane boundary. The green car is now assigned to lane $L_{1,1}$ and simply follows the yellow car, which has stopped. None of the shown cars executes a trash maneuver. But the target has somehow to handle the risk and does so in an analogue way to the trash crossing risk (see above).

A similar situation may occur, when a long vehicle, like a truck or a bus, has to perform a narrow turn maneuver. Due to the long wheel base and the width of the vehicle, it will inevitably swing into adjacent lanes. In case of a following car, the system will handle this risk in the same way as above. But the present system does not yet properly predict the correct behavior of the long vehicle, which has to pay special intention before executing narrow turns.

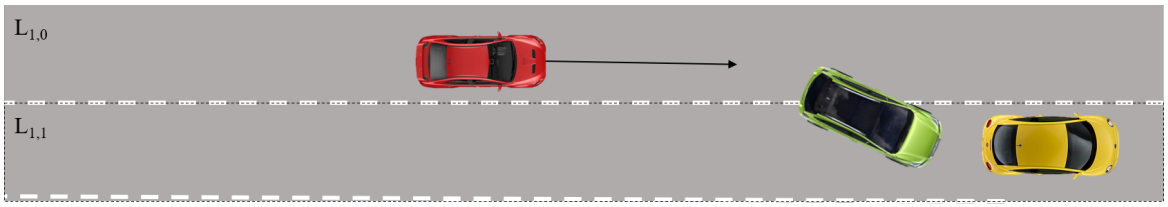


Fig. 5.15 Risk induced by incomplete lane change. The centroid of the green car has passed the lane marking, but it had to stop before completing the lane change.

5.7.7 Pseudo Risks in Curves

Figure 5.16 shows two cars driving close to each other in parallel lanes. The blue ellipses symbolize the Gaussian uncertainty distribution (see Subsection 5.8.3). The calculated collision risk should be below the criticality threshold in this case. But the Gaussian distribution was calculated, while the trajectory was predicted in Frenet coordinates. The risk calculation must be done in Cartesian coordinates to be able to consider also risks between vehicles on different, unrelated lanes. Assuming that the cars in Figure 5.16 are driving through a curve, the situation after transformation to Cartesian coordinates may look like in Figure 5.17. During the transformation, the covariance matrix of the Gaussian distribution has simply been rotated (see Subsection 2.4.3). The result is that the risk calculation will now report a significantly higher collision risk.

A proper transformation of the Gaussian to the curve would result in some kind of bent banana distribution, for which no parametric representation exists. In [39], this problem has been analyzed and a Monte Carlo representation for the resulting distribution was proposed. But the present system has to evaluate more than 100.000 risks per second and doing this with a Monte Carlo representation is computationally not feasible.

The solution is therefore to accept a higher collision risk, if two vehicles are driving in a certain configuration through a curve.

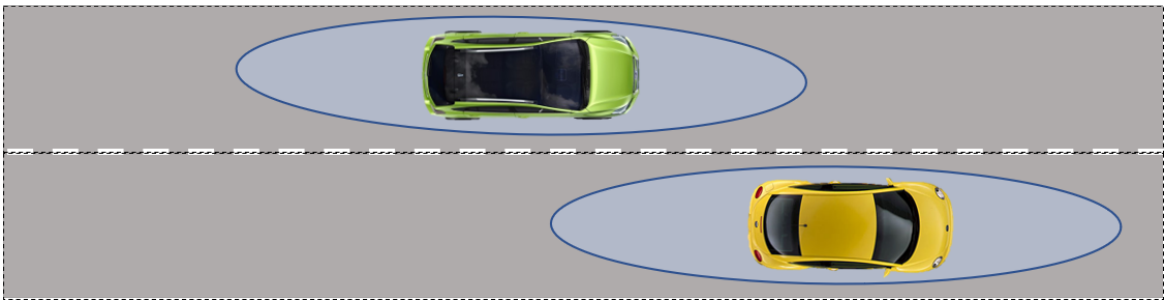


Fig. 5.16 Risk between cars on parallel lanes in Frenet frame. The ellipses symbolize the uncertainty about the positions.

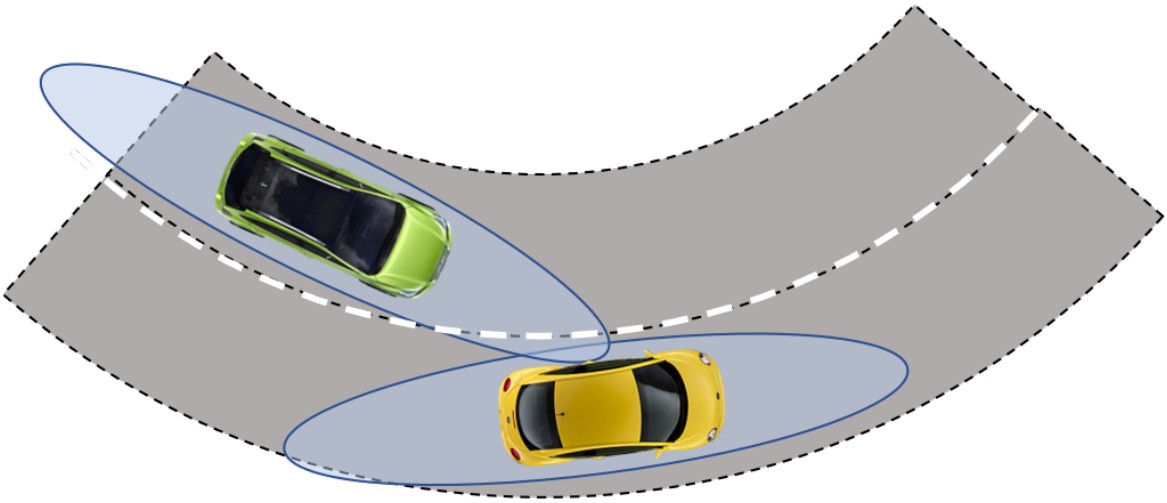


Fig. 5.17 Risk between cars on parallel lanes in Cartesian frame. The ellipses symbolize the uncertainty about the positions.

5.7.8 Forwarding Relevant Risks

All the previous subsections about interaction based motion constraints were based on the calculated collision risks of a trajectory at time step k. After having analyzed the type of

the risk and assigning it to the responsible agent, the knowledge about the risk is used in the subsequent prediction to mitigate the collision probability by the appropriate measures. Consequently, the next trajectory starting at time $k+1$ is expected to be risk free, provided that no new risks appear.

The risks, which have been handled in trajectory $k+1$, must now be forwarded to the next prediction since in the next collision risk analysis, they will appear as sub-critical. Otherwise, the system would start to oscillate between braking and accelerations. Before the risks are forwarded, it is checked whether they were relevant in the previous prediction. Relevant means that the specific risk has at least once triggered the highest deceleration or has inhibited a lane change at least once. Non relevant risks are discarded. The relevant risks have to undergo further checks before being forwarded, as testing whether the corresponding obstacles and maneuvers still exist.

5.8 Motion Prediction

This section describes the motion prediction. For each maneuver of an object, one trajectory is generated. The trajectory of the trash maneuver is creating by predicting the future states using the constant velocity or constant acceleration model without any regard of the roadmap, traffic rules or obstacles. For the lane bound maneuvers the longitudinal components of the predicted states are calculated using an extended Intelligent Driver Model (IDM). The center line of the lane is used as reference path for all maneuvers except lane change.

The motion prediction is done in discrete time with fixed step time. The step time relates usually to the prediction frequency since the prediction steps of the previous prediction should match the new prediction steps with a fixed offset without further interpolation. The predicted actions at each time step are the continuous longitudinal and lateral acceleration values. The trajectory data includes the acceleration, velocity and position of the object, as well as their covariance matrix. It is assumed that any driver wants to reach its goal as fast as possible under the constraints of safeness, comfort, economy and traffic rules.

The default prediction frequency of the system is 10 Hz and the step duration is $0.1s$. The default length of the predicted trajectory is $10s$, e.g. 100 steps.

The subsections of this section are: The trajectory prediction in Frenet frame is handled in Subsection 5.8.1. The Basic Intelligent Driver Model (IDM) is presented in Subsection 5.8.2. Various extensions to the IDM, which are required especially for multi lane urban scenarios, are given in Subsection 5.8.3. The influence of individual driving styles to the motion behavior of an agent, which relates to the trait estimate mentioned in Section 5.1, is described in Subsection 5.8.4.

5.8.1 Prediction in Frenet Frame

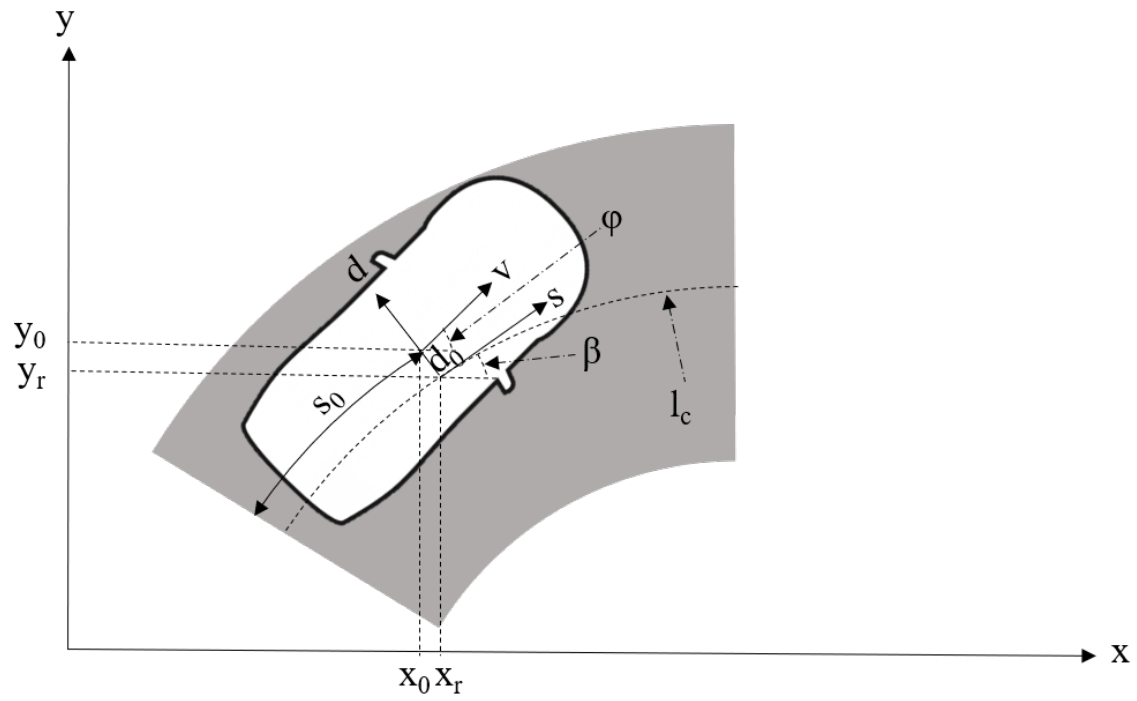


Fig. 5.18 Motion prediction in Frenet coordinates. $[x_0, y_0]$ is the centroid position of the car, $[x_r, y_r]$ is the closest point to the centroid on the center lane L_c.

Prediction and planning of trajectories of lane bound maneuvers are often in the Frenet frame [69] [185]. As depicted in Figure 5.18, the kinematic state of a vehicle is therefore transformed from the map-based Cartesian coordinate frame x-y to a moving reference frame s-d, aligned tangentially to the center line of the lane sequence. The advantage of this approach is that the kinematic state is split into components longitudinal and lateral to the center line l_c of the lane, regardless of the yaw angle and the curvature of the road.

In Figure 5.18, the position $[x_0, y_0]$ is the current vehicle position in map-coordinates. The position $[x_r, y_r]$ is the closest point to the vehicle position on the center line l_c and is the origin of the Frenet frame. The s-axis is aligned tangentially to l_c, which has at this position the yaw angle β. $[s_0, d_0]$ denotes the position in the Frenet frame, where s_0 is the distance from the start of the lane sequence, measured along the line l_c, and d_0 is the lateral offset of the vehicle centroid. The velocity and acceleration have to be rotated by the angle β.

The fully transformed state is

$$[s, d, v_{lon}, v_{lat}, a_{lon}, a_{lat}]^T = f([x, y, v_x, v_y, a_x, a_y]^T) \tag{5.9}$$

To get the proper lateral acceleration, the value of a_{lat} has to be corrected by the centripetal acceleration a_c, induced by the curvature.

$$a_c = \kappa v_{lon}^2 \tag{5.10}$$

with κ being the curvature of l_c at $[x_r, y_r]$.

For a vehicle driving with constant velocity on the center line of a lane (straight or curved), the complete kinematic state in the Frenet frame becomes therefore $[s(t), 0, v_{lon}, 0, 0, 0]$.

5.8.2 Basic Intelligent Driver Model (IDM)

The Intelligent Driver Model (IDM) [169] is a car following model and was initially designed for the simulation of single lane traffic on highways. Other popular car following models are Optimal Velocity Model [18], the Newell Model [138] and the Gibbs Model [76].

The IDM calculates the longitudinal acceleration a_{idm} of a vehicle as a function of the current speed v of the target vehicle as well as the distance s and approach speed Δv to an eventually existing preceding obstacle:

$$a_{idm} = \dot{v} = a \left[1 - \left(\frac{v}{v_0} \right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s} \right)^2 \right] \tag{5.11}$$

The function $s^*()$ calculates the desired distance to the preceding obstacle. It covers also the case of static obstacles or stop signs, where $\Delta v = v$.

$$s^*(v, \Delta v) = s_0 + \max \left(0, vT + \frac{v \Delta v}{2\sqrt{ab}} \right) \tag{5.12}$$

The parameters of the model and their typical values as proposed in [171] are given in Table 5.6.

IDM Parameters		
Parameter	Highway	Urban
Desired speed v_0	120 km/h	50 km/h
Time gap T	1.0 s	1.0 s
Minimum gap s_0	2 m	2 m
Acceleration exponent δ	4	4
Preferred acceleration a	1.0 m/s^2	1.0 m/s^2
Comfortable deceleration b	1.5 m/s^2	1.5 m/s^2

Table 5.6 IDM parameters for highway and urban traffic

The equation balances two components:

$$a_{idm} = a_{free} - a_{int} \tag{5.13}$$

$$\text{Free driving component: } a_{free} = a\left[1 - \left(\frac{v}{v_0}\right)^{\delta}\right] \tag{5.14}$$

$$\text{Interaction component: } a_{int} = a\left[\left(\frac{s^*(v,\Delta v)}{s}\right)^2\right] \tag{5.15}$$

The free driving component can be positive or negative and forces the vehicle to reach the desired velocity v_0. The interaction component is always positive (due to the max() clause in Equation 5.12, see [171] page 189). It implements the so-called Intelligent Braking Strategy:

$$a_{int} = \frac{b_{kin}^2}{b} \tag{5.16}$$

b is the comfortable acceleration (see Table 5.6) and b_{kin} is the kinematic deceleration, e.g. the required constant deceleration to achieve or keep the desired distance. The Intelligent Braking Strategy enforces stronger than kinematically necessary braking in critical situations (b_{kin} > b), and reduces braking in uncritical situations (b_{kin} < b). This conforms with the observation of human drivers. By combining the Equations 5.15 and 5.24 and solving for b_{kin} we get (omitting the max()):

$$b_{kin} = \frac{v\Delta v}{2s} + \sqrt{ab}\frac{s_0 + vT}{s} \tag{5.17}$$

This equation is a heuristic, which has no sound foundation. But it has proven useful and realistic in many experiments for predicting human driving behavior and also for implementing Adaptive Cruise Systems (ACC).

The IDM Equation 5.11 formulates a non-linear second order differential equation (distance s is based on the position) and therefore no analytical solution exists. For the purpose of generating a discrete time trajectory, a numerical integration scheme is needed. [170] proposes to use a simple ballistic scheme to forward the kinematic state of the target vehicle:

$$s_{k+1} = s_k + v_{lon,k}\,\Delta t + \frac{1}{2}\,a_{idm}\,\Delta t^2 \tag{5.18}$$

$$v_{lon,k+1} = v_{lon,k} + a_{idm}\,\Delta t \tag{5.19}$$

5.8.3 Extensions to the IDM

To be usable for the prediction of urban traffic scenarios, the IDM must be extended in several ways. Some extensions to overcome the deficiencies of the model have already been proposed in [171], but these are not sufficient. This subsection describes extensions to the IDM formula, especially: multiple brake reasons, braking for reduced speed limits, braking for crossing obstacles, braking while approaching intersections, lane change decision, braking for lane merges, lateral trajectory prediction and probabilistic trajectories

Multiple Brake Reasons

The standard IDM considers only one brake reason, the preceding vehicle. In real traffic scenarios, two or more brake reasons have to be kept in mind. A simple example is a lane changing vehicle, which has eventually to brake for the preceding car in the old lane and for another one in the destination lane. Another example is a vehicle following a slower obstacle, when both approach a traffic light switching to yellow. Here, the following vehicle has either to brake for the preceding car or, if the preceding car passes at yellow, for the traffic light.

Equation 5.13 is modified to use the maximum brake value calculated for the set of potential brake reasons R.

$$a_{idm} = a_{free} - \max_{r \in R}(\{a_{int,r}\}) \tag{5.20}$$

Braking for Reduced Speed Limit

When generating a trajectory, various kinds of speed limits may have to be observed (see Subsection 5.6.1). Speed limits are handled in standard IDM by manipulating the desired speed v_0. But this results in unrealistic decelerations. In [171], the Improved Intelligent Driver Model (IIDM) has been proposed, which mitigates the problem. But even with IIDM, the deceleration starts only at the point, where the speed limit comes into effect, and not when approaching it. Especially the sharp curvature when turning right requires braking well before the intersection.

When driving with speed v and approaching a speed limit v_{limit} with $v_{limit} < v$ in distance $s_{limit} > 0$, the required kinematic deceleration is calculated as:

$$b_{limit} = \frac{v^2 - v_{limit}^2}{2\, s_{limit}} \tag{5.21}$$

Again, the intelligent braking strategy is applied:

$$a_{int} = \frac{b_{limit}^2}{b} \tag{5.22}$$

Braking for Crossing Obstacles

Another brake reason not properly covered by the IDM are obstacles crossing the lane of the target vehicle. This may be a vehicle on a priority lane at an unsignalized intersection or a pedestrian crossing the road at an arbitrary position. Most planning and prediction algorithms solve this kind of conflict by scheduling a full stop in front of the conflict zone. But human drivers are able to anticipate the point in time, when the obstacle will have left the conflict zone and will try to avoid a full stop.

When driving with speed v and approaching a lane crossing conflict zone in distance s_{crssng}, which is expected to be cleared at time $t_{crssng} > t$, the required kinematic deceleration is

$$b_{crssng} = \begin{cases} \dfrac{2\left(v\left(t_{crssng} - t\right) - s_{crssng}\right)}{\left(t_{crssng} - t\right)^2} & \text{if } s_{crssng} > \frac{v\left(t_{crssng} - t\right)}{2} \\ \dfrac{v^2}{2\, s_{crssng}} & \text{otherwise full stop} \end{cases} \tag{5.23}$$

For this brake reason, the intelligent braking strategy is also applied:

$$a_{int} = \frac{b_{crssng}^2}{b} \tag{5.24}$$

Braking while Approaching Intersections

Even if no collision risk is present, approaching an intersection may require braking:

- Stop sign: The deceleration is calculated using Equation 5.15 with $\Delta v = v$.
- Line of sight: The target has to reduce the speed to be able to halt at the stop line. The assumed speed limit at the line of sight is given by $v_{limit} = \sqrt{2\, b\, \Delta s_{sight}}$. The deceleration for the speed limit is calculated by Equation 5.21. After having crossed the line of sight, the target may accelerate again.
- Traffic light: If a traffic light is registered in the roadmap for the current lane of the target, it will decelerate to halt at the stop line using Equation 5.15, unless the perception system reports green for this traffic light. If green was detected before, but is not anymore (probably switched to yellow), stopping is suppressed, if the distance

to the stop line is $\Delta s_{stop} < v\frac{2}{3}t_{yellow}$. t_{yellow} is the length of the yellow phase, which depends on the legal speed limit.

Lane Change Decision

Braking for leading vehicles in the source and destination lane is done using Equation 5.15.

The required front gap Δs to the new leader is checked by (see [171] P. 249):

$$\Delta s > \frac{s^*(v, \Delta v)}{\sqrt{1 + \frac{b}{a}}} \tag{5.25}$$

$s^*(v, \Delta v)$ is given by Equation 5.12.

The required rear gap to the new follower is calculated by Equation 5.25 using the velocity of the new follower as v. The motion prediction for the LC maneuver calculates the front and rear gap on every prediction step, until they are sufficiently large to start the lane change.

Braking for Lane merges

In [171], the authors claim that merge situations can be handled by the lane change Equation 5.25 when taking the difference of the distances to the merge point as Δs. But this holds only if the merging car has not the priority. The present system solves the problem by using Equation 5.23 to handle the access to the conflict zone in combination with Equation 5.15 for the subsequent car following situation.

Lateral Trajectory Prediction

The main reason for lateral motions in lane bound maneuvers are lane changes. The IDM does not care for any lateral motion. In [171], the discrete lane change decision is handled, but after the decision, the lane change is modeled as taking place instantaneously.

More sophisticated models for the lateral motion during lane changes have been proposed elsewhere. [158] proposes a sine half cycle in road coordinates for the trajectory calculation. The problem with the sine function is the high jerk rate at start and end of the maneuver. Moreover, [158] proposed to use a fixed maneuver duration of 3 seconds for predicting lane changes, which have not yet started. Other authors, like [129], propose 5^{th} order polynomials to generate the lateral components of the trajectory. This approach also assumes a fixed maneuver time, independent of the longitudinal component.

The general requirement for a LC trajectory is to perform the LC as fast as possible without exceeding a given threshold for the lateral acceleration.

The special challenge of LC motion prediction in an urban environment is twofold:

- The speed of the vehicle cannot be considered as constant. Sometimes, vehicles perform an LC when decelerating for a red traffic light or even come to a complete stop during the maneuver. Any trajectory prediction based on a fixed maneuver time will fail in this case.
- For very slow lane changes, the lateral acceleration is not the limiting factor. The trajectory is rather restricted by the maximum steering angle.

This work proposes to use the tanh() function to generate a LC trajectory. Also in [137], the usage of tanh() was proposed, but no details were given.

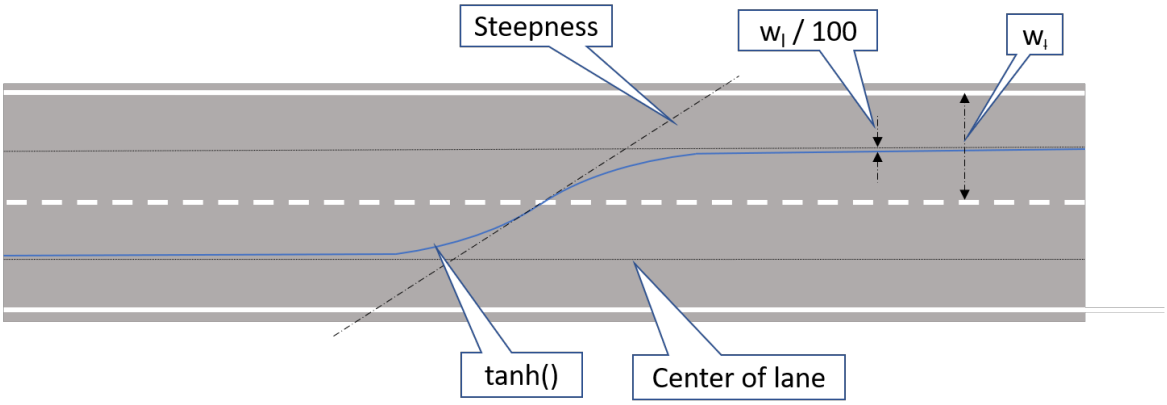


Fig. 5.19 Lane change trajectory at constant velocity modeled by the tanh() function.

Figure 5.19 shows a lane change performed at constant velocity on a tanh() trajectory. Note that the curve will be deformed in spatial coordinates, if the velocity changes during the maneuver. The lateral position $d(t)$ in Frenet frame during a lane change left with $t = k\Delta t$ and a lane width of w_l for both lanes is given by:

$$d(t) = \frac{w_l}{2}\,(\tanh(\beta\, t) + 1) \tag{5.26}$$

The steepness factor β decides about the lateral acceleration of the lane change and therefore about the abruptness of the maneuver. The goal is to find the β value for a desired maximal acceleration a_{maxlat}.

The first three derivatives of Equation 5.26 are:

$$\text{Lateral velocity } d'(t) = \frac{\beta w_l(1-\tanh^2(\beta t))}{2} \tag{5.27}$$

$$\text{Lateral acceleration } d''(t) = -\beta^2 w_l(1-\tanh^2(\beta t))\,\tanh(\beta t) \tag{5.28}$$

$$\text{Lateral jerk } d'''(t) = \beta^3 w_l \, (1 - \tanh^2(\beta t))(3\tanh^2(\beta t) - 1) \tag{5.29}$$

To find the point in time of maximal lateral acceleration, Equation 5.29 is set to 0 and solved for t:

$$t_{amax} = \frac{\operatorname{arctanh}(\sqrt{\frac{1}{3}})}{\beta} \tag{5.30}$$

Inserting Equation 5.30 in Equation 5.28 and solving for β with $a_{maxlat} = d''(t_{amax})$ results in:

$$\beta = \sqrt{\frac{f \, a_{maxlat}}{w_l}} \text{ with } f = \frac{\sqrt{3^3}}{4} \approx 1.3 \tag{5.31}$$

With Equation 5.31, it is possible to calculate a steepness factor β for a given maximal lateral acceleration a_{maxlat} depending on the width of the lane change w_l. But this equation does not yet solve the problem of unfeasible trajectories at low velocities due to limited steering angles. The maximal steering angle of most cars is $\approx 35°$. Since the wheelbase l_{wb} of an observed car is usually hard to measure, it is assumed to be 60 % of the vehicle length l_{veh}. The radius of the minimal turning cycle is therefore estimated with:

$$r_{min} = \frac{0.6 \, l_{veh}}{\sin(35°)} \tag{5.32}$$

With the speed $v_{lon,k}$ at time step k, the lateral acceleration at a maximal steering angle is approximated by:

$$a_{maxsa}(v_{lon,k}) = \frac{v_{lon,k}^2}{r_{min}} \tag{5.33}$$

The final equation for the steepness factor is now time dependent and becomes:

$$\beta_k = \sqrt{\frac{f \, \min(a_{maxlat}, a_{maxsa}(v_{lon,k}))}{w_l}} \tag{5.34}$$

The lateral acceleration a_{lat} at time t is then calculated using Equation 5.28 and propagated to v_{lat} and d_{lat} using zero order hold analogous to Equations 5.19 and 5.18. It remains to calculate the actual time t for tanh() at the begin of the prediction. This is done by solving Equation 5.26 for t:

$$t_0 = \frac{\operatorname{arctanh}\left(\frac{2\ \max(d_0, 0.01\ w_l)}{w_l} - 1\right)}{\beta} \tag{5.35}$$

and rounding t_0 to the nearest multiple of Δt.

The lower bound of $0.01\ w_l$ for the initial lateral offset d_0 is due to the asymptotic nature of tanh at -1. For the same reason, the lane change is considered as completed, when $d(t)$ reaches $w_l - 0.01$.

The equations for the lane change right are analogous.

The trajectory prediction for non LC maneuvers assumes that the vehicle follows the center line of the lane with $d_{lat} = v_{lat} = a_{lat} = 0$. When the measurements report a value of $d_{lat} <> 0$, it is assumed that the vehicle will return to the center line in the same way as it approaches the center line of the destination lane of a lane change after having crossed the lane boundary. Equations 5.26 to 5.35 are used analogously.

Probabilistic Trajectories

To be able to calculate the collision risk between the trajectories of two obstacles, the uncertainty of the predicted states must be estimated. The above algorithms calculate the future states $[s, d, v_{lon}, v_{lat}]^T$ as a result of applying the predicted actions $[a_{lon}, a_{lat}]^T$.

The initial covariance matrix P_0 of the trajectory is initialized from the observed state of the perception system (see Chapter 3):

$$\mathbf{P}_0 = \begin{bmatrix} \sigma_s^2 & 0 & 0 & 0 \\ 0 & \sigma_{v_{lon}}^2 & 0 & 0 \\ 0 & 0 & \sigma_d^2 & 0 \\ 0 & 0 & 0 & \sigma_{v_{lat}}^2 \end{bmatrix} \tag{5.36}$$

Since the longitudinal and lateral motion are predicted independent of each other, the corresponding parts of the covariance matrix are forwarded separately. Both system functions are non-linear and therefore the Linear Kalman Filter approach to predict the covariance matrix of the state cannot be used.

For the longitudinal prediction, the uncertainty has to be propagated through the IDM function. Some authors as [47] propose to use an Uncented Kalman Filter (UKF) for this purpose. The UKF requires to calculate sigma points near the current system state. This results in sigma points with negative longitudinal velocity in case of a standing vehicle. But the IDM function is undefined for negative velocities and the results are meaningless.

In this work, the covariances are propagated using an Extended Kalman Filter. This requires calculating the Jacobian matrix for the system function:

$$\mathbf{J}(s_k, v_{lon,k}) = \begin{bmatrix} \frac{\partial f_s(s,v_{lon})}{\partial s} & \frac{\partial f_s(s,v_{lon})}{\partial v_{lon}} \\ \frac{\partial f_v(s,v_{lon})}{\partial s} & \frac{\partial f_v(s,v_{lon})}{\partial v_{lon}} \end{bmatrix} \tag{5.37}$$

with:

$$f_s = s + v\Delta t + \frac{1}{2}\Delta t^2 \; f_{idm}(s,v) \tag{5.38}$$

$$f_v = v + \Delta t \; f_{idm}(s,v) \tag{5.39}$$

The function f_{idm} is independent of the current longitudinal position s, therefore the Jacobian becomes:

$$\mathbf{J}(v) = \begin{bmatrix} 1 & \Delta t + \frac{1}{2}\Delta t^2 \; \frac{\partial f_{idm}(v)}{\partial v} \\ 0 & 1 + \Delta t \; \frac{\partial f_{idm}(v)}{\partial v} \end{bmatrix} \tag{5.40}$$

with $f_{idm}(v) = f_{free}(v) + f_{int}(v)$

$$\frac{\partial f_{idm}(v)}{\partial v} = -a\,\delta \frac{v^{\delta-1}}{v_0^{\delta}} - \frac{2\,b_{kin}(v)}{b}\frac{\partial b_{kin}(v)}{\partial v} \tag{5.41}$$

Since $b_{kin}(v)$ depends on the actual break reason b_{fllw} (Equation 5.17), b_{spllmt} (Equation 5.21) or b_{crssng} (Equation 5.23), the partial derivative gets:

$$\frac{\partial b_{kin}(v)}{\partial v} = \begin{cases} \dfrac{\Delta v}{2\,\Delta s} + \dfrac{T}{\Delta s} & b_{fllw} \\ \dfrac{v}{\Delta s} & b_{limit} \\ \dfrac{2}{t_{crssng} - t} & b_{crssng} \\ \dfrac{v}{\Delta s} & b_{crssng} \end{cases} \tag{5.42}$$

The covariance matrix for the longitudinal state P_{lon} is forwarded by:

$$\mathbf{P}_{lon}^{k+1} = \mathbf{J}(v) \times \mathbf{P}_{lon}^{k} \times \mathbf{J}(v)^T + \begin{bmatrix} 0.5\,\Delta t^2 \\ \Delta t \end{bmatrix} \times \sigma_a^2 \times \begin{bmatrix} 0.5\,\Delta t^2 \\ \Delta t \end{bmatrix}^T \tag{5.43}$$

The acceleration noise σ_a is assumed to be $0.1\ m/s^2$ (see [171] P. 216). It covers the uncertainty about the driving style and other parameters, like $\Delta v, \Delta s$ and t_{crssng}.

Concerning the lateral motion, [158] P. 158 proposes to model the deviation from the reference trajectory by a continuous-time Ornstein-Uhlenbeck process. This results in forwarding the lateral variance by:

$$P_{dlat}^{k+1} = e^{-2\frac{\Delta t}{T_c}} P_{dlat}^{k} + \sigma_{dlat}^2 (1 - e^{-2\frac{\Delta t}{T_c}}) \tag{5.44}$$

with time constant $T_c = 1.5$. The variance σ_{dlat} results from the assumption that a driver of a vehicle with width w_v will stay inside the lane width w_l with a probability of 3σ:

$$\sigma_{dlat} = \frac{1}{3} \frac{(w_l - w_v)}{2} \tag{5.45}$$

The standard deviation for the lateral velocity is estimated as constant with $\sigma_{vlat} = 0.1 \frac{m}{s}$ and the lateral covariance between position and velocity with $0.5\ \sigma_{vlat} \sqrt{P_{dlat}^{t}}$. The lateral covariance matrix therefore becomes:

$$\mathbf{P}_{lat}^{k+1} = \begin{bmatrix} P_{dlat}^{k+1} & 0.5\ \sigma_{vlat} \sqrt{P_{dlat}^{k+1}} \\ 0.5\ \sigma_{vlat} \sqrt{P_{dlat}^{k+1}} & \sigma_{vlat}^2 \end{bmatrix} \tag{5.46}$$

The complete covariance of trajectory step k is given by:

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{P}_{lon}^{k} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{lat}^{k} \end{bmatrix} \tag{5.47}$$

5.8.4 Influence of the Driving Style

In [35], the trait estimation is considered a core task for driver behavior modeling. In IDM based systems, this task relates to estimate the static parameters of the IDM equation, which may differ from driver to driver. In [93] a system was presented, which tries to estimate all IDM parameters online from the received observations of a vehicle. It turned out that it often requires up to $20s$ of observation time, before the value of a parameter can be inferred. Moreover, most parameters are related to specific driving situations, as free driving or car following, and can be estimated only after having observed the target in such a situation. In [28], the approach was simplified to only estimate the desired velocity parameter v_0.

Human drivers are also not able to predict the desired velocity or braking deceleration of other traffic participants on the fly. But they are often able to categorize the driving style of other drivers as aggressive, normal or defensive. It is therefore proposed to define the driving style as measure of the aggressiveness or defensiveness of the driver. In [61] was shown that the observed longitudinal jerk rate is strongly correlated to the aggressiveness of a driver. It

Parameter	Passenger Car		Bus/Truck		Bicycle	
	p_{def}	p_{aggr}	p_{def}	p_{aggr}	p_{def}	p_{aggr}
Velocity Factor F_{vd}	75 %	125 %	85 %	115 %	70 %	130 %
Time Gap T	1.5 s	0.5 s	2.0 s	1.0 s	2.5 s	0.5 s
Minimum Gap s_0	3.0 m	1.0 m	4.0 m	1.5 m	1.5 m	0.5 m
Preferred Acceleration a	0.8 m/s^2	1.6 m/s^2	0.6 m/s^2	1.2 m/s^2	0.4 m/s^2	1.4 m/s^2
Comfortable Deceleration b	1.0 m/s^2	2.0 m/s^2	0.7 m/s^2	1.3 m/s^2	0.4 m/s^2	1.0 m/s^2

Table 5.7 Ranges for IDM parameters depending on vehicle type

should therefore be possible to get a rough estimate of the driving style from measurements of the vehicle jerk.

The other main influence factor to the parameters of the motion behavior is the type of vehicle. Desired velocity, acceleration rate etc. are very different for passenger cars, bicycles or trucks and buses. It is therefore proposed to use vehicle type dependent ranges for the parameters of the motion type, where p_{aggr} and p_{def} are the values for the most aggressive and defensive drivers.

In Table 5.7, the ranges $\{p_{def} \ldots p_{aggr}\}$ for the IDM parameters depending on the vehicle type are given. The velocity factor F_{vd} replaces the desired velocity v_0, which varies depending on the law and other factors. The velocity factor defines, which percentage of the actual reference velocity is used by the individual driver. For bicycles, which rarely reach the legal speed limit, the reference velocity is set to 18 km/h, for German autobahns without any legal speed limit, it is set to 130 km/h for passenger cars and 80 km/h for buses and trucks. The acceleration exponent δ is fixed to 4, as usual. All values in the table are ad hoc estimates and should be verified empirically.

The actual value p_{act} of each parameter depends on the driving style of the agent $d_s \in \{-1 \ldots +1\}$:

$$p_{act} = p_{def} + \frac{d_s + 1}{2}(p_{aggr} - p_{def}) \tag{5.48}$$

The present prediction system does not yet infer the driving style from the observation and uses the default value 0. In the traffic scenario simulation (see Chapter 6), the driving style is assumed to be a uniformly distributed random variable:

$$d_s \sim \mathcal{U}(-1, +1) \tag{5.49}$$

5.9 Risk Estimate

The approach in the previous section allows to create a Gaussian mixture probability distribution starting at time k for the prediction horizon T over the trajectories of a traffic participant i conditioned on its maneuver intention $M_{i,k}$:

$$p(\mathbf{x}_{i,k:k+T}) = \sum_{j=1}^{r} p(\mathbf{x}_{i,k:k+T}|M_{i,k}=j)p(M_{i,k}=j) \tag{5.50}$$

As a final step, the collision probability between all pairs of maneuvers has to be calculated for all time steps of the prediction. The result of the calculation is the collision matrix C_k.

The calculation is done by the following algorithm:

Algorithm 3 Calculate Collision Matrix

Require: $\mathcal{O}_k, \mathcal{M}_k, \mathcal{T}_k$ ▷ Set of obstacles, maneuvers and trajectories
Ensure: C_k ▷ Collision matrix

1:
2: p $\leftarrow 0$ ▷ Initialize maneuver pair index
3: **for** $i \leftarrow 1$ **to** $|\mathcal{O}_k| - 1$ **do** ▷ For all obstacles
4: **for** $j \leftarrow 1$ **to** $|\mathcal{M}_k^i|$ **do** ▷ For all maneuvers
5: **for** $l \leftarrow i+1$ **to** $|\mathcal{O}_k|$ **do** ▷ For all conflicting obsacles
6: **for** $m \leftarrow 1$ **to** $|\mathcal{M}_k^l|$ **do** ▷ For all conflicting maneuvers
7: $p \leftarrow p+1$ ▷ Update maneuver pair index
8: **for** $t \leftarrow 1$ **to** T **do** ▷ For all trajectroy steps
9: $C_k(p,t) \leftarrow \text{CALCCEP}(T_t^{i,j}, T_t^{l,m})$ ▷ Calculate collsion event density
10: **if** $t > 1$ **then** ▷ If not first step
11: $C_k(p,t) \leftarrow C_k(p,t) + C_k(p,t-1)$ ▷ Accumulate densities
12: **end if**
13: **end for**
14: **end for**
15: **end for**
16: **end for**
17: **end for**

Input are the sets of objects $\mathcal{O}$, maneuvers $\mathcal{M}$ and trajectories $\mathcal{T}$ for time k. The for loops in lines 3-6 iterate through all distinct pairs of maneuvers. Each pair gets assigned a unique index p. The loop in line 9-12 iterates through all prediction time steps and calls procedure CALCCEP for each pair of trajectory points. CALCCEP calculates the collision event rate using Equation 4.9 from Chapter 4. Line 11 adds the rates to get the accumulated collision probability until time step t.

5.10 Evaluation

This section documents the evaluation of the approach based on real-world data. Further evaluations based on simulations are given in Section 6.6.

The evaluation is done for 4 typical urban traffic scenarios. The data was recorded in ROS bag files during test drives with the FU autonomous vehicle MadeInGermany in Berlin Dahlem and Berlin Reinickendorf. During these test drives, MadeInGermany was controlled manually. For each scenario, interaction-aware predictions for all agents are generated with a frequency of 10 Hz. The considered agents are all obstacles on the road, the moving off-road obstacles and the ego vehicle.

The purpose of the evaluation is to compare predicted trajectories with a length of up to 10 seconds with ground truth data. The only agent, for which ground truth data is available, is the ego vehicle. The true trajectory of the ego-vehicle can be reconstructed using Applanix and Odometry data (see Section 2.1). Therefore, the evaluation is done for the predicted trajectory of the ego vehicle.

The four scenarios are:

- Turn and Lane Merge on Thielallee
- Lane Change on Scharnweberstraße
- Intersection Crossing Ehrenbergstraße
- Pedestrian Cross Walk Scharnweberstarße

For each scenario, there is an analysis of the maneuver probability over time and a comparison of the trajectory prediction quality for different prediction methods.

The considered methods are:

- Physical prediction model, using constant velocity.
- Uni-modal prediction based on the roadmap without considering maneuvers (Road follower).
- Multi-modal maneuver-based prediction (Non interaction-aware).
- Multi-modal maneuver-based prediction (Interaction-aware).

For each prediction method is evaluated:

- Position error over the whole prediction horizon starting at different points in time.

- Position likelihood over the whole prediction horizon starting at different points in time (not for road follower).
- Position error over the whole scenario for 1, 3 and 10 seconds prediction horizon.
- Position likelihood over the whole scenario for 1, 3 and 10 seconds prediction horizon (not for road follower).

The position error e_p is calculated from the errors in x- and y-direction:

$$e_p = \sqrt{\tilde{x}^2 + \tilde{y}^2)} \tag{5.51}$$

The position likelihood l_p is calculated from the errors in x- and y-direction and the covariance matrix of the positions Σ_{xy}:

$$l_p = \frac{1}{\sqrt{(2\pi)^2 |\Sigma_{xy}|}} \, e^{-\frac{1}{2}[\tilde{x}\ \tilde{y}]\ \Sigma_{xy}^{-1}\ [\tilde{x}\ \tilde{y}]^T} \tag{5.52}$$

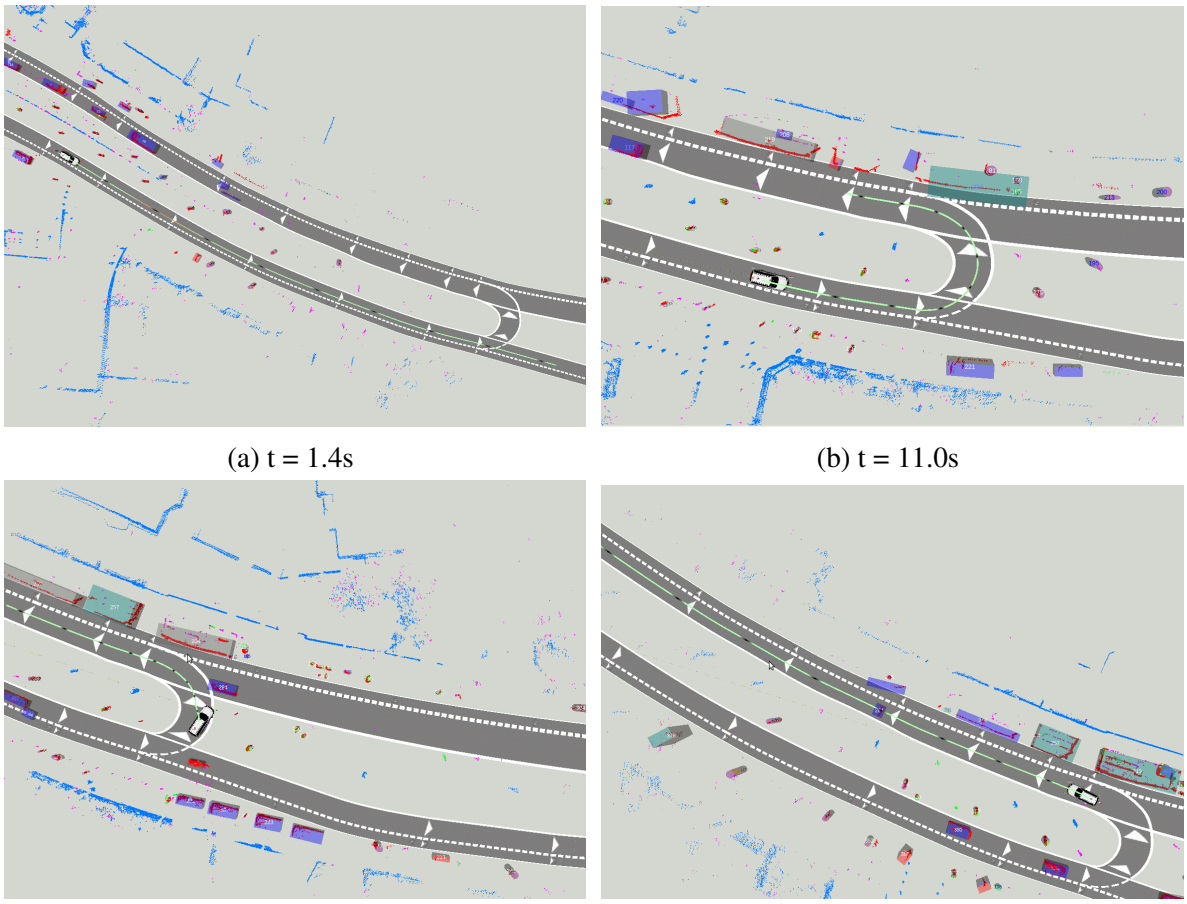
5.10.1 Evaluation of Turn and Lane Merge Scenario

In this scenario, the target drives on the Thielallee in Berlin Dahlem and takes a turn to the reverse direction. During that turn, it has to merge the lane with an oncoming vehicle.

The first Picture 5.20a, taken at $t = 1.4s$, shows the target approaching the turn, but the most probable trajectory is still that of the keep lane maneuver. The Diagram 5.21 shows the probabilities of the keep lane and turn maneuver, the only feasible lane bound maneuvers at the start of the scenario.

The position error of the 10s prediction starting at $t = 1.4s$ (see Figure 5.22a) is in the beginning quite low, but grows for the multi-modal predictions with time, even more than for the simpler road follower and constant velocity predictions. The reason for this is that the target drives at $12m/s$, while $14m/s$ are allowed and the multi-modal methods predict therefore an acceleration. The likelihood in contrast (see Figure 5.22b) is much better for the multi-modal methods due to the characteristics of the covariance matrix. The longitudinal variances are much higher than the lateral ones and the prediction error occurs almost entirely in longitudinal direction. The constant velocity model assumes symmetric uncertainties, while the road follower predictions do not provide any probabilistic information.

At $t = 11s$ (see Figure 5.20b), the turn maneuver becomes the most probable. The vehicle is still about $25m$ away from the start of the turn lane and has so far no lateral deviation from the forward lane. But due to the deceleration of the target, the evidence fits much better to the



(a) t = 1.4s (b) t = 11.0s

(c) t = 20.7s (d) t = 25.0s

Fig. 5.20 Turn maneuver with subsequent lane merge.

turn maneuver. There is no obstacle ahead of the vehicle on the forward lane, which could explain the deceleration.

The position error of the prediction at $t = 11s$ (see Figure 5.22c) for the multi-modal interactive prediction remains very low during the whole prediction horizon, while it increases up to almost $100m$ for non-modal methods, which predict going straight at that time. The multi-modal non-interactive prediction is worse than the interactive one since it is a mixture in which the constant velocity model has a significant weight. The likelihood (Figure 5.22d) shows some strange evolution, but is at least until $t = 17s$ significantly better for the multi-modal predictions.

At about $t = 14s$, the turn maneuver has been executed, i.e. the target is not anymore on the forward lane. The turning lane becomes now the new lane for the keep lane maneuver, turning left becomes unfeasible (see Figure 5.21). The probability of a turn right maneuver at the next intersection (not shown) starts to increase.

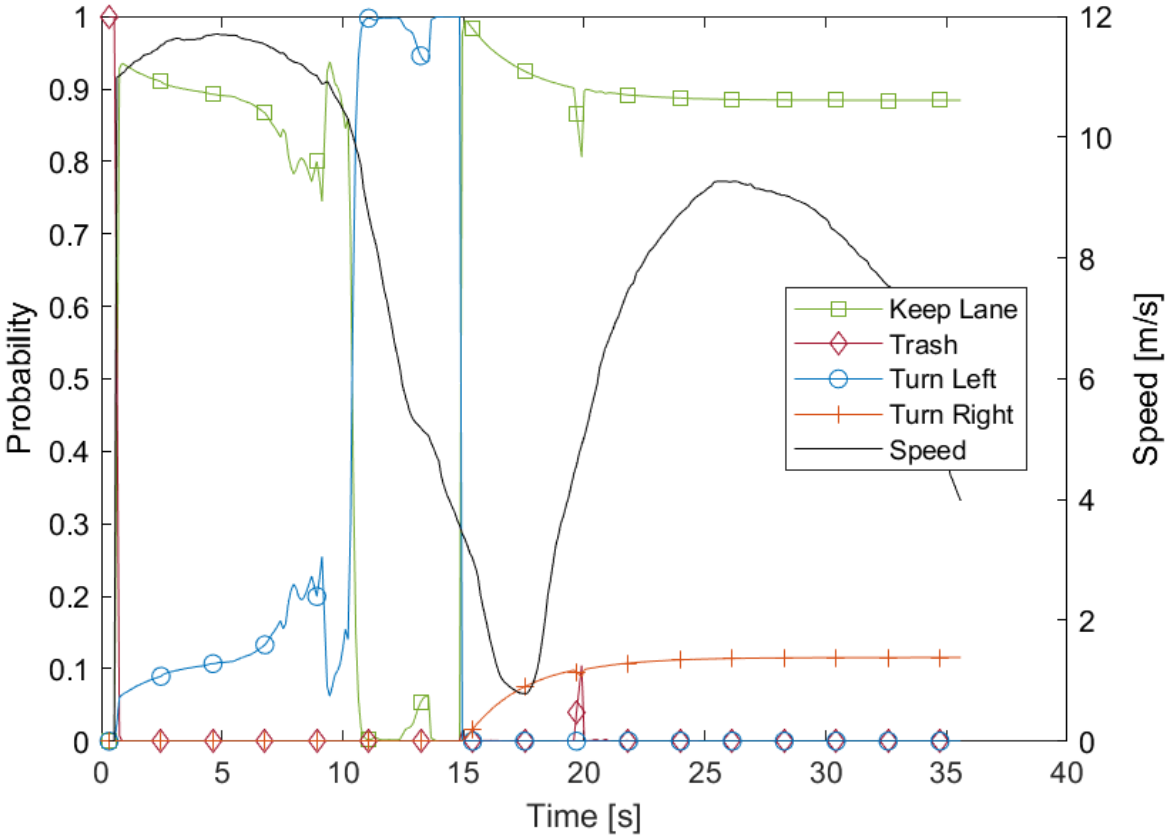
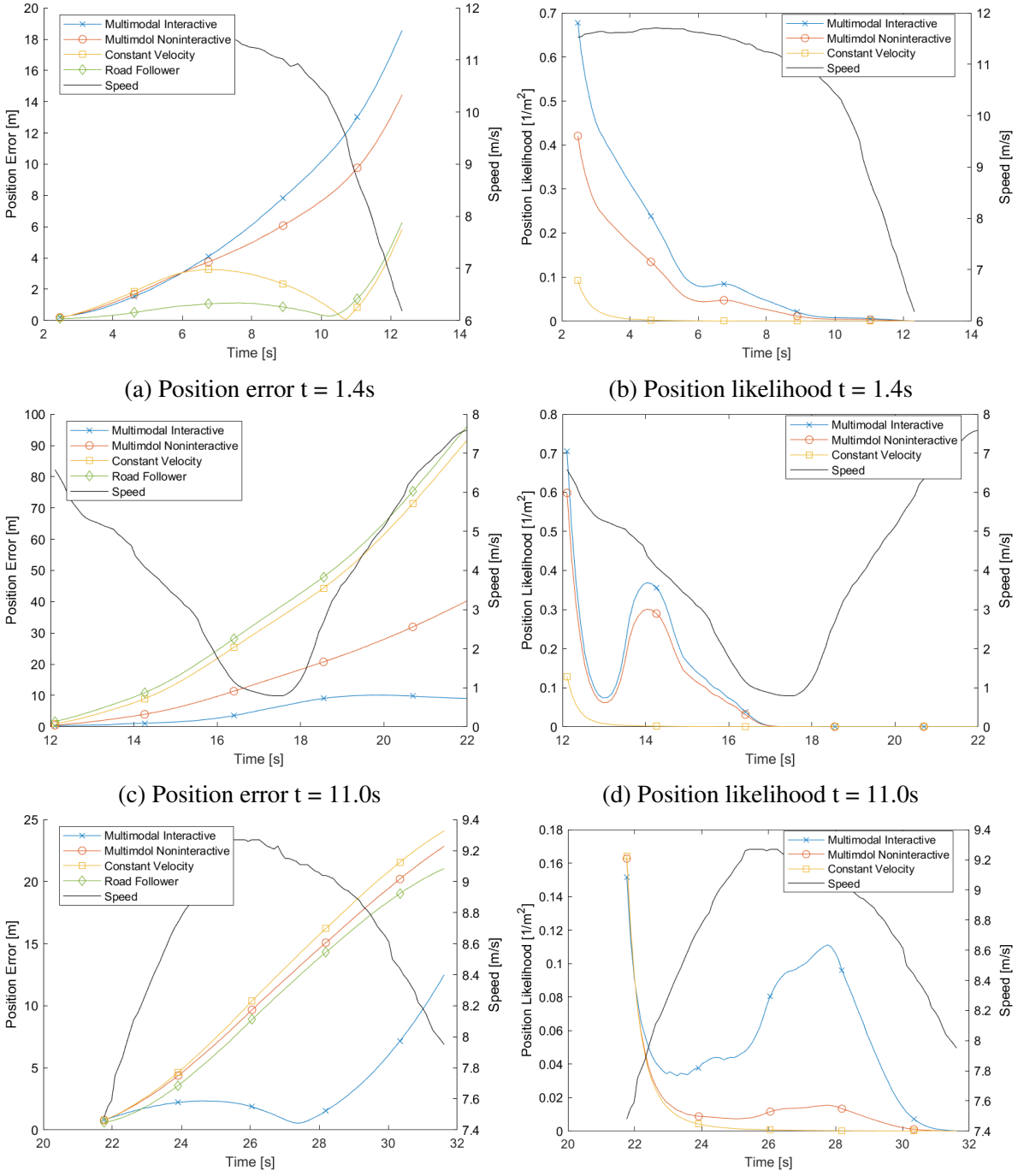


Fig. 5.21 Evolution of maneuver probability over time for the turn and merge scenario. Current speed is shown in black. Turn left becomes the most probable maneuver at $t = 11s$ and is completed at $t = 14s$.

At $t = 20.7s$ (see Figure 5.20c), the turn is almost completed and the target could re-accelerate. But is has to slow down to let pass the obstacle #284, which has the right of way. The interactive prediction (see Figure 5.22e) recognizes this and has therefore at least until $t = 28s$ a very low position error. All other models predict at least constant velocity, which would yield to a crash with the obstacle. The likelihood evolves correspondingly (see Figure 5.22f).

The Figures 5.23a and 5.23b show the evolution of the error and the likelihood for the $\Delta t = 1s$ look-ahead during the whole scenario sequence. The position error is, as expected, very low for all methods, while the likelihood is remarkably better for the multi-modal methods. The $\Delta t = 3s$ prediction error (see Figure 5.23c) increases during the first seconds of all prediction methods, until the multi-modal methods detect the start of the turn maneuver and produce better predictions. A similar effect occurs during the merge with the obstacle. The likelihood (Figure 5.23d) behaves roughly correspondingly with strong variations. The benefits of the interactive method become even more visible in the diagram for the $\Delta t = 10s$

prediction error (see Figure 5.23e), while the 10*s* prediction likelihood (Figure 5.23f) is mostly poor for all methods.



(a) Position error t = 1.4s

(b) Position likelihood t = 1.4s

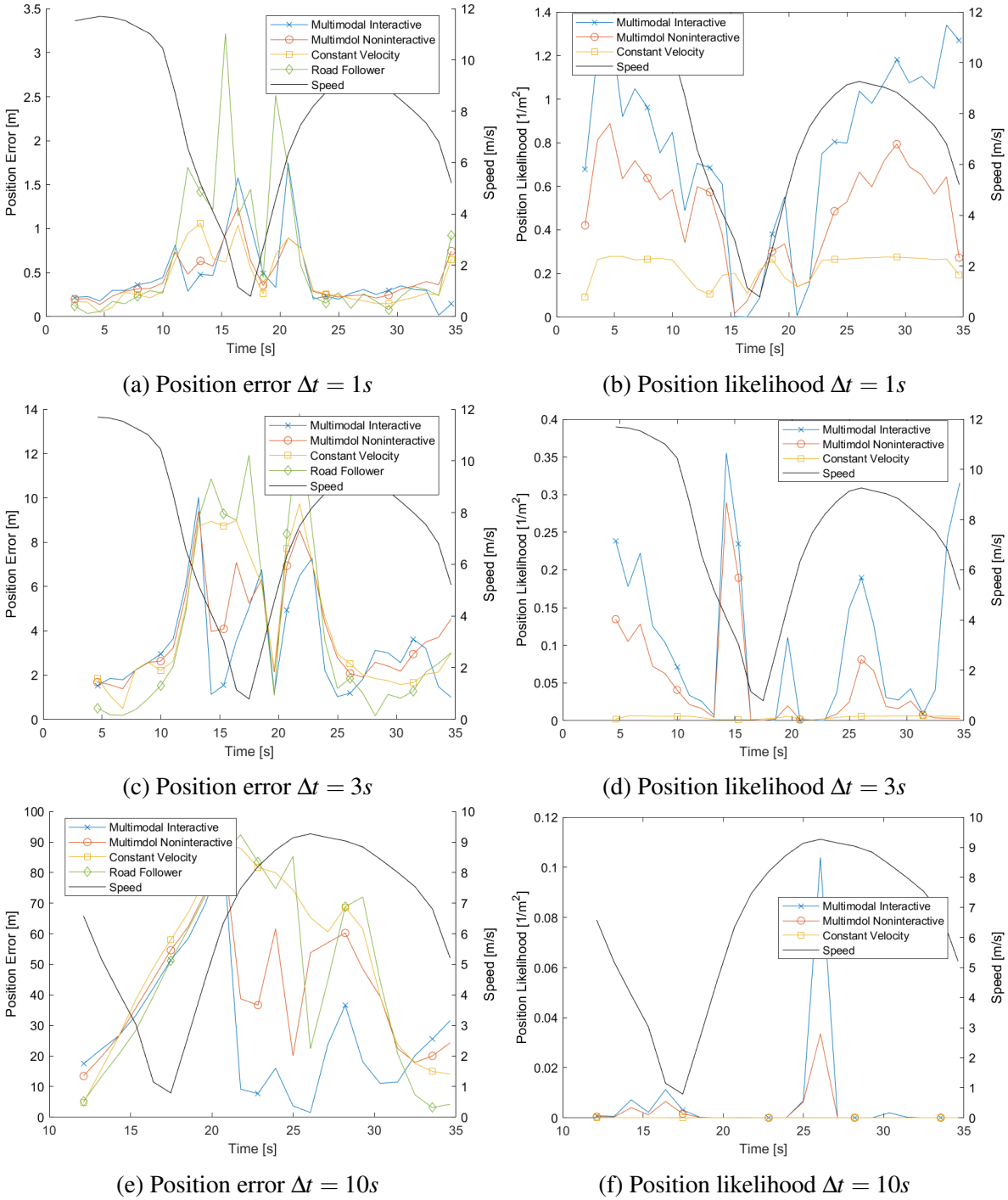
(c) Position error t = 11.0s

(d) Position likelihood t = 11.0s

(e) Position error t = 20.7s

(f) Position likelihood t = 20.7s

Fig. 5.22 Position error and position likelihood over time for the predictions at t=1.4s, t=11.0s and t=20.7s during the turn and merge scenario.



(a) Position error $\Delta t = 1s$

(b) Position likelihood $\Delta t = 1s$

(c) Position error $\Delta t = 3s$

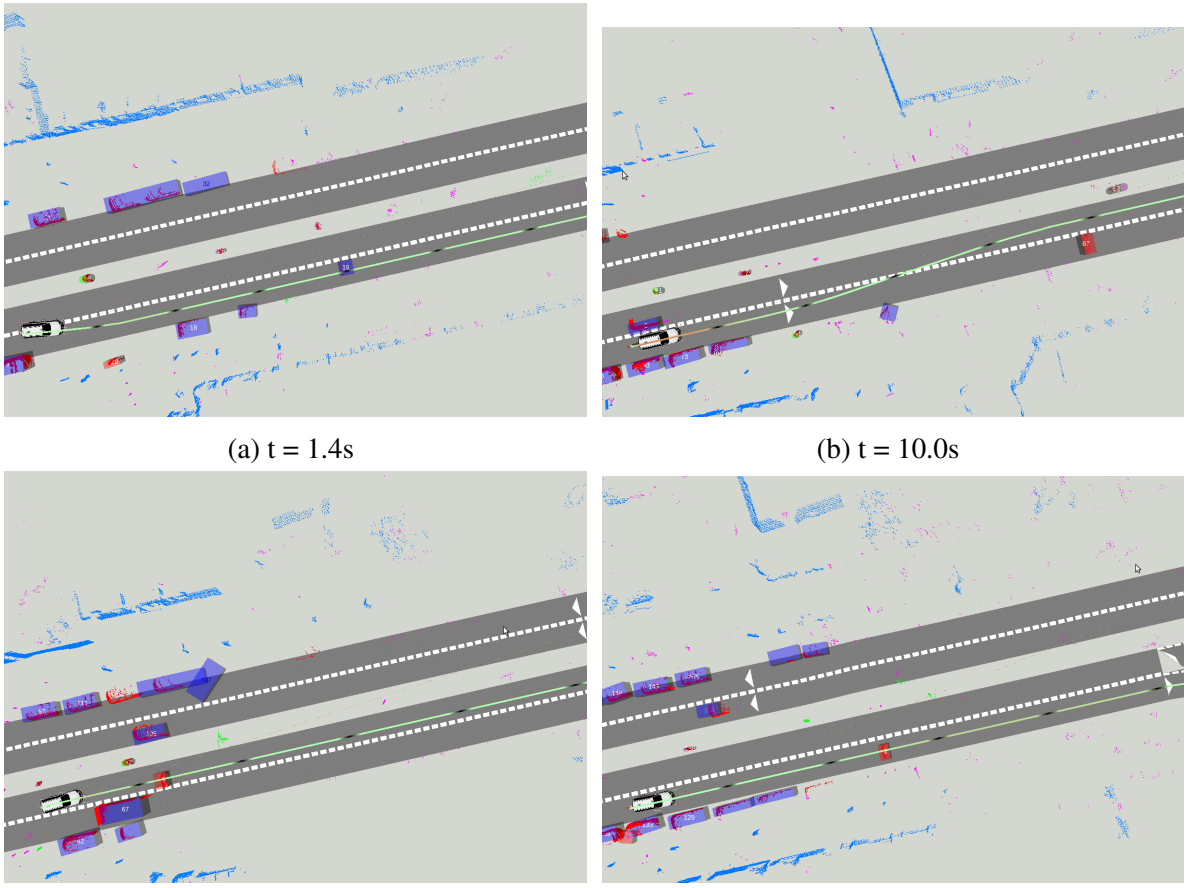
(d) Position likelihood $\Delta t = 3s$

(e) Position error $\Delta t = 10s$

(f) Position likelihood $\Delta t = 10s$

Fig. 5.23 Position error and position likelihood of all predictions of the turn and merge scenario for $\Delta t = 1s$, $\Delta t = 3s$ and $\Delta t = 10s$ look-ahead.

5.10.2 Evaluation of Lane Change Scenario



(a) t = 1.4s (b) t = 10.0s

(c) t = 14.3s (d) t = 20.0s

Fig. 5.24 Lane change maneuver left and right.

In the lane change scenario, the car drives straight on the right lane of the Scharnweberstrasse, a four-lane road with central reservation. When approaching a stopped vehicle on its lane, it changes to the left lane after having found a suitable gap on the destination lane. As soon as the target has passed the stopped vehicle, it changes back to the right lane.

The first Picture 5.24a shows the initial situation when the target has just changed to the right lane and follows the obstacle #19 with an interval of $\approx 3.5s$. The Diagram 5.25 shows the probability of the feasible maneuvers during this scenario. At this initial situation, keep lane is the most probable maneuver.

The evolution of the position error of the prediction at $t = 1.4s$ (see Figure 5.26a) is dominated by the fact that obstacle #19 is decelerating to prepare a right turn. Only the multi-modal interactive prediction considers this and predicts therefore also for the target

vehicle a deceleration. All other methods predict constant and increasing velocity at this point in time and produce therefore high position errors. The likelihoods behave correspondingly.

At $t = 6s$, the probability of a lane change starts to increase strongly and it is about 1 second later already the most probable maneuver. This results from the fact that the current lane of the target is blocked by the stopped obstacle #67 and the lane change incentive (see Subsection 5.5.5) starts to increase. At $t = 10s$, the target starts its lateral movement to the left and the lane change becomes almost sure. Meanwhile the keep lane maneuver is highly unlikely since the target didn't start to brake for the blocking obstacle. At $t = 12s$, the centroid of the target has reached the lane marking and the left lane becomes the current lane.

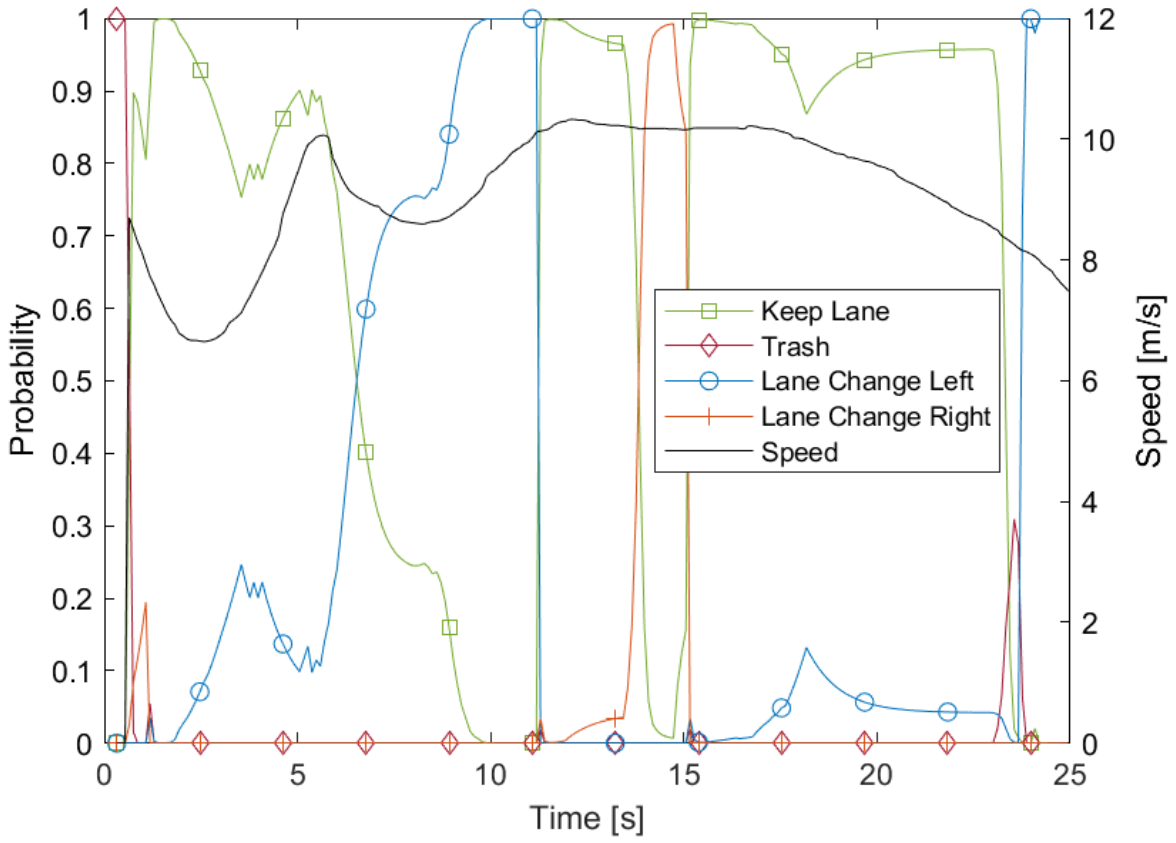


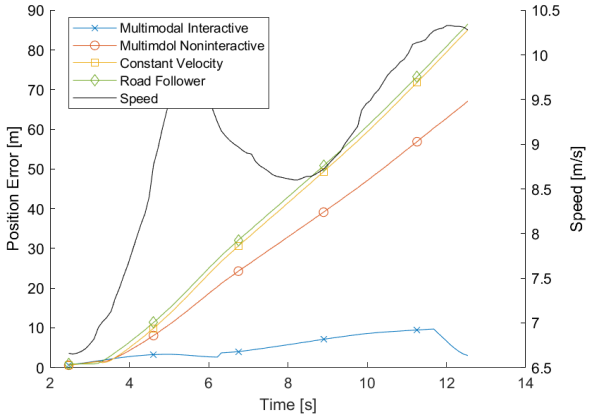
Fig. 5.25 Evolution of maneuver probability over time for the lane change scenario. The probability of the LC left starts to increase at $t = 6s$, the maneuver is completed at $t = 12s$. The LC right starts suddenly at $t = 13.5s$ and the lane marking is passed $1.5s$ later.

The prediction at $t = 10.0s$ (see Diagram 5.26c) initially shows low prediction errors for all methods. After some seconds, the position error of the non-interactive methods increases strongly since their prediction would result in a crash with obstacle #67. But the error for the multi-modal interactive prediction also increases since it doesn't predict at that time the upcoming lane change back to the right lane.

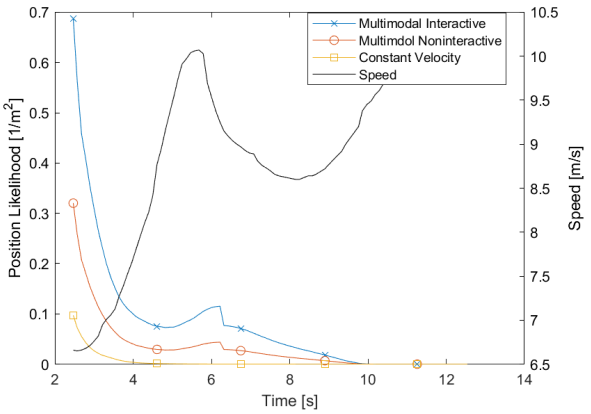
At $t = 13.5s$, the lane change right abruptly becomes the most probable maneuver because the target vehicle starts its lateral movement to the right. This is only about 1.5 s before the passing of the lane marking, while the lane change left was already predicted about 5 s in advance. The reason is that the left lane is free and the car could continue on that lane. The only incentive for the lane change right between $t = 12s$ and $t = 13.5s$ is the "Rechtsfahrgebot", which has only modest influence on the probability.

The prediction at $t = 14.3s$ (Diagram 5.26e) shortly before the lane change back shows first satisfactory position errors, but starting at $t = 17s$, the results become worse. The main reason is the deceleration on behalf of the preceding car, which is not well predicted by any methods. The likelihoods (diagram 5.26f) tend against zero at that time.

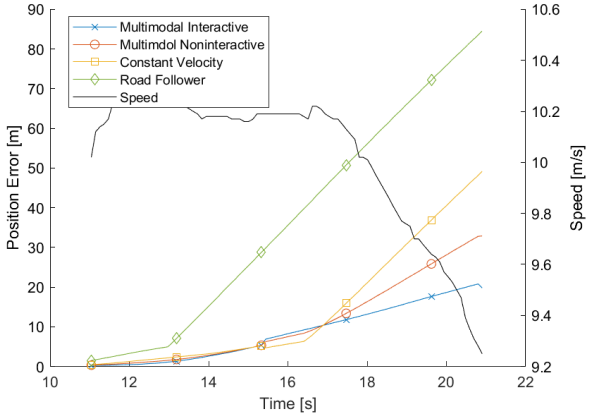
Figure 5.27 shows the position errors and positions likelihoods for every 10th prediction of the whole scenario. The predictions are evaluated for the $\Delta t = 1s$, $\Delta t = 3s$ and $\Delta t = 10s$ look-ahead time. The Diagram 5.27a shows the position error of the $\Delta t = 1s$ look-ahead. The values are constantly below 0.5 m, except for the road follower, which doesn't consider any lateral motion on the road. The likelihood 5.27b for the constant velocity method remains constantly low at $\approx 0.3/m^2$, while the multi-modal methods reach on average $\approx 0.5/m^2$ and $\approx 1.0/m^2$ with strong fluctuations. The position errors for the 3s look-ahead are satisfactory only for the multi-modal interactive method. All other methods suffer from the fact that the longer predictions result in crashes with other traffic participants. The position error and the position likelihood for the $\Delta t = 10s$ look-ahead are unstable for all methods, the multi-modal interactive method is the least bad method.



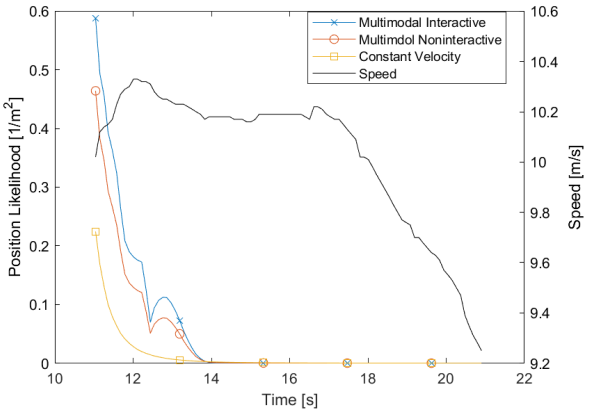
(a) Position error t = 1.4s



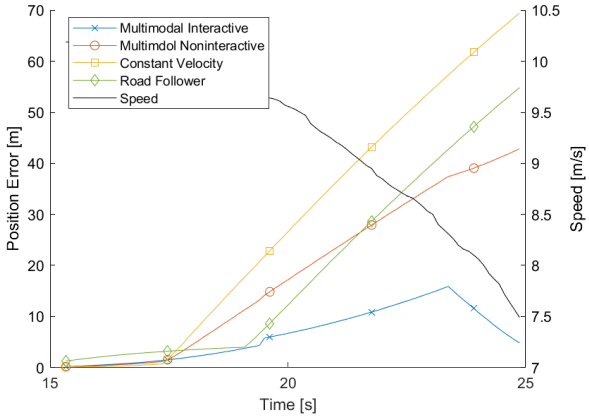
(b) Position likelihood t = 1.4s



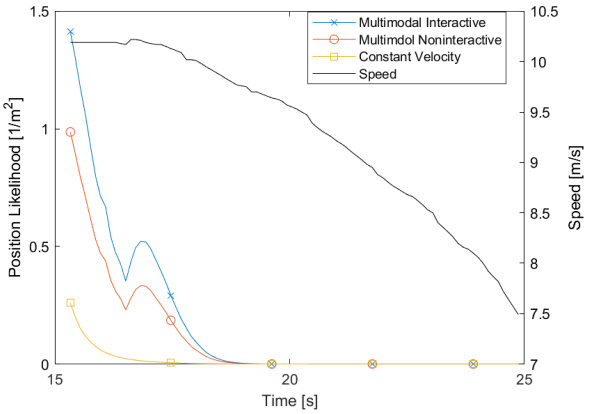
(c) Position error t = 10.0s



(d) Position likelihood t = 10.0s

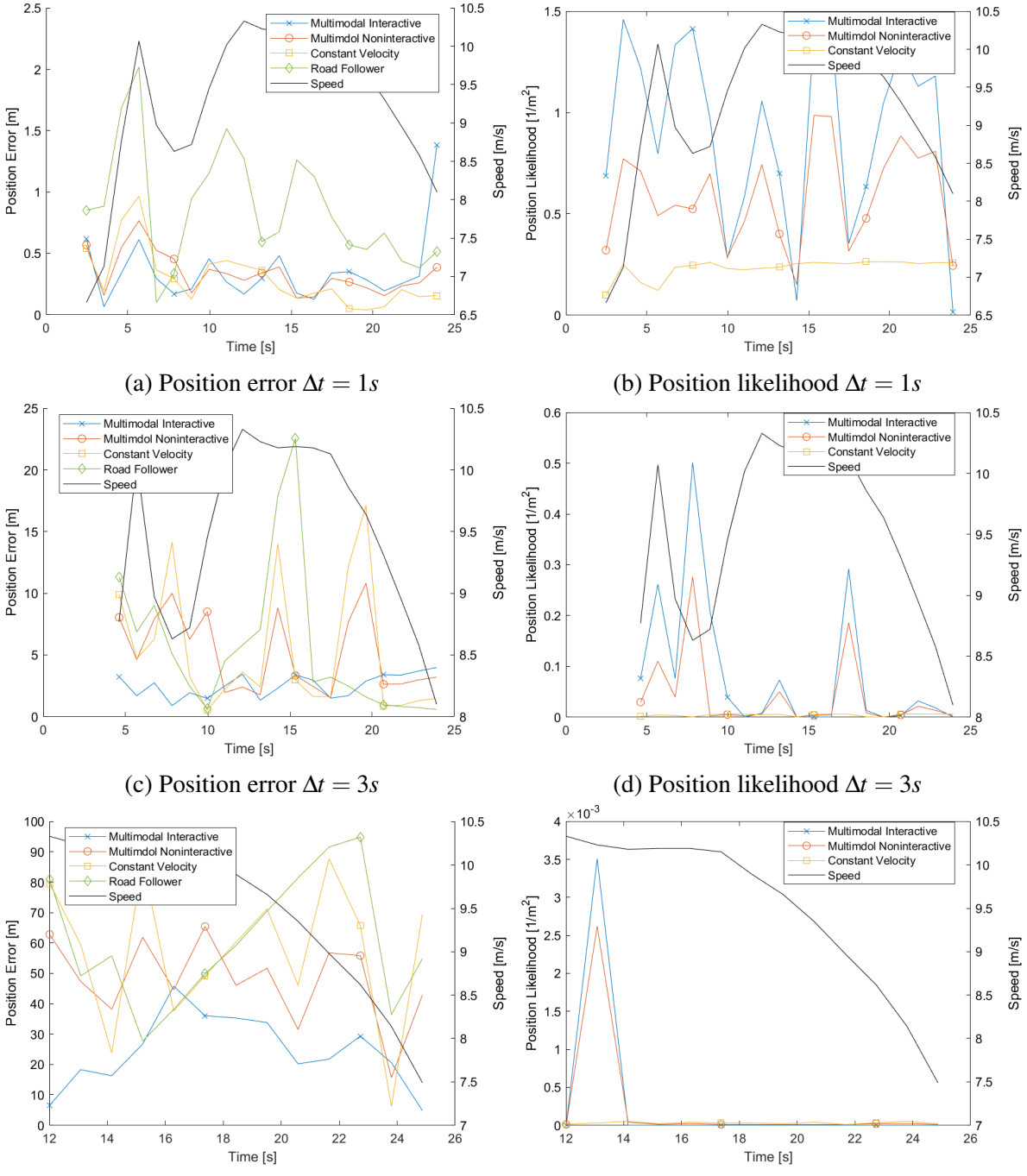


(e) Position error t = 14.3s



(f) Position likelihood t = 14.3s

Fig. 5.26 Position error and position likelihood over time for the predictions at t=1.4s, t=10.0s and t=14.3s during the lane change scenario



(a) Position error $\Delta t = 1s$

(b) Position likelihood $\Delta t = 1s$

(c) Position error $\Delta t = 3s$

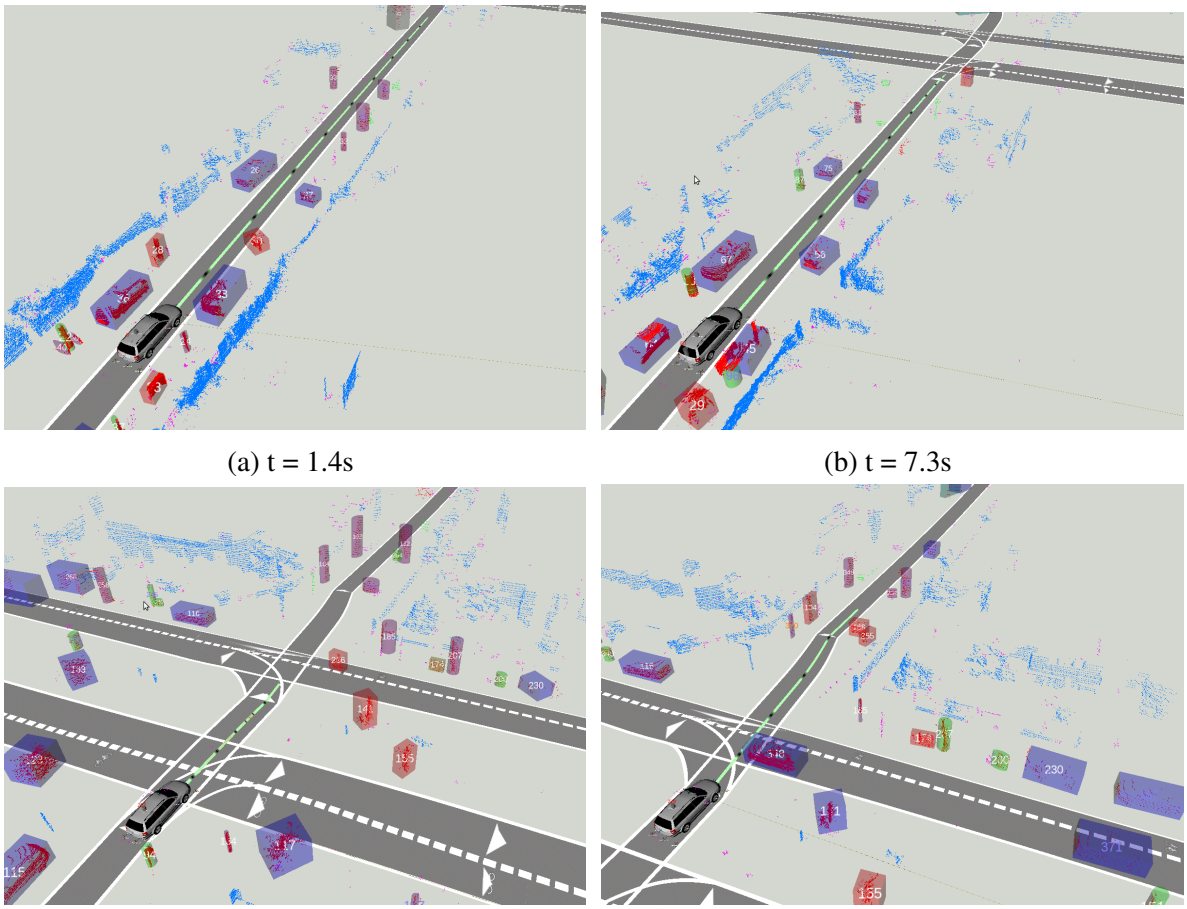
(d) Position likelihood $\Delta t = 3s$

(e) Position error $\Delta t = 10s$

(f) Position likelihood $\Delta t = 10s$

Fig. 5.27 Position error and position likelihood of all predictions of the lane change scenario for $\Delta t = 1s$, $\Delta t = 3s$ and $\Delta t = 10s$ look-ahead.

5.10.3 Evaluation of Intersection Crossing Scenario



(a) t = 1.4s (b) t = 7.3s

(c) t = 23.3s (d) t = 35.0s

Fig. 5.28 Crossing an intersection with central reservation.

In the intersection crossing scenario, the target drives on the single-lane Ehrenbergstrasse, approaching a two-lane priority road with central reservation. The priority road has an additional bicycle lane in each direction. The target vehicle has to stop at first for the traffic from left. Since the central reservation is wide enough for an intermediate stop, the target proceeds before it awaits the traffic from right. After having completed the intersection crossing, the target continues on the Ehrenbergstrasse.

In the beginning of the scenario at $t = 1.4s$ (see Picture 5.28a), the target drives straight at nearly constant velocity. The keep lane has a probability of 80% (see Diagram 5.29). Since the intersection is already in sight, the probability of the turn left and right maneuver reaches $\approx 10\%$. Other maneuvers are not feasible. The position error of the prediction at $t = 1.4s$ (see Diagram 5.30a) increases with the time significantly for all methods except constant

velocity since the target drives slower than the allowed $30km/h$. Accordingly, the position likelihood (Diagram 5.30b) decreases already after 4 s to very low values.

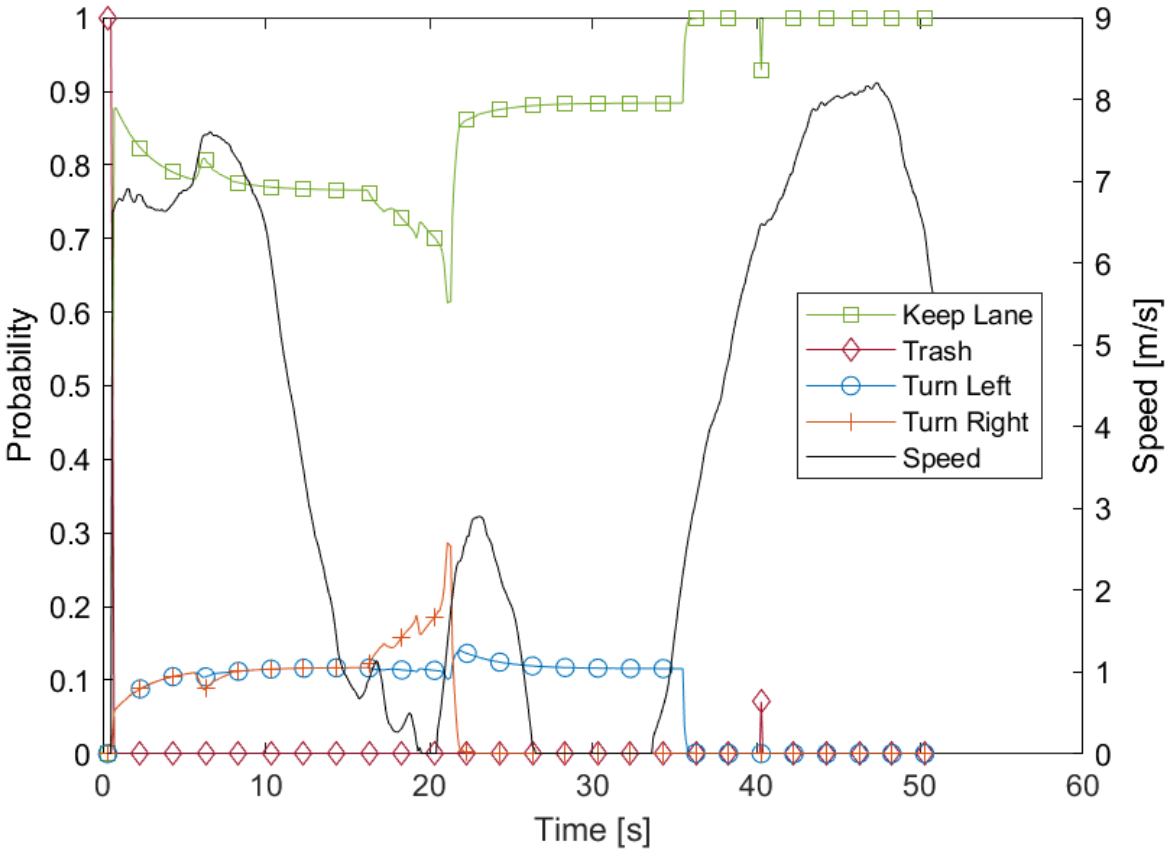


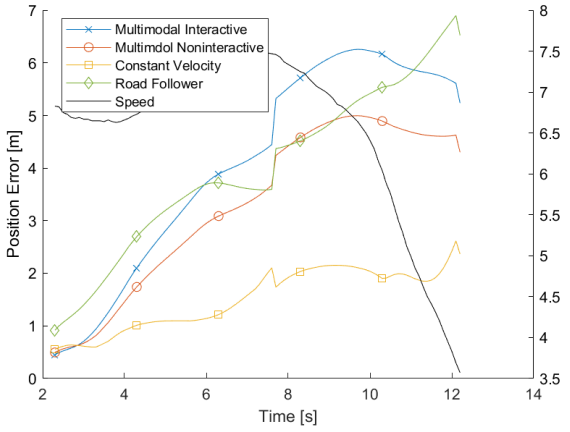
Fig. 5.29 Evolution of maneuver probability over time for the intersection crossing scenario. Initially, turn left and turn right have a certain probability. At $t = 21s$ the turn right becomes unfeasible, at $t = 36s$ keep lane becomes the only possible maneuver within the prediction horizon.

At $t = 7.3s$ (Picture 5.28b), the target has started to decelerate before the intersection. The maneuver probabilities remain unchanged. The position error Diagram (5.30c) looks now completely different. The multi-modal interactive method correctly predicts the deceleration to almost 0 within the next 10s, although no priority vehicle is visible yet. The reason is the line of sight (see Section 5.6) shortly before the intersection, which requires a very low velocity. The other methods predict constant velocities, causing high position errors. The position likelihood for the multi-modal methods is better than for the prediction at $t = 1.4s$.

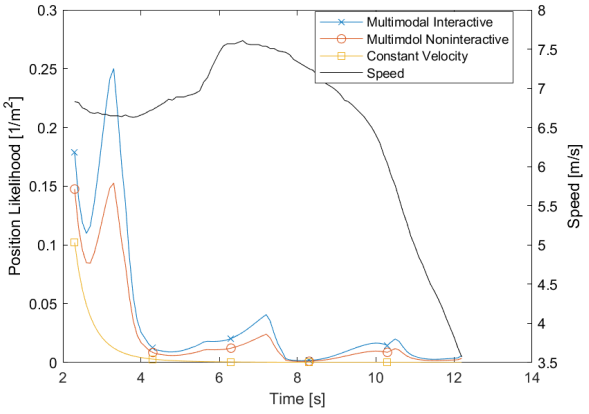
At $t = 23.3s$ (Picture 5.28c), the target has started to accelerate to cross the first lane of the intersection to reach the central reservation. The turn right maneuver now becomes infeasible and the probability of the keep lane maneuver increases. The position errors of this prediction are shown in Diagram 5.30e. Again, only the multi-modal interactive prediction

correctly predicts a stop before the second intersection lane to wait for priority traffic from the right (as the bicycle #216 in Picture 5.28c), while the other methods predict an immediate crossing of the intersection causing a crash. But at $t = 28s$, the multi-modal interactive method also becomes wrong by predicting a premature start of the crossing maneuver. The reasons for this are the trees on the central reservation (obstacles #141 and #155 in Picture 5.28c), which obstruct the view on vehicles farther away. Again, the likelihoods of the multi-modal method are higher than CV, but decrease with time.

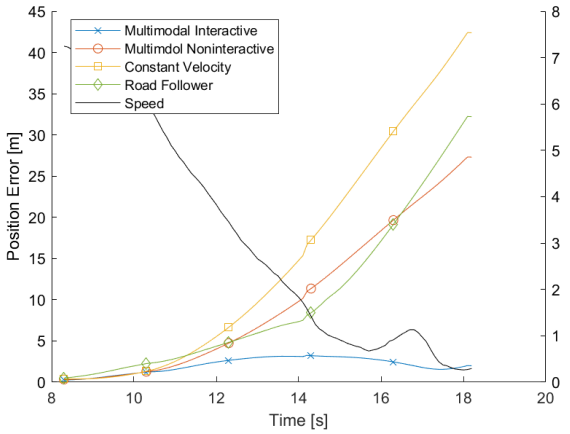
Figure 5.31 shows the position errors and position likelihoods of the whole scenario. The predictions are evaluated for the $\Delta t = 1s$, $\Delta t = 3s$ and $\Delta t = 10s$ look-ahead time. The Diagram 5.31a shows the position error of the $\Delta t = 1s$ look-ahead. The values fluctuate mostly between 0.2m and 0.4m. Only the errors of the road follower method are a little bit higher. The corresponding likelihoods are more stable than in the lane change scenario. Again, the values for the CV method are the lowest. The position errors for the $\Delta t = 3s$ look-ahead are significantly lower than in the LC scenario, caused by the lower velocities. The position likelihoods also benefit from this fact. The same is valid for the $\Delta t = 10s$, where the average error is reduced to 15m compared to the 50m of the LC scenario.



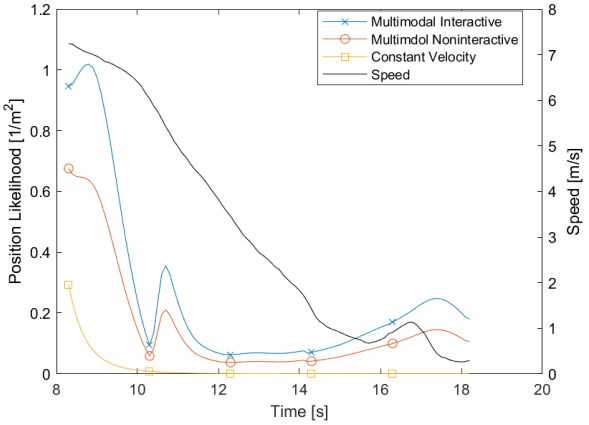
(a) Position error t = 1.4s



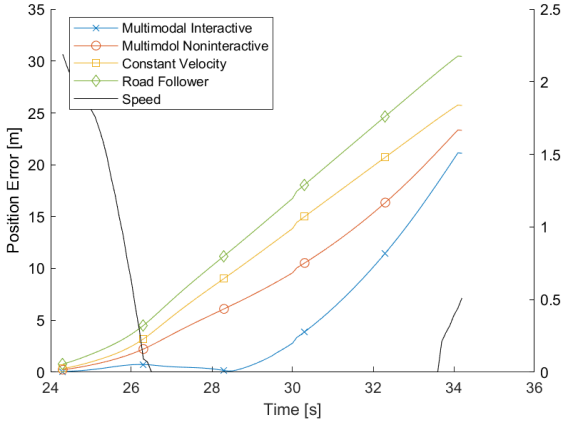
(b) Position likelihood t = 1.4s



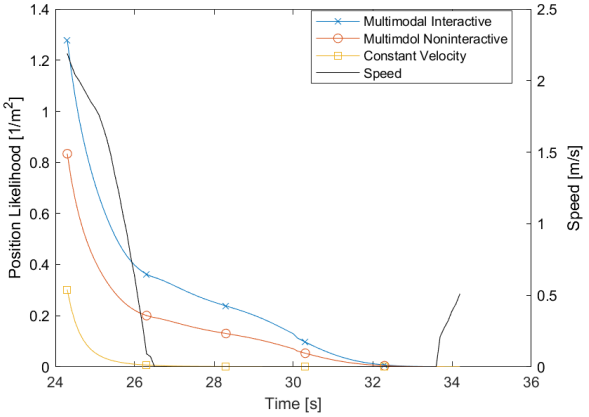
(c) Position error t = 7.3s



(d) Position likelihood 7.3s

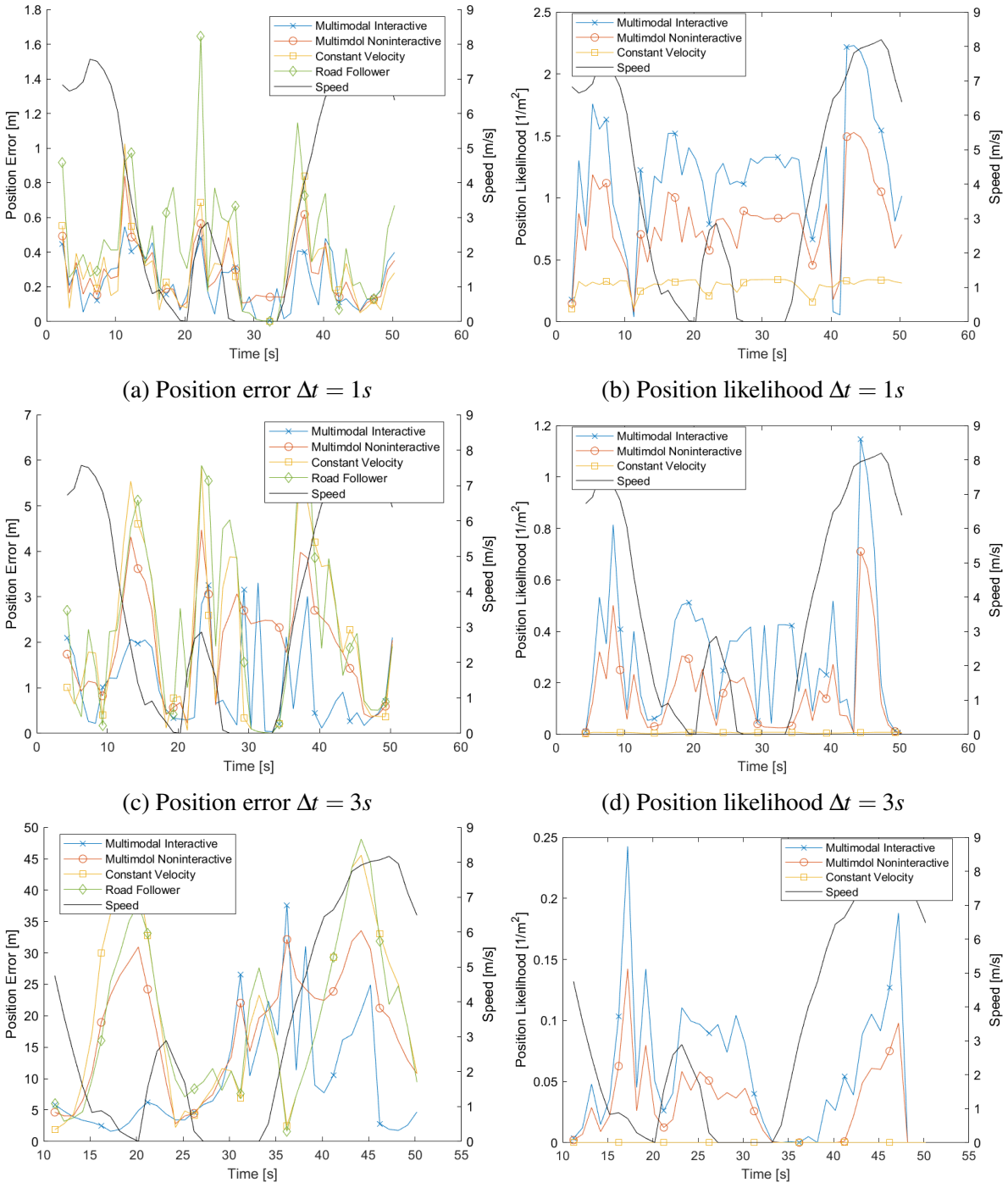


(e) Position error t = 23.3s



(f) Position likelihood t = 23.3s

Fig. 5.30 Position error and position likelihood over time for the predictions at t=1.4s, t=7.3s and t=23.3s during the intersection crossing scenario.



(a) Position error $\Delta t = 1s$

(b) Position likelihood $\Delta t = 1s$

(c) Position error $\Delta t = 3s$

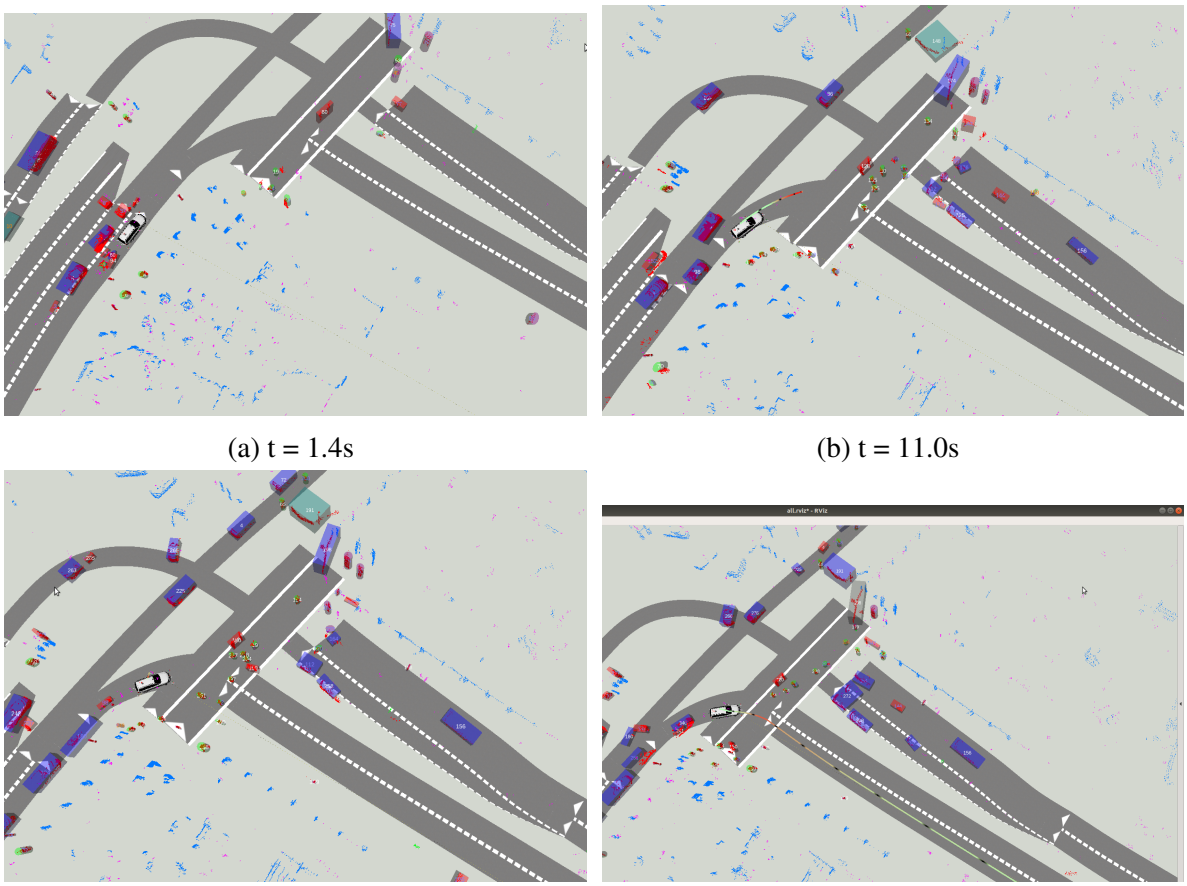
(d) Position likelihood $\Delta t = 3s$

(e) Position error $\Delta t = 10s$

(f) Position likelihood $\Delta t = 10s$

Fig. 5.31 Position error and position likelihood of all predictions of the intersection crossing scenario for $\Delta t = 1s$, $\Delta t = 3s$ and $\Delta t = 10s$ look-ahead.

5.10.4 Evaluation of Pedestrian Cross Walk Scenario



(a) t = 1.4s (b) t = 11.0s

(c) t = 17.5s (d) t = 25.0s

Fig. 5.32 Ego-vehicle at pedestrian cross walk.

In the pedestrian crossing scenario, the target drives on the Ollenhauerstrasse. The initial lane of the target is a dedicated turn right lane with two successor lanes in the Scharnweberstrasse. Since the left successor lane is the lane with the smaller yaw angle deviation from the ancestor lane, it belongs to the keep lane maneuver (see Subsection 5.5.2). The right successor lane belongs to the turn right maneuver. Before entering the Scharnweberstrasse, the target has to pass a bicycle lane and a wide pedestrian crossing, which both have priority.

At the start of the scenario at $t = 1.4s$, the target vehicle is stopped at a red traffic light (Picture 5.32a). The keep lane maneuver has initially the highest probability (Diagram 5.33). There is a $\approx 10\%$ chance for the turn right and until $t = 5s$ also a small chance for a lane change left. The prediction at $t = 1.4s$ shows strongly increasing position errors for all

methods at $t = 3s$ since the target vehicle gets green at this time and starts moving into the intersection. The position likelihoods tend against zero.

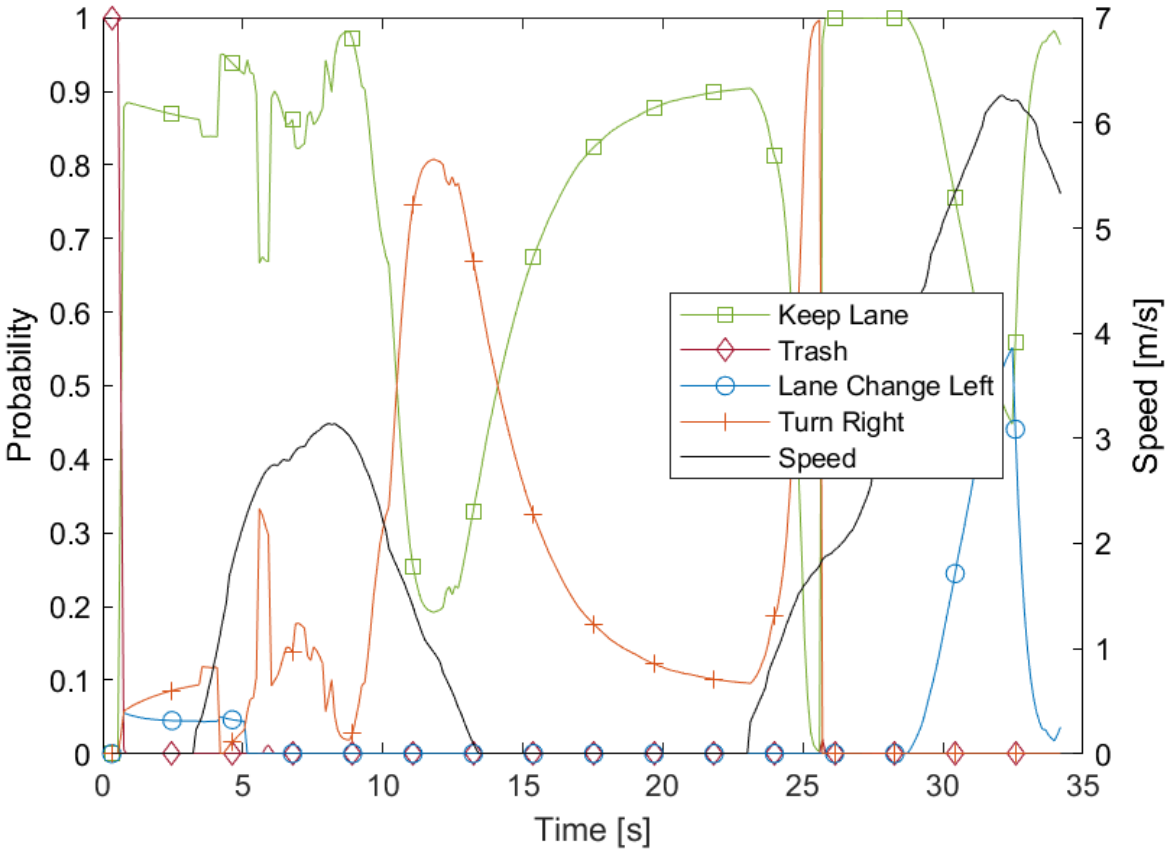


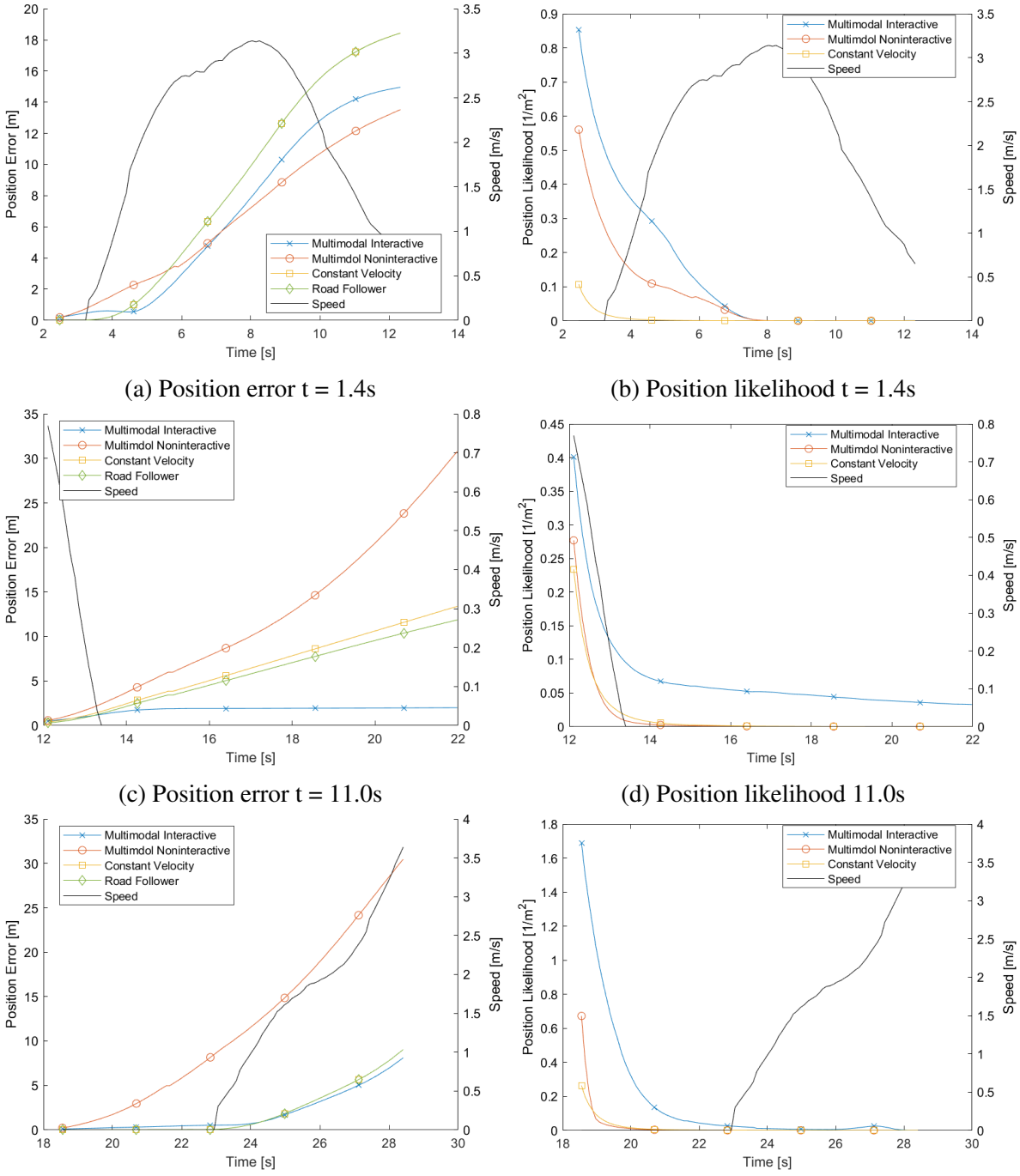
Fig. 5.33 Evolution of maneuver probability over time for the pedestrian cross walk scenario. From $t = 10s$ until $t = 13.5s$, the turn right becomes the most probable maneuver, but while being stopped, the keep lane maneuver gains again. Short after starting again at $t = 24s$, the turn right probability increases again and becomes evident at $t = 26s$. Near the end of the scenario, a lane change left temporarily becomes probable.

At $t = 11.0s$, the target is still moving slowly forward towards the bike and pedestrian crossing. At $t = 13.s$, it stops again at an imaginary stop line before the bike lane even though there is no bike approaching. But there are several pedestrians on the cross walk (see Picture 5.32b), who would prevent the car from passing the complete conflict zone within the prediction horizon of 10s (see Subsection 5.7.3). Diagram 5.34c shows that only the multi-modal interactive method predicts the situation correctly with a low position error, while the other methods predict constant velocity with increasing errors. The likelihood values behave accordingly.

At $t = 17.3s$, the car is still stopped, waiting for the pedestrians to pass (Picture 5.32c). The prediction at that time (Diagram 5.34e) shows that the multi-modal noninteractive

method ignores the pedestrian and predicts immediate acceleration, while the position error for the other methods remains low, until the target starts moving at $t = 23s$. None of the methods has predicted the time of the cross walk clearance correctly and therefore also the likelihoods approach 0 after $t = 23s$. Picture 5.32d shows that the target finally decides to proceed on the right line, which results in steep increase of the turn right probability at $t = 24s$ (Diagram 5.33)

The position errors for the $\Delta t = 1s$ look-ahead in Diagram 5.35a and for the $\Delta t = 3s$ are acceptable, while the $\Delta t = 10s$ look-ahead suffers from the stop and go at the traffic light and the cross walk, even though the velocities are low during most of the scenario.



(a) Position error t = 1.4s

(b) Position likelihood t = 1.4s

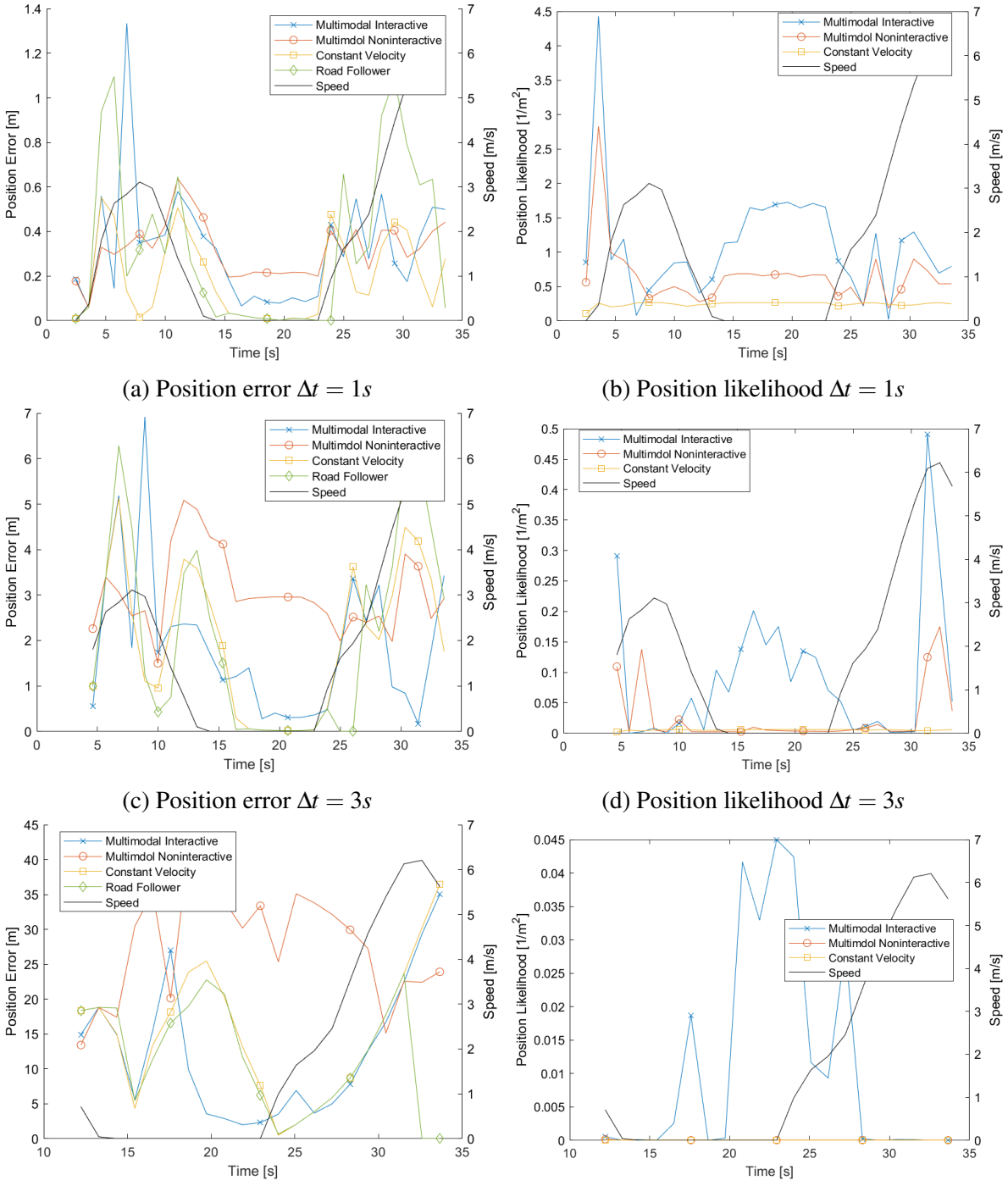
(c) Position error t = 11.0s

(d) Position likelihood 11.0s

(e) Position error t = 17.5s

(f) Position likelihood t = 17.5s

Fig. 5.34 Position error and position likelihood over time for the predictions at t=1.4s, t=11.0s and t=17.5s during the pedestrian cross walk scenario.



(a) Position error $\Delta t = 1s$

(b) Position likelihood $\Delta t = 1s$

(c) Position error $\Delta t = 3s$

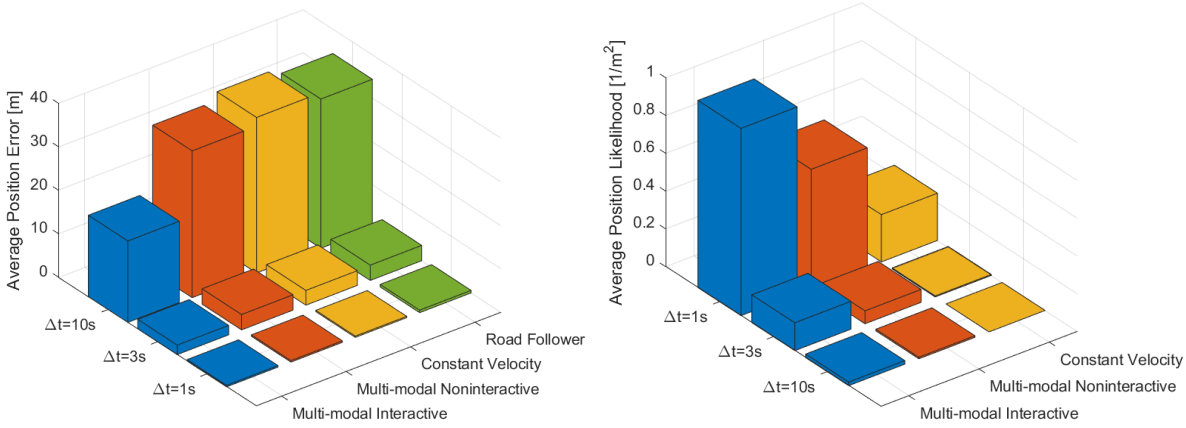
(d) Position likelihood $\Delta t = 3s$

(e) Position error $\Delta t = 10s$

(f) Position likelihood $\Delta t = 10s$

Fig. 5.35 Position error and position likelihood of all predictions of the pedestrian cross walk scenario for $\Delta t = 1s$, $\Delta t = 3s$ and $\Delta t = 10s$ look-ahead.

5.10.5 Summary of Evaluation



(a) Position error (b) Position likelihood

Fig. 5.36 Average position error and likelihood for all four scenarios.

Diagram 5.36 shows the position errors 5.36a and likelihoods 5.36b averaged over all four scenarios. For the $\Delta t = 1s$ look-ahead, the absolute position error is small for all four prediction methods, but the likelihoods for the multi-modal methods, especially for the interactive one, are significantly better than for the constant velocity method. The position error increases for the $\Delta t = 3s$ time span to $\approx 2-3.5m$ for all methods, but the diagram shows already a clear advantage for the multi-modal interactive method. For the $\Delta t = 10s$ look-ahead, the average position error for the interactive method increases to $\approx 18m$, while it is around 35m for the other methods. The likelihood becomes very low for all methods. The high errors are mainly caused by the fact that it is very difficult to predict the velocity of a traffic participant in free driving mode in an urban environment.

More important than the quantitative results of the evaluation are the qualitative findings. When observing a target vehicle driving alone in its lane without any other traffic participant nearby, the accuracy of a long-term prediction of its position is not really important. The decisive factor is the correct prediction of the behavior relative to other obstacles in potential conflict situations. The present evaluation has shown that only the multi-modal interactive method is able to predict the behavior in critical situations, like blocked lanes, merge scenarios, missing gaps in lane change situations, pedestrians crossing the road or unsignalized intersections. All other methods fail in those situations and predict crashes. Even, if the quantitative evaluation has been done only for the ego-vehicle, it has to be stated that the same prediction method is applied for all traffic participants in the scenario. Only in this way, a realistic evolution of the complete situation within the next 10s results from the individual predictions.

	Turn and Merge	Lane Change	Intersection Crossing	Pedestrian Cross Walk
Scenario Length [s]	35.3	24.6	50.6	28.0
Prediction Frequency [Hz]	9.3	9.3	10	9.3
Steps / Prediction	94	94	100	94
∅# of Targets	6.4	7.5	5.2	37.7
∅# of Maneuvers / Prediction	17.1	20.9	13.8	63.1
∅# of Maneuver Pairs / Prediction	117	168	91	1290
∅# of Evaluated Risks / Prediction	4221	6688	3080	31654
∅ System Time [ms] / Prediction	12.5	20.8	36.9	80.4
∅ CPU Time [ms] / Prediction	16.0	26.9	39.1	112.0

Table 5.8 Statistics for Evaluation of Urban Traffic Scenarios

Table 5.8 shows some statistics about the evaluations. The length of the scenarios is in the range of $25-50s$. The prediction frequency is 10Hz for the intersection crossing scenario and 9.3Hz for the other evaluations. This results from the different setups of the Velodyne LIDAR device during the test drives. The number of steps in the predictions are 100 and 94 respectively, resulting in a maximal prediction length of 10s for all scenarios. The average number of targets is in the range of 5 to 7.5 for the first three scenarios, but ≈ 38 for the pedestrian cross walk due to the high visible number of pedestrians.

The average number of feasible maneuvers per prediction ranges from 13.8 to 63.1, resulting in ≈ 2.7 maneuvers per target in the first three scenarios and ≈ 1.7 maneuvers / target for the pedestrian cross walk. The reason for this is that for pedestrians only the trash maneuver is considered, while for vehicles usually a couple of lane bound maneuvers are feasible.

The number of maneuver pairs to be checked per prediction should be $n(n-1)/2$ with n equal to the number of maneuvers. The figures in Table 5.8 are slightly lower due to the fact that the combination of two trash maneuvers doesn't have to be checked for collisions. Since the trash maneuver has always priority (see Section 5.7), a conflict between two trash maneuvers would not have any influence on the trajectory prediction. The collision will simply happen.

The number of evaluated risks per prediction is equal to the number of collision risks calculated according to the method presented in Chapter 4. This should normally be equal to the number of maneuvers pairs times the number of prediction steps. The figures in Table 5.8 are on average three times lower. This results from optimizations to reduce the computational expensive application of the formulars given in Chapter 4. Most important optimization

results from the suppression of the calculation, if the targets are more than five Mahalanobis distances apart from each other.

The elapsed system time for each prediction mainly depends on the number of targets and their maneuvers to be considered. It is on average below 100 ms for all scenarios, proving the real-time capability of the method. The consumed CPU time is about 30-40% higher due to the multi-tasking implementation of the collision risk calculation.

The implementation of the system is anytime-capable. Even if the number of targets temporarily becomes too high, the performance degrades smoothly. This is achieved by two methods:

- Skipping input data: The input data for the prediction is the obstacle list of the perception system, which decides about the prediction frequency. If this frequency cannot be kept up temporarily, individual obstacle lists may be skipped, resulting in longer steps (0.2s instead of 0.1s) between two predictions. Since the algorithm is not bound to a constant input frequency, the accuracy of the results will degrade only slightly.

- Shortening of the prediction horizon: If the system has to skip input data too often, it will start to adapt the number of prediction steps for the trajectories. This will result in an almost linear reduction of computation times. It is then up to the consumer of the prediction results, usually the planner, to cope with the reduced horizon and eventually adapt the speed of the vehicle to ensure safety.

5.11 Summary and Conclusion

This chapter has presented a new method to predict urban traffic scenarios with a horizon of 10 or more seconds at a frequency of 10 Hz in real-time. It does so by analyzing the feasible maneuvers of all perceived traffic participants and rolling the trajectories out into the future. By evaluating the collision risks between all trajectories, the interaction constraints for the next prediction are established. To the knowledge of the author, it is the first system, which not only detects and avoids conflicts between the ego-vehicle and the obstacles, but also the conflicts among the obstacles. Only in this way, a realistic forward projection of a dense urban traffic situation is achievable.

The evaluation using real-world traffic scenarios in this chapter and the simulation in the following chapter proves the thesis that the multi-modal interaction-aware prediction system is able to predict almost arbitrarily complex urban traffic scenarios. This is the main contribution of this chapter.

Further contributions of this chapter are:

- Integration of lane change incentives and turn signal indicators into the probability calculation of maneuver intentions.
- Consideration of multiple brake reasons in Intelligent Driver Model.
- New formulas for approaching speed limits and crossing obstacles for IDM.
- Probabilistic predicted trajectories for IDM.
- A novel efficient method to calculate a lane change trajectory based on the tanh() function.
- The correlation of various driving style parameters to the estimated aggressiveness or defensiveness of the driver.

The output of the traffic scenario prediction are the predicted trajectories of all agents including their probability distribution and the collision matrix, which reflects the probability of conflicts between the trajectories. This output is intended to be used by a planner. In this work, the output is handed over to the traffic scenario simulator described in Chapter 6.

6 Planning and Simulation

6.1 Motivation and Problem Description

The scenario prediction, as presented in Chapter 5, produces a multi-modal distribution of the possible maneuvers of all agents, including the ego vehicle. The predicted trajectory for a given maneuver is an estimate of the motion plan of the corresponding agent. In this way, the prediction system may be used for an ADAS.

Figure 6.1 shows, how the system of Chapter 5 is extended for planning purposes. A global route planner is assumed to produce a route plan, which replaces the intention estimate for the ego-vehicle. Furthermore, the prediction system produces as additional output a planned trajectory for the ego-vehicle, which is a deterministic, non-modal subset of the predicted trajectory. The planned trajectory is forwarded to the vehicle controller, which controls the steering, brake and throttle actuators of the vehicle.

The main purpose of the extension of the prediction system in this work is to test it in a realistic simulation of the environment. Urban traffic scenarios with many traffic participants and complex intersection layouts require to test the software for autonomous vehicles beforehand in a simulation. Especially the behavior of the system in critical situations cannot be evaluated in a real environment. Moreover, the reactions to traffic rule violations of other agents are hard to test without simulation.

Figure 6.2 shows the integration of the simulation environment. The output of the vehicle controller is analyzed by the vehicle simulator, which calculates the reaction of the real hardware to the actuator input. The output of the vehicle simulation is the next state of the ego-vehicle. The scenario simulation takes the new state of the ego-vehicle and the predicted

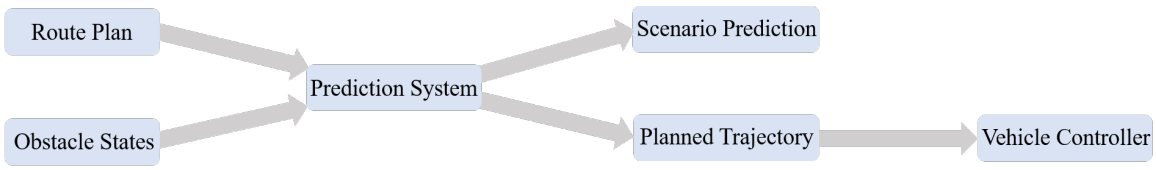


Fig. 6.1 Combined prediction and planning system.

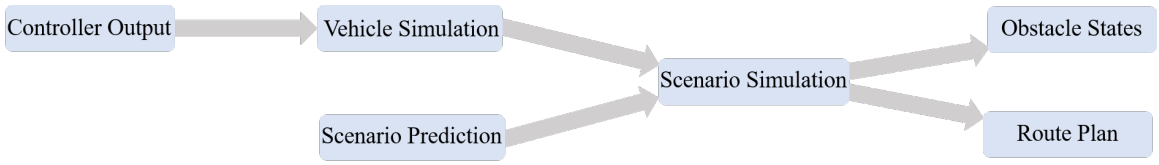


Fig. 6.2 Simulation of traffic scenarios.

trajectories of the obstacles and creates as output the next obstacle list. In this configuration, the simulator acts also as global planner and creates the route plan for the ego-vehicle.

The rest of this chapter is organized as follows: Section 6.2 gives an overview of the existing literature about this theme. In Section 6.3, the definition of a global route as base for the local motion planning is given. The interface between the motion planning and the vehicle control is described in Section 6.4. Section 6.5 documents the traffic scenario simulator. The evaluations presented in Section 6.6 and the summary and conclusion in Section 6.7 complete this chapter.

6.2 Related Work

A survey of different planning methods for autonomous driving is given in [104]. The methods are examined on the levels of path planning, maneuver decision and trajectory optimization. Current constraints and limitations are evaluated and future research directions are discussed. Another survey of motion planning and control techniques with the main emphasis on urban driving is given in [140]. The paper considers the levels of route, behavior and motion planning as well as vehicle control. Especially for motion planning, they evaluate variational methods, graph search methods and incremental search techniques.

The authors of [192] state that prediction methods should be judged on how well they support decision making and planning, instead of pure metrics like RSME, likelihood or Kullback-Leibler (KL) divergence. They propose among others to consider the potential reactions of other vehicles to the own driving decisions and to investigate problems resulting from occlusion. In [123], it is proposed to fuse prediction and planning. They predict the maneuvers of the environmental vehicles using a SVM and generate Gaussian trajectories in the Frenet-Serret frame. These predictions are integrated in the optimization of the ego-vehicle trajectory as an environment potential field.

Trajectory planning based on the Intelligent Driver Model (IDM) is presented in [84]. The calculated reference trajectory is subsequently incorporated into an Optimal Control Problem (OCP) to find an acceleration- and jerk-optimized solution. The approach is limited to plan longitudinal motion. In [82], the method is extended to consider also lateral movements.

This is achieved by integrating the 'Minimizing Overall Braking Induced by Lane Changes' (MOBIL) [105] method for lane change decisions and extending the OCP formulation accordingly. The authors further refine their method by also considering overtake maneuvers with oncoming traffic in [83]. A collision free corridor is generated as constraint for the trajectory optimization.

A classical A*-based algorithm for motion planning is proposed in [5]. They incorporate other vehicles, traffic lights, speed limits and lane markings as constraints into the search space. The search is performed in a 3D-grid of discretized time, and as well as longitudinal and lateral road position. In [31], an extend A* version, the Flexible Unit A-* (FU-A*) algorithm for trajectory planning in structured road maps is proposed. The authors of [134] use a SVM to find a collision free corridor and subsequently apply their own optimization algorithm to generate an optimized trajectory inside the corridor.

Several authors propose Partial Observable Markov Decision Processes (POMDP) or methods based on POMDP for decision making and planning. Classical POMDP approaches consider discrete state, observation and action space, which is not suitable for autonomous driving. In [17], a new algorithm, which handles continuous state and observation spaces, is presented. It computes an optimal policy represented as Generalized Policy Graph (GPG). The authors of [33] propose a similar method for offline calculation of an optimal policy, based on state space representations in form of decision trees and a finite set of linear $\alpha - vectors$. [42] also follows the idea of POMDPs, but presents an approximate approach called Multipolicy Decision-Making (MPDM). It evaluates the policy of the ego-vehicle and other traffic participants. The authors of [176] propose a POMDP with mixed-integer state space for the online calculation of an optimal policy for tactical lane changes. [96] in contrast present an online POMDP for decision making at unsignalized intersections. It uses the point-based Toolkit for approximating and Adapting POMDP solutions In Real time (TAPIR). [133] extends the above methods by extending POMDPs to continuous actions. The proposed planner called Continuous Belief Tree Search (CBTS) uses Bayesian optimization to find optimal actions.

Numerous authors propose learning based methods for planning the actions of autonomous vehicles. In [183], a LSTM neural network is combined with a Conditional Random Field (CRF) to generate human-like decisions for lane change maneuvers. A LSTM is also the base for the solution shown in [155], where the neural network is combined with the IDM prediction model for decision making on highways. In [153] Reinforcement Learning is proposed to train a network for making lane change decisions in dense traffic. A game theoretic approach to find an approximate Nash equilibrium for the task of controlling a vehicle in a racing situation is presented in [182]. People from Waymo recently presented

ChauffeurNet in [19], a RNN architecture of autonomous driving. It is based on imitation learning and leverages a perception system for input reprocessing and a separate controller for actuation to be able to synthesize also unusual situations.

Planning solutions for autonomous vehicles require intensive testing before they may be released for common use. For this purpose, some simulation and testing tools have been developed. [102] presents a survey of publicly available datasets and testing environments for autonomous driving. In [50], Car Learning To Act (CARLA), an open simulator for urban driving is presented. It allows to evaluate classic pipelined architectures as well as end-to-end learning solutions. The authors of [71] present a distributed simulation architecture especially tailored to test cooperative vehicles. In [7], an approach to generate critical test scenarios, which are usually not included in real-world datasets is proposed. It works by artificially reducing the feasible driving space of a real-world scenario to force the vehicle under test into a critical situation.

6.3 Route Planning

The task of a global route planner is to find an optimal path from the start of a trip to its destination. There are many algorithms and many commercial products, like Google Maps [128], which solve the problem. For the purpose of autonomous driving, the calculated route must be more precise than that of most currently available route planners since it must also select the individual lanes to be used by the vehicle.

For single lane roads, the global plan is simply a lane sequence as defined in Section 5.5. If more than one lane is available, lane changes are possible and may be required. There are two types of lane changes [154]:

- Mandatory Lane Changes: These lane changes are required to reach the goal of the mission. They are often prerequisite for a turn maneuver on multi-lane roads.
- Discretionary Lane Change: Lane changes, which are not required to reach the goal. They are applied for overtaking or when the other lane seems to offer better traffic conditions. Near highway on-ramps, discretionary lane changes are sometimes done for courtesy.

For the task of scenario prediction as in Chapter 5, the difference is not important, since the goal of the obstacles is not known.

For local motion planning, the global planner has to advice the mandatory lane changes, but it cannot prescribe the exact position of the lane change.

The global route planner of the MadeInGermany system uses the Dijkstra shortest path algorithm [49] from the Boost Graph Library [164] to find an optimal path through the Atlas roadmap. If several lanes are available, they are weighted. For discretionary lane changes, the lanes get equal weights, otherwise all weight goes to the destination lane(s) of a mandatory lane change.

6.4 Trajectory Planning and Control

If the scenario prediction system of Chapter 5 gets additionally to the obstacles states a route plan for the ego vehicle as input, it works in planning mode. The intention of the ego vehicle is derived from the route plan, except for discretionary lane changes. The decision about discretionary lane changes is made based on the lane change incentive (see Section 5.5).

The motion plan for the ego-vehicle is taken as input for the planned trajectory. The planned trajectory is a discrete time, deterministic sequence of future vehicle poses. It is handed over to the fub_roscar Controller system [77], which uses a PD-controller to generate brake and throttle commands and a pure pursuit controller for the steering wheel actuator.

The motion planner and the controller work in open loop mode, e.g. the planner doesn't take the last measured state of the ego vehicle as start position for the next motion plan, but the planned state. Only in case of a severe difference between the planned and the measured state, the motion planner is reset by starting the next plan with the measured state.

6.5 Simulation of Traffic Scenarios

The purpose of the traffic scenario simulator is to test the results of the prediction and planning solution. It replaces the hardware and the perception system of MadeInGermany and allows to simulate critical traffic situations with 40 and more agents, depending on the processor performance.

The simulator takes an ATLAS roadmap (see Section 2.3) as input. The roadmap is assumed to consist of sequences of road sections, which form endless loops (see Figures 6.3 and 6.8 as examples). On initialization, the simulator places the ego-vehicle and a configurable number of obstacles at random, non-overlapping positions of the roadmap. The size of the obstacles is sampled randomly from a predefined set of vehicle types. The driving style of the obstacle is sampled from a uniform distribution (see Subsection 5.8.4). Optionally, the simulator allows to add pedestrians, which cross the road at a predefined position in an endless loop. The initial velocity of all agents is zero. The initial states of all

obstacles and of the ego-vehicle are packed in an obstacle message and send to the scenario prediction module, simulating in this way the input from the perception system.

The simulator runs at a configurable frequency (default 10 Hz). On every simulation step, the output of the last scenario prediction is analyzed. The new state of all obstacles is taken from the corresponding predicted state of the maneuver with the highest probability. The new state of the ego-vehicle is taken from the vehicle simulation.

If the prediction system is not able to keep the simulation frequency, simulation steps will occasional be skipped. This corresponds to missed messages from the perception system in real operation and the quality of the prediction should only degrade modestly. The disadvantage of this behavior during simulation is that the results depend on the timing of the simulation hardware and are therefore not reproducible. To avoid this, the simulator can be run with dynamic clock simulation, e.g. the system clock of the ROS system is eventually stopped until the prediction has finished.

The simulator checks on every simulation step all obstacles for overlapping poses, e.g. collisions. The state of collided obstacles is optionally frozen during the further simulation.

For evaluation purposes, the simulator allows to record various statistics, like number and type of executed maneuvers, average vehicle speed, number of collisions and emergency brakes. The progress of the simulation is visualized using the fub_riot_visual module and RVIZ.

6.6 Evaluation of Planning and Simulation

The evaluation is done by simulating two typical traffic scenarios:

- Urban Intersection Scenario
- Highway Style Oval Track Scenario

These scenarios allow to simulate all relevant types of situations:

- Car Following
- Lane Change
- Lane Merge
- Intersection Crossing
- Turn Maneuver

- Pedestrian Crossing

There are two quantitative measures for the quality of the planned trajectories:

- Efficiency: Measured by the average speed in m/s during the simulation.
- Safety and Comfort: Measured by the number of emergency brakes per Km performed during the simulation. An emergency brake is a deceleration of the vehicle with a peek rate $< -5m/s^2$.

The evaluation is done for three different prediction models:

- Multi-modal interaction-aware
- Multi-modal non interaction-aware
- Constant velocity

Additionally, two scenario specific evaluations have been done:

- Urban Intersection Scenario: Influence of the prediction horizon in seconds on efficiency and comfort/safety.
- Highway Style Oval Track Scenario: Influence of the speed limit on efficiency and comfort/safety.

6.6.1 Intersection Scenario

Figure 6.3 shows the roadmap for the simulation of the intersection scenario. The intersection is modeled after the so called "Kranzler Eck" in Berlin. The Kurfürstendamm is oriented east-west and has priority, the Joachimthaler Straße is oriented north-south. In the original intersection layout, some lanes are reserved for buses, but these lanes are open for all vehicles in this simulation. The outgoing and incoming road sections are connected to each other, so that the vehicles may circulate in an endless loop without having to be removed or added. The traffic lights at the intersection are assumed to be switched of.

During initialization, the simulator places the ego vehicle and 40 obstacles at random positions of the roadmap. The driving style for each vehicle is sampled from an uniform distribution. The initial velocity is set to $0m/s$. The simulation frequency is set to $10Hz$, the prediction horizon is $10s$ and the prediction step length $\Delta t = 0.1s$. This results in a trajectory of 100 steps per maneuver.

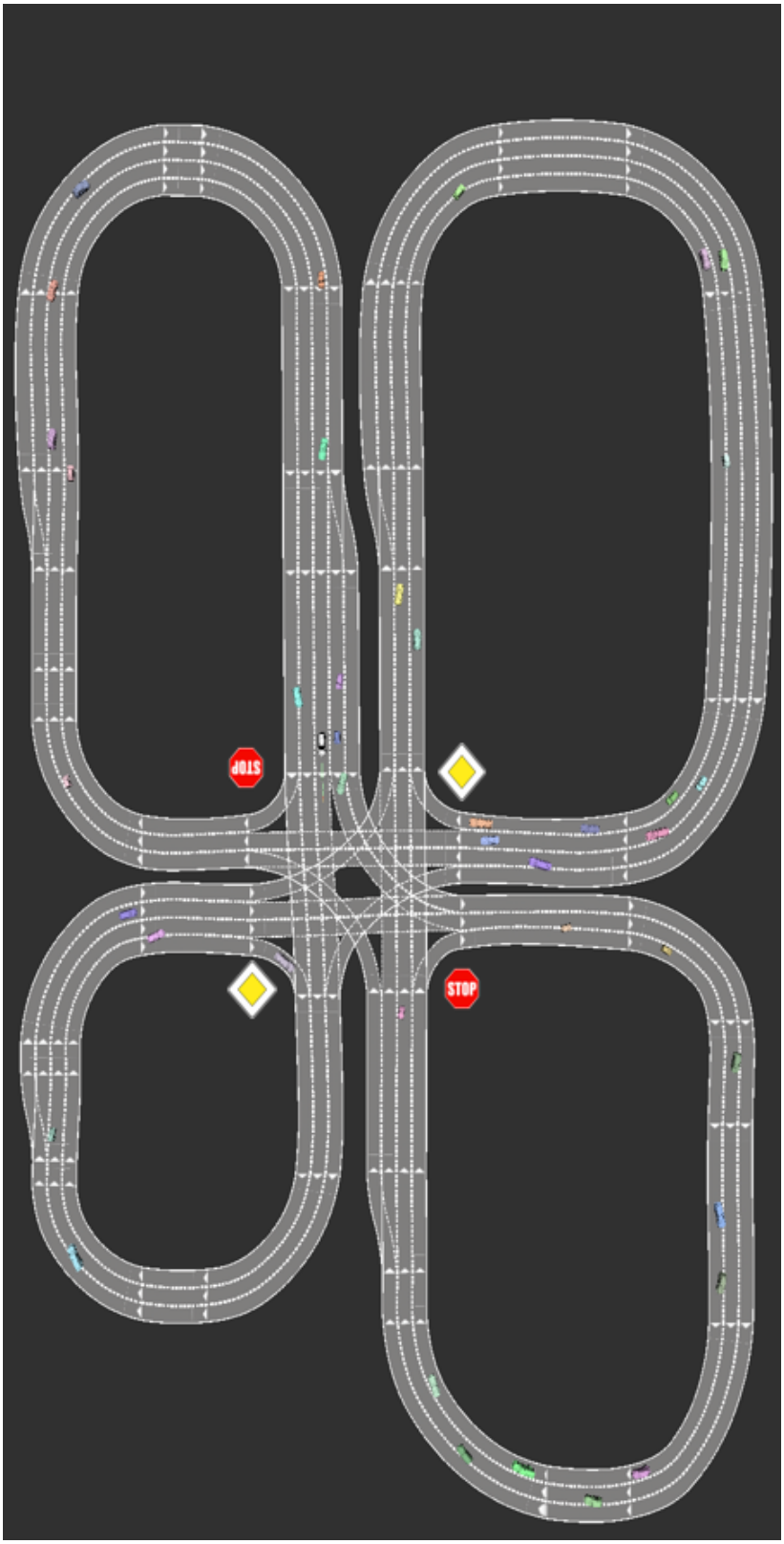


Fig. 6.3 Roadmap for simulation showing an unsignalized multi-lane intersection.

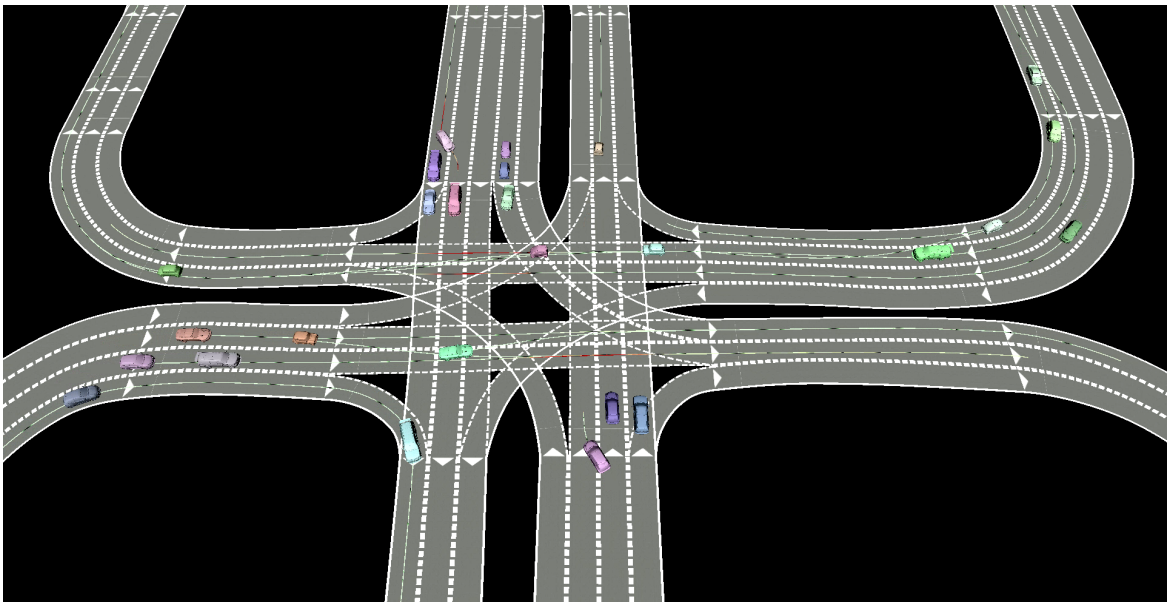


Fig. 6.4 Example traffic situation at intersection. Several cars on the subordinate road have to await priority traffic.

Figure 6.4 shows a situation during the simulation. For each vehicle, the currently most probably planned trajectory of the next 10 seconds is shown. Most of the north and south bound vehicles have no trajectory since they cannot start moving within the next 10 seconds due to the traffic on the priority road.

Figure 6.5 shows the same scenario about 30 seconds later. The priority traffic has crossed the intersection and the north/south bound vehicles can start moving. The three south bound vehicles on the left turning lane are still blocked by the oncoming cars.

To evaluate the influence of the prediction method on efficiency and safety/comfort, the prediction system is run in three different modes:

- Multi-modal interaction-aware: The trajectories of the feasible maneuvers are predicted under consideration of the collision risks with other agents.
- Multi-modal non interaction-aware: The trajectories of the feasible maneuvers are predicted without considering the collision risks.
- Constant velocity: Only one constant velocity trajectory per agent is predicted.

The limited prediction modes are only applied for the ego-vehicle. Planning the trajectories of all agents with limited prediction capabilities would result in chaos after a few seconds.

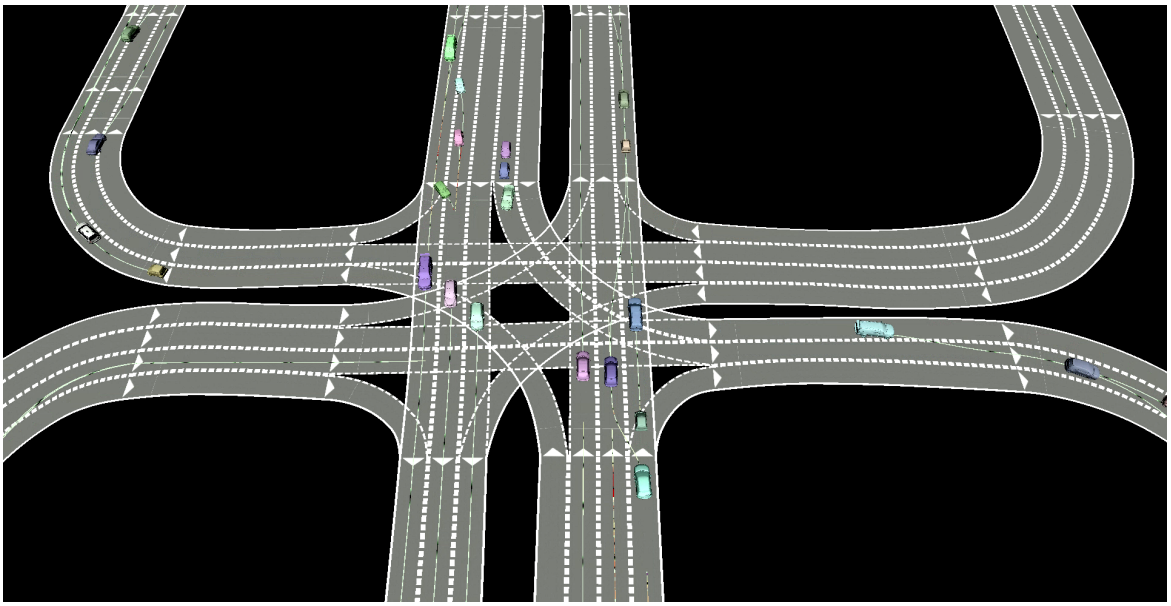
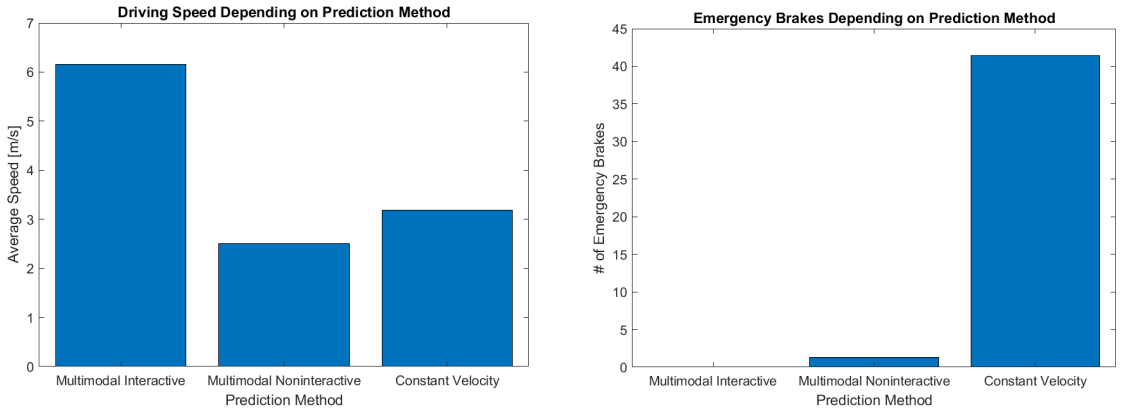


Fig. 6.5 Situation at intersection 30 s later than in Figure 6.4. North/south bound traffic starts to cross the intersection.

For each prediction method, the simulation is run 5 times for 120 seconds. Each run is performed with different initial positions of the agents and with different driving styles. The driving style of the ego-vehicle is always set to neutral (== 0).



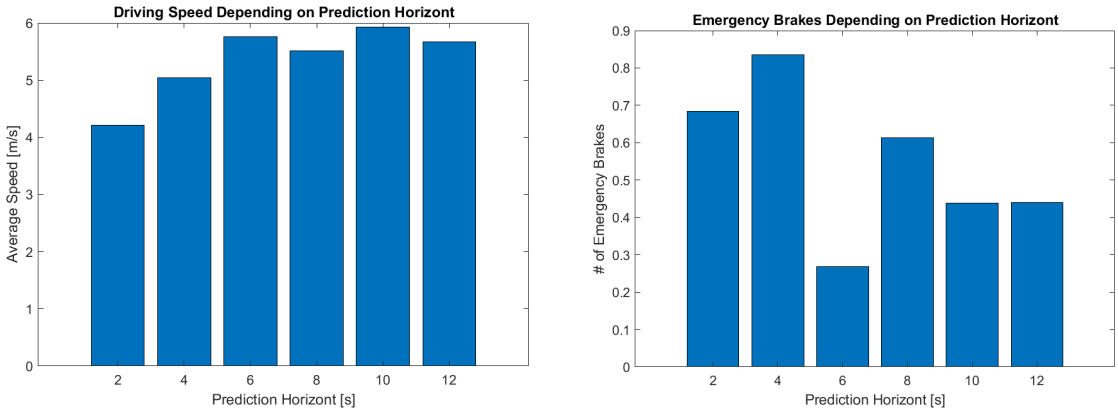
(a) (b)

Fig. 6.6 Efficiency and safety/comfort for different prediction methods in intersection scenario.

Figure 6.6(a) shows the average speed of the ego-vehicle during the 5 simulation runs. As expected, the interaction-aware method shows the best results. The ego-vehicle correctly predicts that the crossing obstacles will stop, if itself has the priority. With the constant velocity method, the ego-vehicle passes the intersection only, if there is no crossing traffic

or the crossing vehicles already have stopped. The performance of the multi-modal non interaction-aware prediction is the worst, since in this mode the ego-vehicle expects the crossing vehicles to accelerate, even if they have stopped.

Figure 6.6(b) shows the number of emergency brakes per km for the three prediction methods. The interaction-aware method has to perform no emergency brake, and also the non interaction-aware method has only a few emergency brakes. The poor performance of the constant velocity method results mainly from driving in parallel lanes through the various curves, which leads to many predicted collisions when using a CV model.



(a) (b)

Fig. 6.7 Efficiency and safety/comfort depending on prediction horizon in intersection scenario.

A further evaluation has been done to examine the influence of the prediction horizon on the efficiency and on safety/comfort. The prediction time was set to 2, 4, 6, 8, 10 and 12 seconds and the simulation has been run 5 times for 120 seconds. Each run is performed with different initial positions of the agents and with different driving-styles.

Figure 6.7(a) shows the results for the average speed of all vehicles, depending on the prediction time. There is a clear tendency to higher average speeds with increased prediction time. Figure 6.7(b) shows the emergency brakes per km depend on the prediction time. There is a tendency to fewer emergency brakes with increasing prediction time. The effect of a short prediction horizon is a more risky behavior, so the increased number of emergency brakes is not surprising. The increased average speed shows that a longer prediction horizon leads to smoother traffic and therefor to increased efficiency.

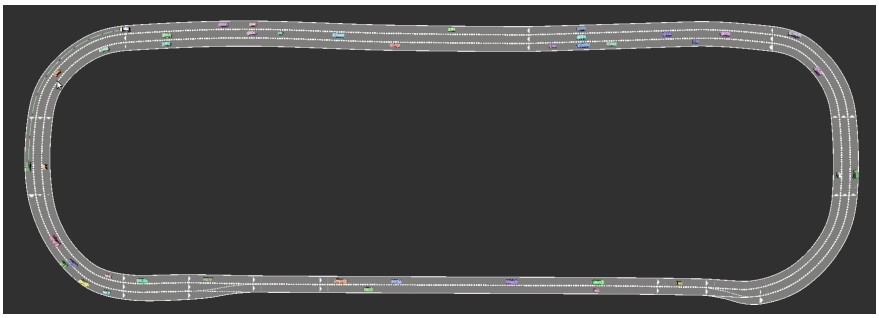


Fig. 6.8 Roadmap for simulation of oval track scenario. The lane merge at the lower left corner models a on-ramp scenery.

6.6.2 Highway Oval Track Scenario

Figure 6.8 shows the roadmap for the simulation of the highway oval track scenario. The most interesting point of the scene is the lane merge at lower left corner, which models a on-ramp situation. As for the intersection scenario in Subsection 6.6.1, the scenario includes the ego-vehicle and 40 obstacles. Simulation frequency is $10Hz$, prediction horizon is $10s$ and prediction step duration $\Delta t = 0.1s$.

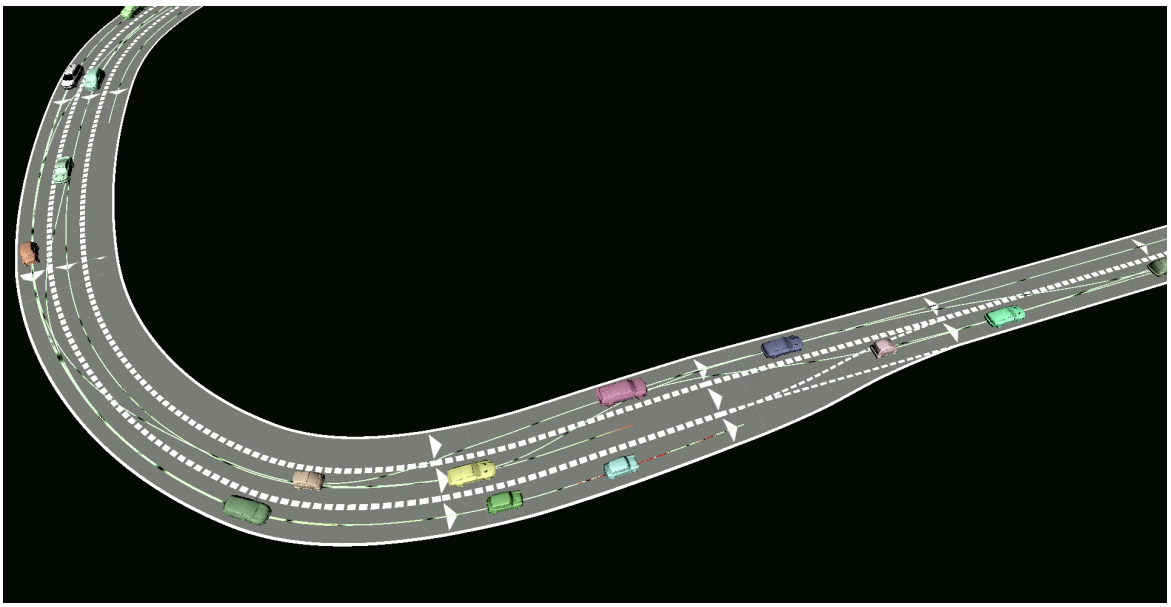


Fig. 6.9 Modest traffic density on the oval track before the lane merge section. The traffic flow is fluid.

Figure 6.9 shows the situation at the lane merge with modest traffic density. For each vehicle, the currently most probably planned trajectory of the next 10 seconds is shown. The black dots in the green trajectories mark each second of the prediction, the distance between

the black dots is proportional to the planned velocity. In this situation, all cars on the right lane are able to merge without stop to the middle lane, but from the length of the prediction trajectory can be seen that the velocity in this road sections is reduced.

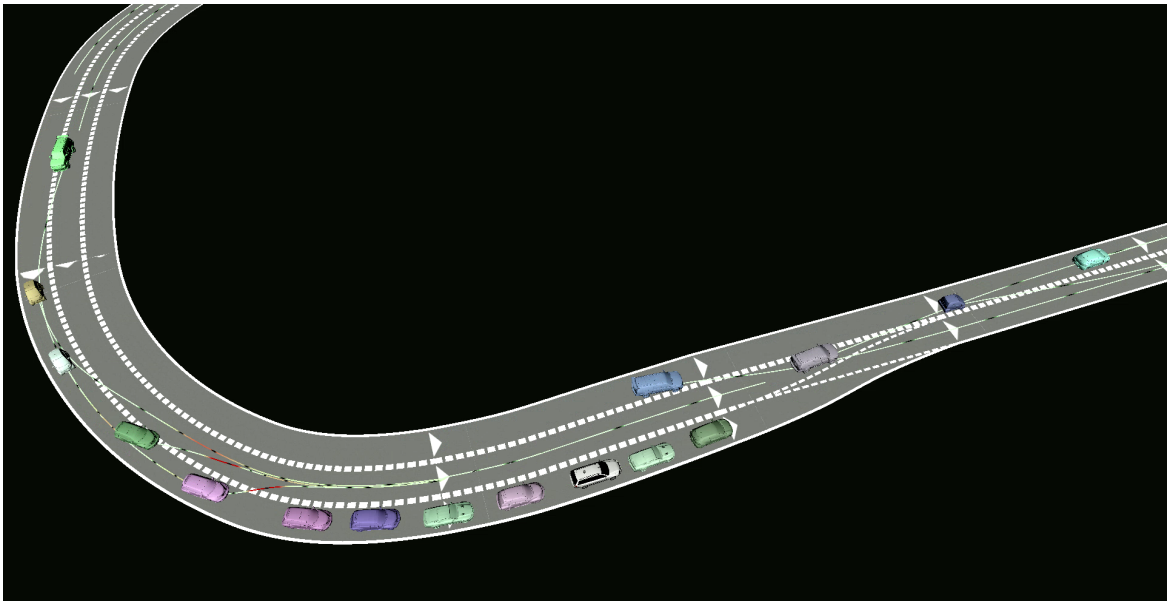


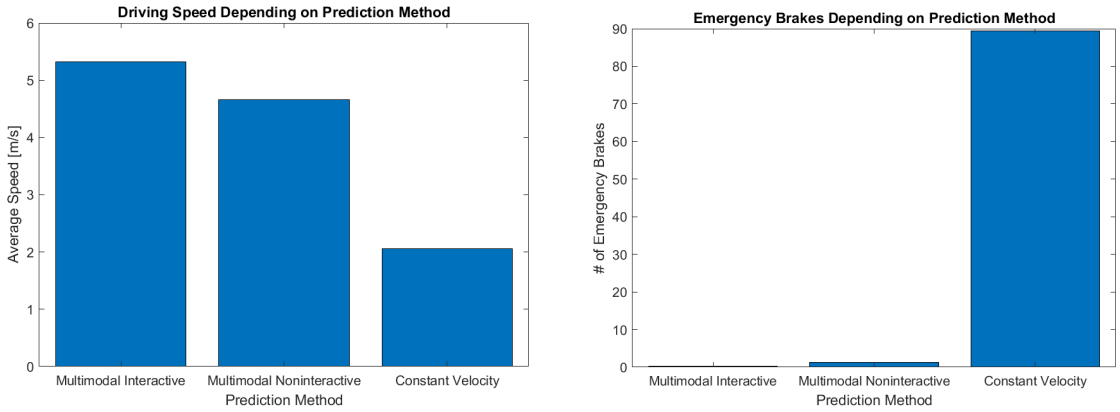
Fig. 6.10 Jammed traffic on oval track before lane merge. A very defensive driver awaits a large gap, blocking other cars.

Figure 6.10 shows the same section of the oval track after a jam has built up on the merging lane. The green car in the front position of the merge lane has a very defensive driving-style and needs therefore a large gap on the priority lane before it starts merging. The eighth and ninth car have a more aggressive driving-style and try to avoid the merge by changing to the middle lane. The blue car on the left lane shows a very dangerous behavior by doing a lane change into the merge section.

As in Subsection 6.6.1, the influence of the prediction method on efficiency and safety/comfort is evaluated. Again, the limited prediction methods are applied only for the ego-vehicle. For each prediction method, the simulation is run 5 times for 120 seconds.

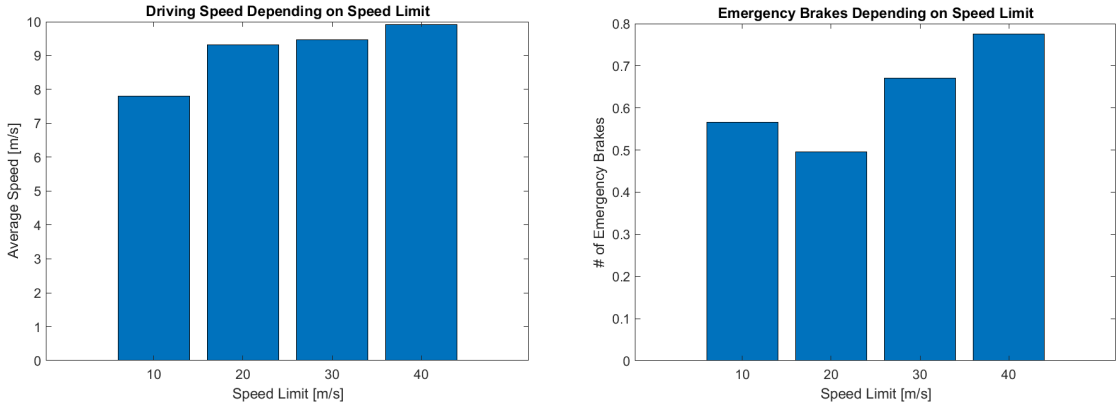
Figure 6.11(a) shows the average speed of the ego-vehicle. As expected, the interaction-aware method shows the best efficiency since it predicts correctly that the other vehicles will yield, when it has the priority. The difference to the achieved speed when using the other prediction methods is not as big as in the intersection scenario, since the merge section is passed not as often as the intersection.

Figure 6.11(b) shows the number of emergency brakes per km. In this scenario, the constant velocity method suffers even more from the high collision risk in the curves than in the intersection scenario.



(a) (b)

Fig. 6.11 Efficiency and safety/comfort for different prediction methods in oval track scenario.



(a) (b)

Fig. 6.12 Efficiency and safety/comfort depending on speed limit in oval track scenario.

A second evaluation is done to examine the influence of the speed limit on efficiency and safety/comfort. The speed limit for the oval track is set to 10, 20, 30 and 40 m/s. Again, the simulation is run 5 times for 120 seconds, each time with different initial positions and different assignment of driving-style.

Figure 6.12(a) shows, how the average speed of all cars evolves with the speed limit. It increases only slightly, which is partly caused be the traffic jams at the lane merge, but which is also the effect of the relatively tight curves of the oval track. Figure 6.12(b) shows the tendency to an increased number of emergency brakes with the speed, which is not surprising.

	Intersection	Oval Track
Scenario Length [s]	300	300
Prediction Frequency [Hz]	10	10
Steps / Prediction	100	100
# of Targets	41	41
# of Intersection Crossings / Scenario	154	0
# of Lane Changes / Scenario	1026	1098
∅# of Maneuvers / Prediction	147.7	136.2
∅# of Maneuver Pairs / Prediction	9854	8261
∅# of Evaluated Risks / Prediction	82378	48225
∅ System Time [ms] / Prediction	127.2	87.1
∅ CPU Time [ms] / Prediction	249.8	159.6

Table 6.1 Statistics for Evaluations

6.7 Summary and Conclusion

To further evaluate the prediction and simulation system, two additional simulation runs for 300 seconds have been done. Table 6.1 summarizes the results. Both runs had a duration of 5 minutes with 40 obstacles + ego-vehicle. Each run produced 3000 scenario predictions over 10 seconds.

In the intersection scenario simulation, there were $\approx$ 1800 crossings / hour, which is an realistic upper limit of an unsignalized intersection. The number of lane changes per vehicle amounts to $\approx$ 5 / min, which is unrealistically high. Obviously, the parameter for the lane change incentive would have to be adjusted for a more realistic driving scenario. But for testing and evaluation purposes, a high lane change frequency is beneficial.

The average number of maneuvers per prediction was $\approx$ 3.7 / obstacle in the intersection scenario simulation and 3.4 / obstacle in the oval track scenario simulation. After subtracting the always included trash and keep lane maneuvers there remain on average 1.7 (1.4) feasible lane change and/or turn maneuvers.

The number of maneuver pairs to be checked for collision amounts to 9854 and 8261, which should result in 985400 and 826100 collision risk calculations per scenario prediction (based on 100 prediction steps). The actual figures are much lower (by the factor of 12 and 17). This is the result of optimizations, which suppresses the risk calculation in obviously risk-free configurations.

The required system time for one prediction of the intersection scenario simulation was 127 ms, which is a little bit slower than real-time, given a frequency of 10 Hz. In the oval track scenario simulation, the lower number of feasible maneuvers enabled a real-time

calculation in 87 ms. The consumed CPU time was approximately twice as high due to the multi-threaded implementation of the risk calculation.

In summary, the evaluation shows that it is theoretically possible to drive with a simple constant velocity prediction model; no collisions occurred during the five simulation runs. But the ride becomes very slow and uncomfortable. Using an interaction-aware multi-modal prediction method leads to higher efficiency and safety and to more comfortable rides.

The main contribution of this chapter was to show that the output of the multi-modal interaction-aware trajectory prediction can be taken as behavior plan of the agents. The simulation proves that the plans are efficient and accident free even in complex merge and intersection scenarios.

7 Summary and Outlook

7.1 Summary

This work has presented a complete system for the perception, prediction and simulation of complex urban traffic scenarios for the purpose of autonomous driving.

Chapter 2: Environment and Prerequisites documents the hard- and software environment for the development and evaluation of the thesis. In addition, Gaussian distributions and covariance matrices, which are frequently used in subsequent chapters, are introduced.

In **Chapter 3: Perception and Multi Object Tracking** of the relevant objects in the environment of an autonomous vehicle based on the input data of a multi-beam LIDAR is presented. As part of the analysis of the LIDAR data, one major contribution of this work is the proof of the first thesis:

Thesis 1: To separate ground and obstacle pixels in LIDAR data, an algorithm based on a range image representation can achieve better results than conventional solutions such as RANSAC.

The thesis is proven qualitatively and quantitatively by comparing it to a RANSAC implementation for point clouds.

It is also shown how object pixels are clustered, how a bounding box around the pixels is found and how the measurements are assigned to the tracks. Tracking requires application of filters. Various types of filters have been implemented and the evaluation of these filters in real-world scenarios proves a further thesis of this work:

Thesis 2: For object tracking in urban environments, Kalman filters with curvilinear motion models are not suitable due to their instability at low velocities.

The evaluation proves the instability of these filters in urban traffic scenarios and discusses the reasons for this.

The perception system is completed with methods for occlusion handling, motion type estimate, object classification and track existence estimate.

Chapter 4: Collision Risk Calculation presents two innovative algorithms to calculate the probability of a collision between two rectangular objects. The evaluation in a simulation and a real-world traffic scenario proves the following thesis:

Thesis 3: he calculation of the future collision risk between two rectangular moving objects based on an analytic algorithm can be efficient enough to check several thousand trajectory pairs per second.

The accuracy of the analytic algorithms is comparable to the results of a Monte Carlo simulation, while being up to 800 times faster.

In **Chapter 5: Traffic Scenario Prediction** an interaction-aware prediction system is presented. It handles most of the relevant urban traffic situations and is efficient enough to cope with 40 or more traffic participants in real-time. By rolling out all feasible trajectory hypotheses of all agents, future collision risks can be detected and iteratively avoided during subsequent rollouts. The quality and efficiency of the predictions is evaluated and compared to other methods in four real-world scenarios. Additionally, the prediction of very complex and potentially dangerous traffic scenarios are evaluated in a simulation environment. From this follows the proof of thesis 4:

Thesis 4: A rule based multi-modal interaction-aware prediction system is able to predict urban traffic scenarios of almost any complexity for up to 10 seconds or longer.

In **Chapter 6: Planning and Simulation** it is shown that the predicted trajectory of the ego-vehicle may be taken as motion plan and fed into the controller of the test system. The output of the controller is then processed by a traffic scenario simulator which allows to evaluate the performance of the predictions system in confusing and dangerous situations. In this way, the following thesis is proven:

Thesis 5: Local behavior planning for the ego-vehicle can benefit from interaction-aware trajectory predictions.

The proofs for the above theses are the major contributions of this work. Additional contributions are documented in the summaries of the Chapters 3 and 5.

7.2 Outlook

While the presented system already achieves good results in predicting traffic scenarios, there are many opportunities to improve the overall performance of the solution. The following developments could provide valuable extensions to the system:

RFS based Extended Object Tracking. As already mentioned in section 3.7, Random Finite Set based methods are proposed for tracking of multiple extended objects. These methods solve the problems of clustering, measurements assignment, clutter, missed detection,

object birth and death simultaneously in a strictly probabilistic manor. But handling 100.000 or more Laser returns as separate measurements is computational unfeasible. A viable option could be to pre-cluster those LIDAR measurements, which undoubtedly originate from the same object into pseudo measurements to reduce the computational burden. Moreover, it could be studied, whether a highly parallel implementation of the proposed algorithms could make them applicable for the purpose of autonomous driving.

Fusion of LIDAR and Camera Data. The presented system uses only LIDAR data for environment perception. In the last years, object detection and classification based on RGB images has made impressive progress. But due to the low reliability of those methods to verify the distance of the detected objects, camera only tracking is not safe enough. Fusing RGB and LIDAR data should result in better segmentation and classification performance while being reliable enough for autonomous driving. Fusion can be applied on raw data level, resulting in a RGB-D image, or on object level. The latter approach may be combined with the previously mentions RFS based tracking methods and replace the burden of extended object handling.

Leveraging Beam Steering Capabilities of Solid State LIDARS. Since several years, solid state LIDARs are under development and partially also already available. As main advantage of this technology are considered the reduced costs of the devices. But they also offer the possibility of controlling the direction of the laser beam by software. Conventional LIDARs scan the whole field of view (FOV) with the same resolution and a constant frequency. Beam steering would allow to scan subregions of the FOV with increased resolution to get more precise information about far away objects after they have been detected. Moreover, regions of the FOV containing fast moving objects could be scanned with higher frequency for an enhanced state estimate. This approach could be implemented by providing feedback of the tracking results to the LIDAR controller.

Collision Risk Analysis Depending on Temporal Aspects and Collision Severity. The current system uses a fixed probability threshold to decide, whether a subordinate traffic participants will react on a collision risk or proceed for the moment without change in his motion plan. This approach is a little simplified. Further research should evaluate the influence of the predicted temporal evolution on the reaction. While imminent collisions require immediate reaction, even if the probability of the collision is low, the same collision risk in a more distant future may be acceptable for the time being. Moreover, the predicted severity of a potential collision must be taken into account. Colliding with high velocity with a vulnerable traffic participant must be avoided in all cases, while a moderate risk for a collision with low velocity difference between cars, for example during parking, may be acceptable.

Completing Behavior Types for Scenario Prediction. The presented prediction system uses the trash maneuver type (see Subsection 5.5.1) as fallback for all unrecognized motion behavior. But there are a couple of scenarios, for which the trash maneuver provides only a rough approximation and which should be handled explicitly. Among these are swerving and usage of the oncoming lane for overtake and obstacle avoidance. Also additional intersection types, like traffic light crossings and 4-way-stops must be considered. For lane changes and lane merges, courtesy and speed synchronization should be taken into account.

Leveraging Turn Signal Detection for Prediction. Even if the turn signal indicator is not always activated by human drivers as required, it provides in many situations valuable information about the intention of the driver. It is therefore already included in the calculation of the intention estimate (see Subsection 5.5.5), but a detection of the indicator signal by the perception system is still pending. The signal cannot be detected in LIDAR data, but must be extracted from the RGB image of a camera. The pose and classification estimate of the perception system may provide the basis to find regions of interest for the localization of the turn signals. Since the color information of the RGB images is too unreliable to distinguish a turn signal form a tail light or a brake light, the flashing of the turn signal must be analyzed. In [70], a Fast Fourier Transform is proposed to detect a turn signal in the frequency domain. Other approaches, such as machine learning-based classifiers, may also be viable.

Driving Style Analysis and Recognition. As mentioned in Subsection 5.8.4, the static parameters of the IDM equation have strong influence on the motion prediction of the agent. In this work, the expected values of these parameters and their variation width are estimated ad hoc depending on the vehicle type. Future research should use observations of real-world traffic scenarios to evaluate the mean and the variance of the parameters. Moreover, this works assumes, that the parameters are perfectly correlated to the driving style and therefore also to each other. Future analysis of real-world traffic data should provide more realistic correlations coefficients, resulting in a covariance matrix for the IDM parameters. Furthermore, the individual driving style of a specific agent should be estimated online as an additional hidden variable in the scenario prediction.

Motion Planning for Ego-Vehicle. As shown in Chapter 6, the motion plan prediction may also be used as basis for the planned trajectory of the ego-vehicle. But planning has additional requirements. At first, the prediction of discrete decisions, like keep lane or change lane, pass or yield at intersection etc. may be easily corrected, when new evidence arrives. The decisions for the planning of the ego-vehicle on the other hand should only be revised in emergency cases. Moreover, the planned trajectory of the ego-vehicle must be smoother than the predicted trajectory of the obstacles. Therefore, additional effort must be taken to create the planned trajectory. A valuable approach could be the Partial Observable Markov

Decision Process (POMDP) framework to realize sequential decision making. There exist extensions for continuous states, observations and actions and efficient tools are available to solve those models.

REFERENCES

[1] M. Aeberhard. *Object-level fusion for surround environment perception in automated driving applications*. VDI Verlag, 2017.

[2] G. Agamennoni, J. I. Nieto, and E. M. Nebot. A Bayesian approach for driving behavior inference. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 595–600. IEEE, 2011.

[3] G. Agamennoni, J. I. Nieto, and E. M. Nebot. Estimation of multivehicle dynamics by considering contextual information. *IEEE Transactions on Robotics*, 28(4):855–870, 2012.

[4] N. A. Ahmed and D. Gokhale. Entropy expressions and their estimators for multivariate distributions. *IEEE Transactions on Information Theory*, 35(3):688–692, 1989.

[5] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn. Search-based optimal motion planning for automated driving. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4523–4530. IEEE, 2018.

[6] R. Altendorfer and C. Wilkmann. What is the collision probability and how to compute it. *arXiv preprint arXiv:1711.07060*, 2017.

[7] M. Althoff and S. Lutz. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1326–1333. IEEE, 2018.

[8] M. Althoff and S. Magdici. Set-based prediction of traffic participants on arbitrary road networks. *IEEE Transactions on Intelligent Vehicles*, 1(2):187–202, 2016.

[9] S. Annell, A. Gratner, and L. Svensson. Probabilistic collision estimation system for autonomous vehicles. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 473–478. IEEE, 2016.

[10] Applanix. Datasheet for the Applanix Pos LV 220. //https://www.applanix.com/downloads/products/specs/POS-LV-Datasheet.pdf, 2019. [Online; accessed 2021-01-03].

[11] I. Arasaratnam and S. Haykin. Cubature Kalman filters. *IEEE Transactions on automatic control*, 54(6):1254–1269, 2009.

[12] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes. 3d Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes. *Robotics and Autonomous Systems*, 83:299–311, 2016.

[13] Autonomos Labs. MadeInGermany. http://www.autonomos.inf.fu-berlin.de/made-in-germany, 2013. [Online; accessed 2021-01-03].

[14] S. Awan, M. Muhamad, K. Kusevic, P. Mrstik, and M. Greenspan. Object class recognition in mobile urban LiDAR data using global shape descriptors. In *2013 International Conference on 3D Vision-3DV 2013*, pages 350–357. IEEE, 2013.

[15] 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*, 17(6):1538–1550, 2016.

[16] M. Bahram, A. Lawitzky, J. Friedrichs, M. Aeberhard, and D. Wollherr. A game-theoretic approach to replanning-aware interactive scene prediction and planning. *IEEE Transactions on Vehicular Technology*, 65(6):3981–3992, 2015.

[17] H. Bai, D. Hsu, and W. S. Lee. Integrated perception and planning in the continuous space: A POMDP approach. *The International Journal of Robotics Research*, 33(9):1288–1302, 2014.

[18] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama. Dynamical model of traffic congestion and numerical simulation. *Physical review E*, 51(2):1035, 1995.

[19] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[20] Y. Bar-Shalom, F. Daum, and J. Huang. The probabilistic data association filter. *IEEE Control Systems Magazine*, 29(6):82–100, 2009.

[21] Y. Bar-Shalom and X.-R. Li. *Multitarget-multisensor tracking: principles and techniques*, volume 19. YBs Storrs, CT, 1995.

[22] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[23] R. Barea, C. Pérez, L. M. Bergasa, E. López-Guillén, E. Romera, E. Molinos, M. Ocana, and J. López. Vehicle detection and localization using 3d LiDAR point cloud and image semantic segmentation. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3481–3486. IEEE, 2018.

[24] U. Baumann, C. Guiser, M. Herman, and J. M. Zollner. Predicting ego-vehicle paths from environmental observations with a deep neural network. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4709–4716. IEEE, 2018.

[25] M. Beard, S. Reuter, K. Granström, B.-T. Vo, B.-N. Vo, and A. Scheel. Multiple extended target tracking with labeled random finite sets. *IEEE Transactions on Signal Processing*, 64(7):1638–1653, 2015.

[26] Y. K. Belyaev. On the number of exits across the boundary of a region by a vector stochastic process. *Theory of Probability & Its Applications*, 13(2):320–324, 1968.

[27] P. Bender, J. Ziegler, and C. Stiller. Lanelets: Efficient map representation for autonomous driving. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 420–425. IEEE, 2014.

[28] R. Bhattacharyya, R. Senanayake, K. Brown, and M. Kochenderfer. Online parameter estimation for human driver behavior prediction. *arXiv preprint arXiv:2005.02597*, 2020.

[29] I. Bogoslavskyi and C. Stachniss. Fast range image-based segmentation of sparse 3D laser scans for online operation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 163–169. IEEE, 2016.

[30] I. Bogoslavskyi and C. Stachniss. Efficient online segmentation for sparse 3d laser scans. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 85(1):41–52, 2017.

[31] Z. Boroujeni, D. Goehring, F. Ulbrich, D. Neumann, and R. Rojas. Flexible unit A-star trajectory planning for autonomous vehicles on structured road maps. In *Vehicular Electronics and Safety (ICVES), 2017 IEEE International Conference on*, pages 7–12. IEEE, 2017.

[32] C. Braeuchle, J. Ruenz, F. Flehmig, W. Rosenstiel, and T. Kropf. Situation analysis and decision making for active pedestrian protection using Bayesian networks. In *6. Tagung Fahrerassistenzsysteme*, 2013.

[33] S. Brechtel, T. Gindele, and R. Dillmann. Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs. In *17th international IEEE conference on intelligent transportation systems (ITSC)*, pages 392–399. IEEE, 2014.

[34] P. Broßeit, M. Rapp, N. Appenrodt, and J. Dickmann. Probabilistic rectangular-shape estimation for extended object tracking. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 279–285. IEEE, 2016.

[35] K. Brown, K. Driggs-Campbell, and M. J. Kochenderfer. Modeling and prediction of human driver behavior: A survey. *arXiv preprint arXiv:2006.08832*, 2020.

[36] S. Challa, M. R. Morelande, D. Mušicki, and R. J. Evans. *Fundamentals of object tracking*. Cambridge University Press, 2011.

[37] D. U. Challenge. Route network definition file (RNDF) and mission data file (MDF) formats. *Tech. Rep., Defense Advanced Research Projects Agency, Tech. Rep.*, 2007.

[38] Z. Cheng, G. Ren, and Y. Zhang. Ground segmentation algorithm based on 3D Lidar Point Cloud. In *2018 International Conference on Mechanical, Electrical, Electronic Engineering & Science (MEEES 2018)*, pages 16–21. Atlantis Press, 2018.

[39] T. Christopher. *Analysis of dynamic scenes: application to driving assistance*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2009.

[40] S. O.-R. A. V. S. Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE Standard J*, 3016:1–16, 2014.

[41] I. J. Cox and S. L. Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 18(2):138–150, 1996.

[42] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson. MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1670–1677. IEEE, 2015.

[43] I. Dagli, M. Brost, and G. Breuel. Action recognition and prediction for driver assistance systems using Dynamic Belief Networks. In *Net. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World*, pages 179–194. Springer, 2002.

[44] L. D'Alfonso, W. Lucia, P. Muraca, and P. Pugliese. Mobile robot localization via EKF and UKF: A comparison based on real data. *Robotics and Autonomous Systems*, 74:122–127, 2015.

[45] F. Damerow and J. Eggert. Risk-aversive behavior planning under multiple situations with uncertainty. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 656–663. IEEE, 2015.

[46] F. Damerow, B. Flade, and J. Eggert. Extensions for the foresighted driver model: Tactical lane change, overtaking and continuous lateral control. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 186–193. IEEE, 2016.

[47] A. Danzer, F. Gies, and K. Dietmayer. Multi-object tracking with interacting vehicles and road map information. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 589–595. IEEE, 2018.

[48] N. Deo and M. M. Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based LSTMs. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1179–1184. IEEE, 2018.

[49] E. W. Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[50] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[51] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the segmentation of 3D LIDAR point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE, 2011.

[52] N. E. Du Toit and J. W. Burdick. Probabilistic collision checking with chance constraints. *IEEE Transactions on Robotics*, 27(4):809–815, 2011.

[53] F. Ebert and H.-J. Wuensche. Dynamic object tracking and 3D surface estimation using Gaussian Processes and Extended Kalman Filter. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1122–1127. IEEE, 2019.

[54] J. Eggert. Predictive risk estimation for intelligent ADAS functions. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 711–718. IEEE, 2014.

[55] J. Eggert, F. Damerow, and S. Klingelschmitt. The foresighted driver model. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 322–329. IEEE, 2015.

[56] J. Eggert and F. Mueller. A foresighted driver model derived from integral expected risk. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1223–1230. IEEE, 2019.

[57] G. Engels, N. Aranjuelo, I. Arganda-Carreras, M. Nieto, and O. Otaegui. 3d object detection from LiDAR data using distance dependent feature extraction. *arXiv preprint arXiv:2003.00888*, 2020.

[58] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[59] J. Fang, D. Zhou, F. Yan, T. Zhao, F. Zhang, Y. Ma, L. Wang, and R. Yang. Augmented LIDAR simulator for autonomous driving. *IEEE Robotics and Automation Letters*, 5(2):1931–1938, 2020.

[60] D. Feng, L. Rosenbaum, and K. Dietmayer. Towards safe autonomous driving: Capture uncertainty in the deep neural network for LiDAR 3d vehicle detection. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3266–3273. IEEE, 2018.

[61] F. Feng, S. Bao, J. R. Sayer, C. Flannagan, M. Manser, and R. Wunderlich. Can vehicle longitudinal jerk be used to identify aggressive drivers? An examination using naturalistic driving data. *Accident Analysis & Prevention*, 104:125–136, 2017.

[62] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[63] E. Fix. *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine, 1985.

[64] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Multi-target tracking using joint probabilistic data association. In *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, pages 807–812. IEEE, 1980.

[65] C. Foundation. Ubuntu 18.04.5 LTS (Bionic Beaver). https://releases.ubuntu.com/bionic/, 2020. [Online; accessed 2021-01-02].

[66] O. S. R. Foundation. ROS Melodic. http://wiki.ros.org/melodic, 2018. [Online; accessed 2021-01-02].

[67] I. Free Software Foundation. GCC, the GNU Compiler Collection. https://gcc.gnu.org/, 2019. [Online; accessed 2021-01-02].

[68] Freie Universitaet Berlin. Dahlem center for machine learning and robotics. https://www.mi.fu-berlin.de/inf/groups/ag-ki/index.html, 2021. [Online; accessed 2021-01-03].

[69] F. Frenet. Sur les courbes a double courbure. *Journal de mathématiques pures et appliquées*, pages 437–447, 1852.

[70] B. Fröhlich, M. Enzweiler, and U. Franke. Will this car change the lane? turn signal recognition in the frequency domain. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 37–42. IEEE, 2014.

[71] C. Frohn, P. Ilov, S. Kriebel, E. Kusmenko, B. Rumpe, and A. Ryndin. Distributed simulation of cooperatively interacting vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 596–601. IEEE, 2018.

[72] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi. GNU scientific library. *Reference Manual. Edition 1.4, for GSL Version 1.4*, 2003.

[73] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3d traffic scene understanding from movable platforms. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):1012–1025, 2013.

[74] Z. Ghahramani. An introduction to hidden Markov models and Bayesian networks. In *Hidden Markov models: applications in computer vision*, pages 9–41. World Scientific, 2001.

[75] T. Gindele, S. Brechtel, and R. Dillmann. A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1625–1631. IEEE, 2010.

[76] P. G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[77] D. Göhring. Controller architecture for the autonomous cars: MadeInGermany and e-instein. *Technical report, Freie Universitat Berlin*, 2012.

[78] D. S. González, J. S. Dibangoye, and C. Laugier. High-speed highway scene prediction based on driver models learned from demonstrations. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 149–155. IEEE, 2016.

[79] D. S. González, M. Garzón, J. S. Dibangoye, and C. Laugier. Human-like decision-making for automated driving in highways. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2087–2094. IEEE, 2019.

[80] D. S. González, V. Romero-Cano, J. S. Dibangoye, and C. Laugier. Interaction-aware driver maneuver inference in highways using realistic driver models. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2017.

[81] A. Gorji, M. B. Menhaj, and S. Shiry. Multiple target tracking for mobile robots using the JPDAF algorithm. In *Tools and Applications with Artificial Intelligence*, pages 51–68. Springer, 2009.

[82] M. Graf, O. Speidel, and K. Dietmayer. A model based motion planning framework for automated vehicles in structured environments. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 201–206. IEEE, 2019.

[83] M. Graf, O. Speidel, and K. Dietmayer. Trajectory planning for automated vehicles in overtaking scenarios. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1653–1659. IEEE, 2019.

[84] M. Graf, O. Speidel, J. Ziegler, and K. Dietmayer. Trajectory planning for automated vehicles using driver models. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1455–1460. IEEE, 2018.

[85] K. Granström, M. Fatemi, and L. Svensson. Gamma Gaussian inverse-Wishart Poisson multi-Bernoulli filter for extended target tracking. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 893–900. IEEE, 2016.

[86] K. Granström and C. Lundquist. On the use of multiple measurement models for extended target tracking. In *Proceedings of the 16th International Conference on Information Fusion*, pages 1534–1541. IEEE, 2013.

[87] K. Granstrom, C. Lundquist, and O. Orguner. Extended target tracking using a gaussian-mixture PHD filter. *IEEE Transactions on Aerospace and Electronic Systems*, 48(4):3268–3286, 2012.

[88] K. Granström, C. Lundquist, and U. Orguner. A gaussian mixture PHD filter for extended target tracking. In *2010 13th International Conference on Information Fusion*, pages 1–8. IEEE, 2010.

[89] K. Granström, S. Renter, M. Fatemi, and L. Svensson. Pedestrian tracking using Velodyne data—stochastic optimization for extended object tracking. In *2017 ieee intelligent vehicles symposium (iv)*, pages 39–46. IEEE, 2017.

[90] K. Granström, S. Reuter, D. Meissner, and A. Scheel. A multiple model PHD approach to tracking of cars under an assumed rectangular shape. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2014.

[91] K. Granström, L. Svensson, S. Reuter, Y. Xia, and M. Fatemi. Likelihood-based data association for extended object tracking using sampling methods. *IEEE Transactions on intelligent vehicles*, 3(1):30–45, 2017.

[92] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche. Fast segmentation of 3D point clouds for ground vehicles. In *2010 IEEE Intelligent Vehicles Symposium*, pages 560–565. IEEE, 2010.

[93] S. Hoermann, D. Stumper, and K. Dietmayer. Probabilistic long-term prediction for autonomous vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 237–243. IEEE, 2017.

[94] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher. An experimental comparison of range image segmentation algorithms. *IEEE transactions on pattern analysis and machine intelligence*, 18(7):673–689, 1996.

[95] A. Houénou, P. Bonnifait, and V. Cherfaoui. Risk assessment for collision avoidance systems. In *Intelligent Transportation Systems (ITSC), 2014 IEEE International Conference on*, pages 386–391. IEEE, 2014.

[96] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1671–1678. IEEE, 2017.

[97] V. L. Inc. HDL-64-E S3 USER'S MANUAL AND PROGRAMMING GUIDE. https://velodynelidar.com/downloads/manuals, 2019. [Online; accessed 2021-01-02].

[98] K. Jayaraman, D. M. Tilbury, X. J. Yang, A. K. Pradhan, and L. P. Robert. Analysis and prediction of pedestrian crosswalk behavior during automated vehicle interactions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6426–6432. IEEE, 2020.

[99] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[100] E. Julian and F. Damerow. Complex lane change behavior in the foresighted driver model. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 1747–1754. IEEE, 2015.

[101] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[102] Y. Kang, H. Yin, and C. Berger. Test your self-driving algorithm: An overview of publicly available driving datasets and virtual testing environments. *IEEE Transactions on Intelligent Vehicles*, 4(2):171–185, 2019.

[103] J. Karush. On the Chapman-Kolmogorov equation. *The Annals of Mathematical Statistics*, 32(4):1333–1337, 1961.

[104] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416–442, 2015.

[105] A. Kesting, M. Treiber, and D. Helbing. General lane-changing model MOBIL for car-following models. *Transportation Research Record*, 1999(1):86–94, 2007.

[106] P. Kmiotek and Y. Ruichek. Representing and tracking of dynamics objects using oriented bounding box and extended Kalman filter. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 322–328. IEEE, 2008.

[107] P. König, S. Aigner, and M. Körner. Enhancing traffic scene predictions with generative adversarial networks. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1768–1775. IEEE, 2019.

[108] K. C. Kumar and S. Al-Stouhi. Real-time spatial-temporal context approach for 3D object detection using LiDAR. In *VEHITS*, pages 432–439, 2020.

[109] R. Labbe. Kalman and Bayesian filters in Python. *Chap*, 7:246, 2014.

[110] A. Lambert, D. Gruyer, and G. Saint Pierre. A fast Monte Carlo algorithm for collision probability estimation. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 406–411. IEEE, 2008.

[111] S. Lange, F. Ulbrich, and D. Goehring. Online vehicle detection using deep neural networks and LiDAR based preselected image patches. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 954–959. IEEE, 2016.

[112] S. Lefèvre, C. Laugier, and J. Ibañez-Guzmán. Exploiting map information for driver intention estimation at road intersections. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 583–588. IEEE, 2011.

[113] S. Lefèvre, C. Laugier, and J. Ibañez-Guzmàn. Intention-aware risk estimation for general traffic situations, and application to intersection safety. 2013.

[114] S. Lefèvre, C. Laugier, J. Ibañez-Guzmán, and P. Bessiere. Modelling dynamic scenes at unsignalised road intersections. 2011.

[115] S. Lefèvre, D. Vasquez, and C. Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech Journal*, 1(1):1, 2014.

[116] D. Lenz, F. Diehl, M. T. Le, and A. Knoll. Deep neural networks for Markovian interactive scene prediction in highway scenarios. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 685–692. IEEE, 2017.

[117] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking: II. ballistic target models. In *Signal and Data Processing of Small Targets 2001*, volume 4473, pages 559–581. International Society for Optics and Photonics, 2001.

[118] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking: III. measurement models. In *Signal and Data Processing of Small Targets 2001*, volume 4473, pages 423–446. International Society for Optics and Photonics, 2001.

[119] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking: decision-based methods. In *Signal and Data Processing of Small Targets 2002*, volume 4728, pages 511–534. International Society for Optics and Photonics, 2002.

[120] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking. Part I. dynamic models. *IEEE Transactions on aerospace and electronic systems*, 39(4):1333–1364, 2003.

[121] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking. Part V. multiple-model methods. *IEEE Transactions on Aerospace and Electronic Systems*, 41(4):1255–1321, 2005.

[122] M. Liebner, M. Baumann, F. Klanner, and C. Stiller. Driver intent inference at urban intersections using the intelligent driver model. In *2012 IEEE Intelligent Vehicles Symposium*, pages 1162–1167. IEEE, 2012.

[123] C. Lienke, C. Wissing, M. Keller, T. Nattermann, and T. Bertram. Predictive driving: Fusing prediction and planning for automated highway driving. *IEEE Transactions on Intelligent Vehicles*, 4(3):456–467, 2019.

[124] R. Mahler. "statistics 102" for multisource-multitarget detection and tracking. *IEEE Journal of Selected Topics in Signal Processing*, 7(3):376–389, 2013.

[125] R. P. Mahler. Multitarget Bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic systems*, 39(4):1152–1178, 2003.

[126] R. P. Mahler. " statistics 101" for multisensor, multitarget data fusion. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):53–64, 2004.

[127] J. Mänttäri, J. Folkesson, and E. Ward. Learning to predict lane changes in highway scenarios using dynamic filters on a generic traffic representation. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1385–1392. IEEE, 2018.

[128] G. Maps. Google maps. *Dipetik Desember*, 14:2015, 2015.

[129] M. Meghjani, Y. Luo, Q. H. Ho, P. Cai, S. Verma, D. Rus, and D. Hsu. Context and intention aware planning for urban driving. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2891–2898. IEEE, 2019.

[130] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. Rangenet++: Fast and accurate LiDAR semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019.

[131] N. Monot, X. Moreau, A. Benine-Neto, A. Rizzo, and F. Aioun. Comparison of rule-based and machine learning methods for lane change detection. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 198–203. IEEE, 2018.

[132] F. Moosmann, O. Pink, and C. Stiller. Segmentation of 3D LIDAR data in non-flat urban environments using a local convexity criterion. In *2009 IEEE Intelligent Vehicles Symposium*, pages 215–220. IEEE, 2009.

[133] P. Morere, R. Marchant, and F. Ramos. Continuous state-action-observation POMDPs for trajectory planning with Bayesian optimisation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8779–8786. IEEE, 2018.

[134] M. Morsali, J. Åslund, and E. Frisk. Trajectory planning for autonomous vehicles in time varying environments using Support Vector Machines. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–6. IEEE, 2018.

[135] S. Mukherjee, S. Wang, and A. Wallace. Interacting vehicle trajectory prediction with convolutional recurrent neural networks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4336–4342. IEEE, 2020.

[136] B. Naujoks and H.-J. Wuensche. An orientation corrected bounding box fit based on the convex hull under real time constraints. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–6. IEEE, 2018.

[137] M. Naumann, H. Königshof, and C. Stiller. Provably safe and smooth lane changes in mixed trafic. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1832–1837. IEEE, 2019.

[138] G. F. Newell. A simplified car-following theory: a lower order model. *Transportation Research Part B: Methodological*, 36(3):195–205, 2002.

[139] P.-J. Nordlund and F. Gustafsson. Probabilistic conflict detection for piecewise straight paths. 2008.

[140] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[141] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE, 2018.

[142] D. Perdomo López. *Scenario interpretation for automated driving at urban intersections*. PhD thesis, 2018.

[143] A. Petrovskaya and S. Thrun. Model based vehicle tracking in urban environments. In *IEEE International Conference on Robotics and Automation, Workshop on Safe Navigation*, volume 1, pages 1–8, 2009.

[144] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr. Lanelet2: A high-definition map framework for the future of automated driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1672–1679. IEEE, 2018.

[145] J. Quehl, H. Hu, S. Wirges, and M. Lauer. An approach to vehicle trajectory prediction using automatically generated traffic maps. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 544–549. IEEE, 2018.

[146] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[147] A. Rasouli, I. Kotseruba, and J. K. Tsotsos. Understanding pedestrian behavior in complex traffic scenes. *IEEE Transactions on Intelligent Vehicles*, 3(1):61–70, 2017.

[148] S. K. Reddy and P. K. Pal. Segmentation of point cloud from a 3D LIDAR using range difference between neighbouring beams. In *Proceedings of the 2015 Conference on Advances in Robotics*, pages 1–6, 2015.

[149] S. K. Reddy and P. K. Pal. Computing an unevenness field from 3D laser range data to obtain traversable region around a mobile robot. *Robotics and Autonomous Systems*, 84:48–63, 2016.

[150] S. Reuter. *Multi-object tracking using random finite sets*. PhD thesis, Universität Ulm, 2014.

[151] D. Roy, T. Ishizaka, C. K. Mohan, and A. Fukuda. Vehicle trajectory prediction at intersections using interaction based generative adversarial networks. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2318–2323. IEEE, 2019.

[152] S. Russell and P. Norvig. Artificial intelligence: a modern approach. 2002.

[153] D. M. Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev. Driving in dense traffic with model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5385–5392. IEEE, 2020.

[154] W. J. Schakel, V. L. Knoop, and B. van Arem. Integrated lane change model with relaxation and synchronization. *Transportation Research Record*, 2316(1):47–57, 2012.

[155] O. Scheel, L. Schwarz, N. Navab, and F. Tombari. Situation assessment for planning lane changes: Combining recurrent models and prediction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2082–2088. IEEE, 2018.

[156] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[157] M. Schreier. *Bayesian environment representation, prediction, and criticality assessment for driver assistance systems*. PhD thesis, Technischen Universitaet Darmstadt, 2015.

[158] M. Schreier. *Bayesian environment representation, prediction, and criticality assessment for driver assistance systems M. Sc. Matthias Schreier, Darmstadt*. PhD thesis, Dissertation, Technische Universität Darmstadt, 2016. Fortschritt-Berichte . . . , 2016.

[159] R. Schubert, E. Richter, and G. Wanielik. Comparison and evaluation of advanced motion models for vehicle tracking. In *2008 11th international conference on information fusion*, pages 1–6. IEEE, 2008.

[160] J. Schulz, C. Hubmann, J. Löchner, and D. Burschka. Interaction-aware probabilistic behavior prediction in urban environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3999–4006. IEEE, 2018.

[161] J. Schulz, C. Hubmann, J. Löchner, and D. Burschka. Multiple model unscented Kalman filtering in dynamic Bayesian networks for intention estimation and trajectory prediction. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1467–1474. IEEE, 2018.

[162] J. Schulz, C. Hubmann, N. Morin, J. Löchner, and D. Burschka. Learning interaction-aware probabilistic driver behavior models from urban scenarios. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1326–1333. IEEE, 2019.

[163] T. J. Sheskin. A Markov chain partitioning algorithm for computing steady state probabilities. *Operations Research*, 33(1):228–235, 1985.

[164] J. Siek, A. Lumsdaine, and L.-Q. Lee. *The Boost graph library: user guide and reference manual*. Addison-Wesley, 2002.

[165] Statistisches Bundesamt. Road traffic accidents. https://www.destatis.de/EN/Themes/Society-Environment/Traffic-Accidents/_node.html, 2021. [Online; accessed 2021-01-03].

[166] S. Thrun, W. Burgard, D. Fox, and R. Arkin. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents series. MIT Press, 2005.

[167] G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE Melecon*, volume 83, page A10, 1983.

[168] Q. Tran and J. Firl. Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 918–923. IEEE, 2014.

[169] M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[170] M. Treiber and V. Kanagaraj. Comparing numerical integration schemes for time-continuous car-following models. *Physica A: Statistical Mechanics and its Applications*, 419:183–195, 2015.

[171] M. Treiber and A. Kesting. Traffic flow dynamics. *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg*, 2013.

[172] M. Treiber and A. Kesting. Automatic and efficient driving strategies while approaching a traffic light. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1122–1128. IEEE, 2014.

[173] M. Treiber and A. Kesting. The intelligent driver model with stochasticity-new insights into traffic flow oscillations. *Transportation research procedia*, 23:174–187, 2017.

[174] M. Treiber, A. Kesting, and D. Helbing. Delays, inaccuracies and anticipation in microscopic traffic models. *Physica A: Statistical Mechanics and its Applications*, 360(1):71–88, 2006.

[175] M. Tsogas, A. Polychronopoulos, and A. Amditis. Unscented Kalman filter design for curvilinear motion models suitable for automotive safety applications. In *2005 7th International Conference on Information Fusion*, volume 2, pages 8–pp. IEEE, 2005.

[176] S. Ulbrich and M. Maurer. Towards tactical lane change behavior planning for automated vehicles. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 989–995. IEEE, 2015.

[177] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer. Defining and substantiating the terms scene, situation, and scenario for automated driving. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 982–988. IEEE, 2015.

[178] B.-N. Vo and W.-K. Ma. The Gaussian mixture probability hypothesis density filter. *IEEE Transactions on signal processing*, 54(11):4091–4104, 2006.

[179] B.-T. Vo and B.-N. Vo. Labeled random finite sets and multi-object conjugate priors. *IEEE Transactions on Signal Processing*, 61(13):3460–3475, 2013.

[180] B. Völz, K. Behrendt, H. Mielenz, I. Gilitschenski, R. Siegwart, and J. Nieto. A data-driven approach for pedestrian intention estimation. In *2016 ieee 19th international conference on intelligent transportation systems (itsc)*, pages 2607–2612. IEEE, 2016.

[181] E. A. Wan and R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[182] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager. Game theoretic planning for self-driving cars in competitive scenarios. In *Robotics: Science and Systems*, 2019.

[183] X. Wang, J. Wu, Y. Gu, H. Sun, L. Xu, S. Kamijo, and N. Zheng. Human-like maneuver decision using LSTM-CRF model for on-road self-driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 210–216. IEEE, 2018.

[184] G. Welch, G. Bishop, et al. An introduction to the Kalman filter, 1995.

[185] M. Werling, J. Ziegler, S. Kammel, and S. Thrun. Optimal trajectory generation for dynamic street scenarios in a Frenet frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993. IEEE, 2010.

[186] J. Wiest, M. Höffken, U. Kreßel, and K. Dietmayer. Probabilistic trajectory prediction with Gaussian mixture models. In *2012 IEEE Intelligent Vehicles Symposium*, pages 141–146. IEEE, 2012.

[187] J. Wiest, M. Karg, F. Kunz, S. Reuter, U. Kreßel, and K. Dietmayer. A probabilistic maneuver prediction framework for self-learning vehicles with application to intersections. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 349–355. IEEE, 2015.

[188] Wikipedia contributors. Conjugate prior — Wikipedia, the free encyclopedia, 2021. [Online; accessed 30-April-2021].

[189] B. Wu, A. Wan, X. Yue, and K. Keutzer. Squeezeseg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LIDAR point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[190] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang. Vehicle trajectory prediction by integrating physics-and maneuver-based approaches using interactive multiple models. *IEEE Transactions on Industrial Electronics*, 65(7):5999–6008, 2017.

[191] H. Yi, S. Shi, M. Ding, J. Sun, K. Xu, H. Zhou, Z. Wang, S. Li, and G. Wang. Segvoxelnet: Exploring semantic context and depth-aware features for 3d vehicle detection from point cloud. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2274–2280. IEEE, 2020.

[192] W. Zhan, A. de La Fortelle, Y.-T. Chen, C.-Y. Chan, and M. Tomizuka. Probabilistic prediction from planning perspective: Problem formulation, representation simplification and evaluation metric. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 1150–1156. IEEE, 2018.

[193] Y. Zhang, J. Wang, X. Wang, C. Li, and L. Wang. 3d lidar-based intersection recognition and road boundary detection method for unmanned ground vehicle. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 499–504. IEEE, 2015.

SELBSTÄNDIGKEITSERKLÄRUNG

Name: Philipp
Vorname: Andreas

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

Datum: Unterschrift: