Conception of Control Paradigms for Teleoperated Driving Tasks in Urban Environments

Dissertation

zur
Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

der Fakultät für Maschinenbau und Schiffstechnik
an der Universität Rostock

vorgelegt von
Dmitrij Schitz, M. Sc.
geb. am 14.07.1994 in Taschkent, Usbekistan

Gutachter: Prof. Dr.-Ing. Harald Aschemann
Lehrstuhl für Mechatronik, Universität Rostock

Prof. Dr.-Ing. Dr. h.c. Oliver Sawodny
Institut für Systemdynamik, Universität Stuttgart

Tag der mündlichen Prüfung: 22. Juni 2022

Lehrstuhl für Mechatronik, Universität Rostock
2022

https://doi.org/10.18453/rosdok_id00004020

Preface

This thesis was written as part of my scientific work as a doctoral student at BMW AG in Munich.

My special thanks go to Prof. Dr.-Ing. Harald Aschemann, head of the Chair of Mechatronics at the University of Rostock, for taking over the scientific supervision and for the valuable suggestions and advice on this work. The numerous discussions with him and his professional suggestions challenged and encouraged me in equal measure and contributed significantly to the success of this work.

I further want to thank Prof. Dr.-Ing. Dr. h.c. Oliver Sawodny for serving as second reviewer of this thesis and Dr.-Ing. Dominik Rieth, who initiated this work, for his support and supervision of the scientific research on the part of BMW AG.

I would like to further express my gratitude to all colleagues. I will have enduring memories of the successful demonstrations of the developed algorithms as well as the many hours of testing in the experimental vehicles.

Finally, I would like to thank my family for always encouraging and supporting me in my life, education and career.

Munich, October 9, 2022 Dmitrij Schitz

Conception of Control Paradigms for Teleoperated Driving Tasks in Urban Environments

– Abstract –

Teleoperated driving enables a fallback solution for autonomous driving. A human operator remotely takes over the control of the automated system for some time-span. By keeping the operator in the vehicles control loop, this mobility concept benefits from the humans cognitive abilities. However, teleoperation poses its own challenges. To overcome these, this contribution presents several teleoperated driving concepts. The presented concepts raise the level of autonomy in different ways, while the human operator remains the main decision-maker.

For this purpose, real-time capable algorithms for path and trajectory planning are developed that aim for safe and comfortable motions of the remote controlled vehicle. For the planning of collision-free and smooth paths in real-time, this work modifies the Constrained CHOMP algorithm, a local path optimization technique. A second optimization step, the domain step, is introduced in order to prevent the path to get stuck in poor local minima. Additionally, the original algorithm is extended in order to take the vehicle dimensions, the vehicles non-holonomic constraints and a reference path into account in the optimization. For the planning of optimal trajectories in real-time, efficient formulations of constrained linear-quadratic optimal control problems are derived separately for the longitudinal and lateral dynamics. They are solved by means of a time-variant, linear MPC scheme using Quadratic Programming. To address the solvability of the optimal control problems as well as to meet the requirements w.r.t. comfort and safety that arise for automated driving, this thesis introduces a two-stage constraint softening using slack variables.

To promote comfort and safety for direct control, the model-predictive cruise control for direct teleoperated driving tasks is presented in this work. This assistance concept uses the presented trajectory optimization, in order to adapt on-board the operator's control commands for the longitudinal dynamics in real-time, for example due to an incorrect assessment of the driving space or due to the delayed or even interrupted vehicle communication. To overcome high communication time delays, this work presents the corridor-based motion planning concept. The operator is enabled to specify an area – the corridor – towards a desired destination, within the automated vehicle then calculates an optimal motion by itself using the novel hybrid motion planning, combining the modified, two-step Constrained CHOMP algorithm and the model-predictive trajectory generation for the longitudinal and lateral dynamics. To support the operator in the planning task, this thesis presents a supervisory control concept. A modified RRT algorithm continuously generates feasible paths that the operator can choose from. The path selected by the operator is then forwarded to the novel hybrid motion planning algorithm.

Finally, this thesis introduces a novel concept for path optimization based on deep learning. For this purpose different problem statements together with different neural network architectures are examined based on data generated using the modified, two-step Constrained CHOMP algorithm.

The proposed teleoperated driving concepts as well as the developed algorithms are evaluated using real driving experiments. The results show an increase in comfort and safety using the teleoperated driving concepts and a significant reduction in computational effort using the developed algorithms.

Contents

List of Figures V

List of Tables IX

Abbreviations XI

List of Symbols XIII

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 State-of-the-Art . . . 2
  - 1.2.1 Teleoperated Driving . . . 2
  - 1.2.2 Automated Driving . . . 4
  - 1.2.3 Motion Planning . . . 9
- 1.3 Challenges in Teleoperated Driving . . . 16
- 1.4 Objectives . . . 17
- 1.5 Outline . . . 19

I Theory 21

2 Path Optimization Using Constrained CHOMP 23
- 2.1 Objective Functional . . . 23
- 2.2 Dual Projected Newton Method . . . 26
- 2.3 Complement of the Objective Functional with Curvature Penalty Function . . . 27
- 2.4 Complement of the Objective Functional with Reference Path Objective . . . 28
- 2.5 Local Minima Avoidance . . . 29

3 Model-Predictive Trajectory Optimization Using Quadratic Programming 33
- 3.1 Linear Vehicle Dynamics . . . 33
- 3.2 Discrete-Time Problem Representation . . . 34
- 3.3 Formulation of the Constrained Quadratic Program . . . 36
- 3.4 Constraint Softening: Quadratic Program Extension through Slack-Variables . . . 37
- 3.5 Longitudinal Trajectory Optimization . . . 39
  - 3.5.1 Safety and Comfort-Related Constraints . . . 40
- 3.6 Lateral Trajectory Optimization . . . 42
  - 3.6.1 Safety and Comfort-Related Constraints . . . 43

II Control Concepts for Teleoperated Driving Tasks in Urban Environments 47

4 Model-Predictive Cruise Control for Direct Teleoperated Driving Tasks 49
4.1 Overview on the Approach 49
4.2 Experimental Results 50
4.2.1 Reaction to a Right-of-Way Vehicle at an Intersection 51
4.2.2 Reaction to a Suddenly Braking Vehicle in Front 51
4.3 Conclusion 52

5 Corridor-Based Motion Planning for Shared Control 55
5.1 Overview on the Approach 55
5.1.1 Corridor Specification 57
5.2 Experimental Results 57
5.2.1 Obstacle passing in a narrow, static Environment 57
5.2.2 Avoiding suddenly appearing static Obstacle in a turning Scenario . . 59
5.2.3 Obstacle passing dealing with oncoming Traffic 61
5.2.4 Keeping distance to a Pedestrian on the Road 61
5.3 Conclusion 64

6 Automatic Path Generation for Supervisory Control 65
6.1 Overview on the Approach 65
6.1.1 RRT-based Path Search 66
6.2 Experimental Results 68
6.2.1 Path Exploration and Clustering Results at an Intersection 69
6.2.2 Interactive Path Planning in a Driving Maneuver 69
6.3 Conclusion 72

III Machine Learning Approaches for Motion Planning 73

7 Path Optimization for Autonomous Driving using Deep Learning 75
7.1 Related Work 75
7.2 Data 76
7.3 Methods 78
7.3.1 Problem Statement 78
7.3.2 Network Architectures 78
7.3.3 Redefining the Problem Statement 81
7.3.4 Trained Models based on the redefined Problem Statement 81
7.4 Neural Network Training 83
7.4.1 Hyperparameter Tuning 85
7.5 Evaluation 87
7.5.1 Quantitative Results 87
7.5.2 Qualitative Results 87
7.5.3 Robustness Evaluation 90
7.6 Conclusion 90

8 Conclusion and Outlook **93**
8.1 Conclusion 93
8.2 Outlook 95

APPENDICES **96**

A Mathematical Explanations **97**
A.1 The Curvature Vector 97
A.2 The Dynamics of a non-holonomic Vehicle in a Frenét Frame 99

Bibliography **101**

List of Figures

1.1 Schematic representation of the data transmission for teleoperated driving of autonomous vechiles (AVs). 2

1.2 Adaption of the diagram of automation level for teleoperation according to [120]: (a) direct control, (b) shared control, (c) supervisory control. 3

1.3 SAE standard J3016: Taxonomy and definitions for terms related to driving automation systems for road-vehicles adapted according to [95] and [67]. . . 5

1.4 Past and potential future evolution of driver assistance systems, adapted from [9]. 6

1.5 Overview of a typical software architecture for self-driving cars, adapted from [11]. 7

1.6 Optimal state trajectory $\mathbf{x}^*(t)$ of an optimal control problem taking into account the inequality constraint $h \leq 0$ for x_2, the end condition $\mathbf{g} = \mathbf{0}$ as well as the fixed end time t_f, according to [131]. 12

1.7 Basic principle of model-predictive control according to [131]; thick, blue: evolution of the optimized state vector and control input at time t_j; thin, blue: optimized evolution in the past; thick, gray: actual evolution of the closed control loop. 13

1.8 General problem of the environment-related non-convex motion planning with multiple options (a,b), adapted according to [58]. 14

1.9 Measured time delay during teleoperated driving using a LTE mobile network, including time for the transmission of the camera images from the vehicle to the operator's working station as well as the time for the transmission of the operator's control commands [32]. 17

2.1 Vehicle body approximation using a set of three circles. 25

2.2 Illustration of the workspace cost function $c(\mathbf{x})$. 25

2.3 Simulation results regarding the curvature constraints. 27

2.4 Illustration of the reference path cost calculation. 28

2.5 Simulation results regarding the reference path objective. 29

2.6 Local minima avoidance using Constrained CHOMP with: (a) original formulation, (b) domain step only. 31

3.1 Relative kinematics of the vehicle motion in a Frenét frame, according to [131]. 34

3.2 Comparison of the total costs with different additional cost terms $l(\epsilon)$, adapted according to [131]. 38

3.3 Multi-stage increase in total costs with two independent slack variables ϵ_1 and ϵ_2, adapted according to [131]. 38

3.4 Constrained longitudinal position $s(k)$ of the ego vehicle (dark blue) at time step k, adapted according to [118] 41
3.5 Representation of the system outputs for lateral collision-avoidance. 43
3.6 Lateral system constraints of the ego vehicle (blue) at time t_j and after the time interval Δt along the reference path Γ. Another road vehicle and its prediction are colored in gray and approximated by a rectangle, adapted according to [118]. 45

4.1 Image of the operator workplace: A human operator controls the vehicle using a gamepad. 49
4.2 Reaction to a passing road participant at two consecutive times, where the graphs indicate: the optimal trajectory (in green), acceleration constraints (in red), and measurements (in black). 53
4.3 Vehicle following with sudden braking maneuvers at two consecutive times, where the graphs indicate: the optimal trajectory (in green), comfort and safety constraints (in orange and red), and measurements (in black). 54

5.1 Illustration of the corridor-based control approach: (a) specified corridor with a collision-afflicted initial path (red) as well as the collision-free optimized path (green), (b) calculation and automated execution of the collision-free motion. 56
5.2 Simulation results: (a) road scenario with four obstacles, (b) - (d) progress of the vehicle along the optimized path (green dotted line) within the specified corridor (red dotted line) with its centre line (in red) and obstacle bounds (in yellow). 58
5.3 Evolution of objective functions over time belonging to the scenario in Fig. 5.2 59
5.4 Real driving results: (a) optimal path (green dotted) located within the specified corridor (red dotted) with obstacle bounds (yellow), (b) Path adaption based on new sensor information. 60
5.5 Camera images belonging to the real driving experiment in Fig. 5.4: (a) View of the side street into which the vehicle should turn, (b) sudden appearance of a previously hidden obstacle. 60
5.6 Obstacle passing dealing with oncoming traffic at two consecutive times, where the graphs indicate: the optimal trajectory (in green), comfort and safety constraints (in orange and red), and measurements (in black). 62
5.7 Keep a comfortable distance to pedestrian on road at two consecutive times, where the graphs indicate: the optimal trajectory (in green), comfort and safety constraints (in orange and red), and the reference curve (black dotted). 63

6.1 Generated paths that the operator can choose from based on the operator's perception. 65
6.2 Sampling range of the modified RRT. 67
6.3 Tree expansion: (a) a single step with one control input, (b) one expansion step with added nodes. 68
6.4 Results of clustered RRT path search: (a) all found paths, (b) clustered results, (c) path suggestions for operator. 70
6.5 Real driving results at two consecutive time stamps. 71

6.6 Image sections of the camera view belonging to Fig. 6.5. 71
6.7 Time evolution of the objective functions corresponding to the scenario in Fig. 6.5. 72

7.1 Data sample: Binary Occupancy grid map in the background, Initial path (in green), optimized path avoiding obstacles (in yellow), ego vehicle position (red star). 77
7.2 Illustration of the supervised learning problem statement for the path optimization task: The neural network takes the occupancy grid map and the initial path as inputs and predicts an optimized path. 78
7.3 Hybrid artificial neural network: Combining CNN and MLP. 79
7.4 Hybrid artificial neural network: Combining CNN and LSTM network. . . . 80
7.5 Illustration of feature addition as extra channels, proposed in [88]. h,w=1536,1536 and c=1, post feature concatenation, number of channels are $c + 2 = 3$ 80
7.6 Illustration of the redefined supervised learning problem statement for the path optimization task: The neural network takes a occupancy feature matrix as input and outputs the predicted deviation between the initial path and optimized path. 82
7.7 Illustration of the occupancy feature matrix extraction. 83
7.8 Illustration of the distributed dense (DD) layer. 84
7.9 Neural network with DD layers. 84
7.10 Neural network with LSTM layers. 85
7.11 Neural network with LSTM and DD layers. 85
7.12 Training loss and validation loss of network architectures discussed in Section 7.3.4. 85
7.13 Hyperparameter tuning for the LSTM-based neural network using Bayesian optimization. 86
7.14 Path prediction results using two scenarios indicating: the initial path in green (operator specified), the optimal path in yellow (generated by Constrained CHOMP), the predicted path in orange (generated by neural network) and the ego vehicle position as a red star. 89
7.15 Robustness evaluation of the LSTM neural network using iterative predictions in two scenarios. 91

A.1 The osculating circle for the point $\mathbf{q}$ on the path $\xi(s)$. The tangent of the path at the given point is represented by the vector $\mathbf{t}$. 97
A.2 Relative kinematics of the vehicle motion in a Frenét frame. 99

List of Tables

7.1 Model complexity in terms of parameters 84
7.2 Hyperparameters and their ranges for the LSTM-based neural network. . . . 86
7.3 Best hyperparameter set for the LSTM-based neural network. 87
7.4 Average Displacement Error of various neural network architectures. 88

Abbreviations

ABS	Anti-lock Braking System
ADAS	Advanced Driver Assistance System
ADA*	Anytime Dynamic A*
ADE	Average Displacement Error
AEB	Autonomous Emergency Braking
APF	Artificial Potential Field
AV	Autonomous Vehicle
ARA*	Anytime Repairing A*
BIT*	Batch Informed Tree
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
CNN	Convolutional Neural Network
DD	Distributed Dense
DRL	Deep Reinforcement Learning
ESP	Electronic Stability Program
FCNN	Fully Connected Neural Network
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
LTE	Long-Term Evolution
LSTM	Long short-term memory
MAE	Mean Absolute Error
MLP	multilayer perceptron
MPC	Model-Predictive Control
MSE	Mean Square Error
NLP	nonlinear programming
PRM	Probabilistic Roadmap
QP	quadratic program
RNN	Recurrent Neural Network
RRT	Rapidly-exploring Random Tree
SQP	sequential quadratic programming
w.r.t.	with respect to

List of Symbols

An overview on used notations in this work shall be given in the following. In general, small letters express scalars (e.g. $p \in \mathbb{R}$) or, if written in boldface, vectors (e.g. $\mathbf{x}$). Boldfaced, capital letters (e.g. $\mathbf{A}$) are used for matrices. Furthermore, the list is split into two parts according to the categories of developed motion planning algorithms: path planning and trajectory planning. Note that supplementary, intermediate, and auxiliary variables are not explicitly listed, but specified at the place of their first usage.

Path Planning

$\mathbf{A}$	Smoothness matrix
$\mathbf{B}$	Binding set vector
$\mathbf{C}$	Matrix of linear inequality constraints expression
c	Workspace cost function
$\mathbf{d}$	Vector of linear inequality constraints expression
d	Distance between paths
$\tilde{d}$	Nearest distance from vehicle's body to obstacle
$\mathbf{e}$	Boundary condition vector of smoothness objective
$\mathbf{F}$	Free set vector
$\mathbf{F}_{\text{occ}}$	Occupancy feature matrix
G	Occupancy grid map
$\mathbf{J}$	Jacobian matrix
$\mathbf{K}$	Finite difference matrix of smoothness objective
f	Objective function
$\mathbf{q}$	Waypoint
r	radius
$\mathbf{S}$	Submatrix of inverse Hessian along the direction of free variables
s	Integration variable of path integral
U	Objective functional
$\boldsymbol{\lambda}$	Lagrange multipliers
u	Body element of vehicle body approximation
$\mathbf{x}$	Workspace point of a body element
x	Coordinate in two-dimensional space
y	Coordinate in two-dimensional space
ε	Threshold
η	Regularization coefficient
$\boldsymbol{\kappa}$	Curvature vector
κ	Curvature

w	weighing factor
$\bar{\nabla}$	Functional gradient operator
θ	Path orientation
$\boldsymbol{\Upsilon}$	Domain step matrix
ξ	Path, optimization parameter

Trajectory Planning

$\mathbf{A}$	System matrix of a state space representation
$\mathcal{A}$	Summarized system matrix of a state space representation
a	Vehicle acceleration
$\mathfrak{B}$	Summarized input matrix of a state space representation
$\mathbf{b}$	Input vector of a state space representation
$\mathbf{C}$	Output matrix of a state space representation
$\mathfrak{C}$	Summarized output matrix of a state space representation
$\mathfrak{D}$	Summarized disturbance matrix of a state space representation
$\mathbf{d}$	Disturbance vector in the system equation
d	Coordinate in Frenét frame
$\mathfrak{E}$	Summarized output matrix of a state space representation
$\mathbf{e}$	Disturbance vector in the output equation
$\mathbf{I}$	Identity matrix
$\mathbf{K}$	Weighing matrix of linear-quadratic cost function
$\mathbf{k}$	Weighing vector of linear-quadratic cost function
l	Cost function
l_{bumper}	Length from vehicle rear axle to bumper
$\mathbf{Q}$	Weighing matrix of quadratic objective function
$\mathfrak{Q}$	Summarized weighing matrix of quadratic objective function
$\mathfrak{R}$	Summarized weighing matrix of quadratic objective function
r	Weighing factor of quadratic objective function
s	Coordinate in Frenét frame
T_s	Step size
t	Time
t_{h}	Constant time headway
$\mathbf{u}$, u	Input of a state space representation
v	Vehicle velocity
w	Weighing factors
$\mathbf{X}$	Sequence of vectors for system state $\mathbf{x}$
$\mathbf{x}$	System state of a state space representation
$\mathbf{Y}$	Sequence of vectors for output $\mathbf{y}$
$\mathbf{y}$	Output of a state space representation
$\mathbf{z}$	Vector sequence of disturbance term z
z	Disturbance of a state space representation
Γ	Reference path
δ	Steering angle

$\boldsymbol{\epsilon}$	Vector of slack variables
ϵ	Slack variable
κ	Curvature
τ	Integration variable
θ	Orientation (angular)

1 Introduction

The development and improvement of the vehicle's intelligence has undergone a rapid process in recent years. While previous research focused on promoting the driver's safety and comfort, current efforts target at completely removing the driver from the control loop of a vehicle. Self-driving systems aim to improve road safety and offer people new mobility services. In addition to passenger transport, autonomous vehicles (AVs) might be used to carry goods. Many new technology companies as well as traditional automobile manufacturers are making immense technological efforts to advance automated driving.

1.1 Motivation

Although extensive research has been carried out towards autonomous driving, this technology remains highly challenging. The greatest challenge is self-driving operation in an urban environment. Autonomous transportation systems with the current level of machine intelligence cannot fully process complex, dynamic environments and make appropriate decisions. Reasons may have different origins, such as traffic regulations that do not correspond with the actual traffic situation, for instance when a human regulation is required. Often construction sites can be too complex for the AVs. This may occur, if they do not meet the standard or are not completely installed, i.e. when sign posts are missing or lane markings are only partially existent. Road constructions may contain narrow spots that require special attention to prevent collisions with other road participants. Other vehicles, such as buses or trucks, may require more space to maneuver at intersections. In such situations, the AVs have to react and reset their position to allow other vehicles to turn. In the case of crowds at public events or high traffic conditions at entrances and exits of main roads, special consideration must be given, but assertive threading may also be required.

To overcome these functional limitations of autonomous transportation systems with the current level of machine intelligence, teleoperated driving offers a high potential and represents a promising fallback option [37]. By transfering data via a mobile network connection with a high bandwidth, the autonomous vehicle (AV) is remotely controlled by a human operator from a stationary working station [136]. The human operator receives the vehicle's environment data provided by sensors, interprets them and transmits strategic and operational commands to the vehicle. Fig. 1.1 shows a schematic representation of this concept. The operator's working station is equipped with control elements, such as the conventional steering wheel and pedal combination or a computer mouse and keybord. Furthermore, the images and other sensor data received from the vehicle are displayed to the operator using several monitors. The key advantage of teleoperation is that the assisting human is able to remotely take over control for some timespan but stays locally independent. In situations in which AVs are not able to process its environment adequately or to make a robust decision, the human

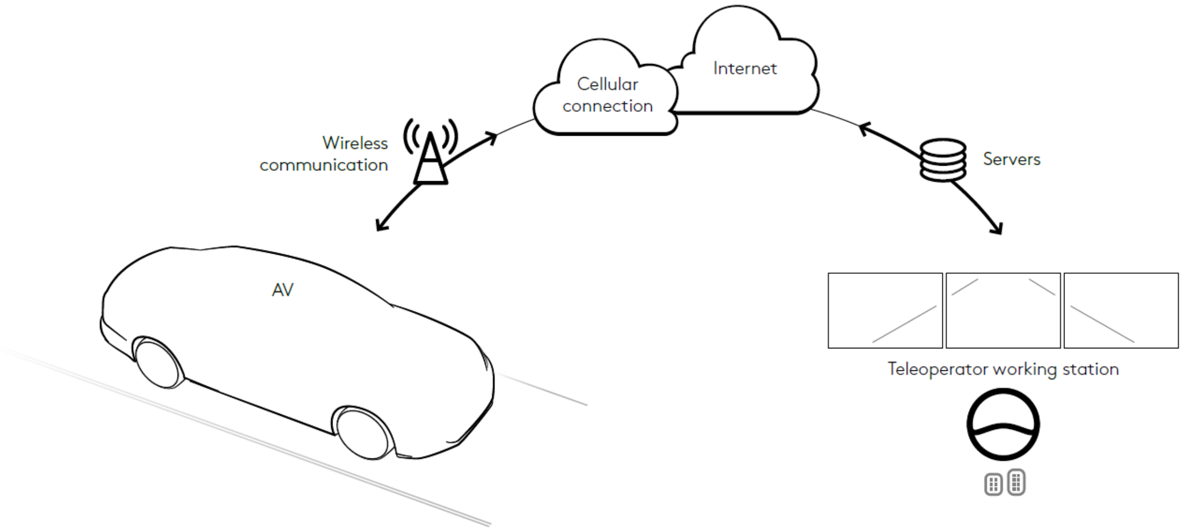


Figure 1.1: Schematic representation of the data transmission for teleoperated driving of autonomous vechiles (AVs).

operator can provide robust driving behavior using its decision-making and environmental awareness abilities.

1.2 State-of-the-Art

1.2.1 Teleoperated Driving

The unavoidable communication link in teleoperation always introduces certain time delays [119]. As pointed out in [104], round trip times of more than 200 ms already decrease human task performance. In particular, a variable time delay has a significant impact. To overcome communication time delays in teleoperation, the authors in [120] distinguish between two main categories: direct and indirect control. Indirect control approaches are additionally divided into shared controls and supervisory controls depending on the automation level, see Fig. 1.2.

Direct Control

The most widespread teleoperated driving paradigm is direct control. By using control inputs such as the conventional steering wheel and pedal combination, the human operator is responsible for observing and perceiving the vehicle environment at all times and deciding appropriate driving commands. The human operator may be supported by augmented reality approaches that provide visual feedback about a time delay or communication failure [68].

The free corridor, for instance, is a safety concept for teleoperated driving using direct control. The approach, proposed in [21], visualizes the human operator a path of full braking of the vehicle, based on parameters such as the current velocity, traction conditions and steering position. In the event of a communication loss to the operator, the vehicle would follow the predetermined trajectory and stop. Therefore, the human operator is always responsible for keeping this path free from obstacles.

Another augmented reality approach for teleoperated driving is the predictive display as proposed in [24] or most recently in [49]. This approach predicts the motions of the teleoperated vehicle and other road participants to forecast not only possible critical situations but also to overcome the communication time delay. The predictions are then visualized to the human operator as rectangles within the camera images.

Indirect Control

As pointed out in [74], direct control – with the human in the closed control loop – may be impossible in certain teleoperation situations due to variable time delays and limited bandwith in mobile network connections. For this reason, indirect controls have been developed. In contrast to direct control, indirect control approaches include automated driving functions such as path control or obstacle avoidance. The guidance loop is closed autonomously by the vehicle. Since the human operator is not directly part of the closed control, in indirect control methods the closed control loop is insensitive w.r.t. any time delays.

Shared Control

In shared control, the human operator is involved in planning tasks: The operator specifies

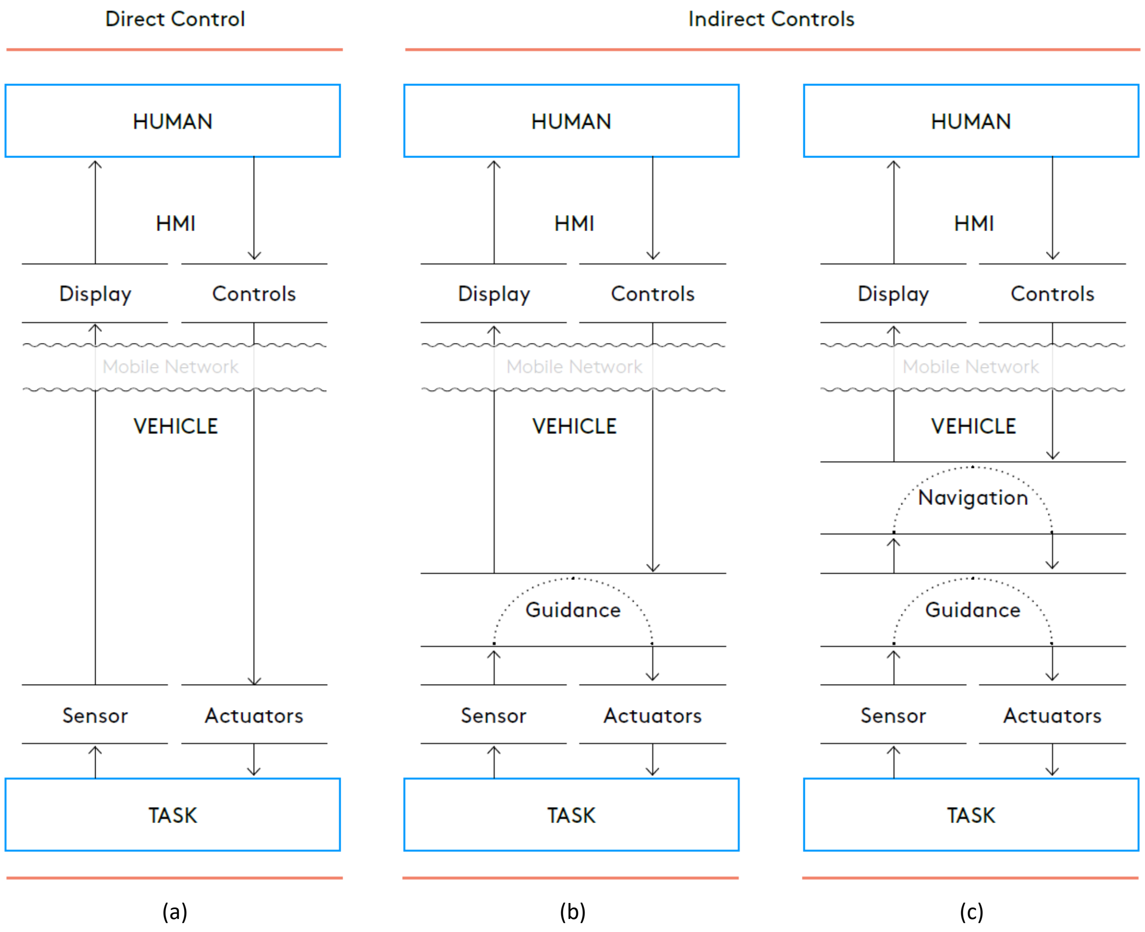


Figure 1.2: Adaption of the diagram of automation level for teleoperation according to [120]: (a) direct control, (b) shared control, (c) supervisory control.

high-level objectives over the delayed mobile network that the automated vehicle executes on its own.

As shown in [74], such goals might be waypoints. Here, the proposed system is based on the transmission of individual images from a camera attached to the front of the vehicle. The human operator defines a sequence of waypoints in the respective camera image and transmits these to the vehicle. Using a cubic spline function, a continuous path is interpolated based on the individually specified waypoints and is then forwarded to the vehicle for tracking purposes. While the vehicle automatically drives to the end of the specified path, a new camera image is already transmitted to the human operator. NASA's Mars rovers are also conducted by a waypoint-based approach [27]. Their operators face very large time delays that result from the distance to Mars. Moreover, the limited communication highly depends on favourable Earth-Mars constellations, hence, the operators must send entire sequences of motions for a Mars day in advance to the rover. By using the on-board autonomy, the rover then accomplishes the path independently.

Alternatively, in trajectory-based control [45], the human operator defines a sequence of trajectories in a video stream while the car is driving. Both, the length and the curvature of the trajectories are edited by the operator using the conventional steering wheel and pedal combination. The concept has proven to be advantageous for the operator compared to direct control with straight trajectories even with latencies of up to 600ms. The manual generation of suitable trajectories in real-time for curved paths or in turning scenarios, however, turned out to be too challenging for the human operator.

Supervisory Control

Supervisory control entails a high level of automation. The control strategy may be delivered before execution or within the progress. The automated system is responsible for the vehicle's guidance and navigation tasks. The human operator supervises the process and may support it in decision making [37].

A higher automated concept for teleoperated driving has been proposed in [68]. The human operator receives path suggestions generated by the automated vehicle. The vehicle then follows the path chosen by the operator using automated driving functions. The proposed approach, however, is only suitable for well structured and static environments where all necessary obstacles are known. This is not the case in almost every urban scenario [118].

1.2.2 Automated Driving

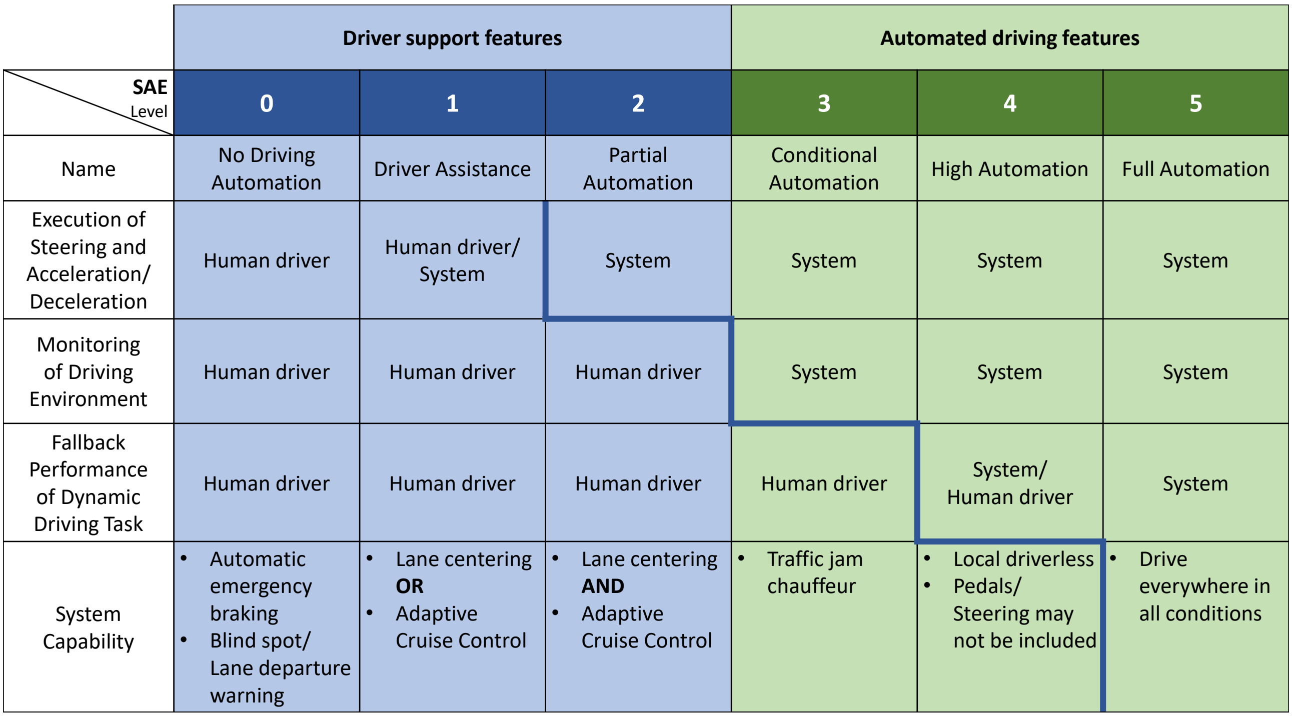
As a classification of driving automation for on-road vehicles, the Society of Automotive Engineers (SAE) published a standardized categorization – J3016 – in 2014. This standard classifies the amount of driver interventions and required attentiveness. As illustrated in Fig. 1.3, driving automation can be differentiated into six levels, which span a spectrum from fully manual driving (SAE Level 0) to full automation (SAE Level 5).

Most vehicles are currently classified in SAE Level 0. These are controlled manually by humans. Although there are systems that help the driver, humans take on the dynamic driving task. One example is the Autonomous Emergency Braking (AEB) system. Since it technically does not drive the vehicle, it is not considered automation in this context. SAE Level 1 represents the lowest level of driving automation. The driver is supported by a single Advanced Driver Assistance System (ADAS, see Section 1.2.2) such as the steering for lane centering or the adaptive cruise control (ACC), but retains complete responsibility for all driving tasks. At the level of partial driving automation (SAE Level 2), the vehicle is able to control both lateral and longitudinal dynamics. For this purpose, several assistance systems are often combined with one another. However, the driver must remain alert and be able to take control of the vehicle at any time.

With the conditional automation level (SAE Level 3), the vehicle enters the world of highly automated driving. Drivers may temporarily detach their attention from the driving task and take their hands off the steering wheel accordingly. However, the human driver has to take control if requested by the system. At the level of high driving automation (SAE Level 4), the vehicle takes on all aspects of the dynamic driving task. In the event of a system failure, the vehicle is able intervene even if the driver does not respond appropriately to a control request. In most cases, the system does not require human interaction. The level of full driving automation (SAE Level 5) can be viewed as a long-term goal for automated driving. Systems at this level do not require human interaction and are able to drive everywhere in any conditions. Vehicles will have neither a steering wheel nor pedals. Human users simply indicate where they would like to be picked up by the vehicle and the destination to which the vehicle should drive them.

Advanced Driver Assistance Systems

Nowadays, Advanced Driver Assistance Systems are indispensable in road vehicles. They aim to reduce the consequences of accidents, prevent traffic accidents and enable autonomous



	Driver support features			Automated driving features		
SAE Level	**0**	**1**	**2**	**3**	**4**	**5**
Name	No Driving Automation	Driver Assistance	Partial Automation	Conditional Automation	High Automation	Full Automation
Execution of Steering and Acceleration/ Deceleration	Human driver	Human driver/ System	System	System	System	System
Monitoring of Driving Environment	Human driver	Human driver	Human driver	System	System	System
Fallback Performance of Dynamic Driving Task	Human driver	Human driver	Human driver	Human driver	System/ Human driver	System
System Capability	• Automatic emergency braking • Blind spot/ Lane departure warning	• Lane centering **OR** • Adaptive Cruise Control	• Lane centering **AND** • Adaptive Cruise Control	• Traffic jam chauffeur	• Local driverless • Pedals/ Steering may not be included	• Drive everywhere in all conditions

Figure 1.3: SAE standard J3016: Taxonomy and definitions for terms related to driving automation systems for road-vehicles adapted according to [95] and [67].

driving in the future. The development of driver assistance systems initially began with vehicle stabilization systems. First the Anti-lock Braking System (ABS) and later the Electronic Stability Program (ESP) were introduced in serial vehicles [140].

The first ADAS to use environmental measurements in addition to the vehicle's driving dynamics is Adaptive Cruise Control (ACC). The ACC system detects vehicles in front and adapts the vehicle's velocity to that of the vehicle in front while maintaining a desired distance [137]. Current ACC systems can brake to a standstill and start again automatically [127]. This is especially intended as a comfort system for traffic jams on highways. However, the functionality in urban environments is limited due to narrow lane widths and tight curves. With the introduction of the ACC system, the development of frontal collision protection systems has been encouraged. Effective collision detection, even at close range and in urban traffic, has been made possible in particular by the additional electronics and sensors that have been integrated into series vehicles through ACC. In this way, the driver can be warned of a potential accident or, if an unavoidable collision is detected by the system, an Autonomous Emergency Braking can be initiated [35].

In the area of lateral vehicle control, the Lane Keeping Assistant (LKA) supports the driver in keeping in lane and automatically steers the vehicle back into the lane if the system detects an unintentional departure. Due to a limited steering torque, the driver can overrule the system at any time [102]. For legal reasons, however, the actual steering is the responsibility of the driver. The systems are therefore equipped with a hands-off detection. If hands-free driving is detected, the system deactivates itself. The areas of application of today's systems are limited to highways with several lanes and clearly visible lane markings.

An overview of the evolutionary progress from ADAS to automated driving is shown schematically in Fig. 1.4. The next generation of ADAS in series vehicles will be integrated longitudinal and lateral control systems. These will be SAE Level 2 systems for partial driving

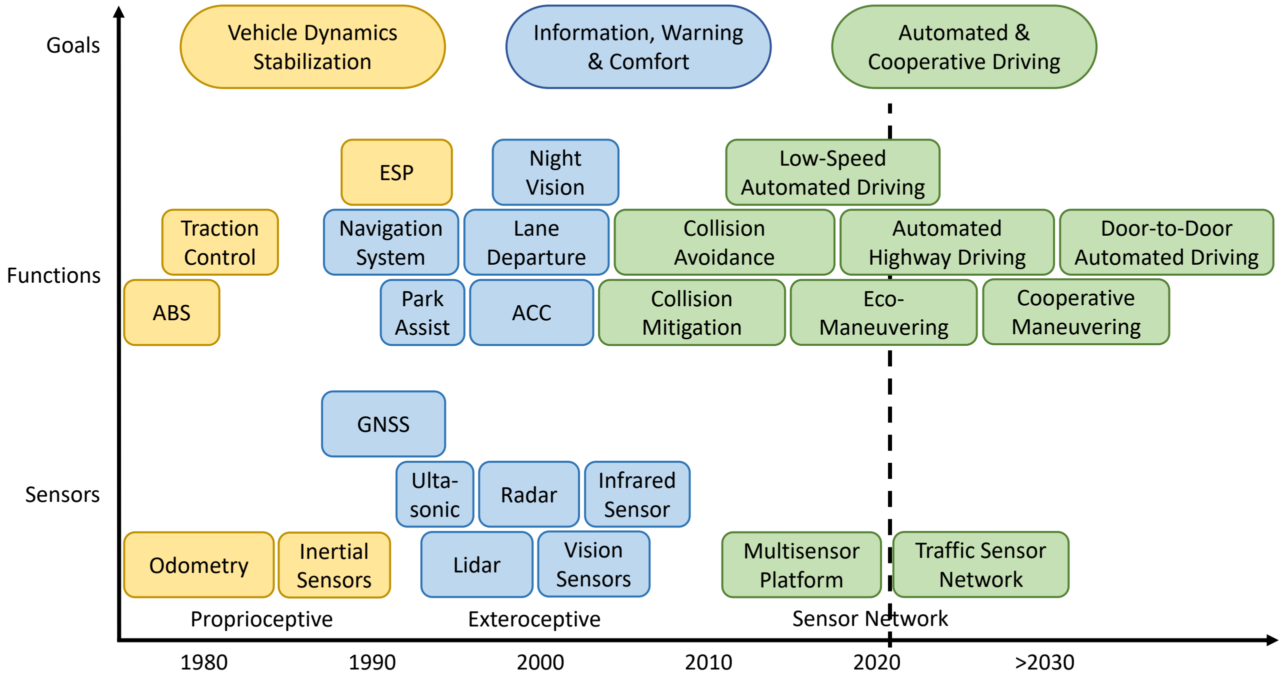


Figure 1.4: Past and potential future evolution of driver assistance systems, adapted from [9].

automation, mainly for highways. Many automobile manufacturers and software development companies are testing their SAE Level 2 systems intensively. Some companies are already offering their first systems with limited functions. Common AI, for instance, provides its Openpilot – an open source software to improve the existing driver assistance in most new road vehicles [26], [112]. The Openpilot software runs on a mobile phone based on the phone's camera data. By connecting the mobile phone to the car, the software enables the car to automatically steer, accelerate and brake within its lane [16]. The driver only monitors the function, but has to remain attentive and be ready to take over the full control of the vehicle at any time.

Software Architecture of Autonomous Driving

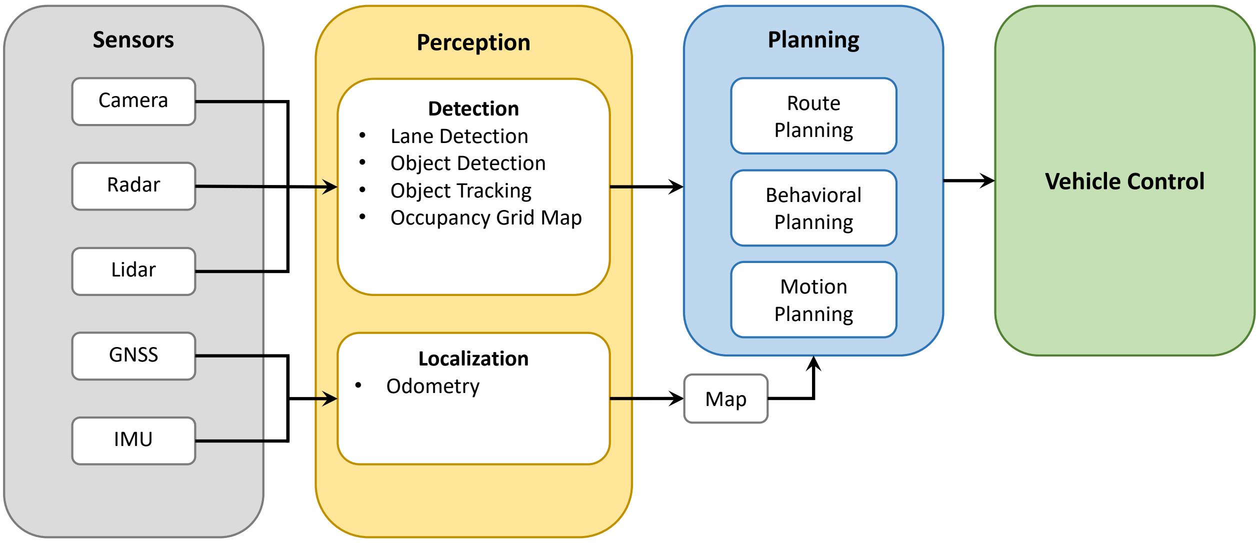


Figure 1.5: Overview of a typical software architecture for self-driving cars, adapted from [11].

The software architecture of autonomous driving can be broadly divided into three categories: perception, planning and control [99]. A detailed overview is depicted in Fig. 1.5.

Perception

In the perception functions, different sensor data have to be processed first. The sensor data are used to extract relevant knowledge about the vehicle environment. In the first step, the vehicle is localized on a digital map using Global navigation satellite system (GNSS) and Inertial Measurement Unit (IMU) sensor data. An environmental perception takes place in parallel. The perception of the environment is a fundamental function in order to provide the system with crucial information about the driving environment [4]. A lane detection function ([64], [39]), an algorithm for object detection and prediction ([110], [100], [138]) and free space detection ([78], [130]) lead to holistic information about the current environment in which the AV is currently driving. The free space function may be used to generate the

Occupancy Grid Map (OGM). The OGM is one of the most common representations of the environment of an AV. First proposed in [92], the OGM informs the system about areas of the vehicle environment, in which the AV cannot drive collision-free. The occupancy grid map describes the vehicle environment as a multidimensional spatial lattice [105]. Each cell of the grid represents a portion of the environment. The cells contain information about whether the space they are representing is currently occupied. However, the OGM only provides a sparse representation of the vehicle environment, since only cells reached by the sensors are updated [75]. The OGM forms the basis for any online motion planning and control operation [99].

Planning

Although the partitioning of the planning functions are somewhat blurred with variations of the scheme that occur in the literature or in successfully implemented systems, such as the first [125], second [91] and third placed [3] in the DARPA Urban Challenge, the planning subsystem can in most cases be roughly divided into three main functions: route planning, behavioral planning and motion planning.

The route planning function calculates a route through the road network from the current position of the AV to the user-defined destination. The route search task is formulated by assigning edge weights according to the cost of traversing a road segment and applying graph search algorithms [99]. Classical graph search algorithms, like Dijkstra's algorithm [29] or the A^* algorithm [62] are effective for small networks. However, there are more advanced approaches that improve the efficiency over large networks. A detailed overview of these approaches with a focus on route planning is given in [6].

The behavioral planning function is responsible for time-critical decisions, e.g. overtaking, lane change, safety braking, etc. [11]. The decisions are made on board of the AV in the form of Finite State Machines (FSM) to specify actions in response to specific perceived driving contexts [91], [12], [5]. The scope of use of FSM is naturally limited as they are designed manually for a number of specific situations. The AV may react unexpectedly in situations that were not explicitly taken into account and, for instance, finds itself in a livelock in which processes constantly repeat the same interaction in response to changes in other processes without doing useful work [87]. Approaches to improve the organization in large decision-making structures in order to manage large sets of rules that occur in urban driving scenarios, have been proposed in [15] and [40]. In [18] and [19] decision making approaches for road rules enforcement using Linear-Temporal Logic variants with successful real-world overtaking experiment were proposed, where the dynamical system systematically picks which safety rules to violate and minimizes the level of unsafety.

The motion planning function generates a motion from the current state of the AV to the next local target state, which is defined by the behavioral planning function. Furthermore, the motion planning function satisfies kinematic and dynamic constraints of the AV, avoids collisions with static and dynamic obstacles and provides comfort to passengers. The motion planning task for AVs has proven to be computationally complex [99]. The task may require an exhaustive search of all possible paths. Motion planning algorithms are usually compared and rated based on their computational efficiency and completeness [47]. The computational efficiency indicates the process run time, while completeness relates to whether the algorithm terminates in a finite time, returns a solution if one exists, or otherwise indicates that no solution exists [79]. Various approaches for motion planning have been proposed in the literature. A detailed review to path and trajectory planning algorithms is given separately in Section 1.2.3.

Vehicle Control

The actions defined in the planning subsystem are carried out in the vehicle control subsystem. The purpose of this task – also known as motion control – is to provide the necessary inputs to the actuators that will generate the desired motions. While planning algorithms often deal with positions and velocities of the AV in relation to its environment, motion controllers map the interaction in form of energy and forces. The system performance is assessed with measurements inside the control system. This enables the motion controller to react to disturbances and thus alter the system dynamics into the desired state [99]. In order to achieve a satisfactory execution of the planned motion, model-based control approaches that describe the desired system dynamics in detail are advantageous. For more comprehensive reviews on motion control techniques, readers are referred to [97], [4] and [99].

1.2.3 Motion Planning

The topic motion planning encompasses a very broad field of research. The areas of applications range from manipulating arms, for instance in manufacturing or medical, to mobile robots and transportation systems. The aim of the motion planning function in the context of automated driving is to generate a motion from the current state of the AV to the next local target state, which is defined by a behavioral planning function. The main task is to avoid collisions with static and dynamic obstacles and simultaneously meeting kinematic and dynamic constraints on the motion of the vehicle. Furthermore, the function should provide comfort for the passengers in the dynamic driving task. The motion plan can be defined by a path or a trajectory. A path is a sequence of coordinates in the plane, whereas the trajectory is a path that additionally specifies the evolution of the vehicle's states – velocity, acceleration and jerk for instance – through time.

Path Planning

For the planning of suitable paths in automated driving a great amount of navigation techniques have been modified from mobile robotics to face the challenges of road networks and driving rules. The utility and performance of these approaches are typically evaluated according to the class of problems to which they apply and by their guarantees of convergence to an optimal solution [97]. The algorithms for path planning can be broadly divided into three categories:

Graph-search methods:

Graph-search based planners search for feasible paths based on a discretized representation of the environment – the graph. The graph is often represented as an occupancy grid, that depicts where objects are located in the environment.

Probably the most widely known algorithm that finds the shortest path in the graph is Dijkstra's algorithm [29]. The algorithm builds up a tree representing the shortest paths from a given source node to all other nodes in the graph by performing the best first search. If only a path to a single target node in the graph is intended, the search can be guided by a heuristic in order to speed up the process. The heuristic function uses the information on the target node in order to estimate the costs, which are necessary to extend a path to the target node. The most recognized heuristic search algorithm is A* [62]. The algorithm has proven to return an optimal path according to the discretized resolution, if the implemented heuristic function is permissble and never overestimates the cost-to-go. A* is suitable for searching spaces that are known a priori, however, it is expensive in terms of memory and speed in vast areas. Each time the occupancy grid is updated with sensory data, the shortest path from the vehicle's current position is recalculated repeatedly. Since such updates often only affect a minor portion of the graph, it can be wasteful to run the entire search each time.

In order to efficiently recalculate the shortest path each time the graph is updated, the family of real-time replanning graph search algorithms such as Dynamic A* (D*) [121], Focussed D* [122] and D* Lite [76] has been designed. These algorithms reduce replanning time by taking advantage of information from previous search efforts.

Anytime search algorithms try to quickly generate and provide a first suboptimal path, which is then continuously improved with more computing time [97]. To find a first solution, the Anytime A* algorithm [61] uses a weighted heuristic. The Anytime behaviour is achieved by continuing the search with the cost of the first suboptimal solution as an upper bound. The Anytime Repairing A* (ARA*) approach [85] performs multiple searches with increasingly smaller weighted heuristic while reusing the information from previous iterations. The Anytime Dynamic A* (ADA*) algorithm [84] represents a dynamic modification of (ARA*) in terms of the D* algortihms. The combination of the methods behind D* Lite and ARA* creates an anytime search algorithm for real-time replanning in dynamic environments [97].

Sampling-based methods:

Sampling-based planners aim to solve timing constraints for the planning in high-dimensional or vast spaces. The most prominent approaches are Probabilistic Roadmap (PRM) [73] and Rapidly-exploring Random Tree (RRT) [80]. The latter has been extensively studied for automated driving. The RRT algorithm is an efficient method for finding paths online. The rapid exploration is achieved by random sampling in the environment and extending a tree of nodes towards the random sample. In addition, non-holonomic constraints can be taken into account in the algorithm. The authors in [72], however, demonstrated that the RRT converges to a non-optimal solution, since the existing tree structure biases future expansions. Furthermore, they introduced an asymptotically optimal modification of the RRT, called RRT* and proved the same computational efficiency in finding non-optimal solutions and probabilistic completeness as the original RRT. The asymptotically optimal convergence is achieved by an additional incremental rewiring function. New nodes are not only added to the tree structure, but they are also considered as replacement parents for existing nearby nodes in the tree, if that results in a lower cost path from the initial node. With uniform global sampling, the RRT* asymptotically finds the optimal solution to the planning problem by asymptotically finding the optimal paths from the initial node to each node in the problem area. This contradicts its single-query nature and becomes expensive in large spaces. In order to improve the convergence rate and final solution quality while maintaining the same probabilistic guarantees on completeness and optimality as RRT*, the authors in [42] present the Informed RRT*. This algorithm uses the information of a target node to perform a focused search by modifying the sampling function. Until a first solution has been found, the Informed RRT* behaves the same as RRT*. Thereafter, Informed RRT* only samples within an admissible heuristic, an ellipsoidal subset of the planning domain which includes the initial node and target node as well as the first non-optimal solution. With an increasingly better solution, the ellipsoidal subset further reduces the search space.

A novel planning method which balances the advantages of graph-search-based planners and sampling-based planners is Batch Informed Tree (BIT*) [41]. BIT* uses batches of samples to perform an ordered search around the minimum solution proposed by a heuristic, as in A*. However, the search is carried out on a continuous planning domain. By processing several sample batches, which allow subsequent searches to be focused on the subproblem that might contain a better solution, the algorithm asymptotically converges towards the global optimum, as in Informed RRT*.

Numerical optimization:

Numerical optimization methods aim to minimize a cost function subject to constraint variables. In path planning, these methods are often used to optimize the solutions of graph search or sampling-based algorithms to remove jerky or redundant motions from the paths that such planners may generate. Nevertheless, the optimization of a complex cost function can require complicated and time-consuming calculations that may not be bearable for real-time applications.

A representative approach with practical applications in automated driving is the Artificial Potential Field (APF) [129]. This algorithm defines a virtual potential in the environment. The potential can be designed according to the desired path property. Possible potentials can be the attraction to a target point or the repulsion from obstacles.

A relatively novel numerical optimization for motion planning is Constrained CHOMP [22]. CHOMP stands for Covariant Hamiltonian Optimization for Motion Planning. The algorithm is well suited to generate a smooth and collision-free solution by iteratively improving an initial path. A major advantage of this optimization technique is that the initial path does not have to be collision-free [108]. The objective function, which represents a trade-off between smoothness and collision avoidance, is minimized by iteratively solving a sequential quadratic problem subject to linear inequality constraints. A detailed description on Constrained CHOMP and the necessary modification for the application in automated driving are given in Section 2.

Trajectory Planning

In general, the trajectory planning task is to determine a time course for the transition of system states, which takes system-related constraints into account and is as optimal as possible with regard to predefined criteria [90]. The fundamental requirement for the trajectory planning function is to ensure that no collision in future vehicle motions occurs. The motions of other road users have a particular influence on the trajectory planning of the automated vehicle. The available driving space is not only constrained according to the current obstacle positions, but also by their future positions, which vary over time depending on their velocity.

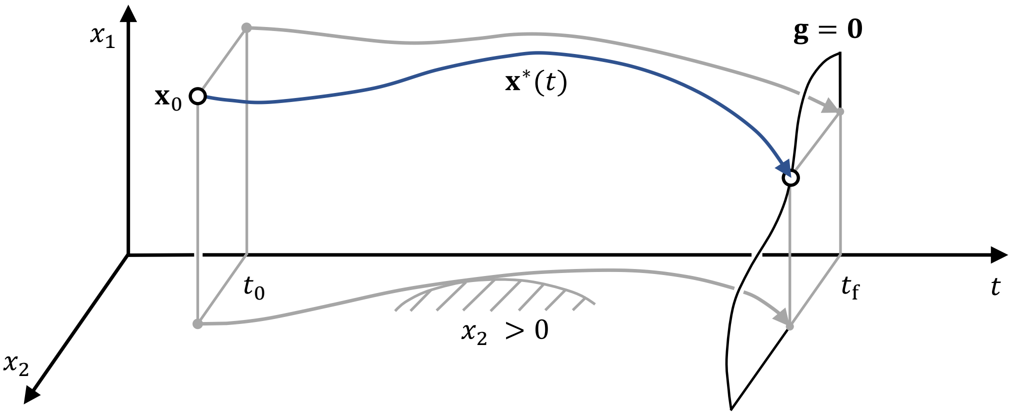


Figure 1.6: Optimal state trajectory $\mathbf{x}^*(t)$ of an optimal control problem taking into account the inequality constraint $h \leq 0$ for x_2, the end condition $\mathbf{g} = \mathbf{0}$ as well as the fixed end time t_f, according to [131].

For this reason, the explicit consideration of time in the trajectory planning is indispensable for dynamic obstacles. In order to timely adapt the vehicle motion to changing traffic situations and to take comfort aspects into account, the trajectory planning has to generate future motions with a sufficient planning horizon. However, the requirement for a planning horizon that is as large as possible contradicts a preferably short calculation time for the trajectory planning problem. This is crucial in order to react quickly to sudden changes in the vehicle environment through cyclical replanning of the vehicle's trajectory. To avoid undesired interactions between the motion control and the motion planning, the trajectory planning has to comply with physical driving limitations, even in critical situations.

Due to the above mentioned requirements in automated driving, the theory of optimization with differential equations has proven to be particularly advantageous and has thus established for trajectory planning [1], [55], [57], [86]. Since the planning of the trajectories is carried out by solving an optimization problem, the term trajectory optimization is used in this context.

Mathematical formulation of optimal control problems

In contrast to static optimization, in which the optimization variables $\mathbf{x}$ are elements of the Euclidean space $\mathbb{R}^n$, dynamic optimization is dedicated to the determination of functions $\mathbf{x}(t)$ of an independent variable t – time for instance – that minimizes a cost functional. Optimal control problems represent a special case of dynamic optimization, for which the function to be determined is the input trajectory of a dynamic system. The system dynamics are taken into account in the optimization in the form of differential equations. A structure of the optimal control problem often used in the literature [58], [82], [98], [131] is:

Minimize the cost functional

$$J(u) = \int_{t_0}^{t_\mathrm{f}} l(\mathbf{x}, u, t)\,\mathrm{d}t + V(\mathbf{x}_\mathrm{f}, t_\mathrm{f}) \tag{1.1a}$$

taking into account the system dynamics

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u, t)\,, \quad \mathbf{x}(t_0) = \mathbf{x}_0\,, \quad \mathbf{x}(t_\mathrm{f}) = \mathbf{x}_\mathrm{f}\,, \tag{1.1b}$$

as well as the end conditions and inequality constraints

$$\mathbf{g}(\mathbf{x}_\mathrm{f}, t_\mathrm{f}) = \mathbf{0}, \tag{1.1c}$$

$$\mathbf{h}(\mathbf{x}, u, t) \leq \mathbf{0}, \tag{1.1d}$$

with respect to the control signal $u(t)$ for $t \in [t_0, t_\mathrm{f}]$.

The problem of optimal control consists in generating a control trajectory $u^*(t)$ on the interval $t \in [t_0, t_\mathrm{f}]$ for the dynamic system (1.1b) with state vector $\mathbf{x}(t)$ and input $u(t)$, so that the state trajectory $\mathbf{x}^*(t)$ reaches the target state $\mathbf{x}_\mathrm{f}$ starting from the initial state $\mathbf{x}_0$, the inequality constraints $\mathbf{h} \leq \mathbf{0}$ and end conditions $\mathbf{g} = \mathbf{0}$ are satisfied and at the same time the cost functional J is minimized, as schematically visualized in Fig. 1.6. The final costs are taken into account in the cost function using $V(\mathbf{x}_\mathrm{f}, t_\mathrm{f})$.

In the context of automated driving, the vehicles motion dynamics are described using the system model (1.1b). The target of the optimization is to determine a control trajectory in the form of future courses for steering, accelerator and brake. Undesired vehicle motions such as deviations from the lane center or hectic accelerations can be penalized using the cost function (1.1a). The inequality constraints (1.1d), as indicated in Fig. 1.6, can be used to constrain the driving space in such a way that obstacles are considered in the optimization process. Additionally, the end conditions (1.1c) offer the option of specifying a defined target state, such as a desired velocity at the end of a maneuver.

Model-predictive trajectory optimization

According to the previously described requirements, a permanent adaptation of the vehicle motion according to current sensor information is necessary in order to take into account the constantly changing driving environment and driving condition. This can be achieved by cyclically solving the optimal control problem at short intervals Δt on a progressive optimization horizon of length T in the sense of the basic principle of Model-Predictive Control

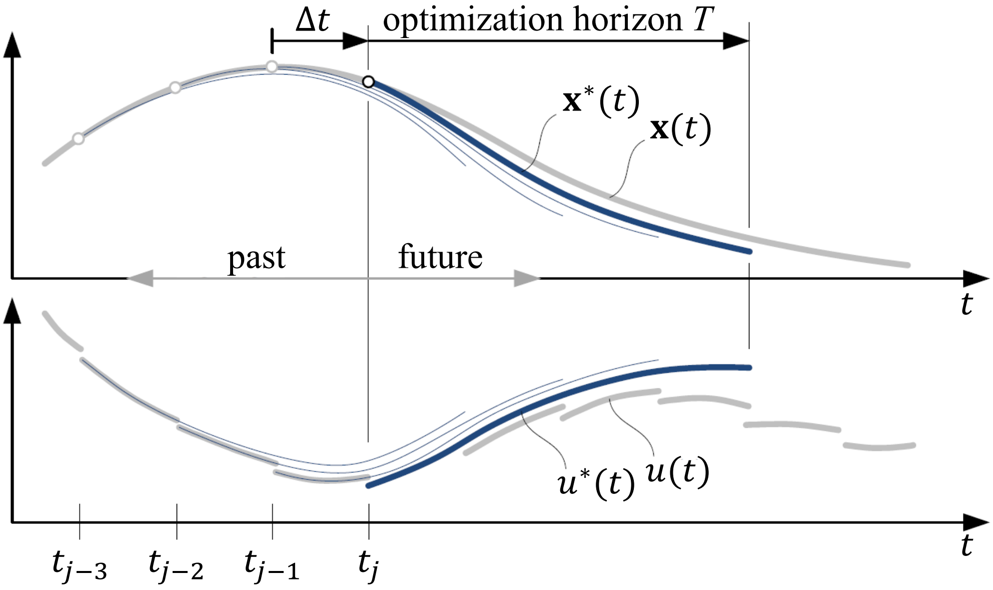


Figure 1.7: Basic principle of model-predictive control according to [131]; thick, blue: evolution of the optimized state vector and control input at time t_j; thin, blue: optimized evolution in the past; thick, gray: actual evolution of the closed control loop.

(MPC) [36], [54], [123]. As visualized in Fig. 1.7, in each step, the control trajectory $u^*(t)$, $t \in \left[t_j, t_j + T\right]$ optimized on the basis of the system state $\mathbf{x}(t_j)$ is applied exclusively on the interval $\left[t_j, t_j + \Delta t\right]$, since the result of the next optimization step is already available at the following point in time.

Due to the constantly changing initial conditions, the optimal control problem (1.1) to be solved on the optimization horizon T is therefore reformulated by the substitution of the time interval from $t \in [t_0, t_\mathrm{f}]$ to $t \in \left[t_j, t_j + T\right]$.

General problem in trajectory optimization

Although the formulation of optimal control problems and their cyclical solution on a finite optimization horizon offers a high potential to meet the requirements for all traffic situations, however, the methods known from optimization theory do not offer the possibility to solve non-linear and non-convex optimization problems in real-time without simplifications. The combination of the high-order non-linear system model resulting from driving dynamics and kinematics [107] as well as the non-convex structure of the optimization problem caused by the driving environment, see Fig. 1.8, prohibits the unrestricted use of existing optimization methods.

The general problem of trajectory optimization is therefore to formulate the non-linear and non-convex optimal control problem in such a way that existing optimization methods can be used effectively according to their properties. Taking into account the computing power required for the solution, simplifying assumptions must be made that convert the general problem into efficiently computable sub-problems without the solution of the general problem losing its validity. Since simplifying assumptions always lead to compromises, the previously mentioned requirements for automated driving offer the opportunity to compare the various optimization methods and to evaluate them objectively.

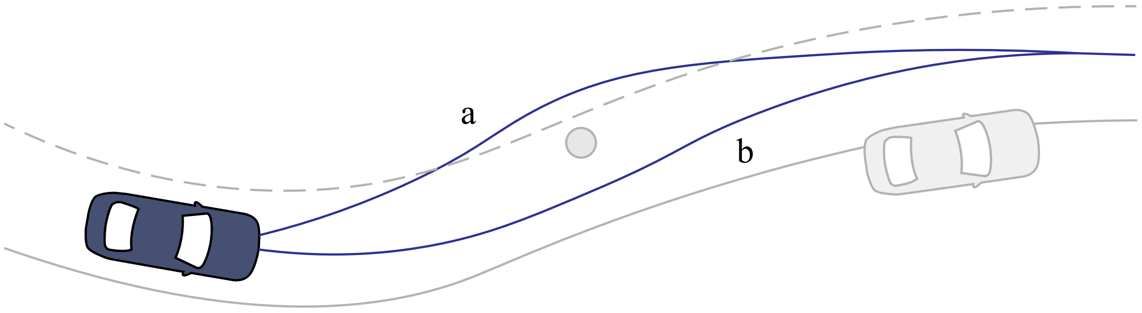


Figure 1.8: General problem of the environment-related non-convex motion planning with multiple options (a,b), adapted according to [58].

Optimization methods

For solving dynamic optimization problems, various methods are known from optimization theory. Typically, they are divided into three categories: dynamic programming, indirect and direct optimization methods [94].

Dynamic programming

This method is directly based on Bellman's Principle of Optimality [7]. Dynamic programming is characterized by a combinatorial solution finding and is hence suitable for global, albeit approximate, solution finding of non-convex optimization problems, including non-linear system models. The cost functional can be composed flexibly of several terms and a large number of system constraints can be defined. By adding many sub-problems, however, the complexity of the optimization problem increases significantly, which is why the use of dynamic programming is limited to systems of low order due to the curse of dimensionality [69]. Despite the limitation, dynamic programming offers advantages for vehicle motion planning due to its particular property for global solution finding with non-linear system dynamics [83]. As suggested in [56], for instance, the non-convex motion planning problem can be solved in a simplified manner for the vehicle kinematics as a partial dynamics of the vehicle model. The resulting trajectory, which is not necessarily drivable, serves as a reference in a two-stage approach for a subsequent optimization of higher system order. Therefore, the original non-convex problem formulation can be successfully solved by a combination of different optimization methods. As shown in [31], this strategy offers clear advantages, especially in semi-structured environments such as parking lots or construction sites. However, real-time planning of future vehicle motions with a continuously differentiable steering course is not possible due to the high system order required for this. Since the requirement for comfortable motion planning cannot be adequately met, this method is no longer taken into account in this work.

Indirect optimization methods

Approaches that are based on the evaluation of the necessary conditions for optimal control problems are called indirect optimization methods [94]. In the multivariable case, the optimality conditions result in a system of equations, the solution of which leads indirectly to the minima or maxima of the original problem [128]. Although the indirect optimization methods can only be used to solve local optimization problems, in special cases of problem formulation they are particularly attractive for trajectory optimization, as they allow insight into the structure of the optimal solution. As shown in [60], the solutions of the optimization problem can be determined extremely efficiently for simple motion models under certain conditions. Using these methods, however, difficulties arise while taking into account equality or inequality constraints, since they make an analytical solution of the optimization problem practically impossible even for simple motion models. This severely limits the use of indirect methods for trajectory optimization. An approach with which some of the previously described requirements for trajectory planning can nevertheless be met, given a sufficiently fine discretization of the driving space in combination with sampling over a large number of motion candidates, is described in [135].

Direct optimization methods

To avoid the issues of indirect methods, direct optimization methods convert the optimal control problem into a nonlinear programming (NLP) problem, which can be efficiently solved using static optimization methods [82]. In particular, the possibility of integrating a large number of system constraints the requirements for motion planning makes direct optimization methods interesting in automated driving, as can be seen from numerous implementations known from research [43], [51], [134].

A widely used method for solving NLP is sequential quadratic programming (SQP). Due to the iterative solution finding, however, the convergence behavior is highly dependent on an initial starting guess for the numerical algorithm. Its determination is generally not trivial. Furthermore, solving NLPs may be time-consuming due to the sequence of approximating linear quadratic programs (QPs) that are solved iteratively using a forward simulation of the system dynamics [59].

The authors in [34] or [59] for instance, propose the formulation of linear-quadratic optimal control problems in order to reduce the computational effort. By combining linear-quadratic optimal control problems with direct optimization methods, no iterations and no initial starting guess are required. By solving only a single QP in each optimization step, the computational effort can be significantly reduced. Using Linear Model-Predictive Control in combination with quadratic programming is advantageous for real-time trajectory optimization with high replanning frequencies. Applications can therefore be found in various fields of research such as marine vessels [96], aviation [8] and vehicle automation [17], [59].

1.3 Challenges in Teleoperated Driving

Although teleoperated driving offers several advantages by keeping the human operator in the control loop, this technique still poses its own challenges as briefly mentioned in Section 1.2.

Reduced situational awareness of the operator

Due to the limited information that the operator receives about the vehicle environment as a result of the physical absence and the limited senor data, the operator's situation awareness is reduced compared to controlling from within the vehicle [67]. A reduced situation awareness may affect the driving performance of the operator.

Communication loss

An important fact is the possible loss of communication within a teleoperation mission. This aspect is always present in wireless communication systems and there is no practical method of preventing it [21]. Therefore, a suitable solution is required that ensures the safety of a teleoperated vehicle in the event of a loss of communication to the operator.

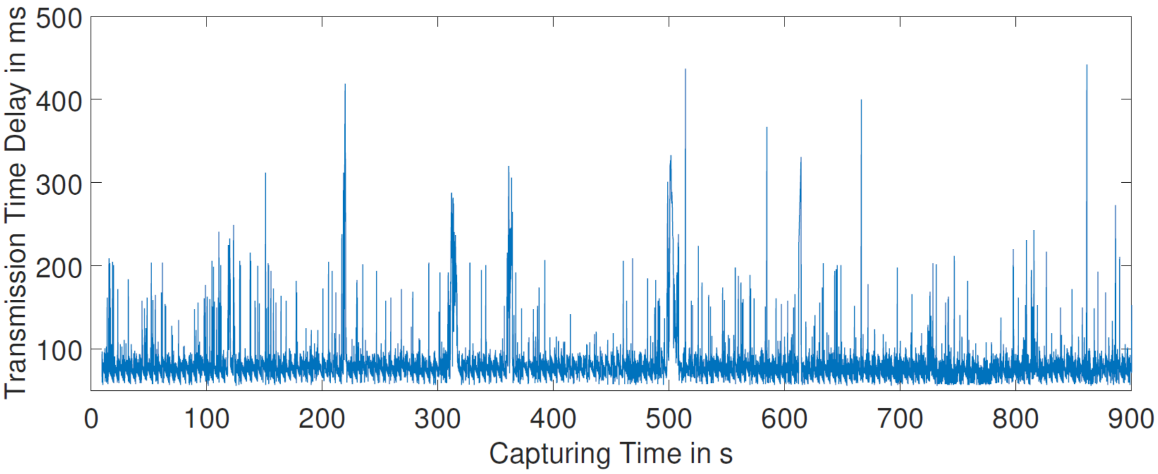


Figure 1.9: Measured time delay during teleoperated driving using a LTE mobile network, including time for the transmission of the camera images from the vehicle to the operator's working station as well as the time for the transmission of the operator's control commands [32].

Communication time delay

In teleoperation, the unavoidable communication link between operator and AV always introduces certain time delays [119]. Furthermore, mobile networks have limited bandwidth capacities, hence, camera images have to be encoded before transmission and decoded thereafter. The authors in [32] took measurements of the transmission of three encoded camera images at the speed of 3 Mbit/s from an AV to the operator's working station using a conventional LTE (4G) network. The results show an average time delay of 138 ms and a maximum time delay of 450 ms, as can be seen in Fig. 1.9. Before visualization, the images are usually buffered up to a constant time delay in order to offer the operator an appropriate streaming quality [32] [67]. Freezing of images during a teleoperation mission can thus be prevented. To take possible communication jitters into account and to ensure a smooth display of the camera images for the operator, the received images must be buffered at least 500 ms at the working station [25], [32]. Controlling a vehicle remotely in an urban environment with such long delays is one of the greatest challenges in teleoperated driving using state-of-the-art methods.

1.4 Objectives

The above mentioned challenges affect the performance of a teleoperated driving task in different ways. Therefore, the main objective of this thesis is to develop control concepts for teleoperated driving that address these challenges. Since a single solution cannot meet all aspects, this thesis aims at providing situation-dependent teleoperated driving concepts for the human operator. The concepts raise the level of autonomy in different ways, while the human operator still remains the main decision maker in all driving tasks.

Model-predictive cruise control for direct control. To relieve the human operator and to promote comfort and safety for direct control, an assistance system is to be devised: the model-predictive cruise control for direct teleoperated driving tasks. Using this concept, the operator's control commands for the longitudinal dynamics are adapted on-board in real-time in order to consider other road participants. This concept aims in particular to support operators in critical situations, when they do not react appropriately in a teleoperated driving maneuver, for example, due to an incorrect assessment of the allowable driving space or due to a delayed or even interrupted communication with the ego vehicle. For this purpose, this assistance system is based on a trajectory optimization for the longitudinal guidance that is to be developed in this thesis.

Corridor-based motion planning for shared control. To overcome high communication time delays, in this thesis a corridor-based motion planning concept is envisaged. This shared control approach is meant to introduce highly automated driving functions for teleoperated driving. In contrast to direct control, in this indirect control concept the operator is kept outside the closed control loop. The guidance loop is closed autonomously by the vehicle. Therefore, the closed control loop is insensitive w.r.t. any communication time delays. By taking advantage of the human abilities in decision making, the operator is enabled to specify an area – the corridor – towards a desired destination. Therefore, the operator is able to interactively take into account, for instance, missing or inadequate lane markings or untracked obstacles, not detected by the system, in the motion planning. The operator decides between specifying a complete corridor to the target destination in advance or initially a sub-corridor using this method. The automated vehicle then calculates an optimal motion by itself within the corridor. In this thesis, a novel hybrid motion planning, combining trajectory and path optimization algorithms, is to be developed.

Automatic path generation for supervisory control. To support the operator in the planning task, this thesis aims at an automatic path generation approach. Without the presence of a target position, this concept continuously generates further feasible paths and then forwards these to the operator. The operator decides which path will be followed by vehicle. For this purpose, a global path search algorithm, the RRT algorithm, is to be modified in such a way that the vehicle environment is explored and reasonable paths are generated. The selected path is then forwarded to the hybrid motion planning algorithm, combining trajectory and path optimization algorithms developed in this work.

Model-predictive trajectory optimization. For the planning of optimal trajectories in real-time, efficient formulations of constrained linear-quadratic optimal control problems are to be derived separately for the longitudinal and lateral dynamics. To keep the order of the system representation and thus the calculation effort as low as possible, the control problem for the lateral dynamics are to be derived with a disturbance term, specifically the reference path orientation. The optimal control problems are solved by means of a time-variant, linear MPC scheme using Quadratic Programming. To address the solvability of the optimal control problems as well as to meet the requirements w.r.t. comfort and safety that arise for automated driving, this thesis introduces a two-stage constraint softening using slack variables. The safety-related longitudinal as well as lateral obstacle distances are linked to a heavily weighted slack variable to guarantee the solvability of the optimization problems. In addition to hard constraints regarding the corresponding accelerations, comfort-related longitudinal and lateral distances to obstacles are to be defined and linked to a less weighted slack variable in order to account for comfort in the trajectory optimization. By introducing a two-stage constraint softening for the longitudinal as well as the lateral obstacle distance constraints in combination with a suitable choice of the weighting of the comfort slack variable, a compromise is to be achieved between a comfortable distance to obstacles and a smooth trajectory profile in a dynamic maneuver. The proposed trajectory optimization design only enables the formulation of locally optimal problems. By combining the trajectory optimization with the path optimization developed is this thesis, the resulting hybrid motion planning envisages globally optimal trajectories.

Path optimization generating globally optimal solutions. For the planning of collision-free and smooth paths in real-time, the Constrained CHOMP algorithm, a local path optimization technique, is to be modified. A second optimization step, the domain step, is to be introduced in this thesis to prevent the path to get stuck in poor local minima. The subsequent smoothing step, including the original formulation of the objective functional, is to be modified in order to take the vehicle dimensions into account. Furthermore, additional objective functions are to be introduced that address the vehicle non-holonomic constraints and penalize the distance to a reference path that represents a guidance planned by the human operator. As a result, the proposed modified, two-step Constrained CHOMP algorithm should allow for the generation of optimal solution including the global minimum in real-time for the application of automated driving.

Path optimization using deep learning. Aiming at a reduction of the computational effort and the memory usage, a concept for path optimization based on deep learning is to be developed. Based on data generated by the modified, two-step Constrained CHOMP algorithm, different problem statements are to be examined together with different neural network architectures. Investigations are to be carried out in which, in addition to the path to be optimized, the entire occupancy grid or preselected sections of the occupancy grid are considered as input of the neural network.

1.5 Outline

The remainder of this thesis is structured as follows:
In **Chapter 2**, the Constrained CHOMP algorithm, a local path optimization, is modified to take the vehicle dimensions and non-holononmic constraints into account. Furthermore, an additional optimization step is introduced so that globally optimal solutions are generated.

Chapter 3 presents an efficient trajectory generation that addresses the requirements arising from automated driving. A two-stage constraint softening is introduced to take comfort and safety into account in the optimization and to ensure the solvability of the linear-quadratic optimal control problems for the longitudinal and lateral dynamics. They are efficiently solved by means of a time-variant, linear MPC scheme using Quadratic Programming.

To promote comfort and safety for direct control, **Chapter 4** proposes a model-predictive cruise control for direct teleoperated driving tasks. This assistance concept uses the presented trajectory optimization in order to adapt on-board the operator's control commands for the longitudinal dynamics in real-time. This concept is validated using two real driving scenarios:

a suddenly appearing road participant at an intersection with right of way and an abruptly braking vehicle in front.

A shared control concept, the corridor-based motion planning, is presented in **Chapter 5**. The human operator is enabled to specify a corridor towards the desired destination, in which the automated vehicle itself calculates an optimal motion. For this purpose, the modified, two-step Constrained CHOMP algorithm and the developed model-predictive trajectory generation are combined. This concept is validated using complex simulations and several real driving scenarios including dynamic obstacles.

Chapter 6 introduces a supervisory control concept. A modified RRT algorithm continuously generates feasible paths, which are forwarded to the operator for selection. The operator decides which path will be followed by the vehicle. The concept is validated in real driving experiments using complex urban scenarios.

Based on deep learning, a novel path optimization concept is presented in **Chapter 7**. For this purpose, different problem statements together with different neural network architectures are examined based on data generated using the modified, two-step Constrained CHOMP algorithm.

Finally, **Chapter 8** concludes this thesis and gives an outlook on possible future work.

Part I

Theory

2 Path Optimization Using Constrained CHOMP

Motion planning represents an essential element in automated driving. In this thesis this is done as a combination of separate path and trajectory planning. While in a less automated teleoperated driving paradigm, i.e. direct control, the path planning results directly from the specifications of the human operator using a gamepad for instance, the path planning is taken over by the system in higher automated approaches (c.f. Sec. 1.2.1). To meet the requirements of real-time path planning, this work modifies the Constrained CHOMP algorithm for the use in automated driving. CHOMP stand for Covariant Hamiltonian Optimization for Motion Planning. Originally, the algorithm has been developed as a trajectory optimization technique in high dimensional spaces, see [108]. As trajectory parameterization, the algorithm uses a uniform discretization, which samples the trajectory $\boldsymbol{\xi}(s) = [x(s), y(s)]$ over equal steps of length Δs: $\boldsymbol{\xi} \approx (\mathbf{q}_1^\top, \mathbf{q}_2^\top, \ldots, \mathbf{q}_n^\top)^\top \in \mathbb{R}^{n\times 2}$, with $\mathbf{q}_0 = [x_0, y_0]$ as the fixed starting point [142]. By using this waypoint parameterization taking into account a constant waypoint distance, the trajectory planning problem is identical to a path planning problem. Therefore, $\boldsymbol{\xi}$ is referred to as path in the following.

2.1 Objective Functional

The Constrained CHOMP algorithm in its original form iteratively improves the quality of an initial path $\boldsymbol{\xi}_0$ by optimizing an objective functional

$$U(\boldsymbol{\xi}) = f_{obs}(\boldsymbol{\xi}) + w \cdot f_{sm}(\boldsymbol{\xi}), \tag{2.1}$$

which represents a trade-off between obstacle avoidance and path smoothness. Here, w denotes the weighing factor, cf. [108].

Smoothness Objective

The term $f_{sm}(\boldsymbol{\xi})$ measures dynamical quantities across the path as the integral over squared velocity norms [142]:

$$f_{sm}(\boldsymbol{\xi}) = \frac{1}{2}\int_0^1 \left\| \frac{\mathrm{d}}{\mathrm{d}s}\boldsymbol{\xi}(s) \right\|^2 \mathrm{d}s, \quad \text{where} \quad s \in [0,1]\,. \tag{2.2}$$

Using the previously described waypoint parameterization, the smoothness objective is therefore reformulated as the sum of squared first-order difference quotients to

$$f_{sm}(\boldsymbol{\xi}) = \frac{1}{2}\sum_{k=0}^{n} \left\| \frac{\mathbf{q}_{k+1} - \mathbf{q}_k}{\Delta s} \right\|^2 . \tag{2.3}$$

By transforming the difference quotients into a matrix notation and introducing the finite difference matrix $\mathbf{K}$, the smoothness objective is rewritten as

$$f_{sm}(\boldsymbol{\xi}) = \frac{1}{2}||\mathbf{K}\,\boldsymbol{\xi} + \mathbf{e}||^2 = \frac{1}{2}\boldsymbol{\xi}^\top \mathbf{A}\,\boldsymbol{\xi} + \boldsymbol{\xi}^\top \mathbf{b} + g\,, \tag{2.4}$$

with $\mathbf{A} = \mathbf{K}^\top\mathbf{K}$, $\mathbf{b} = \mathbf{K}^\top\mathbf{e}$ and $g = \mathbf{e}^\top\mathbf{e}/2$. The vector $\mathbf{e}$ addresses the boundary condition $\mathbf{q}_0$. Since this optimization technique is based on a projected Newton method, the functional gradients of the objective functions are required [115]. Considering the quadratic form of the smoothness objective (2.4), the functional gradient of the smoothness objective becomes straightforward

$$\bar{\nabla} f_{sm}(\boldsymbol{\xi}) = \mathbf{A}\,\boldsymbol{\xi} + \mathbf{b}\,. \tag{2.5}$$

Obstacle Objective

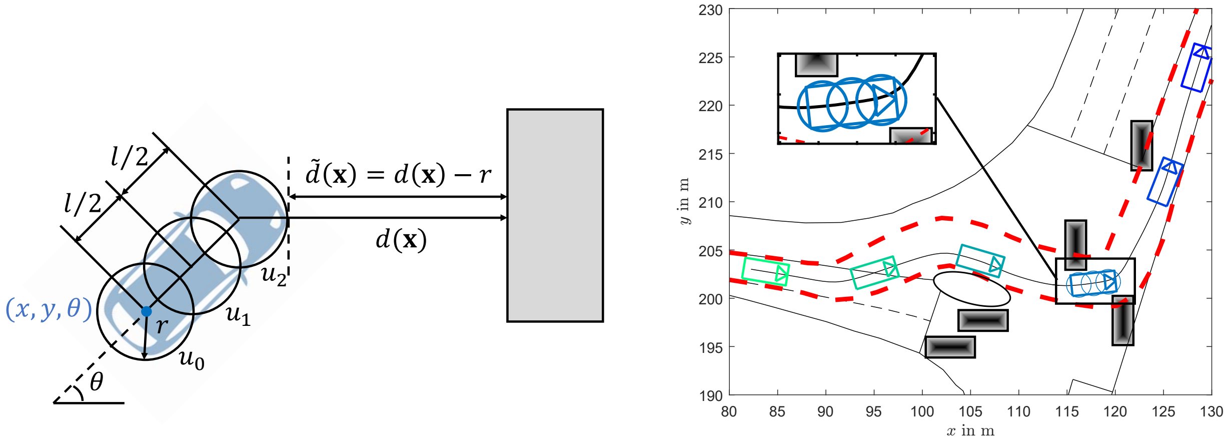
While $f_{sm}(\boldsymbol{\xi})$ encourages smooth paths, the term $f_{obs}(\boldsymbol{\xi})$ penalizes the vehicle's proximity to any obstacle in the environment. In order to make the optimization tractable, an efficient representation of the vehicle body is required [142]. Taking into account each body element $u \in B$ of the set B of the vehicle's body approximation, the obstacle objective $f_{obs}(\boldsymbol{\xi})$ integrates the cost encountered by each body element u along the path. Specifically, the obstacle objective calculates an arc length parameterized line integral of each body element's path through a workspace cost field and integrates those values over all body elements [142]:

$$f_{obs}(\boldsymbol{\xi}) = \int_B \int_0^1 c\big(\mathbf{x}(\boldsymbol{\xi}(s), u)\big)\; ||\frac{\mathrm{d}}{\mathrm{d}s}\mathbf{x}(\boldsymbol{\xi}(s), u)||\;\mathrm{d}s\,\mathrm{d}u, \quad \text{where} \quad s \in [0, 1]\,. \tag{2.6}$$

The term $\mathbf{x}(\boldsymbol{\xi}(s), u)$ denotes the forward kinematics, mapping a robot's configuration $\mathbf{q}$ and a particular body element $u \in B$ to a point $\mathbf{x}(\mathbf{q}, u)$ in the workspace. The workspace cost function $c\big(\mathbf{x}(\boldsymbol{\xi}(s), u)\big)$ quantifies the cost of a body element u of residing at a particular wokspace point $\mathbf{x}(\mathbf{q}, u)$. Furthermore, the workspace cost function $c\big(\mathbf{x}(\boldsymbol{\xi}(s), u)\big)$ is multiplied by the norm of the workspace velocity of each body element. This leads to the arc length parameterization, which ensures that the optimization does not alter the waypoint distance. An arbitrary vehicle motion can be attained depending on the definition of the workspace cost function [116]. As in [108], the workspace cost function is defined as

$$c(\mathbf{x}) = \begin{cases} -\tilde{d}(\mathbf{x}) + \frac{1}{2}\varepsilon, & \text{if } \tilde{d}(\mathbf{x}) < 0 \\ \frac{1}{2\varepsilon}(\tilde{d}(\mathbf{x}) - \varepsilon)^2, & \text{if } 0 < \tilde{d}(\mathbf{x}) \leq \varepsilon \\ 0 & \text{otherwise} \end{cases}\,. \tag{2.7}$$

The function $\tilde{d}(\mathbf{x})$ provides the nearest distance from the vehicle's body to an obstacle. For an efficient computation of $\tilde{d}(\mathbf{x})$, an appropriate representation of the vehicle body is required. As can be seen in Fig. 2.1, the vehicle's body is over-approximated by a set of three circles B in this thesis. By using this approximation, hence, the distance of the vehicle to any obstacle can be calculated efficiently. In the case of a circle, the distance $\tilde{d}(\mathbf{x})$ to any point in the plane results from the distance $d(\mathbf{x})$ to the centre of the corresponding circle subtracted by its radius r. The cost of a workspace point smoothly drops to zero as a distance of the allowable threshold ε to an obstacle is reached, see Fig. 2.2.



(a) **(b)**

Figure 2.1: Vehicle body approximation using a set of three circles.

Since the obstacle objective $f_{obs}(\boldsymbol{\xi})$ depends only on workspace positions and velocities without higher order derivatives, according to [106] the functional gradient of (2.6) can be derived as $\bar{\nabla} f_{obs}(\boldsymbol{\xi}) = \frac{\partial v(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} - \frac{\mathrm{d}}{\mathrm{d}s} \frac{\partial v(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}'}$. Here, $v(\boldsymbol{\xi})$ denotes everything inside the time integral. Applying this formula to (2.6), the functional gradient is given by

$$\bar{\nabla} f_{obs}(\boldsymbol{\xi}) = \int_B \mathbf{J}^\top ||\mathbf{x}'|| \left[(\mathbf{I} - \hat{\mathbf{x}}' \, \hat{\mathbf{x}}'^\top) \nabla c - c \boldsymbol{\kappa} \right] \mathrm{d}u \,, \tag{2.8}$$

where $\boldsymbol{\kappa}$ denotes the curvature vector of path $\boldsymbol{\xi}$ (see Appendix A.1) defined by

$$\boldsymbol{\kappa} = \frac{1}{||\mathbf{x}'||^2} \left(\mathbf{I} - \hat{\mathbf{x}}' \, \hat{\mathbf{x}}'^\top \right) \cdot \mathbf{x}'' \,. \tag{2.9}$$

The term $\hat{\mathbf{x}}'$ in equation denotes the normalized velocity vector $\mathbf{x}'/||\mathbf{x}'||$ and $\mathbf{J}$ represents the kinematic Jacobian at a specific body point u. To simplify the notation, the dependance of $\mathbf{J}$, $\mathbf{x}$ and c on the integration variables u and t has been suppressed.

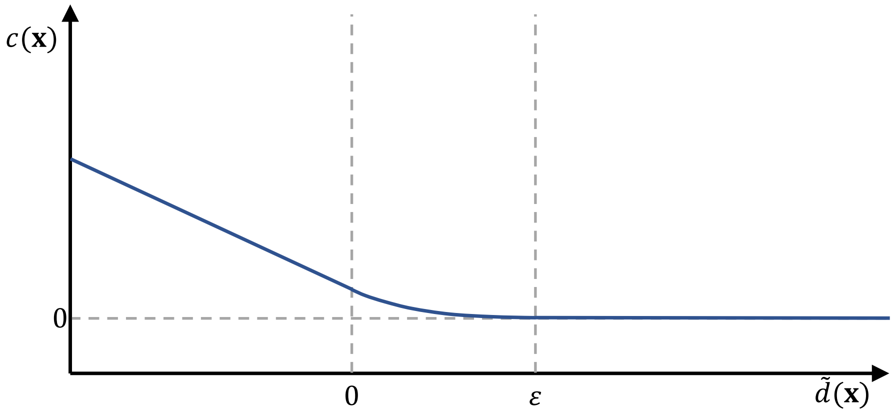


Figure 2.2: Illustration of the workspace cost function $c(\mathbf{x})$.

Using the same discretization as for $f_{sm}(\boldsymbol{\xi})$, the obstacle objective is reformulated to

$$f_{obs}(\boldsymbol{\xi}) = \sum_{u\in B}\sum_{k=1}^{n} c\big(\mathbf{x}(\mathbf{q}_k, u)\big)\ ||\mathbf{x}'(\mathbf{q}_k, u)||\,. \tag{2.10}$$

Equivalent to the obstacle objective, the corresponding functional gradient components $\bar{\nabla} f_{obs,k}(\boldsymbol{\xi}) \in \bar{\nabla} f_{obs}(\boldsymbol{\xi})$ for $k = 1 \dots n$ are reformulated as the sum of the set of body elements:

$$\bar{\nabla} f_{obs,k}(\boldsymbol{\xi}) = \sum_{u\in B} \mathbf{J}_k^{\top} ||\mathbf{x}'_k||\ \left[(\mathbf{I} - \hat{\mathbf{x}}'_k\, \hat{\mathbf{x}}_k'^{\top})\nabla c_k - c_k \kappa_k\right]\,. \tag{2.11}$$

2.2 Dual Projected Newton Method

Given the objective functional $U(\boldsymbol{\xi})$ and the corresponding functional gradient $\bar{\nabla} U(\boldsymbol{\xi})$, the optimization of the initial path $\boldsymbol{\xi}_0$ is performed according to [22] by iteratively solving a sequential quadratic problem subject to linear inequality constraints:

$$\begin{aligned} \boldsymbol{\xi}_{i+1} = \arg\min_{\boldsymbol{\xi}} &\left[U(\boldsymbol{\xi}_i) + (\boldsymbol{\xi} - \boldsymbol{\xi}_i)^{\top}\bar{\nabla} U(\boldsymbol{\xi}_i) + \tfrac{\eta_i}{2}||\boldsymbol{\xi} - \boldsymbol{\xi}_i||_{\mathbf{A}}\right] \\ &\text{s.t}\ \mathbf{C}\,\boldsymbol{\xi} \le \mathbf{d}\,. \end{aligned} \tag{2.12}$$

Here, i represents the iteration and the notation $||\boldsymbol{\xi} - \boldsymbol{\xi}_i||_{\mathbf{A}}$ denotes the norm of the displacement taken with respect to the smoothness matrix $\mathbf{A}$[1]. Furthermore, the admissible path $\boldsymbol{\xi}$ can be restricted by using the linear inequality constraints $\mathbf{C}\,\boldsymbol{\xi} \le \mathbf{d}$.

With the initial path $\boldsymbol{\xi}_0$ and Lagrange multipliers $\boldsymbol{\lambda}_0$, the iterative optimization is stated as follows:

1. Perform an iteration of the projected Newton method

$$\begin{aligned} \boldsymbol{\lambda}_{i+1} &= \mathbf{P}_{\ge 0}\left(\boldsymbol{\lambda}_i - \alpha_i \begin{bmatrix} \mathbf{S}_i & 0 \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} [\bar{\nabla}\mathbf{G}(u_i))]_{\mathbf{F}_i} \\ [\bar{\nabla}\mathbf{G}(u_i))]_{\mathbf{B}_i} \end{bmatrix}\right), \\ \text{where}\ \ \mathbf{G}(\boldsymbol{\lambda}) &= \tfrac{1}{2\eta_i}\boldsymbol{\lambda}^{\top}\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^{\top}\boldsymbol{\lambda} - \dots \\ &\quad \boldsymbol{\lambda}^{\top}\left(\mathbf{C}\boldsymbol{\xi}_i - \mathbf{d} - \tfrac{1}{\eta_i}\mathbf{C}\mathbf{A}^{-1}\bar{\nabla} U(\boldsymbol{\xi}_i)\right)\,. \end{aligned} \tag{2.13}$$

2. Perform a primal path update

$$\boldsymbol{\xi}_{i+1} = \boldsymbol{\xi} - \frac{1}{\eta_i}\mathbf{A}^{-1}\bar{\nabla} U(\boldsymbol{\xi}_i) - \frac{1}{\eta_i}\mathbf{A}^{-1}\mathbf{C}^{\top}\boldsymbol{\lambda}_{i+1}\,. \tag{2.14}$$

The update rule (2.14) is derived from the primal solution (2.12) using a Lagrangian dual method, c.f. [22]. The matrix $\mathbf{P}_{\ge 0}$ projects each variable to the positive half-plane. The term η is a regularization coefficient that specifies the trade-off between the step size α and the minimization of the objective functional $U(\boldsymbol{\xi})$. Furthermore, the term $\mathbf{B}_i$ and $\mathbf{F}_i$ define the binding set and the free set, respectively. The matrix $\mathbf{S}_i = [\bar{\nabla}^2\mathbf{G}(\boldsymbol{\lambda}_i)]_{\mathbf{F}_i}^{-1}$ represents a submatrix of the inverse Hessian along the direction of free variables $\mathbf{F}_i$, cf. [22].

For optimization two stop criteria are used in parallel in this work: One stop criterion applies in the event that the differential change of the objective functional $U(\boldsymbol{\xi})$ becomes smaller than a threshold, whereas the other one represents the maximum number of iterations.

---

[1] $||\boldsymbol{\xi} - \boldsymbol{\xi}_i||_{\mathbf{A}} = \sqrt{(\boldsymbol{\xi} - \boldsymbol{\xi}_i)^{\top}\,\mathbf{A}\,(\boldsymbol{\xi} - \boldsymbol{\xi}_i)}$

2.3 Complement of the Objective Functional with Curvature Penalty Function

Although the Constrained CHOMP algorithm is well suited to generate smooth and collision-free paths, the vehicle non-holonomic constraints are not guaranteed to be satisfied. As described in Sec. 3.1, however, the subsequent trajectory optimization of the two-stage motion planning method developed in this thesis – consisting of a decoupled path and trajectory optimization – requires a reference path with sufficient quality. Due the limitation of the vehicle's maximum steering angle, the curvature of the path must therefore be restricted in the optimization. The integration of the curvature into the inequality constraint of the optimization process is not straightforward, as can be seen in (2.12). The resulting non-linear inequality constraints would require more complicated and therefore more computationally expensive optimization methods compared to the advantageous properties of the Dual Projected Newton method.

Alternatively, a penalty function in the form of the curvature objective

$$\begin{aligned} f_{crv}(\xi) &= \sum_{k=1}^{n} f_{curv,k}(\xi)\,, \text{ where} \\ f_{curv,k}(\xi) &= \begin{cases} 0 & \text{if } \|\kappa_k\| \le \kappa_{max} \\ \frac{1}{2}(\kappa_k - \kappa_{max})^2 & \text{else} \end{cases} \end{aligned} \tag{2.15}$$

and the corresponding gradient components

$$\begin{aligned} \bar{\nabla} f_{curv,k}(\xi) &= \begin{cases} 0 & \text{if } \|\kappa_k\| \le \kappa_{max} \\ (\kappa_k - \kappa_{max})\nabla\kappa_k & \text{else} \end{cases} \\ &\text{for } k = 1 \dots n\,,\ \bar{\nabla} f_{curv,k}(\xi) \in \bar{\nabla} f_{crv}(\xi) \end{aligned} \tag{2.16}$$

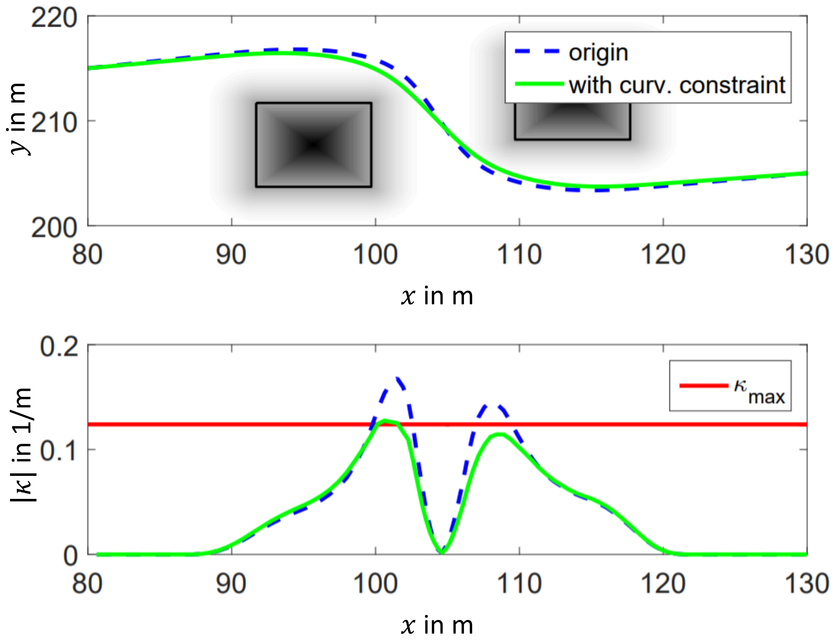


Figure 2.3: Simulation results regarding the curvature constraints.

is integrated in the Constrained CHOMP framework in this work. The gradient $\nabla\kappa$ may be derived as the partial derivative of (2.9) for all path points. As examined in [14], however, this leads to a relatively complex calculation formula. Instead, in this work the curvature vector defined in [50] is used as the update direction, whereby nearly the same result is achieved, but with less calculation and implementation effort. In order not to influence the optimization unnecessarily, the penalty function only takes effect when the curvature limit κ_{max} is exceeded. In particular, if the limit κ_{max} is exceeded in an optimization step, the curvature penalty function pulls the path back within the restrictions, see Fig. 2.3. The term κ_{max} is defined as the vehicle's drivable curvature limit subtracted by a threshold.

By including the curvature penalty function into the optimization, the objective function is reformulated to

$$U(\boldsymbol{\xi}) = f_{obs}(\boldsymbol{\xi}) + w_{sm} \cdot f_{sm}(\boldsymbol{\xi}) + w_{crv} \cdot f_{crv}(\boldsymbol{\xi}) \,. \tag{2.17}$$

Although the curvature value can still exceed the physical limit despite the modification, the excess is kept relatively small. As a result, qualitatively sufficient reference paths can be generated for the subsequent trajectory optimization. In order to generate feasible driving motions for the automated vehicle, however, the curvature limitation is taken into account as a hard constraint in the trajectory optimization.

2.4 Complement of the Objective Functional with Reference Path Objective

With the presence of the curvature term in the objective function, the Constrained CHOMP algorithm is able to generate smooth and collision-free paths while taking curvature constraints into account. While the smoothness objective of the optimization generates a path as straight as possible to the end of a given route after evasive maneuvers, natural human driving behaviour shows the intention to return to the given route as quickly as possible. In autonomous driving the route may be generated by the perception system, i.e. the lane center line. In the case of teleoperated driving approaches, developed in this thesis, the route represents a guidance planned or selected by the operator.

To address the above mentioned natural driving behaviour, the objective functional $U(\boldsymbol{\xi})$ is extended by a further term

$$f_{ref}(\boldsymbol{\xi}) = \frac{1}{2} \sum_{k=1}^{n} d_k(\mathbf{q}_k, \boldsymbol{\xi}_{ref})^2 \,, \tag{2.18}$$

which penalizes the distance to the reference path $\boldsymbol{\xi}_{ref}$ – a route specified by the human operator for instance – in the form of the Frenet-transformed coordinate, see Fig. 2.4. The

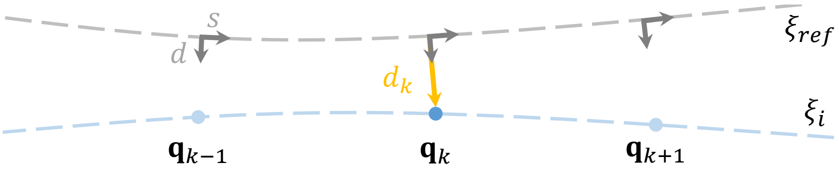


Figure 2.4: Illustration of the reference path cost calculation.

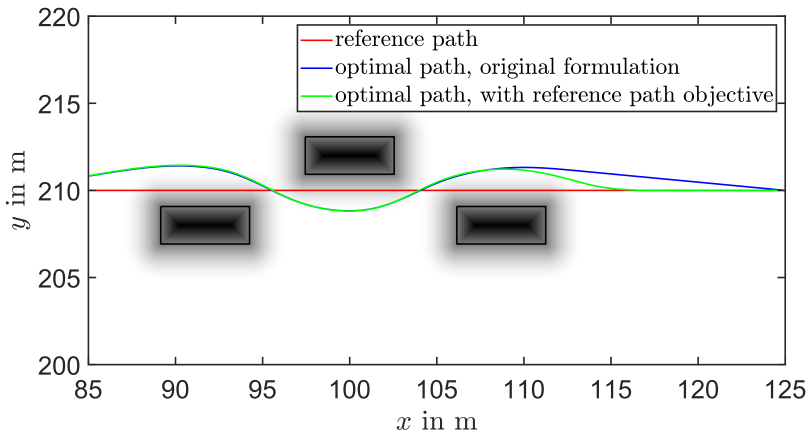


Figure 2.5: Simulation results regarding the reference path objective.

corresponding components of the functional gradient of the reference path objective become

$$\begin{aligned} \bar{\nabla} f_{ref,\,k}(\boldsymbol{\xi}) = d_k(\mathbf{q}_k, \boldsymbol{\xi}_{ref}) \begin{bmatrix} sin(\theta_k) & -cos(\theta_k) \end{bmatrix} \\ \text{for } k = 1 \ldots n\,, \; \bar{\nabla} f_{ref,k}(\boldsymbol{\xi}) \in \bar{\nabla} f_{ref}(\boldsymbol{\xi})\,. \end{aligned} \tag{2.19}$$

As can be seen in Fig. 2.5, the optimized path using the reference path objective smoothly approaches the reference path $\boldsymbol{\xi}_{ref}$ after the evasive maneuvers. Compared to the path without the added reference path objective, the path with the reference path objective returns to the reference path noticeably earlier.

Together with the curvature penalty function, the objective functional $U(\boldsymbol{\xi})$ is therefore reformulated to

$$U(\boldsymbol{\xi}) = f_{obs}(\boldsymbol{\xi}) + w_{sm} \cdot f_{sm}(\boldsymbol{\xi}) + w_{crv} \cdot f_{crv}(\boldsymbol{\xi}) + w_{ref} \cdot f_{ref}(\boldsymbol{\xi})\,. \tag{2.20}$$

2.5 Local Minima Avoidance

Due to the nature of the dual projected newton method, the Constrained CHOMP algorithm has a tendency to get stuck in undesirable local minima. This usually occurs if the initial path is passing through an obstacle including a flat or non-convex outer profile [118]. To be more robust to local minima, the authors in [71] include random pertubations in the optimization process. Therefore, this technique tries to push the optimization result away from a local minima. While this technique enables a better exploration of the search area, it brings significant disadvantages in terms of the computing effort, making the algorithm unsuitable for real-time applications.

As a remedy, this work proposes a two step path optimization including:

- a domain step and

- a smoothing step

using the Constrained CHOMP algorithm. The smoothing step represents the optimization w.r.t. the objective functional (2.20).

If a collision is detected along the initial path $\boldsymbol{\xi}_0$, the domain step is executed first. According to the evaluation of the search area, the path is shifted either orthogonally to the left or the right. For this purpose, the optimization gradient is redefined in the domain step as follows:

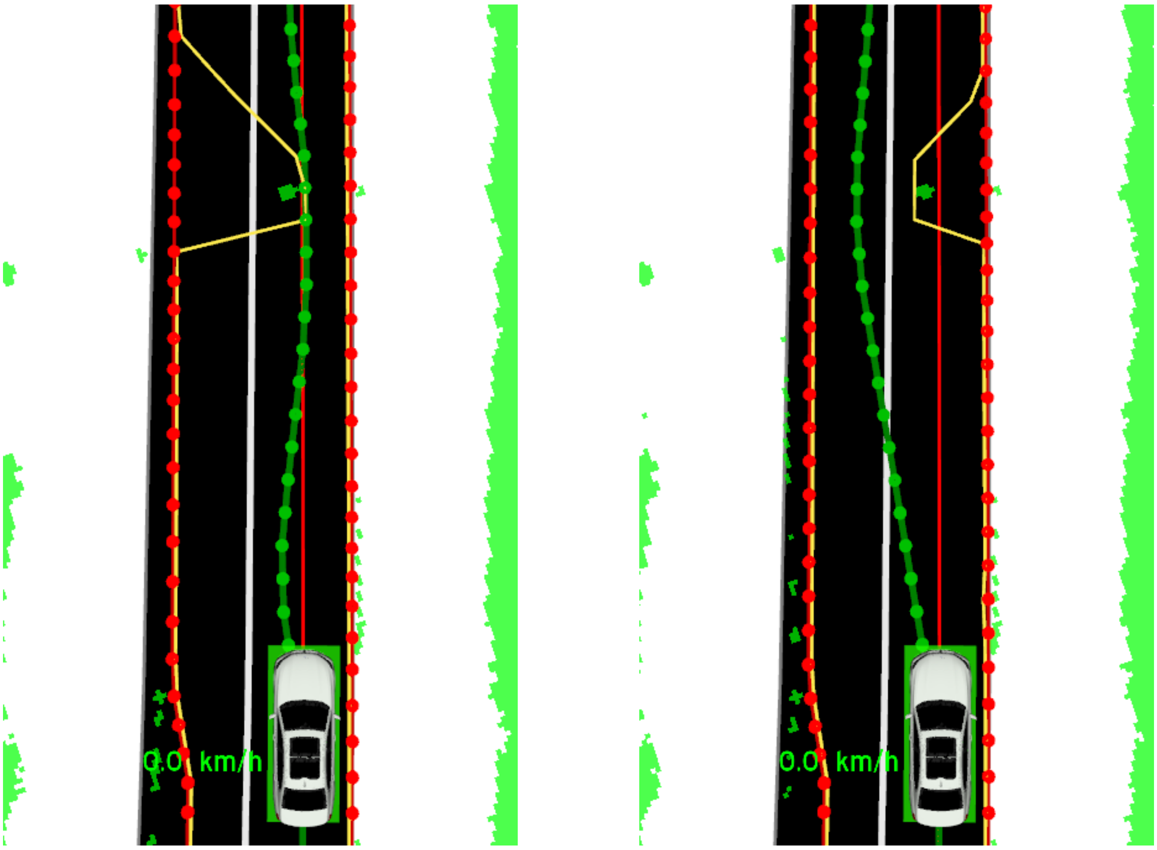
$$
\begin{aligned}
\bar{\nabla} U_k(\boldsymbol{\xi}) &= \begin{cases} \Delta\boldsymbol{\xi}_{\text{norm},k} \cdot \boldsymbol{\Upsilon}_{\text{left/right}} & \text{if collision = true} \\ 0 & \text{else} \end{cases} \\
\boldsymbol{\Upsilon}_{\text{left}} &= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad \boldsymbol{\Upsilon}_{\text{right}} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \\
\text{for } k &= 1 \ldots n, \; \bar{\nabla} U_k(\boldsymbol{\xi}) \in \bar{\nabla} U(\boldsymbol{\xi}),
\end{aligned}
\tag{2.21}
$$

where $\Delta\boldsymbol{\xi}_{\text{norm}}$ denotes the normalized difference vector $\Delta\boldsymbol{\xi}/||\Delta\boldsymbol{\xi}||$. For optimization in the domain step two stop criteria are used in parallel: One stop criterion represents the maximum number of iterations, whereas the other applies if the optimized path becomes collision-free.

Fig. 2.6 shows an example of avoiding local minima using the proposed two-step optimization. Since the vehicle's perception system was not able to correctly perceive barrier tapes in this real-world scenario, the driving area has been manually restricted by the human operator using the teleoperated driving paradigm proposed in Sec. 5 in the form of a corridor. The constraints on the path, i.e. the corridor, are depicted as a red dotted line. For operator feedback purposes, the obstacle bounds, which visualize the free driving space and are based on the optimal path, are represented by yellow lines. The initial path, which is used to initialize the optimization, is displayed as a red line.

As can be seen in the scenario, the initial path is located between an obstacle and the right corridor boundary. The Constrained CHOMP algorithm in its original form tries to push the initial path away from the obstacle, directly into the corridor boundary, resulting in the optimized path (green dotted line) corresponding to a poor local minimum, see Fig. 2.6 (a).

By using the introduced domain step, however, the algorithm manages to generate an optimal result related to the global minimum, see Fig. 2.6 (b). The proposed gradient in the domain step successfully pushes the initial path orthogonally to the left out of the local minimum. As a result, a collision-free path is generated, that circumvents the obstacle. As can also be seen in Fig. 2.6 (b), the path that was generated in the domain step is already smooth. The subsequent smoothing step results in a barely noticeable improvement in smoothness in the given scenario. This is due to the nature of the Constrained CHOMP algorithm, which uses a smoothness matrix $\mathbf{A}$ as the Riemann's metric, which is motivated in [108].



(a) (b)

Figure 2.6: Local minima avoidance using Constrained CHOMP with: (a) original formulation, (b) domain step only.

3 Model-Predictive Trajectory Optimization Using Quadratic Programming

The requirement for a real-time capable trajectory optimization with a high re-planning frequency represents a major challenge in the context of automated systems. The consideration of constraints to meet the requirements defined in Section 1.2.3 is essential in almost all driving situations in order to take into account, for instance, the limits of driving physics during emergency braking and dynamic maneuvers or to ensure collision-free trajectories by constraining the driving space.

By taking advantage of the beneficial propoerties of linear-quadratic optimal control problems in combination with direct optimization methods, this chapter presents an approach for efficient trajectory optimization. The resulting quadratic program allows for a cyclical solution finding on a progressive optimization horizon – in the sense of MPC – with high re-planning frequency in real-time.

3.1 Linear Vehicle Dynamics

The idea of MPC is to optimize a prediction of a system behaviour. As the methodology is based on a system model, the model represents the most important element of the MPC design [109]. For the application of linear-quadratic optimization, the description of the vehicle dynamics in the form of linear system models is essential. Although the vehicle dynamics are generally non-linear, the description of the motion equations in a Frenét frame (Fig. 3.1)

$$\dot{s}(t) = v(t)\frac{\cos\left(\theta(t) - \theta_\Gamma(t)\right)}{1 - d(t)\kappa_\Gamma(t)} \tag{3.1a}$$

$$\dot{d}(t) = v(t)\sin\left(\theta(t) - \theta_\Gamma(t)\right) \tag{3.1b}$$

for the longitudinal distance s and the lateral distance d in relation to a reference path Γ offers the possibility of deriving a simplified system model for the vehicle's relative kinematics [131]. The derivation of the motion equations (3.1) is documented in Appendix A.2. Under the assumption of small differences between the vehicle's orientation $\theta(t)$ and the orientation of the reference path $\theta_\Gamma(t)$ as well as small lateral distances $d(t)$ in relation to the radius of the reference path $r_\Gamma(t)$, so that $\frac{d(t)}{r_\Gamma(t)} = d(t)\kappa_\Gamma(t) \ll 1$ applies, the motion equations are reformulated as follows

$$\dot{s}(t) = v(t) \tag{3.2a}$$

$$\dot{d}(t) = v(t)\left(\theta(t) - \theta_\Gamma(t)\right) . \tag{3.2b}$$

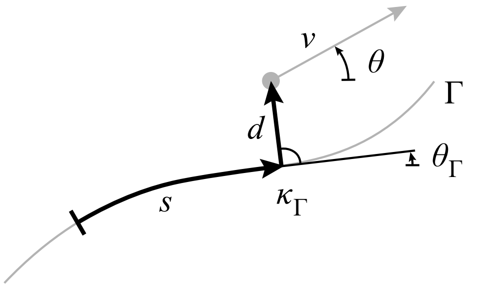


Figure 3.1: Relative kinematics of the vehicle motion in a Frenét frame, according to [131].

Since the optimization of the driving trajectory is based on a reference path Γ and the corresponding vehicle motion is in the vicinity of the reference path, the previously derived assumptions for the linearization of the motion equations (3.1) generally result in negligible effects on the application. Even in critical maneuvers, in which orientation differences $(\theta(t) - \theta_\Gamma(t))$ of up to $25°$ may occur for a short time, the approximation generates a minor deviation of less than 3% in the lateral dynamics $\dot{d}(t)$. For the longitudinal dynamics $\dot{s}(t)$, maximum deviations of 9% result in a conservative overestimation of the positions to be reached, so that possible deviations only lead to an increase in safety reserves.

Despite the simplifying assumptions, the motion equation for the lateral dynamics (3.2b) remains nonlinear. A combined optimization of both dynamics with the state vector $\mathbf{x}(t) = [s(t), d(t)]^\top$ would result in a non-linear optimal control problem for the input vector $\mathbf{u}(t) = [u_1(t), u_2(t)]^\top = [v(t), \theta(t)]^\top$. To determine the optimal trajectory, in this work, the corresponding vehicle dynamics are separated into their two main parts. The optimal trajectory results from a two-stage solution of isolated linear-quadratic optimal control problems for the longitudinal and lateral dynamics. The two-step solution is therefore characterized as follows:

1. optimize $u_1(t)$ based on an initial estimate of $u_2(t)$,
2. optimize $u_2(t)$ using the previously generated $u_1(t)$.

It becomes clear, that a good initial estimate of the lateral dynamics is essential for optimizing the longitudinal dynamics. In this work, this condition is fulfilled in two different ways depending on the teleoperated driving paradigm. In direct control, the vehicle's lateral motion results directly from the steering wheel angle specified by the human operator and is calculated using kinematic relationships. In the case of indirect control, an optimized reference path forms the basis for the initial estimate of the lateral dynamics. The reference path Γ is optimized using the modified Constrained CHOMP algorithm, see Chapter 2. Moreover, the solution of the modified Constrained CHOMP algorithm provides the trajectory optimization the basis for a convex problem formulation including the global minimum.

3.2 Discrete-Time Problem Representation

The discrete-time formulation of the originally continuous-time state space representation

$$\dot{\mathbf{x}}(t) = \mathbf{A}_{cont}(t)\mathbf{x}(t) + \mathbf{b}_{cont}(t)u(t) + \mathbf{d}_{cont}(t)z(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{3.3a}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{e}z(t), \tag{3.3b}$$

forms the basis for the application of direct optimization methods. By considering discrete time steps $t_k = k \cdot T_s$, $k \in \mathbb{N}$ of the continuous-time system with the step size T_s, the continuous-time system can be converted exactly into its discrete-time formulation, assuming a piece-wise constant input $u(t)$ and disturbance $z(t)$ between discrete time steps $kT_s \leq t \leq (k+1)T_s$. The progress of the system state $\mathbf{x}(t_k)$ at discrete time steps t_k is calculated for a given initial state $\mathbf{x}_0$ with the progress of $u(t_k)$ and $z(t_k)$ by successively solving the continuous state differential equation

$$\mathbf{x}(t) = e^{\mathbf{A}_{cont}(t-t_0)}\mathbf{x}_0 + \int_{t_0}^{t} e^{\mathbf{A}_{cont}(t-\tau)}\mathbf{b}_{cont}(\tau)u(\tau)\mathrm{d}\tau + \int_{t_0}^{t} e^{\mathbf{A}_{cont}(t-\tau)}\mathbf{d}_{cont}(\tau)z(\tau)\mathrm{d}\tau \tag{3.4}$$

on the time intervall $t \in [t_k, t_{k+1}]$ for an iterating k starting at $k = 0$ [70].

Assuming a piece-wise constant continuous-time system matrix and system vectors, for the discrete-time system matrix and system vectors the following applies on the discretization interval k:

$$\mathbf{A}(t_k) = e^{\mathbf{A}_{cont}(t_k)T_s}\,, \tag{3.5a}$$

$$\mathbf{b}(t_k) = \int_{t_k}^{t_{k+1}} e^{\mathbf{A}_{cont}(t_k)\tau}\mathbf{b}_{cont}(t_k)\mathrm{d}\tau\,, \tag{3.5b}$$

$$\mathbf{d}(t_k) = \int_{t_k}^{t_{k+1}} e^{\mathbf{A}_{cont}(t_k)\tau}\mathbf{d}_{cont}(t_k)\mathrm{d}\tau\,. \tag{3.5c}$$

The matrix $\mathbf{C}$ and the vector $\mathbf{e}$ are identical in the continuous-time and discrete-time representation. The linear, discrete-time model of the generalized system formulation (3.3) is thus obtained using a simplified representation of the time dependency ($t_k \longrightarrow k$) as

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{b}(k)u(k) + \mathbf{d}(k)z(k)\,, \tag{3.6a}$$

$$\mathbf{y}(k) = \mathbf{C}(k)\mathbf{x}(k) + \mathbf{e}(k)z(k)\,. \tag{3.6b}$$

The optimal trajectory is obtained by minimizing a quadratic objective function

$$J = \sum_{k=1}^{N} [\mathbf{x}(k) - \mathbf{x}_{\mathrm{d}}(k)]^{\top}\, \mathbf{Q}_k\, [\mathbf{x}(k) - \mathbf{x}_{\mathrm{d}}(k)] + \sum_{k=0}^{N-1} r_k u^2(k)\,, \tag{3.7}$$

extended by desired values $\mathbf{x}_{\mathrm{d}}$ and which in the discrete-time case is represented as the sum of all optimization time steps $k = 0, \ldots, N$ on the optimization horizon N [70]. The linear time-variant inequality constraints for the system input u and any system outputs $\mathbf{y}$ are defined at each time step k in the form of:

$$u_{\min}(k) \leq u(k) \leq u_{\max}(k)\,, \tag{3.8a}$$

$$\mathbf{y}_{\min}(k) \leq \mathbf{y}(k) \leq \mathbf{y}_{\max}(k)\,. \tag{3.8b}$$

3.3 Formulation of the Constrained Quadratic Program

For solving constrained linear-quadratic optimal control problems (3.6) - (3.7) subject to linear constraints (3.8), the so-called *batch*-approach has been established in the literature [13]. Using hyper-vectors, this approach compactly represents the future system behaviour based on the initial state $\mathbf{x}_0$. To simplify the notation, the optimization time steps $k = 0, \dots, N$ are shown in the following as an index. Given the sequence of vectors for the system state $\mathbf{x}_k \in \mathbb{R}^n$ as well as the output $\mathbf{y}_k \in \mathbb{R}^p$

$$\mathbf{X} = \left[\mathbf{x}_1^\top, \dots, \mathbf{x}_N^\top\right]^\top, \qquad \mathbf{X} \in \mathbb{R}^{nN} \tag{3.9}$$

$$\mathbf{Y} = \left[\mathbf{y}_1^\top, \dots, \mathbf{y}_N^\top\right]^\top, \qquad \mathbf{Y} \in \mathbb{R}^{pN} \tag{3.10}$$

can be collected. The same is done with the a-priori known vector sequences of the disturbance term $z_k \in \mathbb{R}$

$$\mathbf{z} = \left[z_1, \dots, z_N\right]^\top, \qquad \mathbf{z} \in \mathbb{R}^{N} \tag{3.11}$$

$$\tilde{\mathbf{z}} = \left[z_0, \dots, z_{N-1}\right]^\top, \qquad \tilde{\mathbf{z}} \in \mathbb{R}^{N} \tag{3.12}$$

as well as the sequence of unknown input-vectors $\mathbf{u}_k \in \mathbb{R}$, i.e.,

$$\mathbf{u} = \left[u_0, \dots, u_{N-1}\right]^\top, \qquad \mathbf{u} \in \mathbb{R}^{N}. \tag{3.13}$$

As a result, the system equations (3.6) are reformulated to

$$\mathbf{X} = \mathcal{A}\mathbf{x}_0 + \mathcal{B}\mathbf{u} + \mathcal{D}\tilde{\mathbf{z}} \tag{3.14a}$$

$$\mathbf{Y} = \mathcal{C}\mathbf{X} + \mathcal{E}\mathbf{z}, \tag{3.14b}$$

whereas the summarized system matrices $\mathcal{A}$ and $\mathcal{B}$ are defined by

$$\mathcal{A} = \begin{bmatrix} \mathbf{A}_0 \\ \prod_{l=0}^{1} \mathbf{A}_{1-l} \\ \vdots \\ \prod_{l=0}^{N-1} \mathbf{A}_{N-1-l} \end{bmatrix}, \quad \mathcal{B} = \begin{bmatrix} \mathbf{b}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{A}_1\mathbf{b}_0 & \mathbf{b}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \left(\prod_{l=1}^{N-1} \mathbf{A}_{N+0-l}\right)\mathbf{b}_0 & \cdots & \mathbf{A}_{N-1}\mathbf{b}_{N-2} & \mathbf{b}_{N-1} \end{bmatrix}, \tag{3.15}$$

and $\mathcal{C}$ is a block-diagonal matrix involving N times the output matrix $\mathbf{C}$. Furthermore, the matrices $\mathcal{D}$ and $\mathcal{E}$ can be determined equivalently to $\mathcal{B}$ and $\mathcal{C}$. Accordingly, the objective function (3.7) is summed up for all iterations k over the entire optimization horizon N. Given the input vector sequence $\mathbf{u}$, the state vector sequence $\mathbf{X}$ and the sequence $\mathbf{X}_\mathrm{d}$ of desired values for $\mathbf{x}_{k,\mathrm{d}}$, which is defined analogously to (3.9), the objective function becomes

$$J = \left[\mathbf{X} - \mathbf{X}_\mathrm{d}\right]^\top \mathcal{Q} \left[\mathbf{X} - \mathbf{X}_\mathrm{d}\right] + \mathbf{u}^\top \mathcal{R} \mathbf{u}, \tag{3.16}$$

with $\mathcal{Q} = \mathrm{diag}(\mathbf{Q}_1, \dots, \mathbf{Q}_N)$ and $\mathcal{R} = \mathrm{diag}(r_0, \dots, r_{N-1})$. The compact description of the system dynamics (3.14), which represents the entire future system state as a function of the sequence of the system input u_k to be optimized, can be easily integrated into the objective function (3.16) by inserting (3.14a), so that a static optimization problem arises in which the optimization variables are represented by the vector sequence $\mathbf{u}$. Accordingly, the objective function results in

$$J = \mathbf{u}^\top \mathbf{H} \mathbf{u} + \mathbf{F} \mathbf{u} + \mathrm{o}, \tag{3.17}$$

with $\mathbf{F} = 2\,[\mathcal{A}\mathbf{x}_0 + \mathcal{E}\mathbf{z} - \mathbf{X}_\mathrm{d}]^\top \mathbf{G}$, $\mathbf{G} = \mathcal{Q}\mathcal{B}$ and $\mathbf{H} = \mathcal{B}^\top \mathbf{G} + \mathcal{R}$. The term o represents a constant that is independent of $\mathbf{u}$ and, hence, has no influence on the optimal solution. Equivalently, the output constraints in (3.8b) are reformulated as a function of the optimization vector $\mathbf{u}$

$$\begin{array}{rl} \mathbf{Y} \leq \mathbf{Y}_\mathrm{max} \Rightarrow & \mathcal{C}\mathcal{B}\mathbf{u} \leq \mathbf{Y}_\mathrm{max} - \mathcal{C}\mathcal{A}\mathbf{x}_0 - \mathcal{C}\mathcal{D}\tilde{\mathbf{z}} - \mathcal{E}\mathbf{z} \\ \mathbf{Y} \geq \mathbf{Y}_\mathrm{min} \Rightarrow & \mathcal{C}\mathcal{B}\mathbf{u} \geq \mathbf{Y}_\mathrm{min} - \mathcal{C}\mathcal{A}\mathbf{x}_0 - \mathcal{C}\mathcal{D}\tilde{\mathbf{z}} - \mathcal{E}\mathbf{z} \end{array} \tag{3.18}$$

by inserting (3.14a) and (3.14b), and the corresponding vector sequences for $\mathbf{Y}_\mathrm{min}$, $\mathbf{Y}_\mathrm{max}$ [59]. Both the input and output constraints (3.8) can be compactly represented by

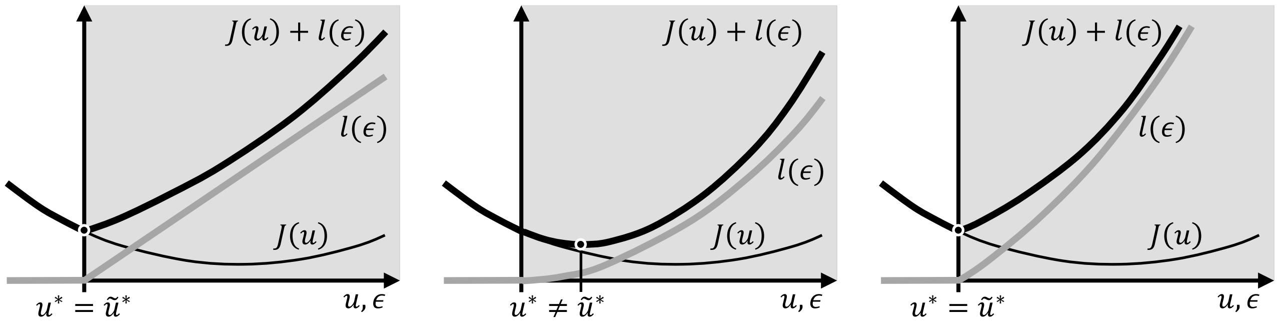
$$\underbrace{\begin{bmatrix} \mathcal{C}\mathcal{B} \\ \mathbf{I}_N \\ -\mathcal{C}\mathcal{B} \\ -\mathbf{I}_N \end{bmatrix}}_{\mathbf{A}_\mathrm{C}} \mathbf{u} \leq \underbrace{\begin{bmatrix} \mathbf{Y}_\mathrm{max} - \mathcal{C}\mathcal{A}\mathbf{x}_0 - \mathcal{C}\mathcal{D}\tilde{\mathbf{z}} - \mathcal{E}\mathbf{z} \\ \mathbf{u}_\mathrm{max} \\ -\mathbf{Y}_\mathrm{max} + \mathcal{C}\mathcal{A}\mathbf{x}_0 + \mathcal{C}\mathcal{D}\tilde{\mathbf{z}} + \mathcal{E}\mathbf{z} \\ -\mathbf{u}_\mathrm{min} \end{bmatrix}}_{\mathbf{b}_\mathrm{C}} \tag{3.19}$$

as a function of the optimization vector $\mathbf{u}$. The objective function (3.17) in combination with the linear inequality constraints (3.19) represents a quadratic program which can be solved efficiently using static optimization methods, cf. [13], [58], [89] and [98].

3.4 Constraint Softening: Quadratic Program Extension through Slack-Variables

When formulating the trajectory optimization problem, physical constraints must be taken into account in order to meet the requirements that arise for automated driving. To avoid infeasibility issues that can be caused by the constraints, cf. [59], and to take comfort for passengers into account, a relaxation is introduced in the form of slack variables. The basic idea is to consider the violation of the constraints as an additional cost $l(\epsilon)$ in the optimization. It is mandatory to formulate sufficiently high costs for the violation of constraints without negatively changing the optimal solution u^* of the original optimization problem. This can be achieved using a linear-quadratic formulation of the additional costs, as shown exemplarily in Fig. 3.2 for the scalar case $r = 1$. By choosing a sufficiently large linear gain factor, it can be ensured that the total costs in the event of constraint violation, i.e. in the impermissible range, are always greater than in the permissible range. Furthermore, the quadratic term enables the violation of constraints to be punished more severely. However, a single quadratic cost formulation may lead to a shift of the optimal solution into the impermissible range in some cases, see Fig. 3.2(b).

By including r independent slack variables $\epsilon = [\epsilon_1, \ldots, \epsilon_r]^\top$ into the inequality constraints (3.8) and the optimization vector $\tilde{\mathbf{u}} = [\mathbf{u}^\top, \epsilon^\top]^\top$, the optimal trajectory is obtained as the



(a) $l(\epsilon) = k_1\epsilon$ **(b)** $l(\epsilon) = k_2\epsilon^2$ **(c)** $l(\epsilon) = k_1\epsilon + k_2\epsilon^2$

Figure 3.2: Comparison of the total costs with different additional cost terms $l(\epsilon)$, adapted according to [131].

trade-off between the violation of constraints and the solvability of the optimization problem using the additional linear-quadratic cost function

$$l(\boldsymbol{\epsilon}) = \mathbf{k}^\top \boldsymbol{\epsilon} + \boldsymbol{\epsilon}^\top \mathbf{K}\, \boldsymbol{\epsilon}\,. \tag{3.20}$$

The additional cost function $l(\boldsymbol{\epsilon})$ is parametrized by $\mathbf{k} = [k_{11}, \dots, k_{1r}]^\top$ and $\mathbf{K} = \operatorname{diag}(k_{21}, \dots, k_{2r})$ with $k_{11,\dots,2r} > 0$. As a result, the objective function (3.17) of the quadratic program is reformulated as

$$\tilde{J} = \tilde{\mathbf{u}}^\top \tilde{\mathbf{H}} \tilde{\mathbf{u}} + \tilde{\mathbf{F}} \tilde{\mathbf{u}} + \mathrm{o} \tag{3.21}$$

with $\tilde{\mathbf{H}} = \operatorname{diag}(\mathbf{H}, \mathbf{K})$ and $\tilde{\mathbf{F}} = \left[2\left[\mathcal{A}\mathbf{x}_0 + \mathcal{E}\mathbf{z} - \mathbf{X}_\mathrm{d}\right]^\top \mathbf{G},\, \mathbf{k}^\top\right]$.

It should be noted that with the expansion of the optimization vector $\tilde{\mathbf{u}}$, the computational effort required for the solution increases with the number of slack variables r. Secondary literature suggests using solely one single slack variable for all inequality constraints in order to guarantee the solvability of the optimization problem in the process of efficient solution finding , c.f. [57] and [86]. In contrast, the present work uses special properties of quadratic programming in combination with static optimization methods in order to solve the optimization problem more efficiently even with several independent slack variables. In addition to ensuring solvability, the use of several independent slack variables offers the possibility of

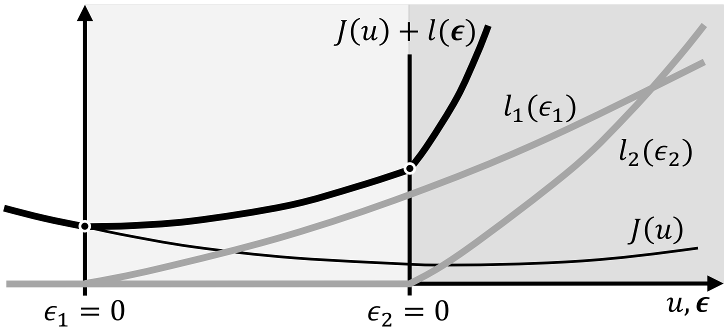


Figure 3.3: Multi-stage increase in total costs with two independent slack variables ϵ_1 and ϵ_2, adapted according to [131].

taking comfort aspects for passengers during the dynamic driving task into account. The violation of various constraints can thus be viewed independently. The different parameterization in $\mathbf{k}$ and $\mathbf{K}$ allows the violation of certain constraints to be prioritized differently, so that a violation of comfort-related constraints can be preferred to the violation of collision-relevant, i.e. safety-related constraints. This results in a multi-stage increase in total costs, as shown in Fig. 3.3 for the case $r = 2$. In order to generate trajectories that are feasible in terms of driving physics, however, there exist constraints that must not be softened. Therefore, the selective use of slack variables for certain constraints takes place by separating those constraints for the outputs in $\mathbf{y}$ and input u that should be softened and those constraints that slack variables should not affect. By using the matrix $\mathbf{S}^y$ and the scalar s^u, those outputs or the input are selected, for which slack variables are defined. Further outputs or the input, which should remain unaffected by the slack variables, are then taken into account using the matrix $\mathbf{H}^y$ and h^u. Accordingly, the vector sequences $\mathbf{u}$ and $\mathbf{Y}$ are divided into

$$\begin{aligned} \mathbf{u}_s &= \mathcal{S}^u \mathbf{u}; & \mathcal{S}^u &= \operatorname{diag}(\mathrm{s}_1^u, \ldots, \mathrm{s}_N^u), \\ \mathbf{Y}_s &= \mathcal{S}^y \mathbf{Y}; & \mathcal{S}^y &= \operatorname{diag}(\mathbf{S}_1^y, \ldots, \mathbf{S}_N^y), \\ \mathbf{u}_h &= \mathcal{H}^u \mathbf{u}; & \mathcal{H}^u &= \operatorname{diag}(\mathrm{h}_1^u, \ldots, \mathrm{h}_N^u), \\ \mathbf{Y}_h &= \mathcal{H}^y \mathbf{Y}; & \mathcal{H}^y &= \operatorname{diag}(\mathbf{H}_1^y, \ldots, \mathbf{H}_N^y), \end{aligned} \tag{3.22}$$

where all constraints are summarized as follows

$$\underbrace{\begin{bmatrix} \begin{bmatrix} \mathcal{H}^y \mathbf{A}'_\mathrm{C} \\ \mathcal{H}^u \\ -\mathcal{H}^y \mathbf{A}'_\mathrm{C} \\ -\mathcal{H}^u \end{bmatrix} & \mathbf{0}_{2n_h \times r} \\ \begin{bmatrix} \mathcal{S}^y \mathbf{A}'_\mathrm{C} \\ \mathcal{S}^u \\ -\mathcal{S}^y \mathbf{A}'_\mathrm{C} \\ -\mathcal{S}^u \end{bmatrix} & -\mathbf{\Sigma}_{2n_s N \times r} \\ \mathbf{0}_{r \times N} & -\mathbf{I}_r \end{bmatrix}}_{\tilde{\mathbf{A}}_\mathrm{C}} \tilde{\mathbf{u}} \leq \underbrace{\begin{bmatrix} \begin{bmatrix} \mathcal{H}^y \left[\mathbf{Y}_\mathrm{max} - \mathbf{B}'_\mathrm{C}\right] \\ \mathcal{H}^u \mathbf{u}_\mathrm{max} \\ -\mathcal{H}^y \left[\mathbf{Y}_\mathrm{min} - \mathbf{B}'_\mathrm{C}\right] \\ -\mathcal{H}^u \mathbf{u}_\mathrm{min} \end{bmatrix} \\ \begin{bmatrix} \mathcal{S}^y \left[\mathbf{Y}_\mathrm{max} - \mathbf{B}'_\mathrm{C}\right] \\ \mathcal{S}^u \mathbf{u}_\mathrm{max} \\ -\mathcal{S}^y \left[\mathbf{Y}_\mathrm{min} - \mathbf{B}'_\mathrm{C}\right] \\ -\mathcal{S}^u \mathbf{u}_\mathrm{min} \end{bmatrix} \\ \mathbf{0}_r \end{bmatrix}}_{\tilde{\mathbf{b}}_\mathrm{C}}, \tag{3.23}$$

with $\mathbf{A}'_\mathrm{C} = \mathcal{C}\mathcal{B}$ and $\mathbf{B}'_\mathrm{C} = \mathcal{C}\mathcal{A}\mathbf{x}_0 + \mathcal{C}\mathcal{D}\tilde{\mathbf{z}} + \mathcal{E}\mathbf{z}$. The matrix $\mathbf{\Sigma}$ links the defined slack variables and the constraints that are to be softened. The objective function (3.21) expanded by slack variables in combination with the modified linear inequality constraints (3.23) remains a quadratic program that can be efficiently solved using static optimization methods despite several independent slack variables.

3.5 Longitudinal Trajectory Optimization

As derived in Section 3.1, the vehicle longitudinal dynamics can be sufficiently described along the reference path Γ in the Frenét frame using a one-dimensional motion. The system dynamics in the longitudinal direction can consequently be efficiently formulated as a time-invariant integrator system

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t) \tag{3.24}$$

with the state vector $\mathbf{x} = [s, v, a, j]^\top$ and input $u = \ddot{a}$. According to (3.1a), $s(t)$ defines the distance, $v(t)$ the vehicle velocity and $a(t)$ the vehicle acceleration along the reference path Γ. Moreover, j represents the jerk in the longitudinal direction. In order to ensure a continuously differentiable acceleration for reasons of comfort despite the necessary time-discretization of the system dynamics, a fourth-order representation for the longitudinal dynamics is chosen in this work. By defining the system outputs

$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}(t), \tag{3.25}$$

collision-relevant restrictions and physical driving limitations of the longitudinal trajectory can be taken into account as linear inequality constraints with respect to $s(t)$ and $a(t)$ in the optimization.

According to the objective function (3.7), the weightings

$$\mathbf{Q}_k = \mathrm{diag}(w_s, w_v, w_a, w_j) \quad \text{and} \quad r_k = w_u, \tag{3.26}$$

including the weighting factors $w_s, w_v, w_a, w_j, w_u > 0$, allow desired trajectory profiles to be defined, taking into account motion specifications with respect to the states of the system dynamics (3.24) as a square in the objective function. The factors w_s, w_v, w_a serve, for instance, to address deviations from a desired motion $\mathbf{x}_{k,\mathrm{d}}$, in particular a desired vehicle velocity. In contrast, the terms w_j and w_u inhibit any change in the driving velocity in order to avoid jerky accelerations caused by sudden changes in the velocity specifications.

3.5.1 Safety and Comfort-Related Constraints

For the planning of safe and comfortable longitudinal trajectories, this section formulates the constraints w.r.t. the previously defined system outputs (3.25). The constraints are derived from the given motion specifications and the prevailing traction conditions. In this way, collisions with obstacles can be avoided during future vehicle motions and physical driving limitations can be observed.

In order to comply with physical driving limitations and thus to ensure feasible longitudinal vehicle motions, constraints that result from the maximum acceleration and deceleration potential are particularly important [58]. Depending on the prevailing tire grip, which varies, for instance, with weather or road conditions, the traction condition is accounted for by time-variant inequality constraints

$$a_{\min}(k) \leq a(k) \leq a_{\max}(k) \tag{3.27}$$

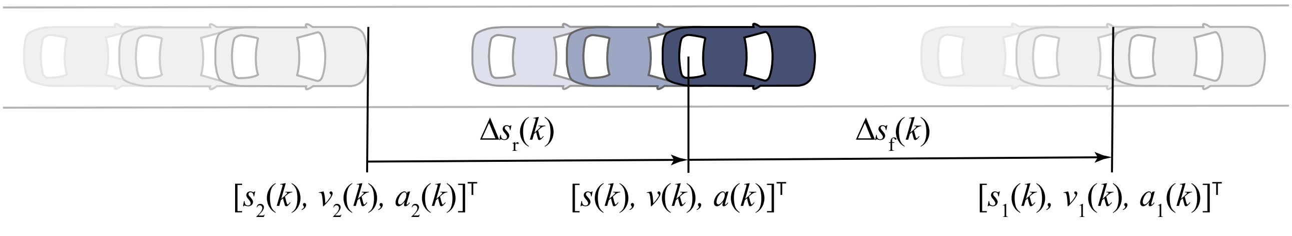


Figure 3.4: Constrained longitudinal position $s(k)$ of the ego vehicle (dark blue) at time step k, adapted according to [118]

as acceleration restrictions in the trajectory optimization.

In addition to taking into account physical driving limitations, the formulation of time-variant position constraints enables collision-relevant obstacles to be taken into account in the motion planning. As shown schematically in Fig. 3.4, the predicted positions of other road participants at future time steps k limit the longitudinal position $s(k)$ of the ego vehicle at the corresponding time steps. In the case of abrupt movements of the other road participants or sensor noise for instance, the definition of a hard constraints of the longitudinal position $s(k)$ may render the optimization problem unsolvable [118]. Therefore, the longitudinal position restrictions for the ego vehicle are taken into account using soft constraints.

In the literature, as for example in [57] or [86], a single slack variable is usually applied for a soft constraint on a safety distance. On the contrary, the optimization formulation in this work accounts for a two-stage longitudinal distance constraint within the optimization horizon N

$$\begin{aligned} s_2(k) + \Delta s_{\mathrm{r,comf}}(v_2(k)) - \epsilon_{\mathrm{comf}} \leq s(k) \leq s_1(k) - \Delta s_{\mathrm{f,comf}}(v_1(k)) + \epsilon_{\mathrm{comf}} \\ s_2(k) + \Delta s_{\mathrm{r,safe}}(v_2(k)) - \epsilon \leq s(k) \leq s_1(k) - \Delta s_{\mathrm{f,safe}}(v_1(k)) + \epsilon \end{aligned} \tag{3.28}$$

for reasons of comfort. The time-variant terms

$$\begin{aligned} \Delta s_{\mathrm{f,comf}}(v_1(k)) &= \Delta s_{\mathrm{f,obs,comf}}(v_1(k)) + l_{\mathrm{f,bumper}} \\ \Delta s_{\mathrm{f,safe}}(v_1(k)) &= \Delta s_{\mathrm{f,obs,safe}}(v_1(k)) + l_{\mathrm{f,bumper}} \\ \Delta s_{\mathrm{r,comf}}(v_2(k)) &= \Delta s_{\mathrm{r,obs,comf}}(v_2(k)) + l_{\mathrm{r,bumper}} \\ \Delta s_{\mathrm{r,safe}}(v_2(k)) &= \Delta s_{\mathrm{r,obs,safe}}(v_2(k)) + l_{\mathrm{r,bumper}} \end{aligned} \tag{3.29}$$

are modeled as a function of comfort and safety-related obstacle distances

$$\begin{aligned} \Delta s_{\mathrm{obs,comf}}(v_p(k)) &= \max\left(v_p(k) \cdot t_{\mathrm{h,comf}}\,,\ \Delta\tilde{s}_{\mathrm{obs,comf}}\right) \\ \Delta s_{\mathrm{obs,safe}}(v_p(k)) &= \max\left(v_p(k) \cdot t_{\mathrm{h,safe}}\,,\ \Delta\tilde{s}_{\mathrm{obs,safe}}\right)\,, \end{aligned} \tag{3.30}$$

which depend on the predicted velocities of the corresponding road participants $v_p(k)$ as well as the constant comfort and safety-related time headways ($t_{\mathrm{h,comf}}$ and $t_{\mathrm{h,safe}}$). In the case of a crossing road participant or a static obstacle, $v_p(k)$ becomes 0. The terms $\tilde{s}_{\mathrm{obs,comf}}$ and $\tilde{s}_{\mathrm{obs,safe}}$ define a minimum distance to obstacles for comfort and safety. The constant terms $l_{\mathrm{f,bumper}}$ and $l_{\mathrm{r,bumper}}$ define the length from the vehicle rear axle to the front bumper and rear bumper, respectively. To ensure the solvability of the quadratic program, the safety distances $\Delta s_{\mathrm{f,safe}}$ and $\Delta s_{\mathrm{r,safe}}$ are linked to the heavily weighted slack variable ϵ. For reasons of comfort, however, the longitudinal comfort distances $\Delta s_{\mathrm{f,comf}}$ and $\Delta s_{\mathrm{r,comf}}$ are linked to the less weighted slack variable ϵ_{comf}. Due to the two-stage longitudinal distance constraint, a compromise between a comfortable distance to an obstacle and a smooth acceleration or deceleration profile in a dynamic maneuver can be achieved through a suitable choice of the weighting of the comfort slack variable ϵ_{comf} [118].

By introducing the independent slack variables ϵ and ϵ_{comf}, the matrices and scalars introduced in Section 3.4 are parametrized by

$$\begin{aligned} \mathrm{s}^u &= 0 \qquad\qquad \mathrm{h}^u = 0 \\ \mathbf{S}^y &= \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \quad \mathbf{H}^y = \begin{bmatrix} 0 & 1 \end{bmatrix} \\ \mathbf{\Sigma}_{4N\times 2} &= \left[\mathbf{I}_2 \ldots \mathbf{I}_2\right]^\top , \end{aligned} \tag{3.31}$$

where $r = 2$ applies.

3.6 Lateral Trajectory Optimization

In contrast to the longitudinal dynamics, which can be adequately described by time-invariant system equations of a one-dimensional motion, there are various opportunities in the Frenét frame to formulate the lateral vehicle dynamics under the assumption of linear system models [59], [131]. The system models differ primarily in the physical aspects taken into account and the choice of the system input.

Based on the relationships shown in Fig. 3.1 between the orientation θ_Γ and the curvature κ_Γ of the reference path as well as the vehicle states of the lateral distance d, the orientation θ and the curvature κ along the longitudinal position s, the kinematic equation (3.2b) is expanded to include higher system states for the formulation of the system model of the lateral motion. As a result, collision-relevant restrictions and physical driving limitations are taken into account. For this purpose, the center of the vehicle rear axle is defined as the reference point, so that the side slip angle can be neglected [107]. As a result, the derived assumptions for the linearization of the motion equations show negligible effects on the motion planning, c.f. Section 3.1.

Whereas the authors in [59], for instance, use the reference path orientation as a state of their system model, in this work, the reference path orientation is defined as a disturbance term $z(t) = \theta_\Gamma$. By choosing the first time derivative of the curvature $\dot{\kappa}$ as a system state and its second time derivative as the system input $u(t) = \ddot{\kappa}$, a fourth order system is modeled, which ensures a continuously differentiable steering angle for comfort reasons despite the necessary time-discretization of the system dynamics.

By using the velocity profile $v(t)$ generated by the longitudinal trajectory optimization as a time-variant parameter, the linear system dynamics for the lateral motion are modelled as follows

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & v(t) & 0 & 0 \\ 0 & 0 & v(t) & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t) + \begin{bmatrix} -v(t) \\ 0 \\ 0 \\ 0 \end{bmatrix} z(t) , \tag{3.32}$$

with the state vector $\mathbf{x} = [d, \theta, \kappa, \dot{\kappa}]^\top$.

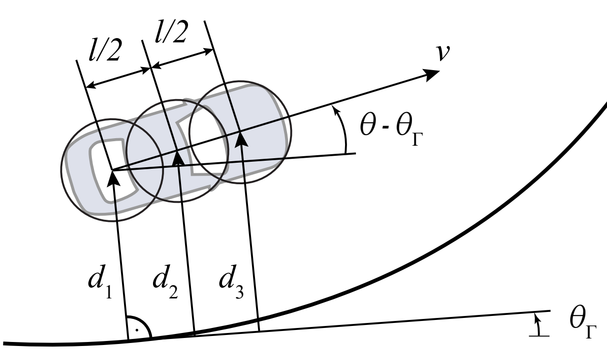


Figure 3.5: Representation of the system outputs for lateral collision-avoidance.

To formulate collision-relevant constraints, as shown in Fig. 3.5, the vehicle geometry is approximated by three circles, which are positioned along the longitudinal axis. The circle center positions are defined in relation to the rear axle center using the distances $l_1 = 0$, $l_2 = l/2$, $l_3 = l$ based on the wheelbase l, so that their distances to the reference path are calculated – analogously to the approximation of the lateral distance d – using

$$d_i = d + l_i \sin(\theta - \theta_\Gamma) \approx d + l_i(\theta - \theta_\Gamma), \quad i = 1, 2, 3\,. \tag{3.33}$$

In combination with the curvature κ to take physical driving limitations into account, the system output is defined by

$$\mathbf{y}(t) = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \kappa \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}l & 0 & 0 \\ 0 & l & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ -\frac{1}{2}l \\ -l \\ 0 \end{bmatrix} z(t)\,. \tag{3.34}$$

For reasons of comfort for the passengers, a natural driving behavior is desirable. The driving behavior of human drivers is essentially characterized as a compromise between driving in the middle of the lane and minimizing the lateral vehicle accelerations and the lateral jerk. This is achieved, according to (3.7), using the weightings

$$\mathbf{Q}_k = \mathrm{diag}(w_d, w_{(\theta-\theta_\Gamma)}, w_\kappa, w_{\dot{\kappa}}) \quad \text{and} \quad r_k = w_u\,, \tag{3.35}$$

including the weighting factors $w_d, w_{(\theta-\theta_\Gamma)}, w_\kappa, w_{\dot{\kappa}}, w_u > 0$. While the terms w_d and $w_{(\theta-\theta_\Gamma)}$ address the avoidance of any deviations from the reference path Γ, the terms w_κ, $w_{\dot{\kappa}}$ and w_u inhibit any changes in direction by minimizing the curvature and its derivatives, so that the vehicle motion is planned as smooth as possible.

3.6.1 Safety and Comfort-Related Constraints

In order to ensure feasible lateral vehicle motions, physical driving limitations must be taken into account, which result mainly from mechanical properties of the steering actuator at low driving speeds in teleoperation missions. The physical limitations due to the maximum steering angle δ are accounted for by time-invariant inequality constraints

$$\kappa_{\min,\delta} \leq \kappa(k) \leq \kappa_{\max,\delta}\,, \tag{3.36}$$

as curvature restrictions in the trajectory optimization. Due to the low lateral dynamics in the selected experiments, the limited steering acceleration and steering speed of the actuator are not discussed further in this thesis. A consideration of these, however, is possible by constraining $\dot{\kappa}$ and the system input $u = \ddot{\kappa}$, respectively.

In addition to the limitations of the driving physics, collision-relevant constraints are required to generate safe lateral trajectories. Other road participants and static obstacles have a significant impact on the planning of future lateral vehicle motions. To avoid collisions, as shown in Fig. 3.6, the driving space can be constrained by restricting the right and left side of the reference path Γ so that the positions of dynamic and static obstacles are taken into account in the lateral trajectory optimization. In the case of abrupt movements of other road participants or sensor noise for instance, the definition of hard constraints of the lateral distances $d_i(k)$ with $i = 1, 2, 3$ may render the optimization unsolvable [118]. Therefore, the lateral distance restrictions of the approximated body representation are taken into account using soft constraints. Analogously to the formulation of the longitudinal distance constraints, the lateral trajectory optimization accounts for two-stage soft constraints for the lateral distances along the vehicle's longitudinal position $s(k)$ aiming at both comfort and solvability:

$$\begin{gathered} d_{i,\text{min, comf}}(k) - \epsilon_{\text{comf}} \leq d_i(k) \leq d_{i,\text{max, comf}}(k) + \epsilon_{\text{comf}} \,, \\ d_{i,\text{min, safe}}(k) - \epsilon \leq d_i(k) \leq d_{i,\text{max, safe}}(k) + \epsilon \,, \\ \text{with } \; i = 1, 2, 3 \,. \end{gathered} \tag{3.37}$$

To ensure the solvability of the quadratic program, the safety distances $d_{i,\text{min,safe}}$ and $d_{i,\text{max,safe}}$ are linked to the heavily weighted slack variable ϵ. For reasons of comfort, however, the lateral comfort distances $d_{i,\text{min,comf}}$ and $d_{i,\text{max,comf}}$ are linked to the less weighted slack variable ϵ_{comf}. Due to the two-stage lateral distance constraint, a compromise between a comfortable distance to an obstacle and a smooth steering profile in a dynamic maneuver can be achieved through a suitable choice of the weighting of the comfort slack variable ϵ_{comf} [118].

By introducing the independent slack variables ϵ and ϵ_{comf}, the matrices and scalars introduced in Section 3.4 are parametrized by

$$\begin{aligned} &\mathrm{s}^u = 0 \quad \mathrm{h}^u = 0 && \mathbf{H}^y = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \\ &\mathbf{S}^y = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top && \mathbf{\Sigma}_{6N\times 2} = \begin{bmatrix} \mathbf{I}_2 \\ \vdots \\ \mathbf{I}_2 \end{bmatrix}, \end{aligned} \tag{3.38}$$

where $r = 2$ applies.

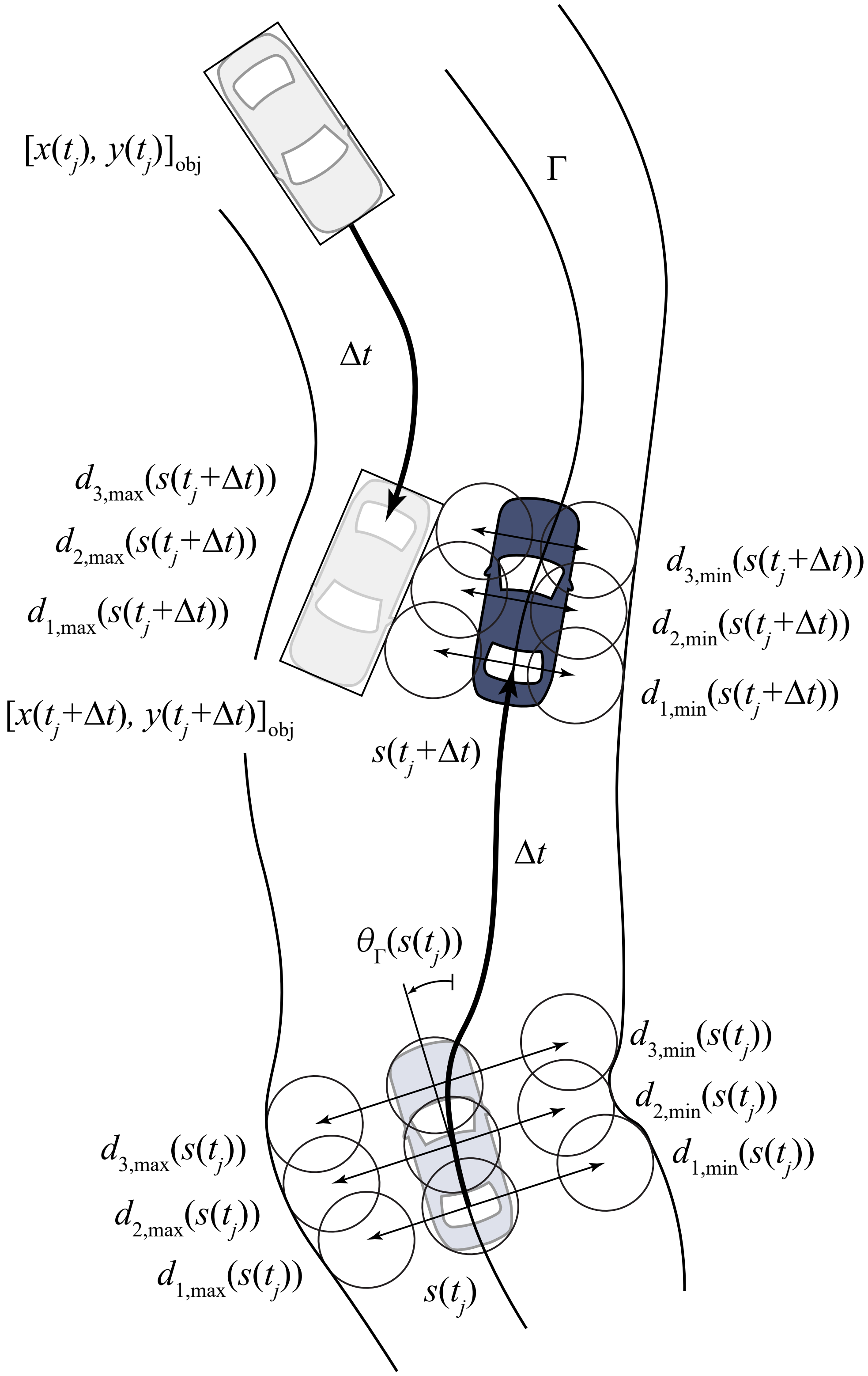


Figure 3.6: Lateral system constraints of the ego vehicle (blue) at time t_j and after the time interval Δt along the reference path Γ. Another road vehicle and its prediction are colored in gray and approximated by a rectangle, adapted according to [118].

Part II

Control Concepts for Teleoperated Driving Tasks in Urban Environments

4 Model-Predictive Cruise Control for Direct Teleoperated Driving Tasks

The following chapter presents an assistance concept for direct control in teleoperated driving tasks. The main content of the concept has been previously presented in [117].

4.1 Overview on the Approach

As described in Section 1.2.1, direct control represents a teleoperated driving paradigm with the lowest level of automation. The operator closes the control loop by directly providing operational commands to the automated vehicle based on the received sensor information about the teleoperation scenario. As can be seen in Fig. 4.1, the operational commands may be generated using a gamepad. Direct teleoperated driving offers several advantages by keeping the human in the closed control loop. Humans still outperform machines in terms of perception and processing tasks [93]. Consequently, situations may occur where the decision system of the AV reaches its limits and unforeseen traffic situations lead, in the worst case, to an accident. This can be the case of automated driving tasks on road particularities such as construction sites or in situations in which a policeman controls the traffic.

As mentioned in Section 1.3, two of the main challenges in teleoperated driving are insufficient remote situational awareness and the communication time delay. State-of-the-art direct control concepts to deal with these issues are visual assistance systems in the form of the

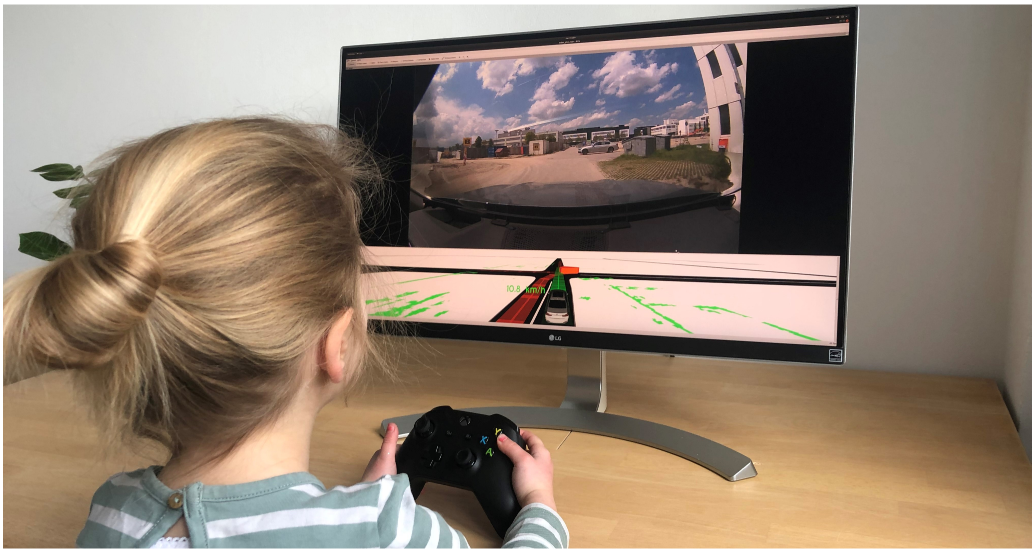


Figure 4.1: Image of the operator workplace: A human operator controls the vehicle using a gamepad.

Predictive Display, where the motions of the ego vehicle and other road users are predicted and visualized in the form of augmented information as rectangles in the camera images, c.f. Section 1.2.1. Any misinterpretation of these additional information, however, may have fatal consequences. Furthermore, due to the delayed transmission of camera images and operational commands, the operator may not have enough time to react appropriately to a sudden change in a traffic situation in a hazardous scenario. As investigated in [126], the reduced field of view due to the characteristics of cameras may lead to incorrect assessments of obstacle positions and thus to insufficient distances to vehicles in front. Although no empirical evidence exists in the literature, passengers complained about unnatural acceleration processes during experiments in scope of this work. All of these factors, which the human operator has to take into account and compensate for, can lead to cognitive overload and hence a performance degradation in complex teleoperation scenarios, as pointed out in [48].

A useful method to address the described issues and thus to increase safety and comfort for direct teleoperated driving, could be an on-board adaption of the operator's control commands. An ADAS known from conventional driving, is the ACC system, see Section 1.2.2. This system measures the distance to vehicles in front and alters the ego vehicle's dynamics in order to maintain a desired distance [137]. Latest ACC approaches, mainly based on a MPC framework such as in [2], [23] or [86], focus on minimizing fuel consumption and vehicle control errors. These systems are mainly designed for highway scenarios as they only consider vehicles in front. However, these approaches are not suited for application in cross traffic that occurs in urban scenarios.

For this purpose, the model-predictive cruise control for direct teleoperated driving tasks is presented in this work. This assistance concept uses the advantageous properties of the quadratic program for longitudinal trajectory optimization developed in Sec 3.5 in order to adapt on-board the operator's control commands for the longitudinal dynamics in real-time to address both safety and comfort. This concept aims in particular to support operators in critical situations, when they do not react appropriately in a driving maneuver, for example due to an incorrect assessment of the possible driving space or due to the delayed or even interrupted communication with the ego vehicle. Critical scenarios in which an operator has to react quickly and carefully can be caused by a sudden change in the traffic situation. Two critical scenarios, which are experimentally examined in more detail in the following section, are: a suddenly appearing road participant at an intersection with right of way and an abruptly braking vehicle in front.

4.2 Experimental Results

The effectiveness of the model-predictive cruise control approach is assessed by means of real-world teleoperated driving use cases. The experimental vehicle is equipped with a LiDAR as well as cameras for operator feedback, object prediction and environmental detection, respectively. For the optimization of the longitudinal trajectory a time step of 200 ms was chosen, so that the selection of $N = 30$ leads to an optimization time horizon of 6 s. For computational efficiency, the proposed approach is implemented in *C*++. The computations run on an Intel i9-9980XE 3 GHz processor with an average calculation time of 5 ms. For reasons of comfort, the maximum acceleration is limited to $a_{\max} = 2\ \mathrm{m/s^2}$. The experiments were carried out on a road with reduced traction conditions caused by gravel. The minimum acceleration is therefore limited to $a_{\text{min}} = -5.5\ \text{m/s}^2$. In order to be able to react comfortably to an abrubt braking of a preceding vehicle, it is recommended, according to [38], to adhere a distance in meters that corresponds to half the value of the current speed in kilometers per hour. For this purpose, the comfort related time headway is defined as $t_{\text{h,comf}} = 1.8$ s, c.f. equation (3.30). For a sufficient safety distance in critical situations, the safety related time headway is defined as $t_{\text{h,safe}} = 0.9$ s.

The optimized trajectory is passed to a stabilizing I/O-linearizing feedback controller which feeds its control signals to the vehicle's actuators. The design and implementation of the motion controller is out of the scope of this thesis.

4.2.1 Reaction to a Right-of-Way Vehicle at an Intersection

In the first scenario, as can be seen in Fig. 4.2, the operator with the ego vehicle wants to cross an intersection. Accordingly, the operator sets the desired acceleration of $a_{\text{d}} = 2\,\text{m/s}^2$ together with a desired maximum velocity of $v_{\text{d}} = 13.89\ \text{m/s} \approx 50\ \text{km/h}$ using a gamepad. Unfortunately, the operator does not react to a vehicle approaching from the right at the intersection. This can have several triggers in such a real-world teleoperation scenario. As can be seen in the image frame of the operator's camera view in Fig. 4.2 (a), the vehicle which has right of way is difficult to perceive, since it is partially obscured and moves behind a static object of a similar color. Another important challenge in teleoperated driving represents the unavoidable time delay in the mobile network connection. If this is comparatively high, as stated in Sect. 1.3, the operator may finally detect the vehicle approaching from the right too late. A manual intervention by the operator might be too late in both situations. The result can be an uncomfortable driving behavior or even a collision.

As can also be seen in Fig. 4.2 (a) in the virtual representation visualized using RViz[1], the object detection perceives the vehicle approaching from the right (orange cuboid) and predicts three possible maneuvers (orange rectangles). The implemented longitudinal trajectory optimization reflects this information and calculates a comfortable trajectory accordingly (green rectangles) down to a standstill in the selected optimization horizon while maintaining a comfort distance. The corresponding optimal trajectories in terms of the vehicle's velocity and the acceleration are depicted in Fig. 4.2 (a).

As the other road participant continues its motion, the longitudinal trajectory optimization generates a smooth acceleration profile in compliance with the desired acceleration, see Fig. 4.2 (b). The link to the video associated with the described scenario can be found in the footnote[2] below.

4.2.2 Reaction to a Suddenly Braking Vehicle in Front

In the second scenario, the operator with the ego vehicle follows a preceding vehicle. For this purpose, the operator leaves the setting of the appropriate vehicle's velocity and distance to the preceding vehicle to the proposed longitudinal trajectory optimization. Accordingly, the

---

[1] 3D visualization tool for Robot Operating System (ROS)

[2] Video of the experiment: https://youtu.be/wnPiXu6C9sg

operator specifies the desired maximum acceleration, i.e. $a_\mathrm{d} = 2\ \mathrm{m/s^2}$ together with a desired maximum velocity of $v_\mathrm{d} = 13.89\ \mathrm{m/s} \approx 50\ \mathrm{km/h}$ using the gamepad. As can be seen in Fig. 4.3 (a), the algorithm smoothly adjusts the velocity profile of the ego vehicle w.r.t. the one of the preceding vehicle. In this way, it is possible for the operator to maneuver the ego vehicle in flowing traffic comfortably while maintaining safety and comfort distance without much effort.

Even if the vehicle in front brakes abruptly, as can be seen in Fig. 4.3 (b), the longitudinal trajectory optimization algorithm generates a smooth trajectory until the ego vehicle comes to a standstill behind the vehicle in front. The compliance with the safety distance and the minimum permitted acceleration are also kept. In this situation as well, manual intervention by the operator might take place too late due to the time-delayed data transmission. Again, the consequences could be an uncomfortable driving behavior or even a collision.

As described in Section 3.5, the longitudinal safety distance constraint $\Delta s_\mathrm{f,obs,safe}$ is linked to the heavily weighted slack variable ϵ to ensure the solvability of the quadratic program. The longitudinal comfort distance constraint $\Delta s_\mathrm{f,obs,comf}$, which is related to the weakly weighted slack variable ϵ_comf, is used to provide a comfortable distance to the vehicle in front. A suitable choice of the weighting of the comfort slack variable ϵ_comf allows for an optimal compromise between a comfortable distance to the vehicle in front and a smooth deceleration process in such a abrupt braking scenario for instance, as depicted in Fig. 4.3 (b).

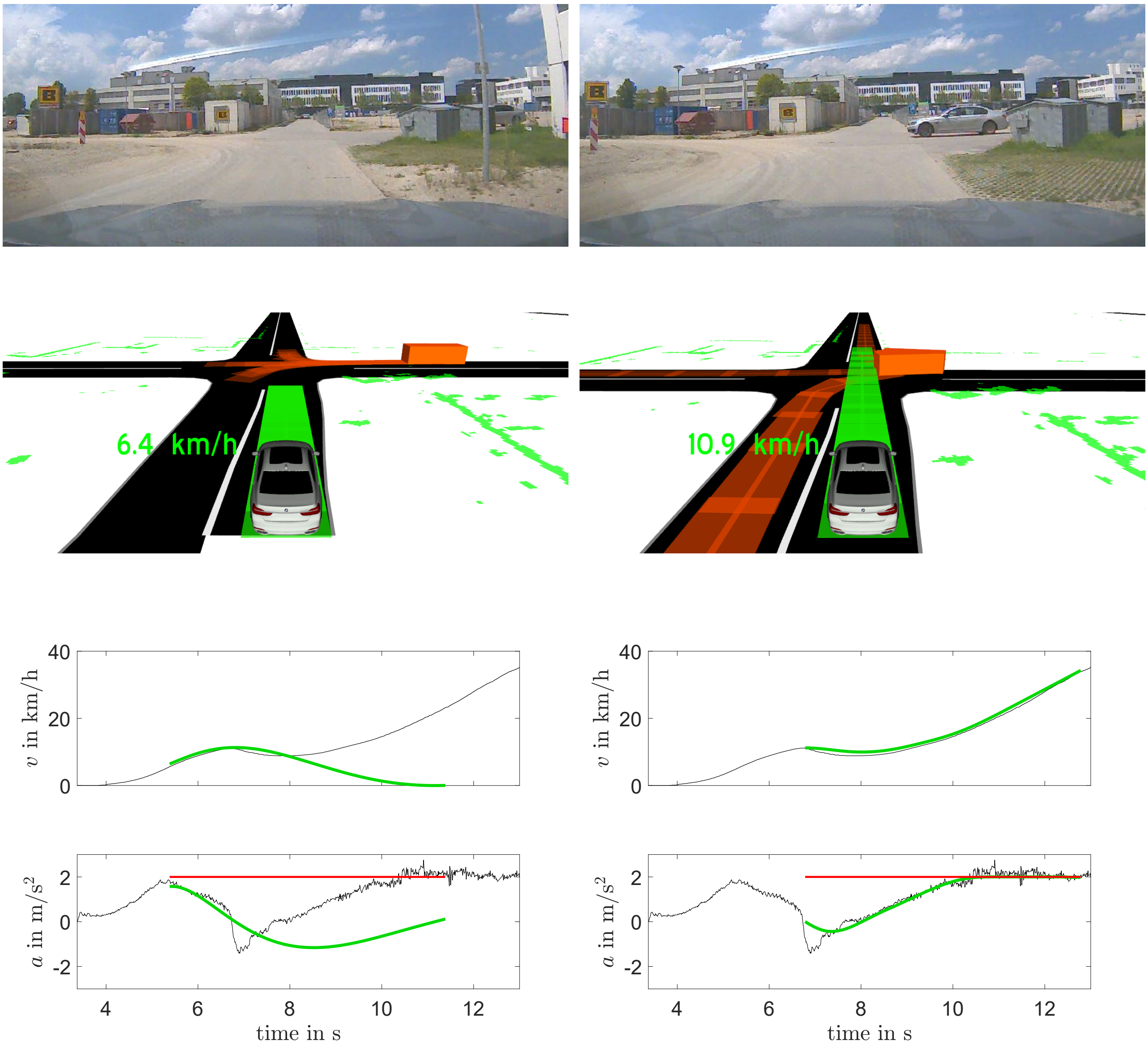
The link to the video associated with the described scenario can be found in the footnote[3] below.

4.3 Conclusion

This chapter presents an assistance system, the model-predictive cruise control for direct teleoperated driving tasks. This approach aims to relieve the human operator in complex teleoperation scenarios in which the human is directly involved in the closed control loop. The model-predictive cruise control approach is based on a novel linear-quadratic problem formulation for the vehicle longitudinal guidance, proposed in Chapter 3. This is efficiently solved by means of a time-variant, linear MPC scheme using Quadratic Programming while at the same time satisfying the requirements for comfort and safety, resulting from automated driving. In the work on ACC, [2], [23] or [57] for instance, the required computing time is not mentioned. However, the studies revealed clear advantages of the proposed longitudinal trajectory optimization in terms of computing effort in comparison to known approaches. Although a direct comparison of the computing effort is only possible qualitatively due to different hardware used, the trajectory optimization time of 100 - 200 ms in [133] or [141] in relation to only 5 ms in this work shows a significant increase in efficiency on a comparable processor.

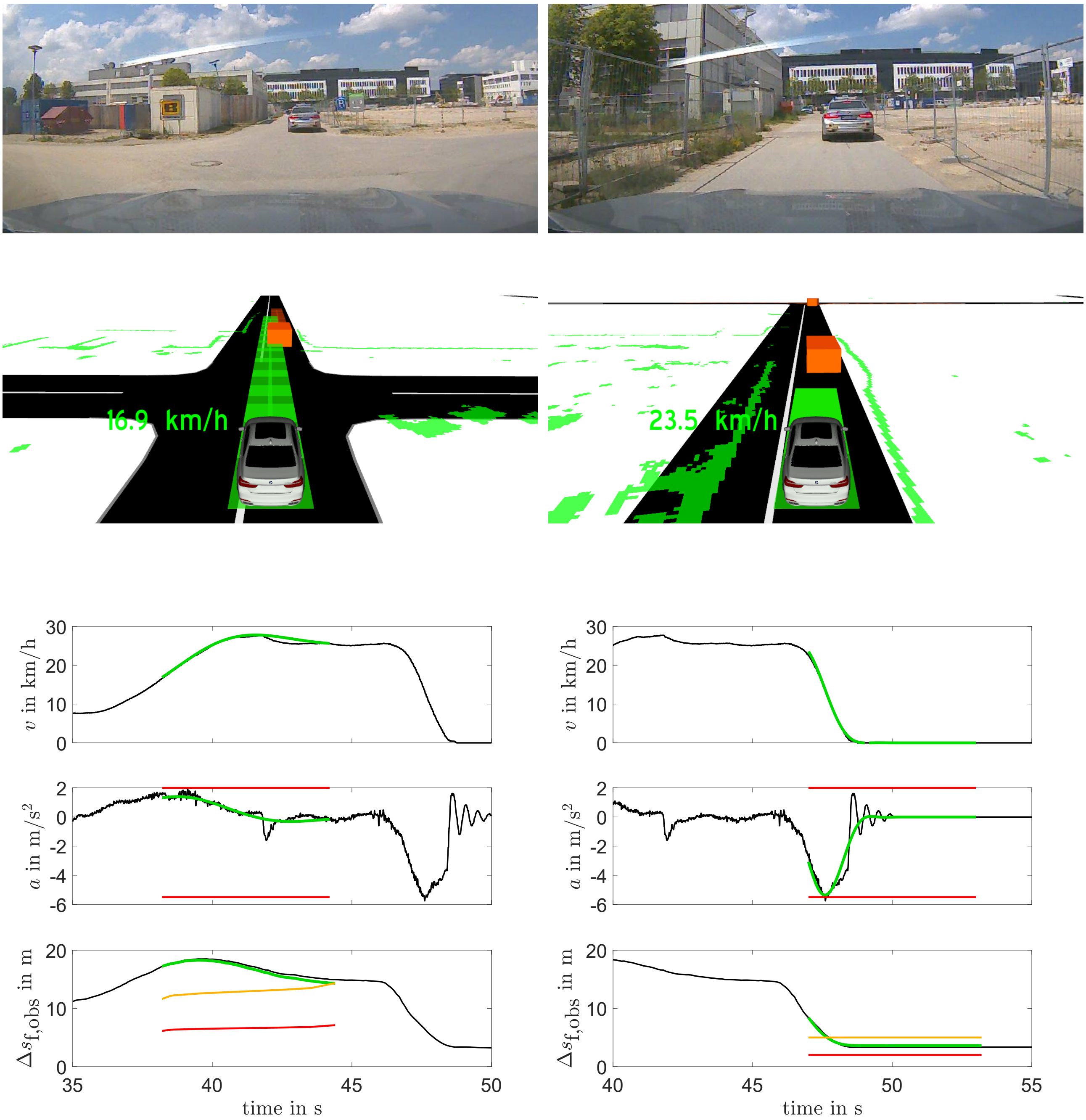
---

[3] Video of the experiment: https://youtu.be/f1u1g6O3JEc



(a) (b)

Figure 4.2: Reaction to a passing road participant at two consecutive times, where the graphs indicate: the optimal trajectory (in green), acceleration constraints (in red), and measurements (in black).



(a) (b)

Figure 4.3: Vehicle following with sudden braking maneuvers at two consecutive times, where the graphs indicate: the optimal trajectory (in green), comfort and safety constraints (in orange and red), and measurements (in black).

5 Corridor-Based Motion Planning for Shared Control

The following chapter presents a shared control concept for indirect teleoperated driving tasks. The main content of the concept has been previously presented in [116] and [118].

5.1 Overview on the Approach

In direct teleoperated driving, the human operator is kept directly in the vehicle's closed control loop. As described in Section 1.2.1 or Section 4.1, possible shortcoming w.r.t. machine perception for instance can be compensated for in this way. However, complex urban scenarios including high or variable communication time delays lead to a high workload to the operator as stated in Section 1.3. This forces the operator to drive more slowly and more carefully than with low communication time delays. As stated in Section 1.2.1, indirect control approaches has been introduced to overcome high and variable communication time delays. The operator is not part of the closed control loop. The guidance loop is closed autonomously by the vehicle. Therefore, the closed control loop is insensitive w.r.t any communication time delays. In shared control, a subclass of indirect control (see Section 1.2.1), the operator is only involved in planning tasks. The operator is enabled to define high level goals over the delayed communication channel that the robot executes on its own afterwards [120]. State-of-the-art shared control approaches are: trajectory-based control and waypoint-based control.

The trajectory-based control proposed in [45] proved to be easier for the operator when driving straight-line paths with communication time delays of over 200 ms compared to direct control. It turned out, however, that specifying appropriate trajectories in turning scenarios or for curved roads become highly challenging. Any attempt to generate trajectories manually in such scenarios resulted in an undesirable stop-and-go driving behavior. Issues regarding an incorrect assessment of distances to obstacles affect both, the trajectory-based and waypoint-based approach. Any misjudgement of the driving space may lead to collision-afflicted paths. Using waypoint-based control, no reliable statement can be made about possible collisions along the resulting path. Only a simulation that takes into account both the vehicle geometry and kinematics can clarify the situation. This approach is therefore time-consuming in complex urban scenarios, since it requires several attempts to set the waypoints properly until a feasible collision-free path is found. Even worse, both approaches are not suitable for dynamic environments including other road participants.

The work in [28] points out clearly that human strengths are related to cognitive tasks, like behavioural decision-making and situation analysis. After years of learning and gaining experience, see [46], humans manage to easily cope with difficult road topologies and confusing traffic situation. Automatized machines, however, have strong capabilities in vehicle stabilization and collision avoidance thanks to their precise sensor equipment. Furthermore, Automation systems are superior in coping with latency effects. Humans often solve delay-free tasks with cognitive challenges faster than automatized systems [124]. However, this relationship reverses with the presence of time delay. An optimal percentage of human tasks taken over by automatized systems, depending on the given time delay, leads to a minimum completion time.

A shared control approach that seeks to distribute human-machine tasks optimally is the corridor-based motion planning concept. This concept enables the operator to specify an area – the corridor – towards the desired destination. The automated vehicle has then to calculate a collision-free motion within the specified corridor. By specifying the corridor, the operator is able to interactively take into account, for instance, missing or inadequate lane markings or untracked obstacles, see Fig. 5.1. For this purpose, this concept is supported by both camera and LiDAR measurements. The operator is able to take advantage of the sensor measurements and to define the boundaries of the corridor according to the perception. The operator decides between specifying a complete corridor to the target destination in advance or initially a sub-corridor using the method described in Section 5.1.1. As the automated vehicle progresses within the sub-corridor, the operator is able to append further corridor segments. Within the specified corridor the automated vehicle calculates an optimal trajectory in real-time. For this purpose, a novel hybrid motion planning method is proposed, that determines, in the first phase, an optimal path for the static environment using the modified, two-step Constrained CHOMP algorithm, developed in Chapter 2. In the second phase, a collision-free trajectory is generated online along the optimal path using two separate linear-quadratic problem formulations for both, longitudinal and lateral dynamics. They are efficiently solved by means of a time-variant, linear MPC scheme using Quadratic Programming taking into account safety and comfort related requirements resulting from automated driving.

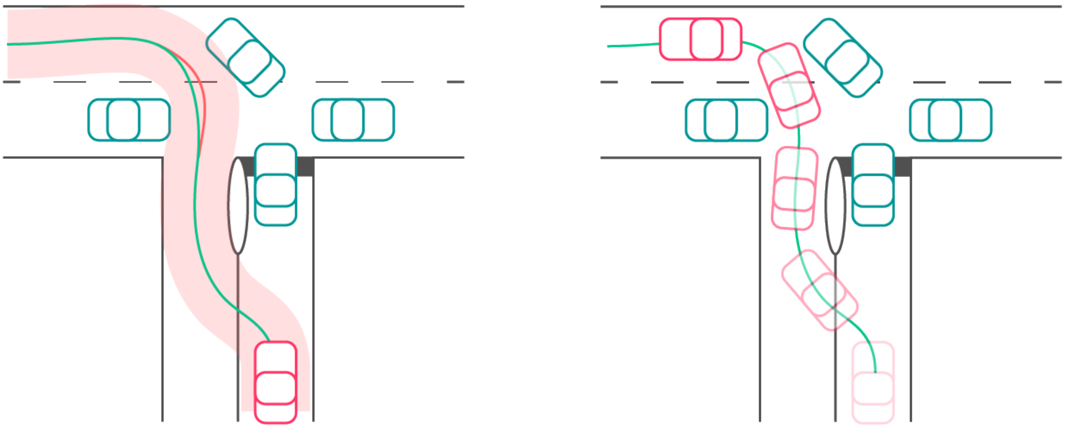


Figure 5.1: Illustration of the corridor-based control approach: (a) specified corridor with a collision-afflicted initial path (red) as well as the collision-free optimized path (green), (b) calculation and automated execution of the collision-free motion.

5.1.1 Corridor Specification

As described in the previous section, the corridor-based motion planning concept for teleoperated driving mainly involves two steps: In the first step, the operator specifies the corridor according the perceived sensor information. Subsequently, a motion planning algorithm generates an optimal motion which the vehicle executes on its own afterwards. The specification of the corridor is done by:

- a spline-based computation of an initial path using specified vehicle poses (x, y, θ) provided by the operator,
- a subsequent computation of the corridor boundaries using a predefined width and
- a optional customization of the boundaries by moving discrete points.

By using spline functions, polynomial curves are generated that interpolate between given poses $p_A = [x_A,\ y_A,\ \theta_A]$, $p_B = [x_B,\ y_B,\ \theta_B]$ with associated scalar curvatures κ_A, κ_B. In order to generate smooth paths, the authors in [111] suggest to take curvature derivatives into account. Therefore, a C^3-spline is required. For this purpose, a polynomial curve of degree seven as presented in [101] is employed in this thesis.

5.2 Experimental Results

The efficiency of the proposed corridor-based control approach is assessed by means of simulations as well as real-world teleoperated driving experiments. The experimental vehicle is equipped with a LiDAR as well as cameras for operator feedback, object prediction and environmental detection, respectively. For the optimization of the longitudinal dynamics as well as the lateral dynamics a time step of 200 ms was chosen so that the selection of $N = 30$ leads to an optimization time horizon of 6 s. For computational efficiency, the algorithms approach are implemented in C++. The computations run on an Intel i9-9980XE 3 GHz processor with an average calculation time per calculation step of 8 ms for the path optimization and 9 ms for the combined trajectory generation.

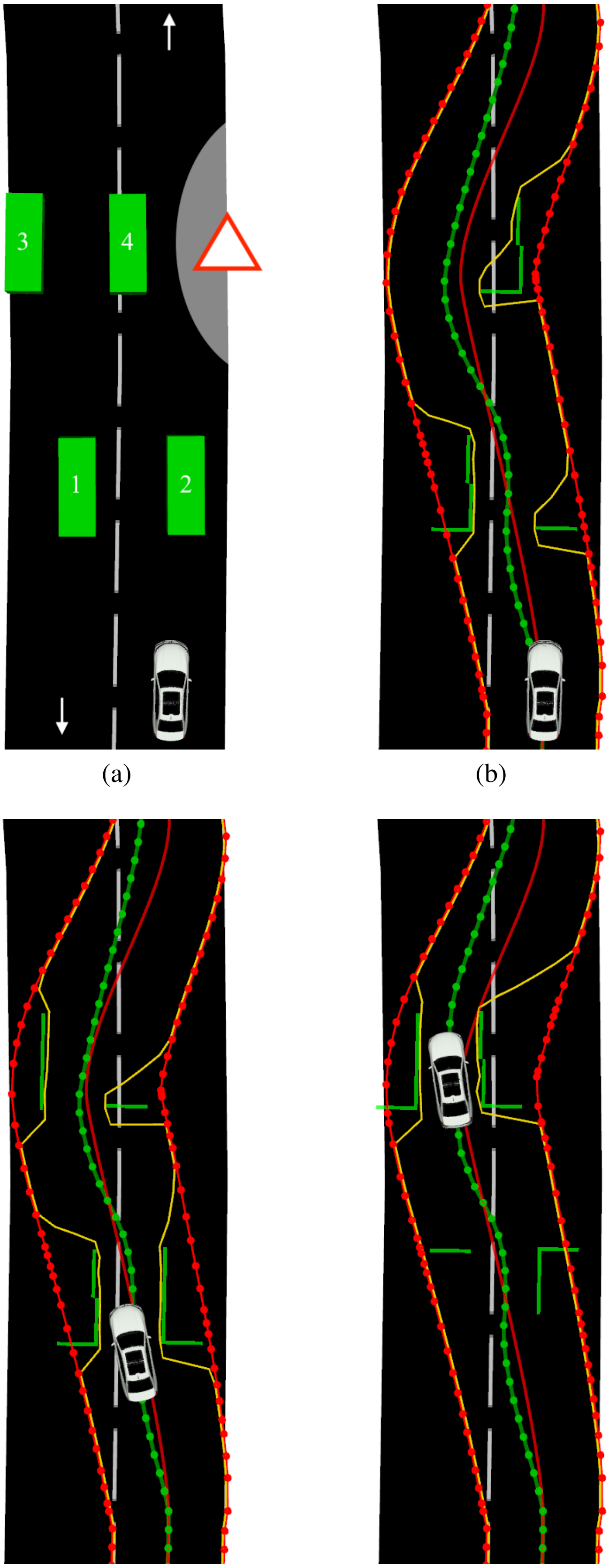
The optimized trajectory is passed to a stabilizing I/O-linearizing feedback controller which feeds its control signals to the vehicle's actuators. The design and implementation of the motion controller is out of the scope of this thesis.

5.2.1 Obstacle passing in a narrow, static Environment

First, simulations were carried out in a relatively complex road scenario with narrow distances to obstacles, created in a ROS[1] environment, in order to investigate the path planning of the corridor-based shared control approach. The scenario for this purpose including four obstacles is depicted in Fig. 5.2 (a). It can be observed that the ego vehicle is blocked by several obstacles along its road lane. The red triangle together with the white area represent a construction site in the lane. In Fig. 5.2 (a) the overall geometries of all obstacles are represented. However, due to the functional principle of a LiDAR sensor, such a view is not possible in reality. It becomes obvious that, depending on the specific poses of the vehicle,

---

[1] Robot Operating System: set of software libraries for robot software development



(a) (b)

(c) (d)

Figure 5.2: Simulation results: (a) road scenario with four obstacles, (b) - (d) progress of the vehicle along the optimized path (green dotted line) within the specified corridor (red dotted line) with its centre line (in red) and obstacle bounds (in yellow).

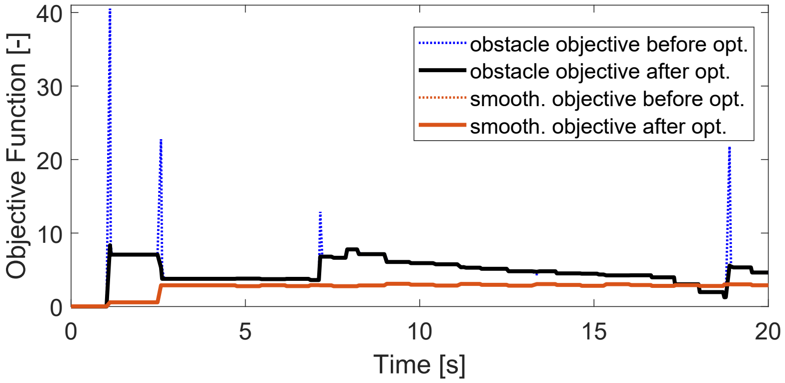


Figure 5.3: Evolution of objective functions over time belonging to the scenario in Fig. 5.2

obstacles are only partially scanned or completely covered by other obstacles during this scenario.

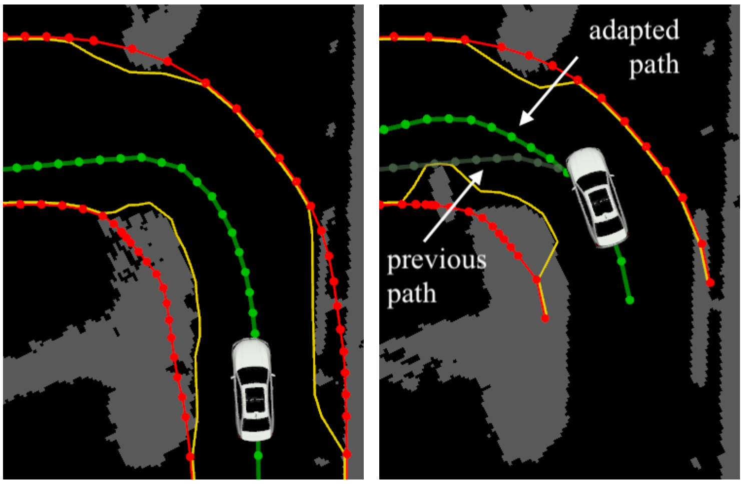
Using the method described in Section 5.1.1, the operator is able to specify a corridor according to the sensor perception. First, the operator places a pose to avoid obstacle 4, which is located at the road centre line next to the construction site. Subsequently, the operator specifies the pose where the ego vehicle should return to the original lane. As can be observed in the video[2], the operator is able to generate a plausible shape of the corridor according to the situation analysis using just a few actions, cf. Fig. 5.2 (b). However, the initial path, i.e. the centre line of the corridor, is collision-afflicted. By using the corridor bounds as inequality constraints, the modified, two-step Constrained CHOMP algorithm optimizes the initial path and generates a collision-free path according to the current sensor information. In Fig. 5.2 (b) it can also be seen, that obstacle 3 is not detected at all by the LiDAR sensor. Accordingly, an optimal path is generated that tries to use the apparently free space. As the vehicle progresses along the generated path, parts of obstacles are detected that were previously not perceived by the LiDAR sensor. The modified, two-step Constrained CHOMP algorithm adapts the path in real time according to new sensor information. As a result, the ego vehicle manages to avoid obstacles 3, cf. Fig. 5.2 (c). Nevertheless, obstacle 4 is not completely scanned. With further progress of the ego vehicle along the corridor and the corresponding scanning of the environment, collision-relevant information on obstacle 4 are also taken into account in the optimization algorithm and the path is adapted accordingly, cf. Fig. 5.2 (d). The evolution of the obstacle objective and smoothness objective during this experiment are depicted in Fig. 5.3. The first two peaks of the obstacle objective represent the optimization of the initial paths from the corridor specification, which comprised two actions.

5.2.2 Avoiding suddenly appearing static Obstacle in a turning Scenario

As part of this work, the proposed approach was implemented in a real experimental vehicle. The path planning of the corridor-based shared control approach was tested in a typical urban scenario. The operator's task was to navigate the vehicle into a parking area. The operator specified the corridor for this purpose into the corresponding side street, which is only partially

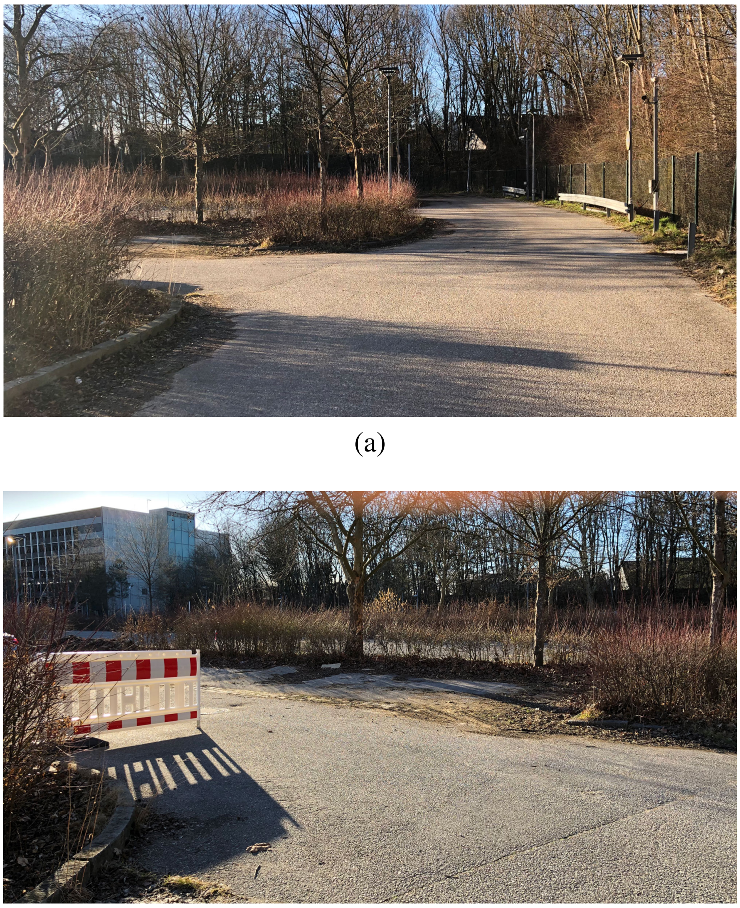
---

[2] Video of the experiment: https://youtu.be/M0R2S4WFciI



(a) (b)

Figure 5.4: Real driving results: (a) optimal path (green dotted) located within the specified corridor (red dotted) with obstacle bounds (yellow), (b) Path adaption based on new sensor information.



(a)

(b)

Figure 5.5: Camera images belonging to the real driving experiment in Fig. 5.4: (a) View of the side street into which the vehicle should turn, (b) sudden appearance of a previously hidden obstacle.

visible from the starting position of the vehicle. The occupancy grid map used for the corridor specification and the associated optimal path are shown in Fig. 5.4 (a). Fig. 5.5 (a) represents the corresponding camera image of the scenario. The modified, two-step Constrained CHOMP algorithm continuously optimizes the path in real time based on newly received sensor information as the vehicle progresses along the specified corridor. As the ego vehicle turns into the side street, an obstacle suddenly appears, see Fig. 5.4 (b) and Fig. 5.5 (b). The path optimization algorithm is able to adapt the path online based on the newly received sensor information and, hence, avoid a collision. The previous and the adapted path are depicted in Fig. 5.4 (b).

5.2.3 Obstacle passing dealing with oncoming Traffic

The following complex urban scenario is used to examine the corridor-based shared control approach in its entirety, considering static and dynamic obstacles. In this scenario, the experimental vehicle is blocked by an obstacle along its road lane. As can be observed from the camera view in Fig. 5.6, the driving space is restricted by barrier tapes. The occupancy grid map, displayed in the virtual representation implemented in RViz, however, shows that the barrier tapes are only partially detected near the experimental vehicle but are not detected at all a little further away. By using the method described in Section 5.1.1, the human operator manages to specify a plausible corridor shape with just a few actions and, consequently, includes the barrier tapes in the motion planning process of the shared control approach. This scenario already represents the basis for the proposed modification, the two-step Constrained CHOMP algorithm, to avoid collision-afflicted paths in poor local minima in Section 2.5. The corresponding optimal path is depicted in Fig. 2.6 (b) in a top view. The optimal path smoothly circumvents the blocking static obstacle. As the ego vehicle starts to drive around the static obstacle, the object detection recognizes a moving vehicle on the opposite road lane (orange cuboid) and predicts its manoeuvre (orange rectangles) – as can be seen from the virtual representation in RViz. The implemented trajectory optimization reflects this information and accordingly calculates a comfortable trajectory (green rectangles) down to a standstill and lets the oncoming vehicle pass, see Fig. 5.6 (a).

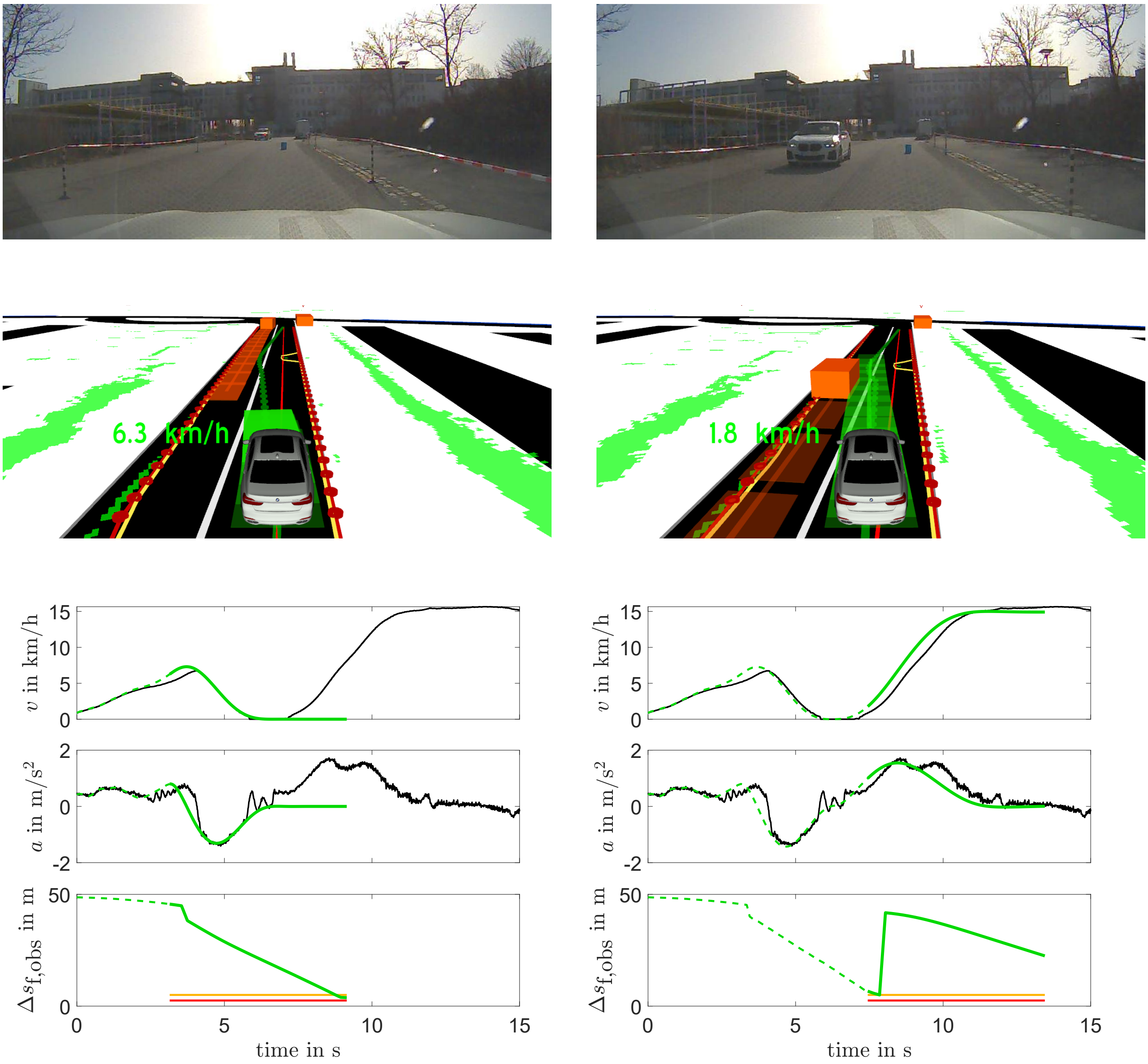
As the other road participant continues its motion, the trajectory optimization generates a smooth acceleration profile in compliance with a desired velocity value of 15 km/h, see Fig. 5.6 (b). In addition to the speed and acceleration curves, the graphs in Fig. 5.6 show the planned distances $\Delta s_{\mathrm{f,obs}}$, which are constrained in this scenario either by the oncoming vehicle or the end of the corridor. The link to the video associated with the described scenario is stated in the footnote[3].

5.2.4 Keeping distance to a Pedestrian on the Road

In another scenario, the operator's task is to specify a corridor along a curved road towards a desired destination. The resulting optimal path, which in our approach only considers the static environment, is planned too close to a dynamic object – a pedestrian at the roadside. Nevertheless, as illustrated by two consecutive time instants in Fig. 5.7, the trajectory

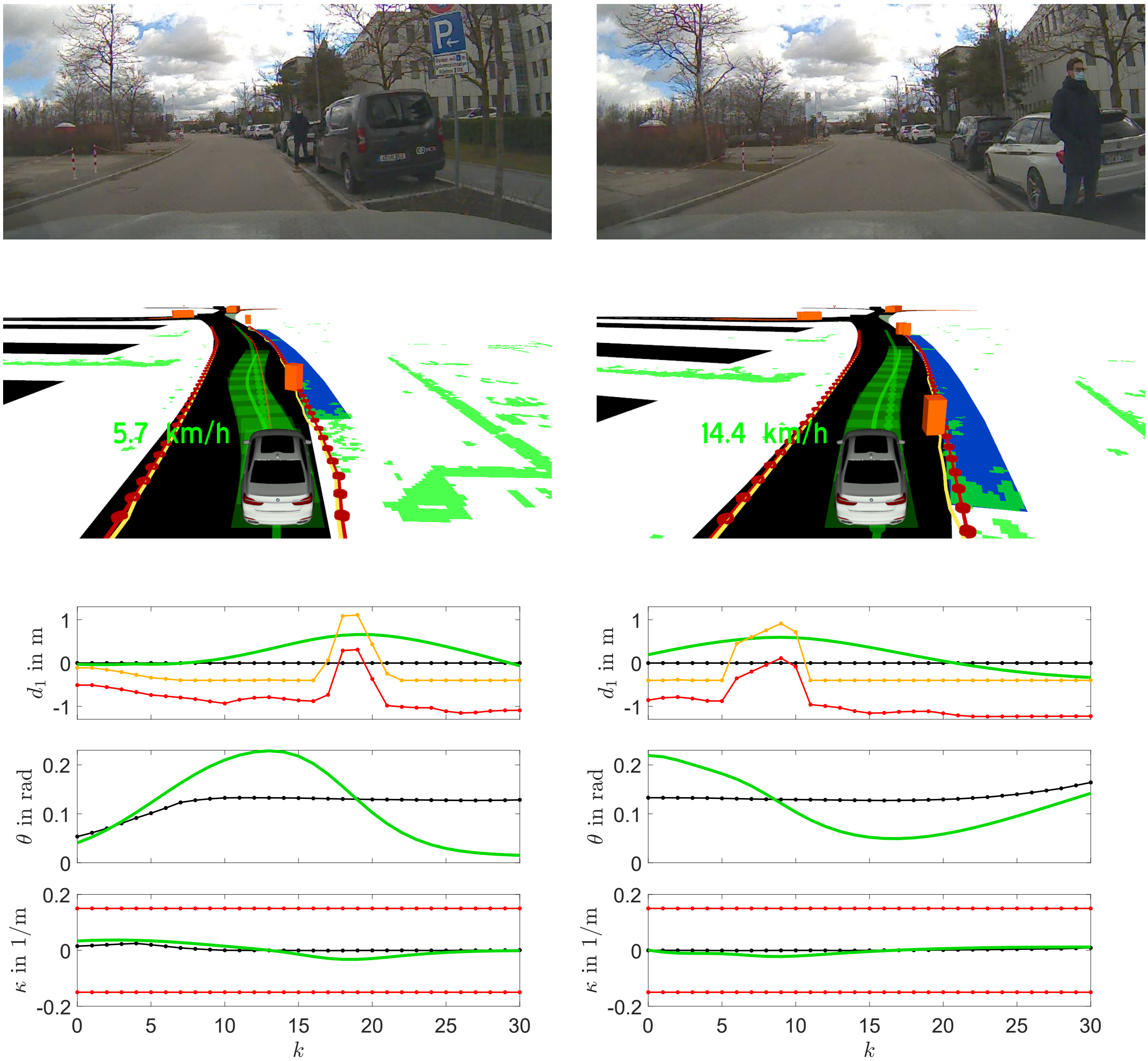
---

[3] Video of the experiment: https://youtu.be/H2VoKepMClw



(a) (b)

Figure 5.6: Obstacle passing dealing with oncoming traffic at two consecutive times, where the graphs indicate: the optimal trajectory (in green), comfort and safety constraints (in orange and red), and measurements (in black).



(a) (b)

Figure 5.7: Keep a comfortable distance to pedestrian on road at two consecutive times, where the graphs indicate: the optimal trajectory (in green), comfort and safety constraints (in orange and red), and the reference curve (black dotted).

optimization plans a smooth trajectory around the pedestrian in order to avoid a collision. The benefits of the proposed two-stage constraint softening are reflected, for example, in the graphs for the lateral position d_1. Here, the display of the distance constraints $d_{1,\text{max, comf}}$ and $d_{1,\text{max, safe}}$ are omitted for clarity. As described in Section 3.6, the lateral safety distance constraints $d_{i,\text{min, safe}}$ and $d_{i,\text{max, safe}}$ are linked to the heavily weighted slack variable ϵ to ensure the solvability of the quadratic program. The lateral comfort distance constraints $d_{i,\text{min, comf}}$ and $d_{i,\text{max, comf}}$, which are related to the weakly weighted slack variable ϵ_{comf}, are used to provide a comfortable distance to the obstacles. A suitable choice of the weighting of the comfort slack variable ϵ_{comf} allows for an optimal compromise between a comfortable distance to obstacles and a smooth steering action, as depicted in Fig. 5.7. The link to the video associated with the described scenario can be found in the footnote[4].

5.3 Conclusion

This chapter presents an indirect control concept for teleoperated driving, the corridor-based motion planning shared control. This approach aims to relieve the human operator in complex urban scenarios including low as well as high or varying communication time delays. In contrast to direct control, in this indirect control concept the operator is kept outside the closed control loop. The guidance loop is closed autonomously by the vehicle. Therefore, the closed control loop is insensitive w.r.t any communication time delays. By taking advantage of the human abilities in decision making, the operator is enabled to specify a corridor towards a desired destination, whereas the automated vehicle has to calculate an optimal motion on its own. Based on linear-quadratic optimization, novel problem formulations are developed for the vehicle longitudinal and lateral dynamics. They can be efficiently solved by means of a time-variant, linear MPC scheme using Quadratic Programming while at the same time satisfying the requirements for comfort and safety, resulting from automated driving. Although the proposed trajectory optimization only enables the formulation of locally optimal problems, the combination with the modified, two-step Constrained CHOMP algorithm leads to global optimal solutions. Although a direct comparison of the computing effort is only possible qualitatively due to different hardware used, the trajectory optimization time of 100 - 200 ms in [133] or [141] in relation to only 8 ms for path planning and 9 ms for trajectory planning in this work shows a significant increase in efficiency on a comparable processor.

---

[4] Video of experiment B: https://youtu.be/u1u52vzy4oI

6 Automatic Path Generation for Supervisory Control

The following chapter presents a supervisory control concept for indirect teleoperated driving tasks. The main content of the concept has been previously presented in [114].

6.1 Overview on the Approach

As pointed out in Section 1.3, remote controlling of a road vehicle in complex urban scenarios leads to a high workload for the operator. Stabilizing the road vehicle and avoiding possible collisions in direct teleoperated driving require both full attention and a high level of cognitive performance. The main issues are in particular the lack of three dimensional perception as well as the communication time delay, i.e. the delayed camera transmission of the vehicle environment and the delayed transmission of the operational commands specified by the operator.

In order to relieve the operator in complex urban scenarios, a useful approach may be to continuously generate further feasible paths and then forward these to the operator, see Fig. 6.1. The work in [28] points out clearly that human strengths are related to cognitive tasks, like behavioural decision-making and situation analysis. After years of learning and gaining experience, see [46], humans manage to easily cope with difficult road topologies and confusing traffic situation. Automatized machines, however, have strong capabilities in vehicle stabilization and collision avoidance thanks to their precise sensor equipment. It is therefore important that the found paths are not selected automatically by the machine. In each command step,

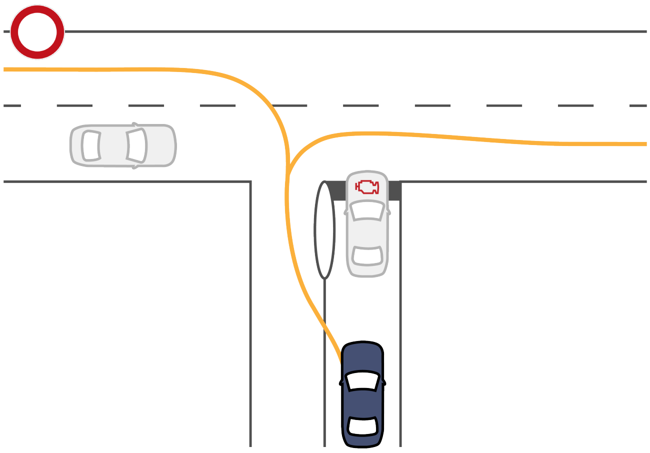


Figure 6.1: Generated paths that the operator can choose from based on the operator's perception.

the generated paths are suggested to the operator. The corresponding path will be followed by the vehicle only after their confirmation by the operator.

Unlike the shared control concept proposed in Chapter 5, the proposed supervisory control concept does not use predefined target points. Instead, a global path search algorithm – a modified RRT – generates reasonable paths and proposes them to the operator. However, in order not to overstrain the operator, the suggestion of too many paths has to be avoided. Before the remote controlled vehicle stops at the end of a selected path, the algorithm generates new path suggestions beforehand. As the vehicle navigates as selected by the operator, however, the vehicle environment changes continuously. Therefore, a very fast reactive local path adopter is required to ensure that no collision occurs. For this purpose, the modified, two-step Constrained CHOMP algorithm – developed in Chapter 2 – is employed in this concept. The major advantage of this modified path optimization is that the path to be optimized can be collision-afflicted. Obstacles exist in almost every urban scenario that are either partly or even completely obscured by other obstacles. The progress of the automatized vehicle on a planned path leads to the discovery of previously undetected obstacles. In fact, the last optimal path that previously appeared to be beneficial and collision-free may turn out to be infeasible.

6.1.1 RRT-based Path Search

In order to explore the vehicle environment using LiDAR sensor information, the RRT algorithm is modified and combined with a clustering algorithm. The RRT algorithm is commonly used to find a path between a starting point and and target point [80]. In the context of the proposed interactive path planning concept for supervisory control, the RRT algorithm is modified to explore the entire occupancy grid map and to build up a tree including all feasible paths. In general, the tree-structured graph of RRT algorithms grows towards stochastic samples, c.f. Section 1.2.3. For the purpose of the supervisory control concept, the stochastic samples are generated in the area of interest around the planning pose (x, y, θ) in a polar coordinate system that is centered on the vehicle rear axis:

$$\begin{bmatrix} x_{sample} \\ y_{sample} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + r \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix} . \tag{6.1}$$

Here, the sampling distance r is a stochastic value in the range of $[r_{min}, r_{max}]$, whereas the stochastic angle α lies in the range of $[\theta - \frac{\beta}{2}, \theta + \frac{\beta}{2}]$. The clearance distance r_{min} neglects the sampling in the vicinity of the planning pose, since this is less helpful for the tree expansion. This would lead to unreasonable U-turns, which are rare maneuvers in urban scenarios. The maximum radial component r_{max} specifies the planning horizon, whereas the angular component takes the the kinematic constraints into account. In order to speed up the exploration process while driving, the forward sampling and backward sampling are subdivided in the search process, see Fig. 6.2. At the moment when no feasible paths in the forward direction can be generated, additional computational effort is accepted in order to explore the environment behind the ego vehicle for backwards driving. Backwards driving scenarios usually arise from a standstill position for the ego vehicle, when the operator has just been contacted to resolve a blocking situation. The additional computational effort at this point, however, is negligible as it takes the operator some time to become aware of the scenario.

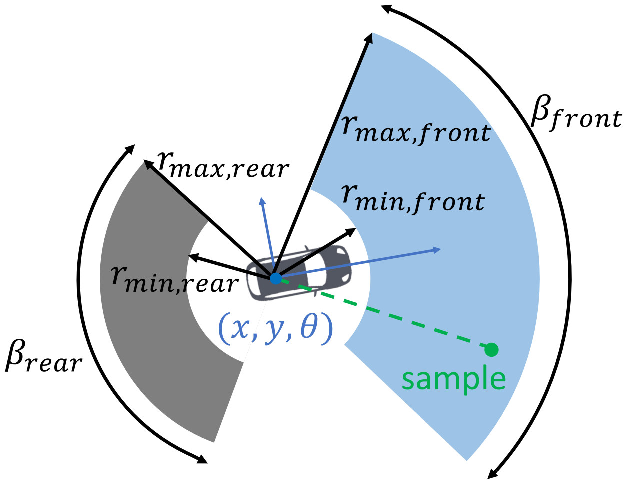


Figure 6.2: Sampling range of the modified RRT.

Heuristics are typical approaches to reduce the computational effort of the original RRT [68]. A well-known improvement is the cost-to-go estimation [77]. However, this heuristic requires a specific target position, which is not present in this concept. Instead, the algorithm decides which node among the existing ones in the tree is selected for the expansion process. For this purpose, a new measurement criterion γ is introduced. The next node for the expansion process is the one with the smallest criterion value. This criterion consists of three components which are added up for each node:

$$\gamma = \gamma_{smooth} + \gamma_{length} + \gamma_{distance} \,. \tag{6.2}$$

The term γ_{smooth} rates the control commands, i.e. the steering angles that were needed from the starting pose to the node. This way, smooth paths are preferred in the expansion process. The second component γ_{length} favors the short branches of the tree to be expanded, which leads to a more thorough search of the vehicle environment. The last component $\gamma_{distance}$ is motivated by the original exploring heuristic, which measures the euclidian distance between sample point and node, c.f. [81].

Usually only one node is expanded per expansion step. In contrast to other RRT variants, in this approach up to five nodes are added to the expansion tree. They are generated using a kinematic single track model of the ego vehicle. The control inputs of the kinematic single track model are a series of steering angles δ. By limiting the value of δ, the vehicle kinematic constraints are accounted for. With a fixed step length l and the previous vehicle state $z_i = (x_i, y_i, \theta_i)$, the subsequent state is calculated as

$$\begin{cases} \Delta\theta = \frac{l \cdot \tan\delta}{L}. \\ R = \frac{l}{\Delta\theta} \\ x_{i+1} = x_i + R \cdot (\sin(\theta_i + \Delta\theta) + \sin\theta_i) \\ y_{i+1} = y_i + R \cdot (\cos\theta_i - \cos(\theta_i + \Delta\theta)) \\ \theta_{i+1} = \theta_i + \Delta\theta \end{cases} , \tag{6.3}$$

where R stands for the turning radius, L is the wheelbase and $\Delta\theta$ represents the change of vehicle orientation, see Fig. 6.3 (a). As depicted in Fig. 6.3 (b), the intermediate states are checked for collision to avoid collision in the segment between two nodes. The corresponding node is only added to the expansion tree if all intermediate states are collision-free.

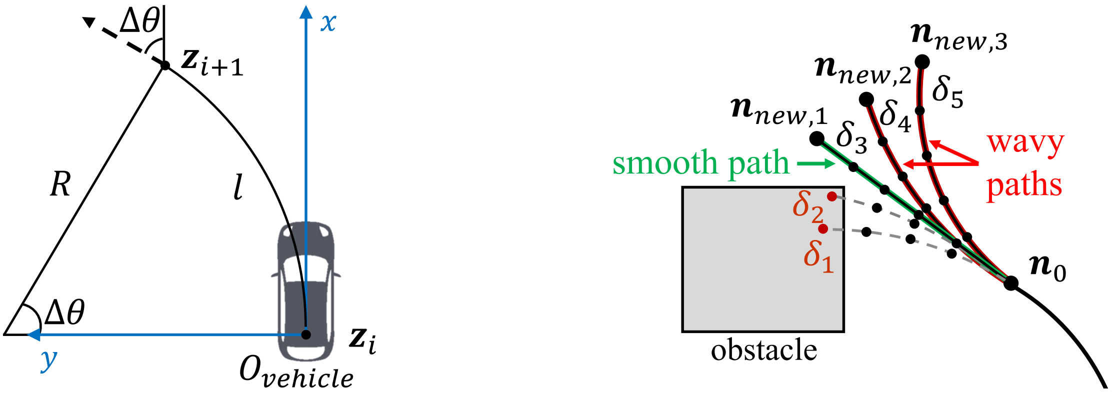


Figure 6.3: Tree expansion: (a) a single step with one control input, (b) one expansion step with added nodes.

The search process of the RRT algorithm usually is terminated after a feasible path from starting point to a target point has been found [81]. Since a target point is not available in this approach, the nodes are expanded until a predefined path length is reached. The path search algorithm is terminated after a specific number of feasible paths are found or maximal sample attempts has been reached.

The result of the modified RRT in this work is a multitude of feasible paths. To avoid overstraining the operator with the path selection task, the number of the paths needs be reduced without discarding relevant ones. For this purpose the found paths are clustered according to their end positions by means of the DBSCAN algorithm, see [33].

6.2 Experimental Results

The efficiency of the proposed interactive path planning concept for supervisory control is assessed by means of real driving experiments in different urban scenarios. As indicated, this concept is evaluated on the level of path planning. The effectiveness of the combined motion planning algorithm including path and trajectory optimization has already been investigated by means of the shared control concept in Chapter 5 and is therefore out of the scope of this evaluation.

The occupancy grid map forms the basis of the interactive path planning process. The required data is provided by a LiDAR sensor. The LiDAR sensor is mounted on top of the experimental vehicle. For operator feedback, a camera transmission is used in addition to the occupancy grid map.

For computational efficiency, the algorithms were implemented in C++. The computations were carried out on an Intel i9-9980XE 3 GHz processor with an average calculation time per calculation step of 11 ms for the path exploration with the modified RRT and 8 ms for the path optimization with the modified, two-step Constrained CHOMP algorithm.

6.2.1 Path Exploration and Clustering Results at an Intersection

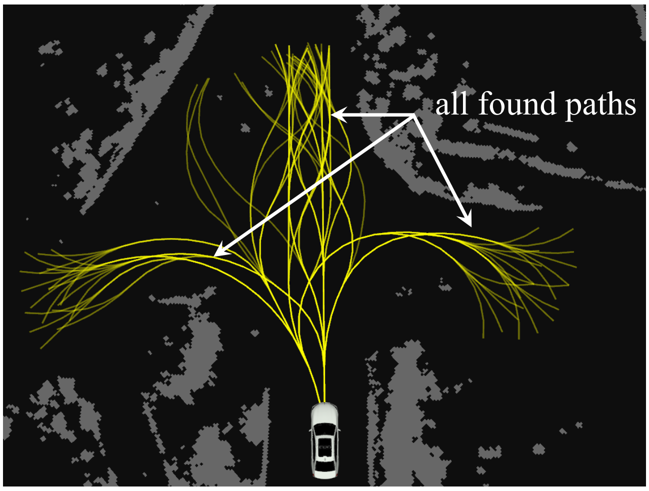
Fig. 6.4 shows an example of the clustered path search using an intersection scenario. This scenario is used to investigate whether all feasible paths are found and only reasonable ones are proposed to the operator. The found paths by the modified RRT algorithm are displayed in Fig. 6.4 (a). It can be observed that the exploration process generates a lot of feasible paths. It would be irresponsible to suggest all the paths found by the RRT algorithm. This is particularly to be avoided while the vehicle is moving, since the human operator will be overwhelmed during the selection task. To reduce the suggested solutions to the relevant ones, the feasible paths are clustered by means of the DBSCAN algorithm. The resulted cluster found in this scenario are shown in Fig. 6.4 (b). For each cluster found, only one path is subsequently proposed to the operator. The path to be suggested to the operator is selected based on the cost criterion value of the last node, see Section 6.1.1. The final result of this experiment is shown in Fig. 6.4 (c). The operator is given three reasonable paths to choose from.

6.2.2 Interactive Path Planning in a Driving Maneuver

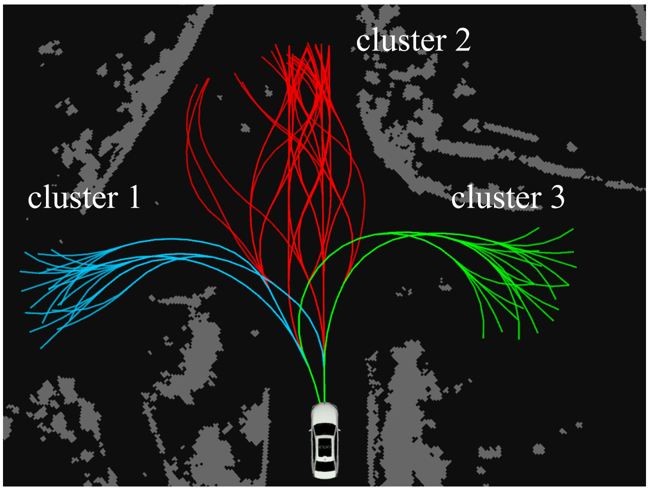
In another urban scenario, the clustered path search using the modified RRT and the DBSCAN algorithm generated two solutions between which the operator could select, see Fig. 6.5 (a). The operator's decision led to the task for the automated vehicle to turn right into a side street. However, the sidestreet is only partially visible from the vehicle's starting position, see Fig. 6.6 (a). As stated in [80], the RRT algorithm does not generate optimal solutions. As can be seen in Fig. 6.5 (a), the modified CHOMP algorithm removes redundant motions of the selected option and generates an optimal path. The optimality at the time step as can be seen in Fig. 6.6 (a) is essentially achieved as a tradeoff between the distance to the reference path – the selected path option – and smoothness. As the automatized vehicle follows the optimal path, previously hidden obstacles appear as the vehicle turns into the side street, see Fig. 6.5 (b) and Fig. 6.6 (b). Based on newly received sensor measurements, the modified CHOMP algorithm adapts the path in real time and, hence, is able to generate a collision-free and optimal path. For this purpose, the end point of the optimal path is shifted away from the reference path in order to reduce the cost value of the obstacle objective. As can be seen in the graph in Fig. 6.7 at approx. 9 sec., the cost value of the reference path objective, however, increased. The curvature objective has been suppressed in Fig. 6.7. The curvature objective had no influence in this scenario, since the corresponding constraint was not violated. Instead of stopping at the end of the optimal path, the clustered path search generates further options for the operator. The optimal path forms the basis for this. In this scenario, as can be seen in Fig. 6.5 (b) or in the associated video[1], the path search algorithm generates a straight path option for the operator.

---

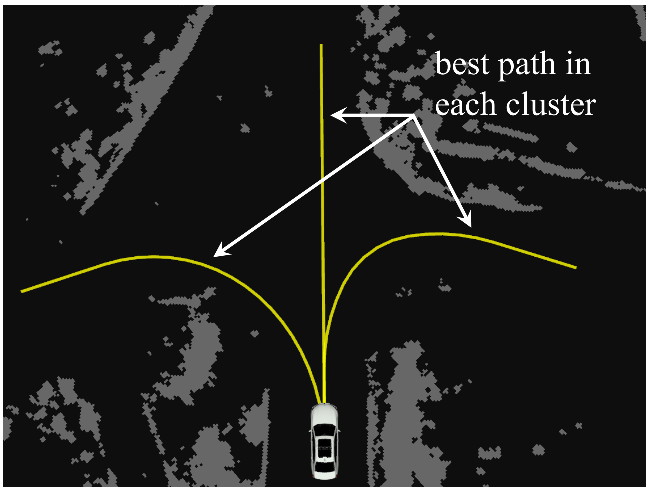
[1] Video of the experiment: https://youtu.be/POhICIi4FJI



(a)

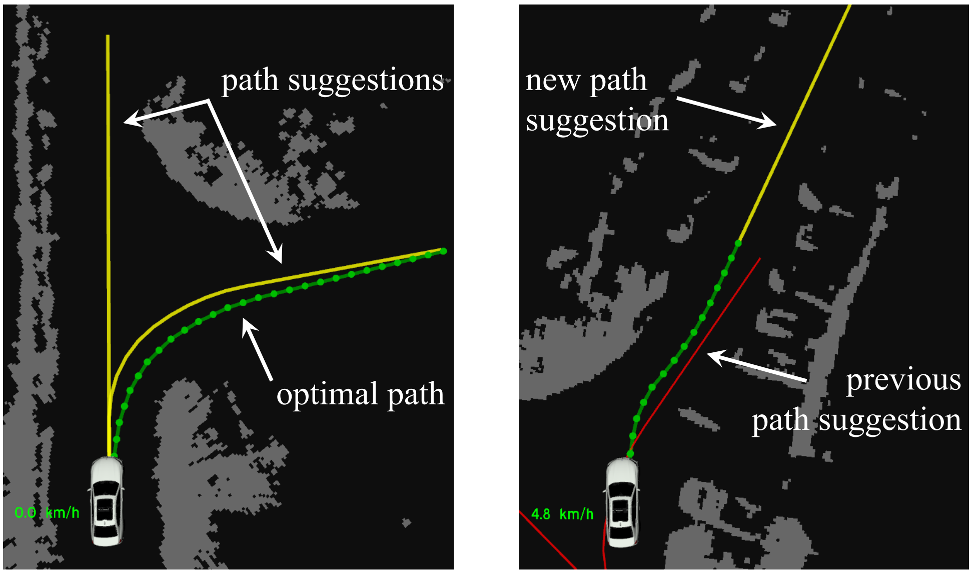


(b)



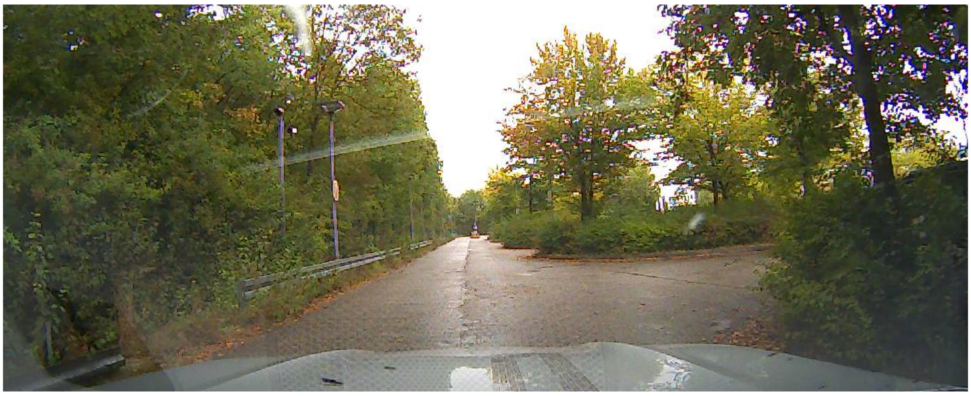
(c)

Figure 6.4: Results of clustered RRT path search: (a) all found paths, (b) clustered results, (c) path suggestions for operator.

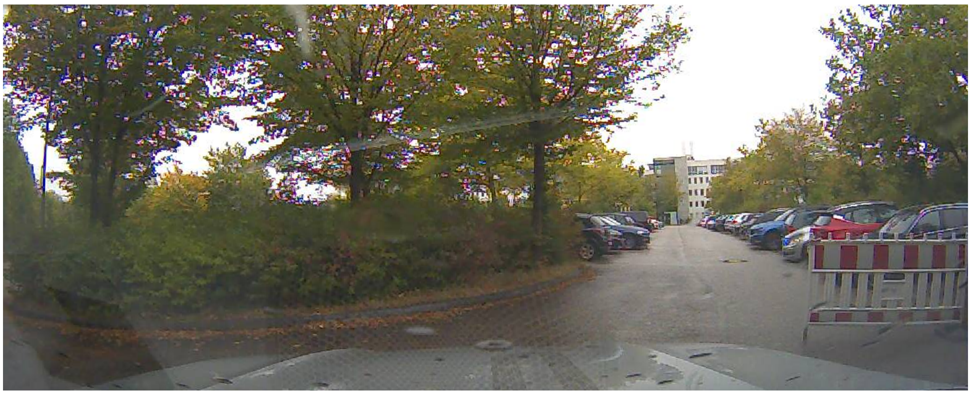


(a) (b)

Figure 6.5: Real driving results at two consecutive time stamps.



(a) Before the turning operation



(b) After the turning operation facing an unexpected obstacle

Figure 6.6: Image sections of the camera view belonging to Fig. 6.5.

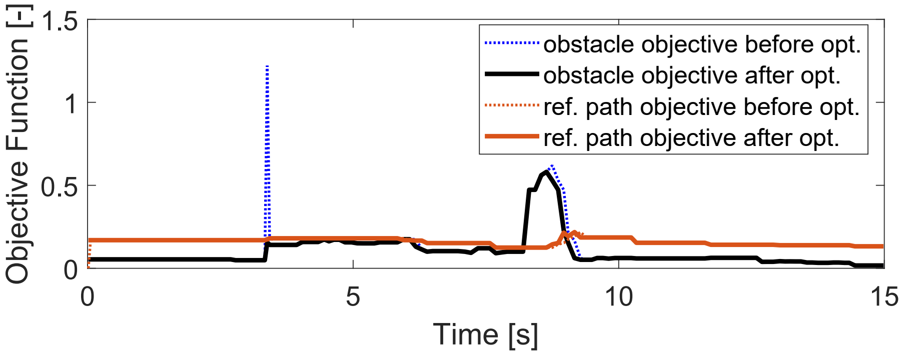


Figure 6.7: Time evolution of the objective functions corresponding to the scenario in Fig. 6.5.

6.3 Conclusion

This chapter presents an indirect control concept for teleoperated driving, an interactive path planning method for supervisory control. This approach aims to relieve the human operator in complex urban scenarios including low as well as high or varying communication time delays. Since in this indirect control concept, the operator is kept outside the closed control loop and the guidance loop is closed autonomously by the vehicle, the closed control loop is insensitive w.r.t any communication time delays. By taking advantage of the human abilities in decision making, the operator is enabled to choose between feasible paths in a given teleoperated driving scenario. The corresponding path will be followed by the vehicle only after the confirmation by the operator. For this purpose, the RRT algorithm is modified in such a way that, without the operator specifying target positions, further paths are generated by the automatized vehicle, between which the operator can select. To generate optimal paths and, hence, to avoid collisions, the selected option generated by the RRT algorithm is optmized in real time using the modified, two-step Constrained CHOMP algorithm. The concept has been evaluated in real driving experiments, where the RRT in combination with the DBSCAN clustering algorithm successfully generates suitable path suggestions and the operator selected option is optimized online to avoid collisions.

Part III

Machine Learning Approaches for Motion Planning

7 Path Optimization for Autonomous Driving using Deep Learning

The following chapter presents a shared control concept for indirect teleoperated driving tasks. The main content of the concept has been previously presented in [113]

7.1 Related Work

Over the past two decades, neural networks have gained popularity for addressing complex, non-linear problems. The majority of prior work in autonomous driving is dominated by end-to-end or Deep Reinforcement Learning (DRL) methods. Already by 1989 neural networks were employed in the first self-driving vehicle as an end-to-end process, in which the algorithm generated a steering angle based on a camera image after recognizing lanes and segmenting the terrain, see [103]. Since end-to-end and DRL methods directly map sensor information to control commands, functional safety is difficult to prove.

As an alternative, the authors in [53] propose a solution for a combined perception-planning deep neural network. In contrast to end-to-end and DRL methods, the neural network is trained to estimate an optimal trajectory over a finite prediction horizon. For prediction of the optimal trajectory, the deep neural network – a combination of convolutional neural network (CNN) and long short-term memory (LSTM) network – is given a sequence of occupancy grid maps as input. The predicted trajectory states are subsequently forwarded to a motion controller for tracking purposes. Although this approach enables the investigation of the functional safety of the generated trajectory by evaluating its states using the occupancy grid maps, the optimality and the quality of the solution is questionable. Since the performance of the neural network highly depends on the used data set, even simple scenarios that were not taken into account in the data set can lead to implausible driving trajectories. Critical traffic situations in which a collision-avoiding trajectory is urgently required can lead to fatal consequences.

A hybrid machine learning approach that is designed for the planning of safe trajectories in complex and dynamic traffic-scenarios is the HARRT+ algorithm proposed in [20]. This algorithm uses a 3D-CNN for predicting longitudinal acceleration and steering wheel angle profiles in combination with a RRT variant. By using a dynamic vehicle model, physical constraints are taken into account in the trajectory planning. The experimental results show the real-time capability of the algorithm without harming safety in most of the investigated scenarios. However, the use of the RRT variant requires a subsequent trajectory optimization in order to remove redundant or jerky motions that such planners may generate.

The motion planning in higher automated teleoperation concepts presented in this work is employed as a combination of a path optimization using a modified Constrained CHOMP algorithm (see Section 2) and a trajectory optimization using quadratic programming (see Section 3). Considering the calculation times of 8 ms and 9 ms resulting from the two algorithms, a further reduction in calculation times with regard to real-time capability with the used hardware is not necessary. However, a reduction in the overall computational effort is reasonable with regard to more cost-effective hardware. The trajectory optimization designed in this thesis takes into account driving physics as well as comfort and collision-relevant constraints. The effectiveness of the trajectory optimization has been investigated in several automated driving scenarios. In order to maintain the optimality of the generated trajectories for reasons of driving safety, a strategy based on neural networks for the optimization of collision-free paths to reduce computational effort is presented in this chapter.

The remainder of this chapter is structured as follows: In Section 7.2, the process of collecting, extraction, transformation data required for training a neural network is mentioned. Section 7.3 explains the problem statement, mentions different neural network architectures which were trained on data collected and transformed in Section 7.2. Section 7.4 details the training parameters and process, Section 7.5 details the evaluation criteria and evaluation metrics for the trained models. Section 7.6 concludes the work.

7.2 Data

For the first investigations of neural networks for possible application of path optimization, measurements were recorded in this work in a standstill position using the corridor-based teleoperation concept proposed in Section 5. The operator specified an initial path with a predefined width of the corridor. Based on the occupancy grid map and the operator specified initial path, the modified Constrained CHOMP algorithm including the domain step, see Section 2, then generated an optimal, collision-free path.

The raw data was collected in ten distinct scenarios, all of which took place in a car park. Each scenario contains a number of path optimization cases. Eight of the ten scenarios are used for training and the remaining two scenarios for validation and testing, respectively. The raw data is recorded in the odometry frame. As examplarly shown in Fig. 7.1, each of the data sample contains the following information:

1) Occupancy grid map G of dimension 1536×1536 that contains information about surrounding occupancy with a grid resolution of $G_{\text{res}} = 0.15\,\text{m}$,
2) Origin of the occupancy grid map $G_{\text{org}} = \left[x_{\text{org}}, y_{\text{org}}\right]$,
3) Ego vehicle position $\mathbf{q}_{\text{ego}} = \left[x_{\text{ego}}, y_{\text{ego}}\right]$,
4) Initial path $\xi_{\text{init,odom}} = \left[\mathbf{q}_1^\top, \ldots, \mathbf{q}_k^\top\right]^\top \in \mathrm{R}^{k\times 2}$ with $\mathbf{q}_0 = [x_0, y_0]$ as the fixed starting point and
5) Optimized path $\xi_{\text{opt,odom}} \in \mathrm{R}^{k\times 2}$.

For optimization, the path specified by the operator is resampled in the path planning module, so that both the initial path and the optimized path have a constant waypoint distance of $2\,\text{m}$ in the raw data set. Furthermore, the optimization horizon is limited to $50\,\text{m}$, which is why $k \leq 25$ applies.

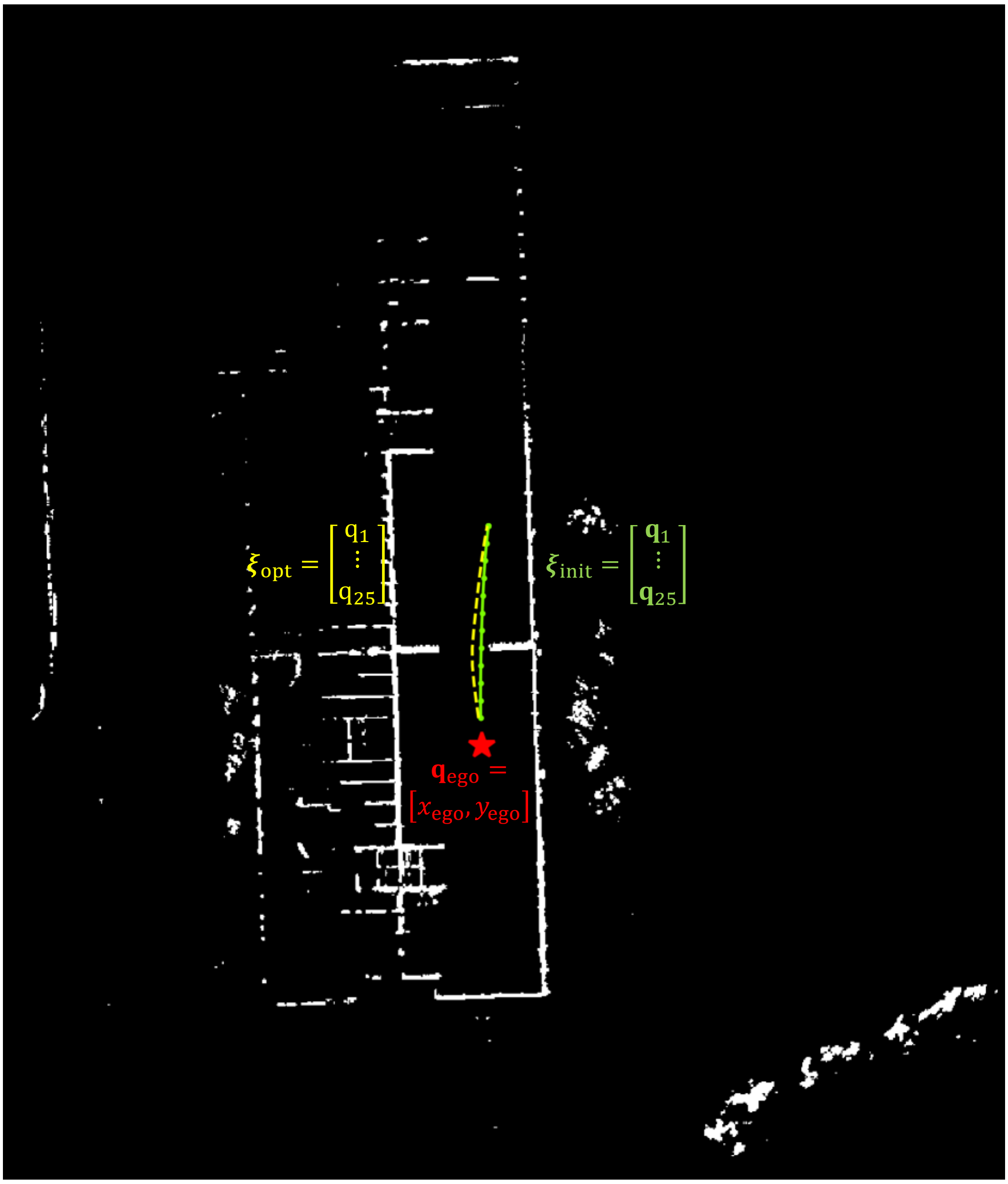

$\xi_{\text{opt}} = \begin{bmatrix} q_1 \\ \vdots \\ q_{25} \end{bmatrix}$
$\xi_{\text{init}} = \begin{bmatrix} q_1 \\ \vdots \\ q_{25} \end{bmatrix}$
$q_{\text{ego}} = [x_{\text{ego}}, y_{\text{ego}}]$

Figure 7.1: Data sample: Binary Occupancy grid map in the background, Initial path (in green), optimized path avoiding obstacles (in yellow), ego vehicle position (red star).

For the investigations roughly 15000 samples of raw data have been recorded.

However, since the following initial path is already optimal and does not need to be optimized further after an optimization step, only 10% of the raw data are cases with optimized paths. Due to the extremely unbalanced nature of the data, training on the entire set resulted in failure in learning the optimization objectives. The data set is thus filtered based on the inequality of the initial path and the optimized path. In order to keep the non-optimizing behavior for the cases in which it is required, a few random samples with non-optimizing behavior are added to the experimental data. This results in an data set with around 2000 samples, 70% of which are instances with optimized paths. For training the neural network, both the initial path and the optimal path are transformed into the occupancy grid map space:

Initial path in occupancy grid map space: $\xi_{\text{init}} = (\xi_{\text{init,odom}} - G_{\text{org}})/G_{\text{res}}$

Optimized path in occupancy grid map space: $\xi_{\text{opt}} = (\xi_{\text{opt,odom}} - G_{\text{org}})/G_{\text{res}}$.

Depending on the operator's specification, the initial path and thus the optimized path might be fewer than 25 waypoints. Since neural networks require a fixed input size, all paths that consist of fewer than 25 waypoints are padded with their last coordinate values to bring them to a fixed length of 25.

7.3 Methods

7.3.1 Problem Statement

Training a model that maps specified inputs to outputs based on data with input-output pairs is referred to as supervised learning. As mentioned in section 7.2, a sample of the training data set includes information about an occupancy grid map G, an initial path ξ_{init} and an optimized path ξ_{opt}. In this case, the problem statement can be described as a supervised learning task with the occupancy grid map and initial path as inputs and the predicted optimized path $\tilde{\xi}_{\text{opt}}$ as output of the model to be trained. An illustration of the problem statement is shown in Fig. 7.2. In general supervised learning problems are subdivided into regression and classification tasks. The fact that the output of the model in this work correlates to a numerical value, makes it a classic instance of a regression problem.

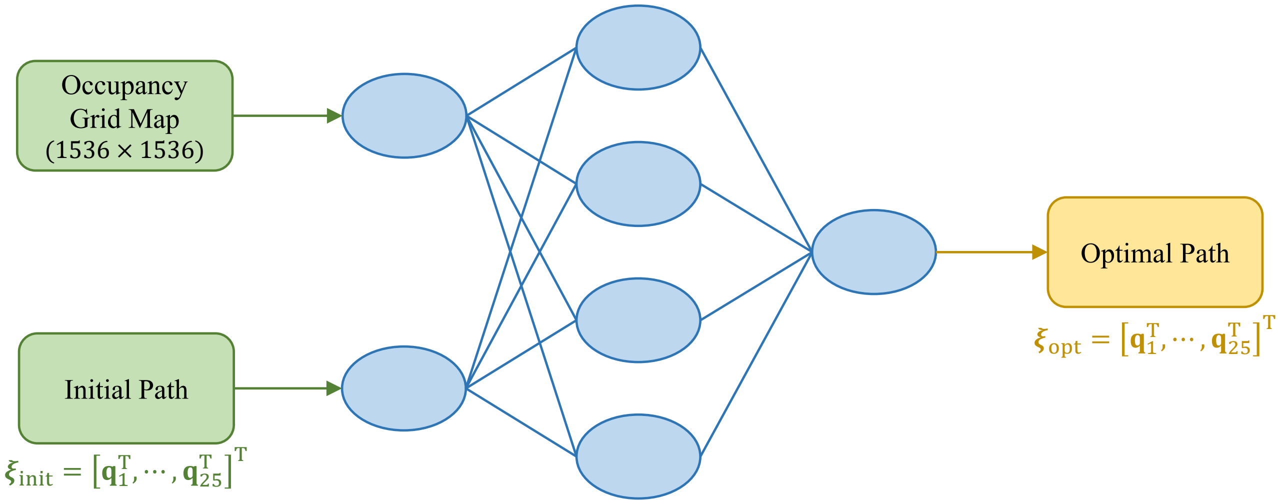

Occupancy Grid Map (1536 × 1536)
Initial Path
$\xi_{\text{init}} = [\mathbf{q}_1^{\text{T}}, \cdots, \mathbf{q}_{25}^{\text{T}}]^{\text{T}}$
Optimal Path
$\xi_{\text{opt}} = [\mathbf{q}_1^{\text{T}}, \cdots, \mathbf{q}_{25}^{\text{T}}]^{\text{T}}$

Figure 7.2: Illustration of the supervised learning problem statement for the path optimization task: The neural network takes the occupancy grid map and the initial path as inputs and predicts an optimized path.

7.3.2 Network Architectures

For classification and regression, artificial neural networks, notably multilayer perceptrons (MLPs), have gained prominence [44]. To utilize MLPs on two-dimensional data, the data needs to be flattened into a one-dimensional long vector, which can then be provided as input to the MLPs. The spatial structure of the data is lost during the flattening process [44]. With the loss of spatial organization of the data, an enormous number of parameters would be required for the application of MLPs to the described problem statement in this thesis. This would require the use of unrealistic GPU resources, making MLPs impractical for the given supervised learning task.

Convolutional neural networks (CNNs) offer an elegant and powerful approach for processing two-dimensional data while maintaining the spatial structure. CNN-based architectures are widely utilized in image classification, object detection, traffic sign detection, semantic segmentation and in many other applications in the field of computer vision and autonomous driving, see [63], [78], [103] and [110]. Convolution in the context of a convolutional neural network is a linear operation involving a dot product of a kernel/filter (two dimensional set of weights) patch by patch throughout the entire two-dimensional input data. During the training phase, this filter learns a certain type of feature in the input data and eventually allowing it to recognize the feature regardless of its position in the related two-dimensional data.

For supervised learning tasks with multiple inputs, including spatial and non-spatial data, CNNs are often combined with other artificial neural networks. The spatial features extracted by a CNN are concatenated with the non-spatial input data and passed on to a further neural network for the classification or regression task. A various number of such hybrid architectures are summarized in [52] for the domain of autonomous driving. However, an application of an artificial neural network in relation to the path optimization task, given in this work, does not exist in the literature.

Hybrid artificial neural networks

According to the survey in [52], a possible neural network architecture to learn the path optimization objectives might be the spatial feature extraction from the occupancy grid map using a CNN followed by the concatenation of the extracted spatial features with the initial path and finally predicting the optimized path from the concatenated features using a MLP. An illustration of this architecture is given in Fig. 7.3. MLPs consist of a series of fully connected layers, i.e. every neuron in one layer is connected to every neuron in the neighboring layers. Due to their fully connected layers, they are considered as structure agnostic. This means that no special assumptions have to be made about the input, which makes them widely applicable.

Another neural network architecture mentioned in [52] shows the same structure as the architecture depicted in Fig. 7.3. Instead of a MLP, however, a recurrent neural network (RNN)

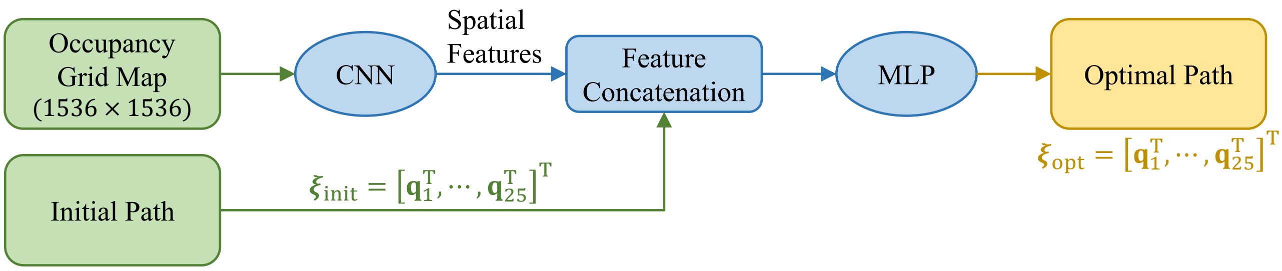

Occupancy Grid Map (1536 × 1536)
CNN
Spatial Features
Feature Concatenation
MLP
Optimal Path
$\xi_{\text{opt}} = [\mathbf{q}_1^{\mathrm{T}}, \cdots, \mathbf{q}_{25}^{\mathrm{T}}]^{\mathrm{T}}$
Initial Path
$\xi_{\text{init}} = [\mathbf{q}_1^{\mathrm{T}}, \cdots, \mathbf{q}_{25}^{\mathrm{T}}]^{\mathrm{T}}$

Figure 7.3: Hybrid artificial neural network: Combining CNN and MLP.

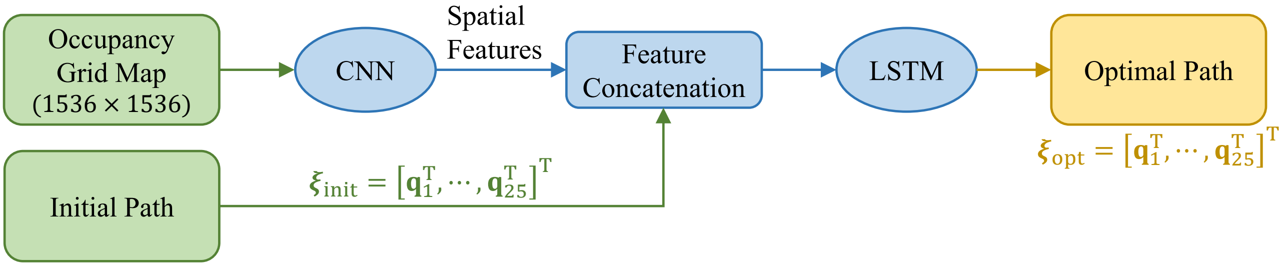

Occupancy Grid Map (1536 × 1536)
CNN
Spatial Features
Feature Concatenation
LSTM
Optimal Path
Initial Path
$\xi_{\text{init}} = [\mathbf{q}_1^\mathrm{T}, \cdots, \mathbf{q}_{25}^\mathrm{T}]^\mathrm{T}$
$\xi_{\text{opt}} = [\mathbf{q}_1^\mathrm{T}, \cdots, \mathbf{q}_{25}^\mathrm{T}]^\mathrm{T}$

Figure 7.4: Hybrid artificial neural network: Combining CNN and LSTM network.

is used. RNNs are characterized by their ability to use information from previous inputs to affect the current input and output. In the path optimization case, sequences of waypoints are to be predicted that are related in a temporal dimension. An approach to predict trajectories using RNNs is proposed in [53]. Although RNNs capture temporal/sequential patterns, they suffer from the problem of vanishing gradients, which inhibits learning especially long data sequences as described in [65]. To address the vanishing gradient problem, long short-term memory (LSTM) networks – a type of RNNs – were proposed in [66]. In this work, supervised learning attempts are made with LSTM networks with the structure described above. The hybrid artificial neural network including LSTMs is shown in Fig. 7.4.

However, none of the above architectures could solve the problem statement described in 7.3.1. Instead of learning the relationship between the occupancy grid map, the initial path and the optimized path, the architectures proposed above almost only interpolate between the initial position and the final position of the initial path. Quantitatively, the loss function employed for training – the mean absolute error – decreased over the epochs. Qualitatively, however, the model predictions were not applicable, since collision avoidance in the output path could not be achieved.

Despite the fact that CNNs have gained popularity as the common method to solve vision-based tasks, they still show a few drawbacks. As discussed in [88], mapping from pixel index

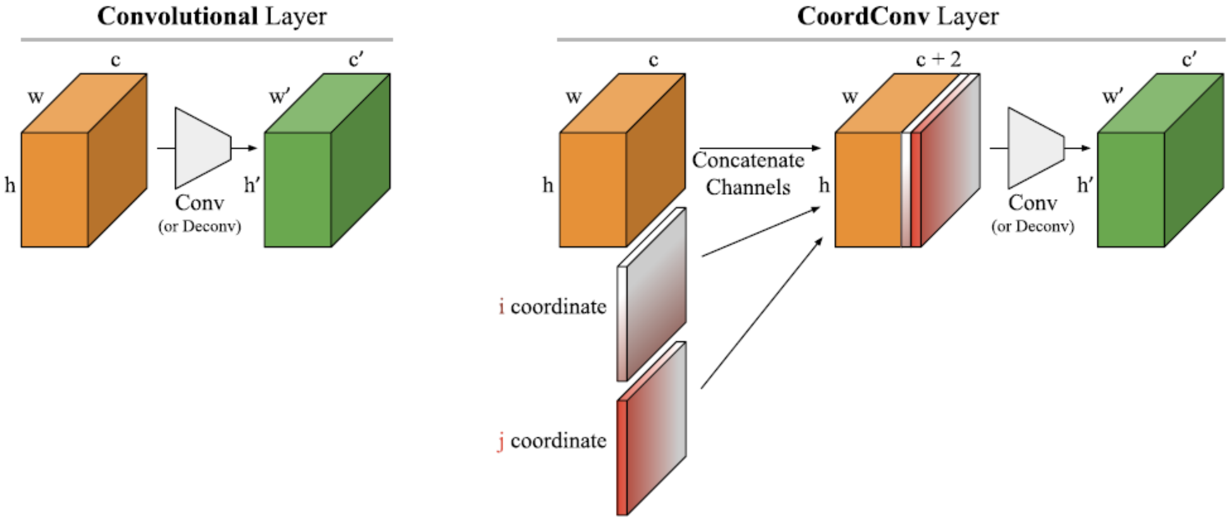


Figure 7.5: Illustration of feature addition as extra channels, proposed in [88]. h,w=1536,1536 and c=1, post feature concatenation, number of channels are $c + 2 = 3$

space to coordinate space proves difficult for CNNs. As with the task given in this thesis, the CNN was not able to learn the mapping between the occupied cell in the binary grid map and the corresponding path indices w.r.t. the occupancy grid map. An idea to address this issue was presented in [88], where the coordinate specific information is manually added as additional features to the input data. This is done by adding two further channels in a hard coded manner to the incoming two dimensional data, see Fig. 7.5. However, this operation of coordinate convolution slowed down the training process in in the context of this work without noticeably improving the results.

7.3.3 Redefining the Problem Statement

In general, the main purpose of path planning algorithms is to generate collision-free paths. In the given path optimization task, the initial path has to be altered if obstacles interfere along it. The degree of path deviation required is determined based on the proximity of obstacles with respect to the corresponding waypoints. The problem statement is therefore reformulated using this statement. The redefined problem statement, see Fig. 7.6, remains a regression task using supervised learning. In contrast to the problem statement in Section 7.3.1, the occupancy grid map G and the initial path ξ_{init} are no longer direct inputs for the neural network. Instead, the occupancy feature matrix $\mathbf{F}_{\text{occ}}$ containing information about the proximity of obstacles along the planned path represents the input. The neural network is trained based on input-output pairs of the occupancy feature matrix $\mathbf{F}_{\text{occ}}$ and the deviation

$$\Delta\xi = \xi_{\text{init}} - \xi_{\text{opt}} \, . \tag{7.1}$$

between the initial path ξ_{init} and the optimized path ξ_{opt}. The output of the neural network is hence the predicted path deviation $\Delta\tilde{\xi}$. The predicted optimized path results as the difference between the initial path and the predicted path deviation:

$$\tilde{\xi}_{\text{opt}} = \xi_{\text{init}} - \Delta\tilde{\xi} \, . \tag{7.2}$$

The information about the proximity of obstacles to each waypoint in the initial path is collected by constructing a matrix including the neighboring obstacle positions using occupancy values for respective locations on the occupancy grid map. To create the occupancy feature matrix, a certain area is defined for each of the 25 waypoints so that sufficient information on surrounding obstacles is taken into account. In this thesis, a rectangular search region of 40 cells long and 26 cells wide is defined, which considers the vehicle dimension with additional threshold values, as can be seen in Fig. 7.7. This results in a surrounding area including 1040 cells per waypoint and thus the occupancy feature matrix is defined with the dimension 25×1040. The redefinition of the problem statement has several advantages. The dimension of the input data is reduced significantly from 1536×1536 to 25×1040, which improves its manageability. Furthermore, all relevant occupancy cells are filtered for each waypoint. As a result, this relationship no longer has to be learned by the neural network.

7.3.4 Trained Models based on the redefined Problem Statement

For the previous problem statement, as described in Section 7.3.1, a CNN was used to extract spatial information based on the occupancy grid map. By introducing the occupancy feature matrix and therefore the redefined problem statement, this step is no longer required.

Distributed Dense Neural Network

As previously mentioned, MLPs have the great advantage of being structure agnostic, which makes them widely applicable. However, flattened data, i.e. one-dimensional data is required. To enable processing two-dimensional data and therefore to keep the spatial information provided by the occupancy feature matrix, a distributed dense (DD) network architecture for path optimization is proposed in this work. In a distributed layer, a single MLP is defined, which, however, is applied to each of the 25 features, see Fig. 7.8. Thus, each waypoint-specific entry in the occupancy feature matrix, i.e. each row, is processed independently.

The output layer of the proposed architecture is again a single MLP wrapped in a distributed layer that consists of two units due to the coordinate-specific output. Since the ground truth deviation $\Delta\xi$ ranges between $[-1, 1]$, due to normalization, the hyperbolic tangent operator (tanh) is defined as activation function.

LSTM based Network

As mentioned earlier, LSTMs belong to the category of recurrent neural networks and are capable of learning long term dependencies in sequential data. The occupancy feature matrix of dimension $(25, 1040)$ defines a sequence of features of 25 way points. Due to the nature of the CHOMP algorithm changes in one waypoint have an impact on subsequent waypoint positions. LSTMs can be used to learn these patterns. Internal gates in LSTMs control the flow of information. These gates learn which data in a sequence should be kept or discarded. It can then relay important information along the chain of sequences to make predictions. The proposed neural network based on LSTM layers consists of one hidden layer, where the hyperbolic tangent is applied as activation function. The output LSTM layer consists of 50 units with tanh as activation function. A subsequent reshaping layer reshapes the output to

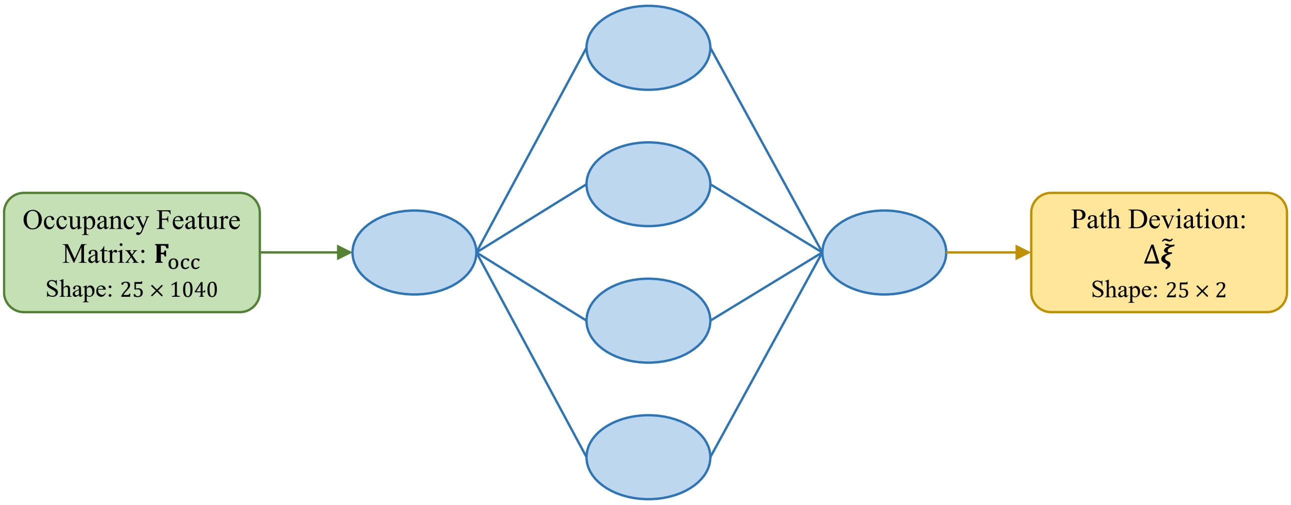

Occupancy Feature Matrix: $\mathbf{F}_{\text{occ}}$
Shape: 25 × 1040
Path Deviation: $\Delta\tilde{\xi}$
Shape: 25 × 2

Figure 7.6: Illustration of the redefined supervised learning problem statement for the path optimization task: The neural network takes a occupancy feature matrix as input and outputs the predicted deviation between the initial path and optimized path.

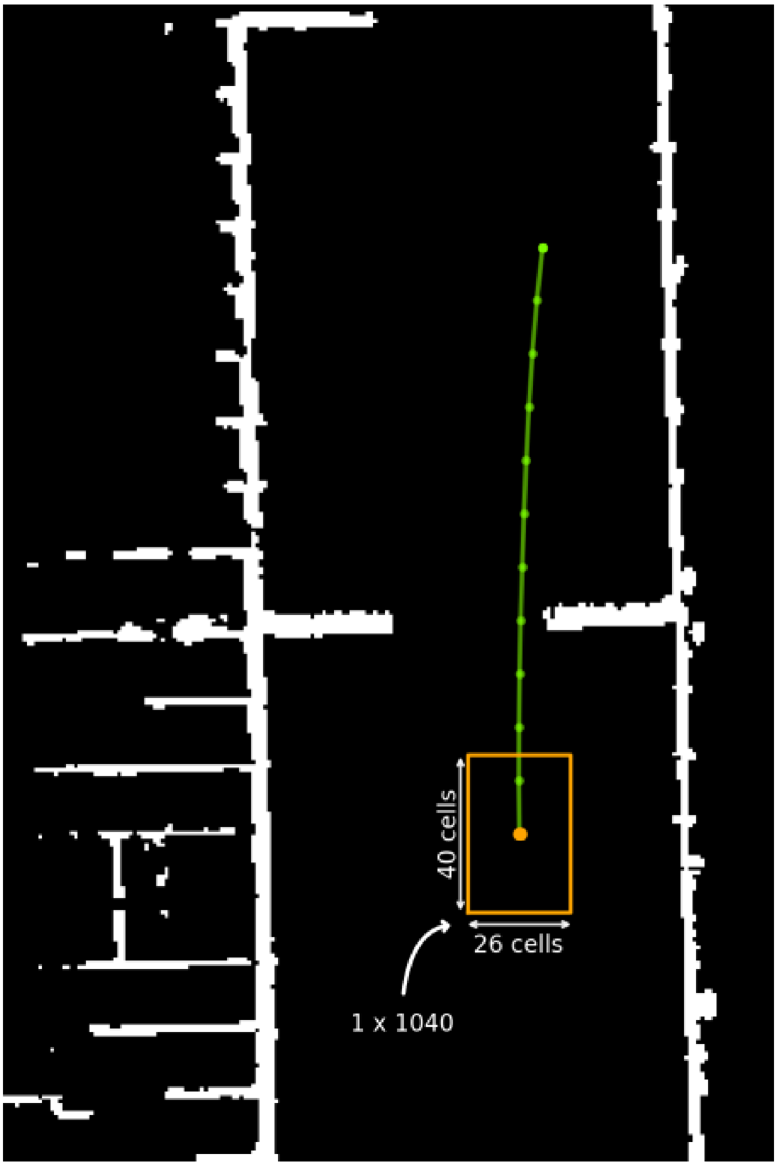


Figure 7.7: Illustration of the occupancy feature matrix extraction.

dimension of 25×2, see Fig. 7.10.

Hybrid Network of LSTM and Distributed Dense Layer

A hybrid neural network architecture is depicted in Fig. 7.11. The LSTM layer consists of one hidden layer, where the hyperbolic tangent function serves as activation function. The output layer of the hybrid architecture is a MLP wrapped in a distributed layer that consist of 2 units with tanh activation, which reshapes the output to dimension of 25×2.

7.4 Neural Network Training

For training the various neural network-based approaches in this thesis TensorFlow 2.3.0 was used. TensorFlow derives its name from operations that neural networks perform on tensors, i.e. multidimensional arrays. It provides a user-friendly front-end API for developing using Python, while the back-end is programmed in C++ for performance reasons.

The presented network architectures for the path optimization task mentioned in 7.3.4 were trained using the mean absolute error (MAE) as loss function and Nadam as optimizer. As can be seen in Fig. 7.12, LSTM based model converges much faster than distributed dense and hybrid based neural networks. Furthermore, it becomes clear that there is no overfitting in the training process as the MAE of training and validation steadily decrease without a large gap in between. That means that the neural network models do not simply memorize the training data, but rather learn the pattern of the path optimization. In terms of model complexity the LSTM-based model has the most parameters, followed by the hybrid network. The DD neural network has the fewest parameters. The model complexity in terms of parameters is directly related to the model size and training duration, see Table 7.1.

Table 7.1: Model complexity in terms of parameters

Neural Network	Number of parameters	Size in KB	Training time
DD	4174	76.5	148 s
LSTM	18944	184.7	215 s
LSTM + DD	8350	95.7	189 s

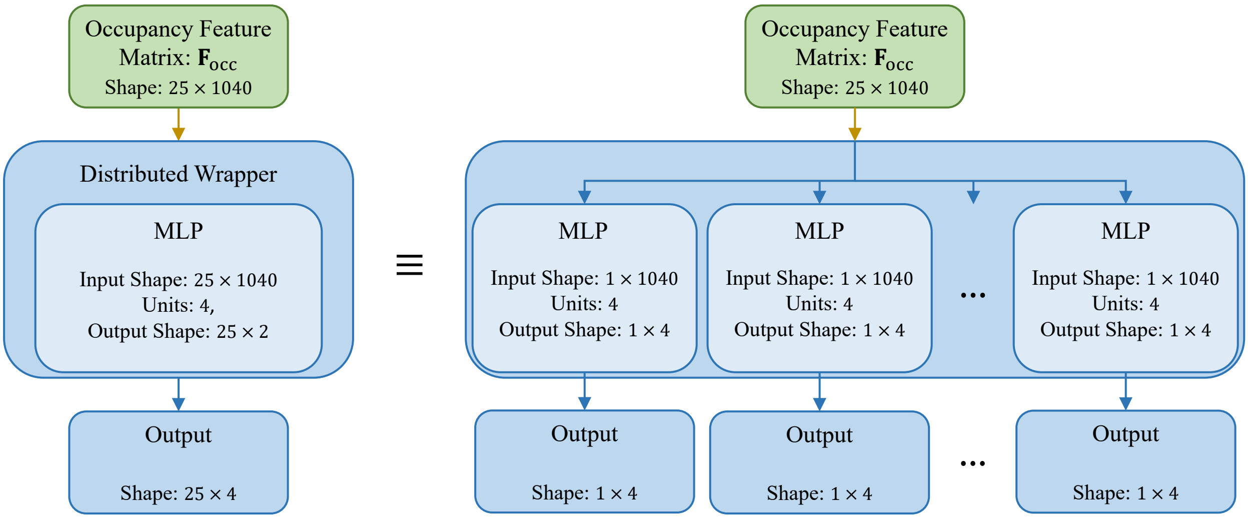


Figure 7.8: Illustration of the distributed dense (DD) layer.

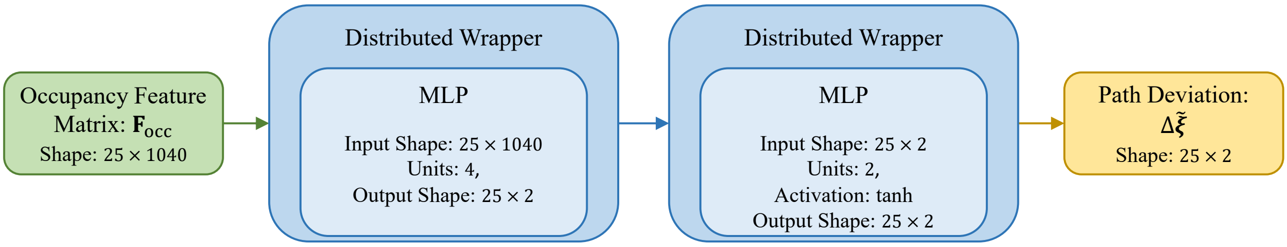


Figure 7.9: Neural network with DD layers.

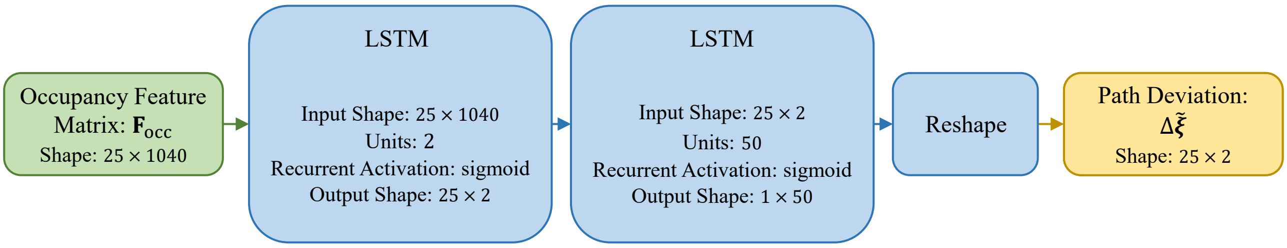


Figure 7.10: Neural network with LSTM layers.

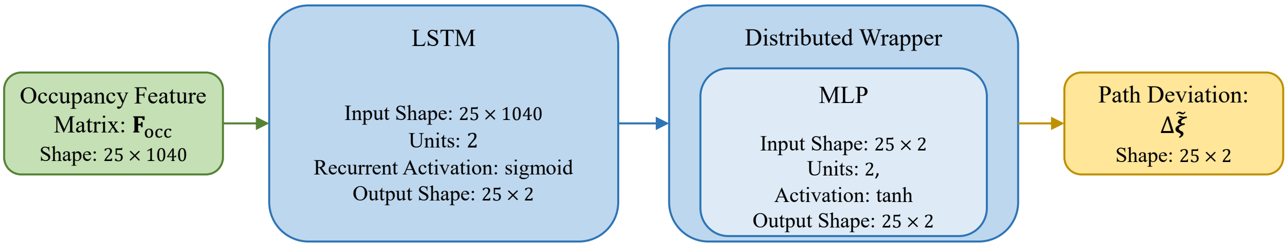


Figure 7.11: Neural network with LSTM and DD layers.

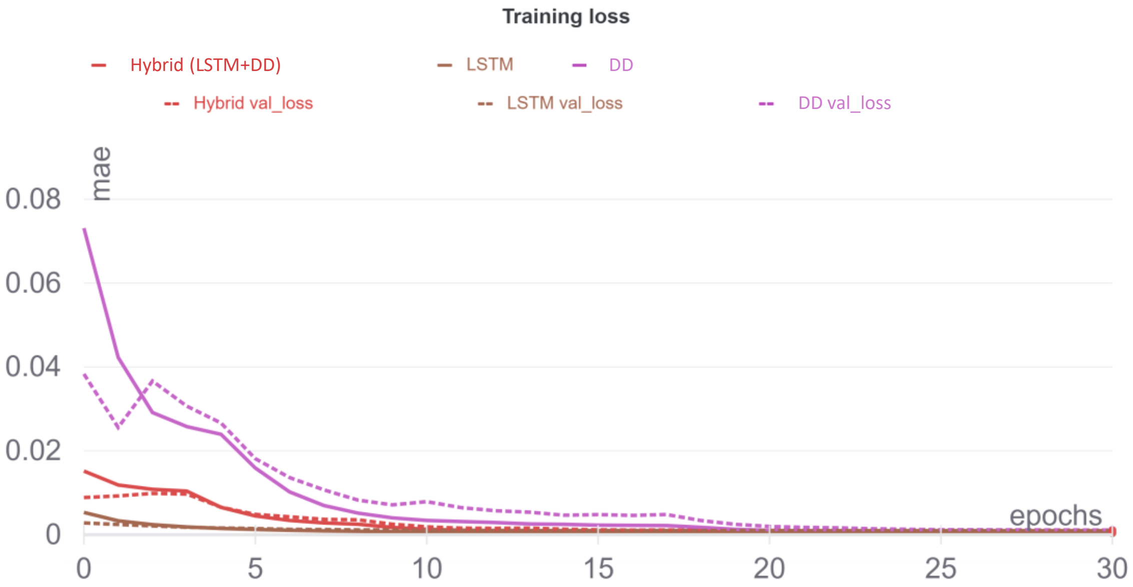


Figure 7.12: Training loss and validation loss of network architectures discussed in Section 7.3.4.

7.4.1 Hyperparameter Tuning

Parameters that influence the learning task of neural networks are referred to as hyperparameters. Hyperparameter tuning is the process of exploring for the ideal hyperparameters to gain the best performing model. In order to generalize various data patterns for the model, the learning rates, weights or different constraints may have to be adapted. Using hyperparameter optimization, an optimal neural network is found that minimizes a specified validation loss function (in our case MAE) on independent data and at the same time finds a tuple of hyperparameters.

Conventional hyperparameter tuning algorithms such as grid search and random search blindly investigate the hyperparameter space, resulting in an exhaustive search process. Bayesian optimization approaches, however, link the model metric and the hyperparameters using Gaussian Processes and, hence, iteratively decide the next hyperparameter candidates based on previous results until the process converges to an optimum [10].

The hyperparameters taken into account for tuning and their ranges are given in Table 7.2 as an example for the LSTM-based neural network. The effects of different hyperparameter combinations regarding the validation loss function are shown in Fig. 7.13. It can be observed that lower learning rates performed better in combination with the nadam optimizer. Satisfactory results were obtained with different amounts of LSTM units in the first layer. Based on this, the model with the fewest units was chosen due to the less required storage capacity of the model and consequently faster calculations. The best number of epochs can be observed between 35 and 70. The best hyperparameter set is stated in Table 7.3.

Table 7.2: Hyperparameters and their ranges for the LSTM-based neural network.

Bayesian optimization		
Parameters	**Ranges**	**Distribution**
LSTM Units in layer1	[2-30]	Integer Uniform
learning rate (lr)	[0.01-0.04]	Uniform
optimizer	[nadam,adam]	Categorical
epochs	[5-100]	Integer Uniform

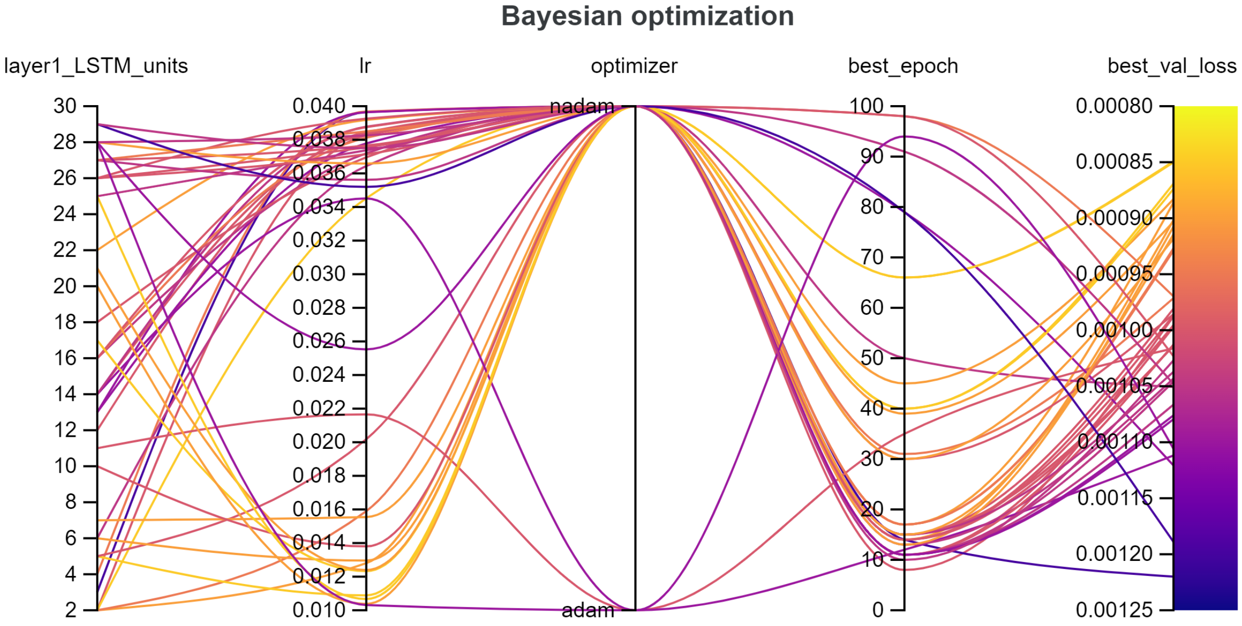


Figure 7.13: Hyperparameter tuning for the LSTM-based neural network using Bayesian optimization.

Table 7.3: Best hyperparameter set for the LSTM-based neural network.

Best hyperparameter set	
Parameters	**Value**
LSTM Units in layer1	2
learning rate (lr)	0.033
optimizer	nadam
epochs	40

7.5 Evaluation

The evaluation of neural networks, in particular the statement that models perform correctly with unknown data, is one of the most critical components of neural network quality. It is crucial to determine a rigorous approach for the evaluation of neural network performance. This is especially important in the context of automated driving, since the system is safety critical. The evaluation is carried out using both: quantitative and qualitative metrics.

7.5.1 Quantitative Results

Quantitative metrics are measurements using certain formulas and are therefore represented numerically. They provide precise information, i.e. facts in mathematical data. For regression tasks, the most applied quantitative metrics are: mean absolute error, mean square error and root mean square error. In this thesis, the evaluation target is to discover how close the predicted path, generated by the neural network, is in relation to the optimal path, generated by the Constrained CHOMP algorithm. This can be determined from distance between the respective waypoints.

Average Displacement Error

A quantitative metric that provides insight into the proximity of given paths is the average displacement error (ADE). This metric refers to the mean square error (MSE) over all waypoints, of the predicted path $\tilde{\xi}_{\mathrm{opt}}$ and the optimal path ξ_{opt}. The ADE is therefore calculated as follows

$$\mathrm{ADE} = \frac{1}{k} \sum_{k=1}^{25} \sqrt{\left(\xi_{\mathrm{opt}}(k) - \tilde{\xi}_{\mathrm{opt}}(k)\right)^2}\,. \tag{7.3}$$

The results of the various model architectures can be seen in Table 7.4.

7.5.2 Qualitative Results

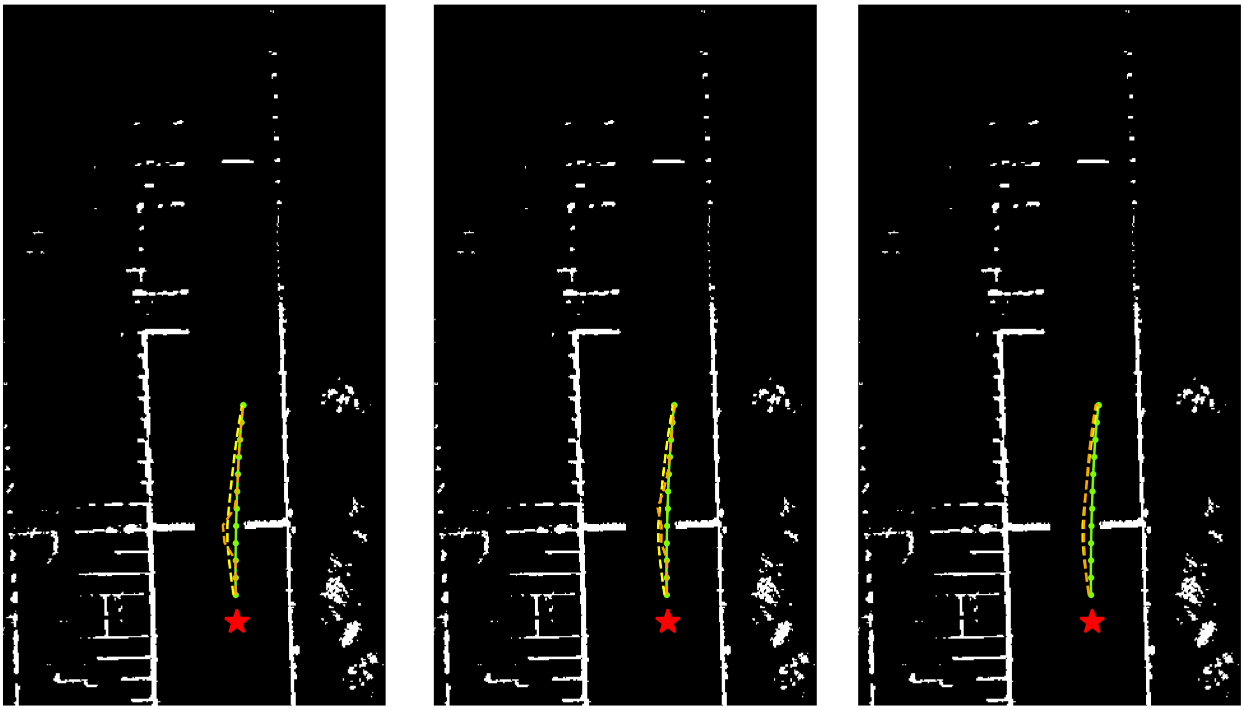
While quantitative metrics measure performance in terms of response time needed, accuracy or likelihood to deviate from, qualitative metrics include subjective assessment, for instance, about usefulness or satisfaction [139]. A visual representation of exemplary results

Table 7.4: Average Displacement Error of various neural network architectures.

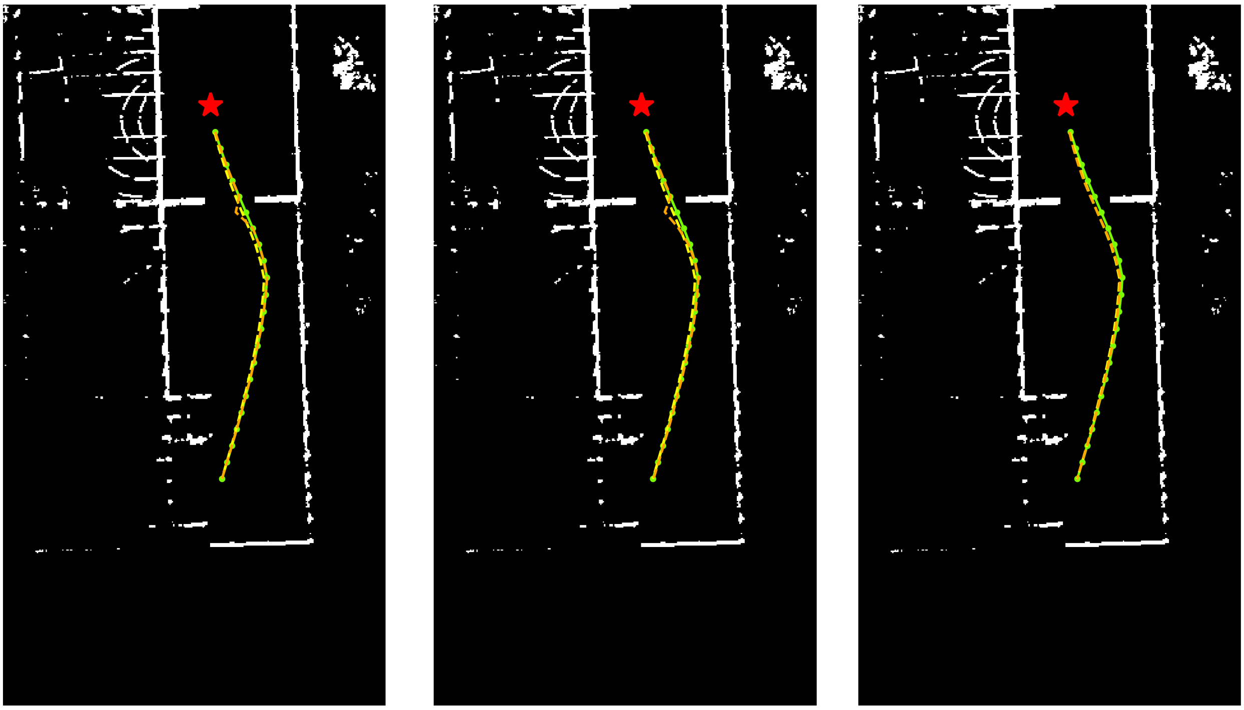
Neural Network	**ADE**
DD	1.094
LSTM	**0.88**
DD + LSTM	1.120

allows even those who are not familiar with the corresponding technical terms to evaluate the model performance qualitatively. While qualitative metrics are simple methods for evaluating a model performance, however, this type of evaluation may need expert supervision and may be exhaustive given large data sets.

The visualization of the optimal path as well as the predicted path and their manual confirmation serve the qualitative evaluation of the model performance in this thesis. Fig. 7.14 shows that the model architecture which contains only MLPs can satisfy the objective of generating collision-free paths. However, this network does not meet the objective of generating smooth paths. The paths generated by model architectures containing MLPs deviate from the initial path when there is a surrounding obstacle, see Fig. 7.14(a). The fact that MLPs are not able to learn sequential relationships between waypoints, as already mentioned, may results in such a behaviour. LSTMs are designed to learn patterns in sequential data, i.e. between waypoints in the path sequence. As a result, the model architecture containing LSTM layers generates smoother paths compared to model architectures based on MLPs as can be seen in Fig. 7.14(c). The hybrid model architecture, which includes a LSTM as the first layer and a distributed dense layer as the second layer, cannot generate smooth paths either, since the output layer is a MLP, see Fig. 7.14(b).



(a) Distributed Dense model (b) Hybrid model (c) LSTM-based model



(d) Distributed Dense model (e) Hybrid model (f) LSTM-based model

Figure 7.14: Path prediction results using two scenarios indicating: the initial path in green (operator specified), the optimal path in yellow (generated by Constrained CHOMP), the predicted path in orange (generated by neural network) and the ego vehicle position as a red star.

7.5.3 Robustness Evaluation

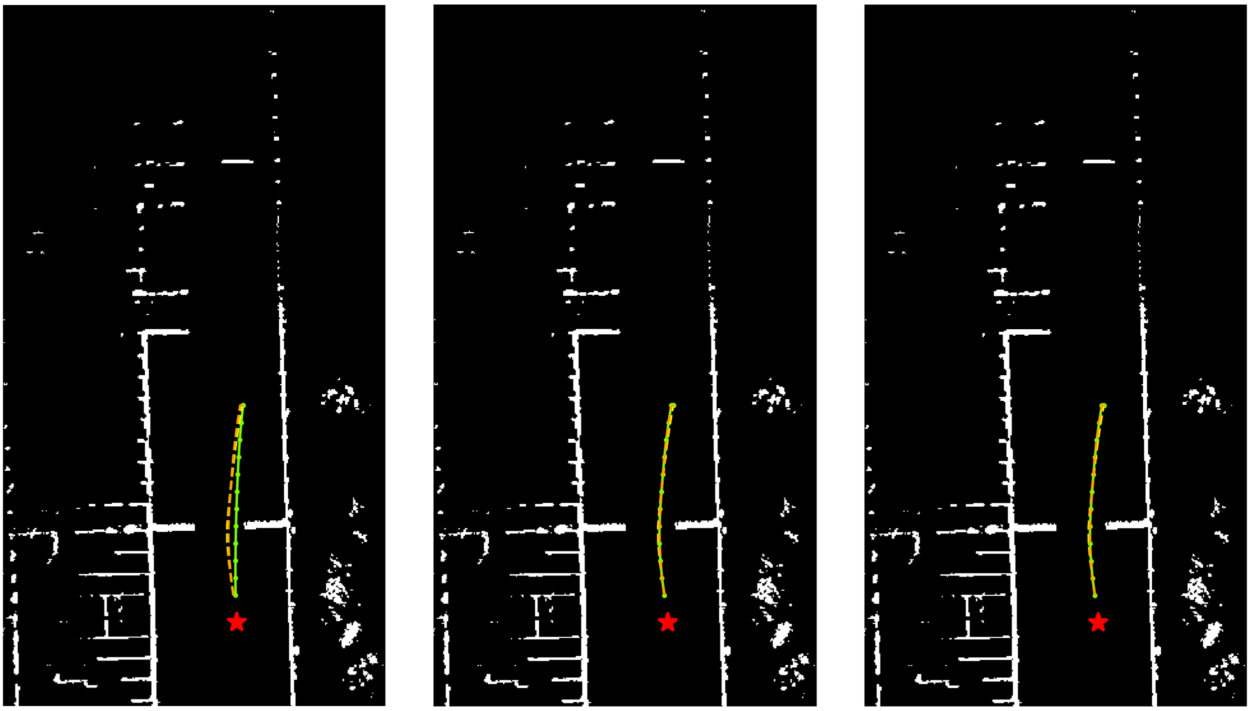
Safety has a crucial role in autonomous driving. It is therefore imperative that the behaviour of the neural network is consistent and explainable. In order to investigate the robustness and the validity of the neural network, iterative predictions are carried out based on paths that were predicted by the neural network in previous steps. Since optimal paths are also generated iteratively using Constrained CHOMP, the robustness evaluation process is defined as follows:

1. Given the occupancy grid map and the initial path ξ_{init}: Calculate the occupancy feature matrix $\mathbf{F}_{\text{occ}}$.
2. Predict the deviation $\Delta\tilde{\xi}$ based on the occupancy feature matrix $\mathbf{F}_{\text{occ}}$, then calculate $\tilde{\xi}_{\text{opt}} = \xi_{\text{init}} - \Delta\tilde{\xi}$.
3. Update the occupancy feature matrix $\mathbf{F}_{\text{occ}}$ based on $\tilde{\xi}_{\text{opt}}$.
4. Back to step 2 and iterate.

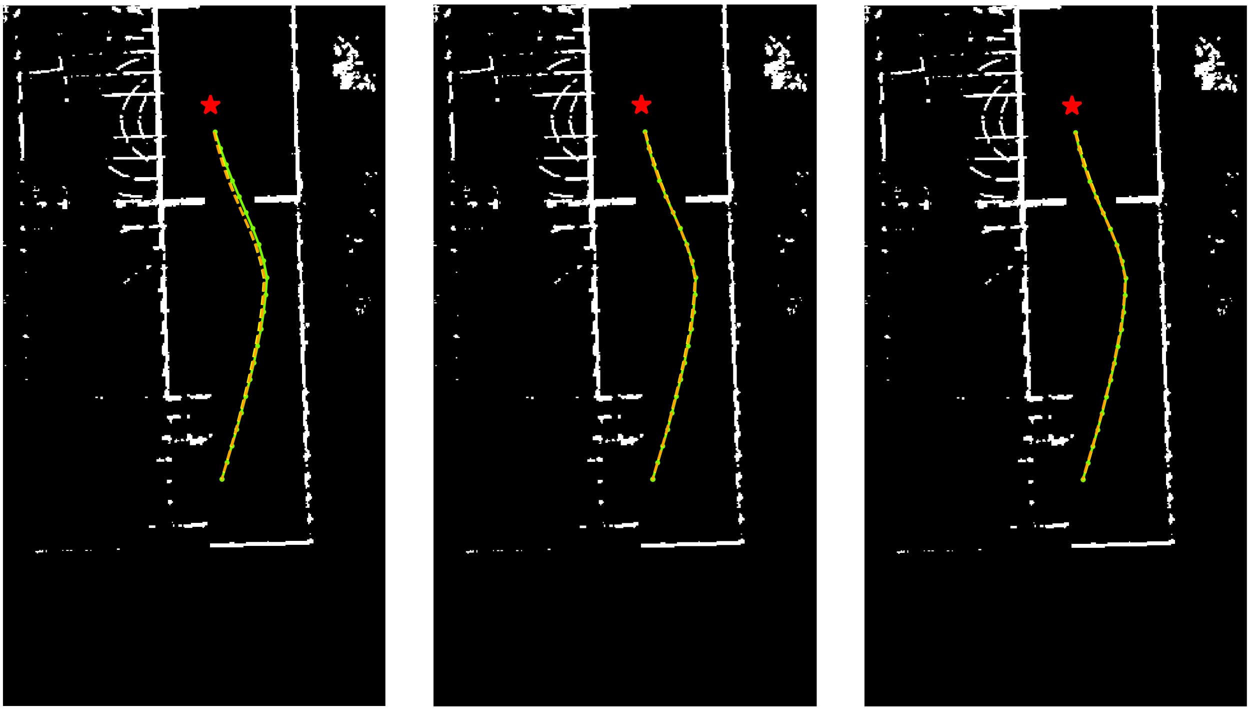
This method allows making a statement about the robustness of the neural network and thus to show whether the model only optimizes the path when it is necessary. Due to the advantageous properties and results, demonstrated by previous quantitative and qualitative metrics, the robustness evaluation is visualized in Fig. 7.15 for the LSTM based neural network. Here, it can be observed, that in the first iteration, Fig. 7.15 (a) and (b), the neural network is able to predict the optimal path sufficiently accurate. The path predicted by the neural network is almost identical to the optimal path. An ADE of less that 0.6 is achieved in both scenarios. In further iterations it can be seen that the path is not optimized further. This underlines its consistency in predicting optimal paths.

7.6 Conclusion

This chapter presents an alternative path planning method based on deep learning to the modified, two-step Constrained CHOMP algorithm, developed in Chapter 2. Different neural network architectures are examined, which are suitable for path optimization taking into account the smoothness factor and collisions. The LSTM based neural network shows the highest model complexity, however, outperforms other models. The required memory and time constraints represent key factors in real-time systems. The LSTM based neural network occupies around 185 kB. The calculations were carried out on an Intel i7-8565U 1.80 HHz processor with an average calculation time per calculation step of 1.8 ms for occupancy feature matrix extraction and 0.7 ms for the path prediction. In summary, the total calculation time resulted in around 2.5 ms per calculation step. Considering the less powerful processor compared to the experiments in Chapter 5 and 6, where the path optimization takes 8 ms using the modified, two-step Constrained CHOMP algorithm, a noticeable improvement in computing effort is achieved using the deep learning approach.



(a) Scenario 1, iteration: 1
(b) Scenario 1, iteration: 2
(c) Scenario 1, iteration: 3



(d) Scenario 2, iteration: 1
(e) Scenario 2, iteration: 2
(f) Scenario 2, iteration: 3

Figure 7.15: Robustness evaluation of the LSTM neural network using iterative predictions in two scenarios.

8 Conclusion and Outlook

8.1 Conclusion

The teleoperation of road vehicles is intended to resolve complex, urban scenarios in which the current intelligence of autonomous vehicles is insufficient. In order to increase the safety and precision in urban environments, several teleoperated driving concepts for direct and indirect control are proposed in this thesis. The proposed concepts raise the level of autonomy in different ways, while the operator remains the main decision maker in all driving tasks. For this purpose, real-time capable algorithms for path and trajectory planning are developed that aim for safe and comfortable motions of the remote controlled vehicle.

The proposed modified, two-step Constrained CHOMP algorithm introduces a second optimization step, the domain step, to prevent the path to get stuck in poor local minima. The subsequent smoothing step, including the original formulation of the objective functional, is modified in order to take the vehicle dimensions into account. Furthermore, additional objective functions are introduced that address the vehicle non-holonomic constraints and penalize the distance to a reference path that represents a guidance planned by the human operator. For the application of automated driving, the proposed modified, two-step Constrained CHOMP algorithm allows the generation of optimal solution including the global minimum. Although global path search algorithms exist for this purpose, the proposed path optimization is able to generate an optimal path with significantly less computational effort.

For the planning of optimal trajectories in real-time, efficient formulations of constrained linear-quadratic optimal control problems are derived separately for the longitudinal and lateral dynamics. They are solved by means of a time-variant, linear MPC scheme using Quadratic Programming. To guarantee the solvability of the optimization problems, constraint softening is applied by linking the safety-related longitudinal as well as lateral obstacle distances to a heavily weighted slack variable. In addition to hard constraints regarding the corresponding accelerations, comfort-related longitudinal and lateral distances to obstacles are defined and linked to a less weighted slack variable in order to account for comfort in the trajectory optimization. By introducing the two-stage constraint softening for the longitudinal as well as the lateral obstacle distance constraints in combination with a suitable choice of the weighting of the comfort slack variable, a compromise between a comfortable distance to obstacles and a smooth trajectory profile in a dynamic maneuver is achieved. Although the proposed trajectory optimization design only enables the formulation of locally optimal problems, the combination with the modified, two-step Constrained CHOMP algorithm leads to global optimal trajectories. Compared with state-of-the-art motion planning algorithms, the results of the novel hybrid motion planning method indicates a significant increase in computational efficiency.

A teleoperated driving concept proposed in this thesis is the model-predictive cruise control for direct teloperated driving tasks. This concept aims to relieve the human operator in complex teleoperation scenarios in which the human is directly involved in the closed control loop. The model-predictive cruise control approach uses the advantageous properties of the developed trajectory optimization for the longitudinal guidance. It adapts on-board the operator's control commands for the longitudinal dynamics in real-time in order to address both safety and comfort while considering other road participants. This concept aims in particular to support operators in critical situations, when they do not react appropriately in a driving maneuver, for example due to an incorrect assessment of the possible driving space or due to the delayed or even interrupted communication with the ego vehicle. This concept is successfully validated using two real driving scenarios: a suddenly appearing road participant at an intersection with right of way and an abruptly braking vehicle in front.

The corridor-based motion planning concept proposed in this thesis represents a novel shared control approach. This approach aims in particular to overcome varying or high communication time delays to the ego vehicle by introducing highly automated driving functions. In contrast to direct control, in this indirect control concept the operator is kept outside the closed control loop. The guidance loop is closed autonomously by the vehicle. Therefore, the closed control loop is insensitive w.r.t any communication time delays. By taking advantage of the human abilities in decision making, the operator is enabled to specify an area – the corridor – towards a desired destination. Therefore, the operator is able to interactively take into account, for instance, missing or inadequate lane markings or untracked obstacles by the system. For this purpose, this concept is supported by both camera and LiDAR measurements. The operator is able to take advantage of the sensor measurements and to define the boundaries of the corridor according to the perception. The operator decides between specifying a complete corridor to the target destination in advance or initially a sub-corridor using this method. The automated vehicle calculates and executes an optimal motion on its own in real-time within the specified corridor. As the automated vehicle progresses within the sub-corridor, the operator is able to append further corridor segments. For motion planning, the novel hybrid motion planning approach is implemented, that determines, in the first phase, an optimal path for the static environment using the modified, two-step Constrained CHOMP algorithm. In the second phase, a collision-free trajectory is generated online along the optimal path using two separate linear-quadratic problem formulations for both, longitudinal and lateral dynamics. They are efficiently solved by means of a time-variant, linear MPC scheme using Quadratic Programming taking into account safety and comfort related requirements resulting from automated driving. This concept is successfully validated using complex simulations and several real driving scenarios including dynamic obstacles.

To support the operator in the planning task, this thesis presents an automatic path generation approach for supervisory control. In this indirect control concept, again, the operator is kept outside the closed control loop. The guidance loop is closed autonomously by the vehicle. Therefore, the closed control loop is insensitive w.r.t any communication time delays. This concept aims to relieve the operator in complex urban scenarios by continuously generating further feasible paths and then forward these to the operator. The operator decides which path will be followed by vehicle. For this purpose a global path search algorithm, the RRT algorithm, is modified in such a way that, without the operator specifying target positions, the vehicle environment is explored and reasonable paths are generated. Before the remote controlled vehicle stops at the end of a selected path, the algorithm generates new path suggestions beforehand. As the vehicle navigates as selected by the operator, the modified, two-step Constrained CHOMP algorithm ensures that no collision occurs by optimizing the followed path in real-time. The concept is successfully validated in real driving experiments using complex urban scenarios.

Finally, this thesis introduces a novel concept for path optimization based on deep learning. For this purpose different problem statements together with different neural network architectures are examined. Based on data generated by the modified, two-step Constrained CHOMP algorithm, a LSTM based neural network is derived that outperforms noticeably the modified, two-step Constrained CHOMP algorithm in terms of memory usage as well as computational effort. The alternative path planning concept using the LSTM based neural network is successfully validated using real data recorded in a parking area.

8.2 Outlook

In future work, the combination of the presented teleoperated driving concepts for the step-by-step solution of a single teleoperated driving task could be investigated. One possibility is that if the operator has just been contacted to resolve a teleoperated driving task, the automated system will suggest several feasible paths to the operator. After confirming a path, the operator specifies corridor boundaries to account for, for instance, missing lane markings or untracked obstacles in the motion planning. If there is no communication latency problem and the operator feels ready, the operator can directly take over the control of the remote-controlled vehicle. In the case of direct control, the automated system further generates new path suggestions that the operator can choose between and can thus hand over the control of the vehicle back to the automated system.

To decrease the computational effort, the path optimization using the modified, two-step Constrained CHOMP algorithm may be replaced by the proposed LSTM based neural network. However, to cover all possible driving scenarios more corresponding data is required. In addition, a neural network could be developed that dispenses the LiDAR data and only uses camera measurements. This could be accomplished by transforming the camera measurements into a two-dimensional grid, comparable to the occupancy grid map. This would eliminate the high costs of the LiDAR sensor.

A Mathematical Explanations

A.1 The Curvature Vector

The curvature of a path characterizes the amount by which the given path deviates from a straight line. From differential geometry, c.f. [30], the curvature $\kappa(s)$ for a path $\boldsymbol{\xi}(s)$ parameterized by arc length is defined as

$$\kappa(s) = \left\| \frac{\mathrm{d}^2\boldsymbol{\xi}(s)}{\mathrm{d}s^2} \right\| = ||\boldsymbol{\xi}(s)''|| \,. \tag{A.1}$$

Considering an osculating circle, see Fig. A.1, that has a second order contact (equal curvature) with the path $\boldsymbol{\xi}(s)$ at given point $\mathbf{q}$: The arc length parameterization for the circle with radius ρ centered at the origin in the plane, is defined by

$$\begin{aligned} x &= \rho \cos \frac{s}{\rho} \,, \\ y &= \rho \sin \frac{s}{\rho} \,. \end{aligned} \tag{A.2}$$

Differentiating (A.2) w.r.t. s results in

$$\begin{aligned} x' &= -\sin \frac{s}{\rho} \,, \\ y' &= \cos \frac{s}{\rho} \,. \end{aligned} \tag{A.3}$$

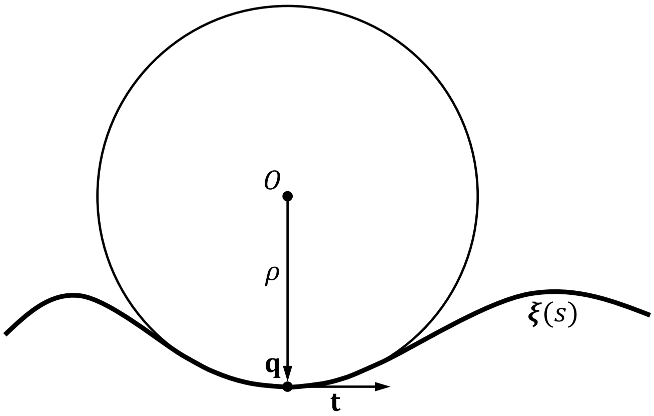


Figure A.1: The osculating circle for the point $\mathbf{q}$ on the path $\boldsymbol{\xi}(s)$. The tangent of the path at the given point is represented by the vector $\mathbf{t}$.

With the arc length parameterization, the magnitude of the velocity vector $[x', y']$ is equal for all s. By differentiating (A.3) w.r.t. s

$$\begin{aligned} x'' &= -\frac{1}{\rho}\cos\frac{s}{\rho}\,, \\ y'' &= -\frac{1}{\rho}\sin\frac{s}{\rho}\,, \end{aligned} \tag{A.4}$$

it can be observed, that the magnitude of the acceleration vector $[x'', y'']$ equals $\frac{1}{\rho}$. Assuming the path is arc length parameterized as well, according to [106] the magnitude of $\boldsymbol{\xi}''$ should equal the magnitude $[x'', y'']$, i.e.

$$\kappa(s) = ||\boldsymbol{\xi}(s)''|| = \frac{1}{\rho(s)}\,. \tag{A.5}$$

The curvature vector $\boldsymbol{\kappa}(s)$ is therefore defined as the vector of magnitude $\kappa(s)$ that points from the point $\mathbf{q}$ on the path towards the center O:

$$\boldsymbol{\kappa}(s) = \boldsymbol{\xi}''(s)\,. \tag{A.6}$$

Considering a path $\boldsymbol{\xi}(s^*)$ that is not parameterized by arc length, differentiating the path twice w.r.t. arc length s, the following applies

$$\begin{aligned} \frac{\mathrm{d}^2\boldsymbol{\xi}}{\mathrm{d}s^2} &= \frac{\mathrm{d}}{\mathrm{d}s}\left(\boldsymbol{\xi}'\frac{\mathrm{d}s^*}{\mathrm{d}s}\right), \\ &= \boldsymbol{\xi}''\left(\frac{\mathrm{d}s^*}{\mathrm{d}s}\right)^2 + \boldsymbol{\xi}'\frac{\mathrm{d}^2 s^*}{\mathrm{d}s^2}\,. \end{aligned} \tag{A.7}$$

Since s represents the arc length of the path $\boldsymbol{\xi}$, the relation of s and s^* is characterized by

$$s(s^*) = \int_0^{s^*} ||\boldsymbol{\xi}'||\mathrm{d}s^*\,. \tag{A.8}$$

Based on the first differentiation of (A.8) w.r.t. s^*

$$\frac{\mathrm{d}s}{\mathrm{d}s^*} = ||\boldsymbol{\xi}'||\,, \tag{A.9}$$

the following expression results:

$$\frac{\mathrm{d}s^*}{\mathrm{d}s} = \frac{1}{||\boldsymbol{\xi}'||}\,. \tag{A.10}$$

By differentiating the given expression (A.10) w.r.t. s

$$\begin{aligned} \frac{\mathrm{d}^2 s^*}{\mathrm{d}s^2} &= \left(\frac{\mathrm{d}}{\mathrm{d}s^*}\frac{1}{||\boldsymbol{\xi}'||}\right)\frac{\mathrm{d}s^*}{\mathrm{d}s} = \left(\frac{\mathrm{d}}{\mathrm{d}s^*}\frac{1}{\sqrt{\boldsymbol{\xi}'^\top\boldsymbol{\xi}'}}\right)\frac{\mathrm{d}s^*}{\mathrm{d}s}\,, \\ &= \left(-\frac{1}{2}\left(\boldsymbol{\xi}'^\top\boldsymbol{\xi}'\right)^{-\frac{3}{2}}\cdot 2\,\boldsymbol{\xi}'^\top\boldsymbol{\xi}''\right)\frac{1}{||\boldsymbol{\xi}'||} = \left(-\frac{\boldsymbol{\xi}'^\top\boldsymbol{\xi}''}{||\boldsymbol{\xi}'||^3}\right)\frac{1}{||\boldsymbol{\xi}'||}\,, \\ &= -\frac{\boldsymbol{\xi}'^\top\,\boldsymbol{\xi}''}{||\boldsymbol{\xi}'||^4} \end{aligned} \tag{A.11}$$

and substituting the obtained result into (A.7), the curvature vector can be calculated by

$$\kappa(s) = \frac{\mathrm{d}^2\xi}{\mathrm{d}s^2} = \frac{\xi''}{||\xi'||^2} - \xi'\xi'^\top \frac{\xi''}{||\xi'||^4}\,. \tag{A.12}$$

A.2 The Dynamics of a non-holonomic Vehicle in a Frenét Frame

To model the vehicle dynamics in the Frenet frame, the relation between the vehicle velocity $v(t)$ and the first derivative of the Frenét coordinate s(t), i.e. $\dot{s}(t)$, is required [132]. As can be seen in Fig. A.2, the point $\mathbf{q}_\Gamma(s(t)) = [x_\Gamma(s(t)),\ y_\Gamma(s(t))]^\top$ on the path Γ represents the shortest distance to the vehicle's rear axle middle point $\mathbf{q}_v(s_v(t)) = [x(s_v(t)),\ y(s_v(t))]^\top$. Therefore, the connection between these two points and the tangent $\mathbf{t}_\Gamma(s(t))$ of the path form a right angle, what can be expressed by

$$\left[\mathbf{q}_v(s_v(t)) - \mathbf{q}_\Gamma(s(t))\right]^\top \mathbf{t}_\Gamma(s(t)) = 0\,. \tag{A.13}$$

With the time derivative of (A.13) applies:

$$\left[\frac{\mathrm{d}\mathbf{q}_v}{\mathrm{d}s_v}\dot{s}_v - \frac{\mathrm{d}\mathbf{q}_\Gamma}{\mathrm{d}s}\dot{s}\right]^\top \mathbf{t}_\Gamma + \left[\mathbf{q}_v - \mathbf{q}_\Gamma\right]^\top \frac{\mathrm{d}\mathbf{t}_\Gamma}{\mathrm{d}s}\dot{s} = 0\,. \tag{A.14}$$

By using Frenét's Formula $\mathbf{t}'_\Gamma = \kappa_\Gamma \mathbf{n}_\Gamma$ and

$$\mathbf{n}_\Gamma = \frac{\mathbf{q}_v - \mathbf{q}_\Gamma}{||\mathbf{q}_v - \mathbf{q}_\Gamma||} = \frac{\mathbf{q}_v - \mathbf{q}_\Gamma}{d}\,, \tag{A.15}$$

(A.14) is reformulated with $\dot{s}_v = v$ as follows

$$\begin{aligned} \left[\mathbf{t}_v v - \mathbf{t}_\Gamma \dot{s}\right]^\top \mathbf{t}_\Gamma - \kappa_\Gamma d\,\dot{s} &= 0\,,\\ v\left(\cos\theta\cos\theta_\Gamma + \sin\theta\sin\theta_\Gamma\right) - \dot{s} - \kappa_\Gamma d\,\dot{s} &= 0\,,\\ v\cos\left(\theta - \theta_\Gamma\right) - \dot{s} - \kappa_\Gamma d\,\dot{s} &= 0\,, \end{aligned} \tag{A.16}$$

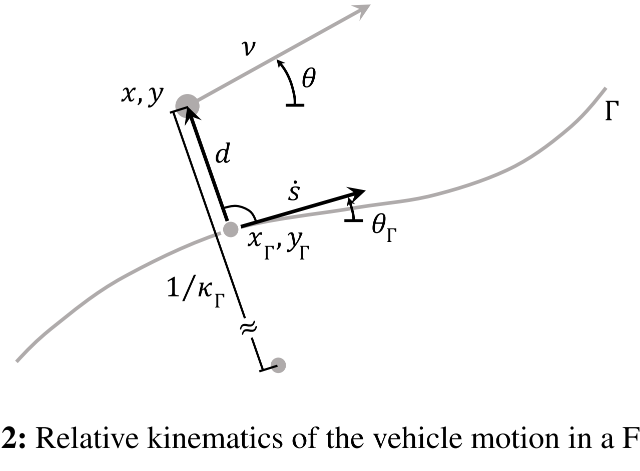


Figure A.2: Relative kinematics of the vehicle motion in a Frenét frame.

which results in

$$\dot{s} = v \frac{\cos(\theta - \theta_\Gamma)}{1 - d\kappa_\Gamma} \,. \tag{A.17}$$

The time derivative of

$$d^2 = \left[\mathbf{q}_v(s_v) - \mathbf{q}_\Gamma(s)\right]^\top \left[\mathbf{q}_v(s_v) - \mathbf{q}_\Gamma(s)\right] \tag{A.18}$$

yields

$$\begin{aligned} 2d\dot{d} &= 2\left[\mathbf{q}_v - \mathbf{q}_\Gamma\right]^\top \left[\mathbf{t}_v v - \mathbf{t}_\Gamma \dot{s}\right] \,, \\ &= 2\, d\, \mathbf{n}_\Gamma^\top \left[\mathbf{t}_v v - \mathbf{t}_\Gamma \dot{s}\right] \,, \\ \dot{d} &= v\, \mathbf{n}_\Gamma^\top \mathbf{t}_v \,, \\ &= v\left(-\sin\theta_\Gamma \cos\theta + \cos\theta_\Gamma \sin\theta\right) \,, \\ &= v \sin(\theta - \theta_\Gamma) \,. \end{aligned} \tag{A.19}$$

Bibliography

[1] Sterling J. Anderson, Steven C. Peters, Tom E. Pilutti, and Karl Iagnemma. An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. *International Journal of Vehicle Autonomous Systems*, 8(2-4):190–216, 2010.

[2] Behrang Asadi and Ardalan Vahidi. Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time. *IEEE transactions on control systems technology*, 19(3):707–714, 2010.

[3] Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles Reinholtz, Dennis Hong, Al Wicks, Thomas Alberi, David Anderson, et al. Odin: Team victortango's entry in the DARPA urban challenge. *Journal of field Robotics*, 25(8):467–492, 2008.

[4] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Meireles Paixão, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, page 113816, 2020.

[5] Christopher R. Baker and John M. Dolan. Traffic interaction in the urban challenge: Putting boss on its best behavior. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1752–1758, 2008.

[6] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. *Algorithm engineering*, pages 19–80, 2016.

[7] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[8] Alberto Bemporad and Claudio Rocchi. Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles. In *50th IEEE conference on decision and control and European control conference*, pages 7488–7493. IEEE, 2011.

[9] Klaus Bengler, Klaus Dietmayer, Berthold Farber, Markus Maurer, Christoph Stiller, and Hermann Winner. Three decades of driver assistance systems: Review and future perspectives. *IEEE Intelligent transportation systems magazine*, 6(4):6–22, 2014.

[10] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *International conference on machine learning*, pages 115–123, 2013.

[11] Johannes Betz, Alexander Wischnewski, Alexander Heilmeier, Felix Nobis, Tim Stahl, Leonhard Hermansdorfer, Boris Lohmann, and Markus Lienkamp. What can we learn from autonomous level-5 motorsport? *9th International Munich Chassis Symposium 2018*, pages 123–146, 2019.

[12] Jonathan Bohren, Tully Foote, Jim Keller, Alex Kushleyev, Daniel Lee, Alex Stewart,

Paul Vernaza, Jason Derenick, John Spletzer, and Brian Satterfield. Little ben: The ben franklin racing team's entry in the 2007 darpa urban challenge. *Journal of Field Robotics*, 25(9):598–614, 2008.

[13] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. Predictive control for linear and hybrid systems. *Cambridge University Press*, 2017.

[14] Pascal Fabian Bosshard. Investigation of trajectory optimization for multiple car-like vehicles. Technical report, School of Information Science, Computer and Electrical Engineering, Halmstad University, 2015.

[15] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps. *17th International Conference on Intelligent Transportation Systems (ITSC)*, pages 392–399, 2014.

[16] Victor Caragiu. Self driving automobiles. *Technical Scientific Conference of undergraduate, master and PhD students*, pages 388–391, 2020.

[17] Ashwin Carvalho, Yiqi Gao, Andrew Gray, H Eric Tseng, and Francesco Borrelli. Predictive control of an autonomous ground vehicle using an iterative linearization approach. In *16th International IEEE conference on intelligent transportation systems (ITSC 2013)*, pages 2335–2340. IEEE, 2013.

[18] Luis I Reyes Castro, Pratik Chaudhari, Jana Tumova, Sertac Karaman, Emilio Frazzoli, and Daniela Rus. Incremental sampling-based algorithm for minimum-violation motion planning. *52nd Conference on Decision and Control*, pages 3217–3224, 2013.

[19] Pratik Chaudhari, Tichakorn Wongpiromsarny, and Emilio Frazzoli. Incremental minimum-violation control synthesis for robots interacting with external agents. *American Control Conference*, pages 1761–1768, 2014.

[20] Amit Chaulwar, Michael Botsch, and Wolfgang Utschick. A machine learning based biased-sampling approach for planning safe trajectories in complex, dynamic traffic-scenarios. *IEEE Intelligent Vehicles Symposium (IV)*, pages 297–303, 2017.

[21] Tang Chen and Tito Lu. *Methods for improving the control of teleoperated vehicles*. PhD thesis, Technical University of Munich, 2015.

[22] Sanjiban Choudhury and Sebastian Scherer. Constrained CHOMP using dual projected newton method. *Carnegie Mellon University, Pittsburgh, Tech. Rep.*, 2016.

[23] Hongqing Chu, Lulu Guo, Bingzhao Gao, Hong Chen, Ning Bian, and Jianguang Zhou. Predictive cruise control using high-definition map and real vehicle implementation. *IEEE Transactions on Vehicular Technology*, 67(12):11377–11389, 2018.

[24] Frederic Chucholowski. *Eine vorausschauende Anzeige zur Teleoperation von Straßenfahrzeugen (in German)*. PhD thesis, Technical University of Munich, 2015.

[25] Frederic Chucholowski, Tito Tang, and Markus Lienkamp. Teleoperiertes Fahren: Sichere und robuste Datenverbindungen (in German). *ATZelektronik*, 9(1):60–63, 2014.

[26] Comma.ai. Openpilot. *https://github.com/commaai/openpilot*, 2019.

[27] B Cooper. Driving on the surface of mars using the rover control workstation. Technical report, Jet Propulsion Laboratory, National Aeronautics and Space, Pasadena, 1998.

[28] Ewart J. de Visser, Richard Pak, and Tyler H. Shaw. From 'automation' to 'auton-

omy': The importance of trust repair in human–machine interaction. *Ergonomics*, 61(10):1409–1427, 2018.

[29] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[30] Manfredo P. Do Carmo. *Differential geometry of curves and surfaces*. Prentice-Hall, 1976.

[31] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The international journal of robotics research*, 29(5):485–501, 2010.

[32] Pedro M d'Orey, Amin Hosseini, José Azevedo, Frank Diermeyer, Michel Ferreira, and Markus Lienkamp. Hail-a-drone: Enabling teleoperated taxi fleets. *IEEE Intelligent Vehicles Symposium (IV)*, pages 774–781, 2016.

[33] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *International Conference on Knowledge Discovery and Data Mining*, 96(34):226–231, 1996.

[34] Paolo Falcone, Manuela Tufo, Francesco Borrelli, Jahan Asgari, and H Eric Tseng. A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems. In *2007 46th IEEE Conference on Decision and Control*, pages 2980–2985. IEEE, 2007.

[35] Brian Fildes, Michael Keall, Niels Bos, Anders Lie, Yves Page, Claus Pastor, Lucia Pennisi, Matteo Rizzi, Pete Thomas, and Claes Tingvall. Effectiveness of low speed autonomous emergency braking in real-world rear-end crashes. *Accident Analysis & Prevention*, 81:24–29, 2015.

[36] Rolf Findeisen and Frank Allgöwer. An introduction to nonlinear model predictive control. *21st Benelux meeting on systems and control*, 11:119–141, 2002.

[37] Terrence Fong and Charles Thorpe. Vehicle teleoperation interfaces. *Autonomous robots*, 11(1):9–18, 2001.

[38] Bernhard Friedrich. The effect of autonomous vehicles on traffic. In *Autonomous Driving*, pages 317–334. Springer, 2016.

[39] Jannik Fritsch, Tobias Kühnl, and Franz Kummert. Monocular road terrain detection by combining visual and spatial information. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1586–1596, 2014.

[40] Andrei Furda and Ljubo Vlacic. Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making. *IEEE Intelligent Transportation Systems Magazine*, 3(1):4–17, 2011.

[41] Jonathan D Gammell, Timothy D Barfoot, and Siddhartha S Srinivasa. Batch informed trees (BIT*): Informed asymptotically optimal anytime search. *The International Journal of Robotics Research*, 39(5):543–567, 2020.

[42] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004, 2014.

[43] Yiqi Gao, Andrew Gray, Janick V Frasch, Theresa Lin, Eric Tseng, J Karl Hedrick, and

Francesco Borrelli. Spatial predictive control for agile semi-autonomous ground vehicles. *Proceedings of the 11th international symposium on advanced vehicle control*, (2):1–6, 2012.

[44] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[45] Sebastian Gnatzig. *Trajektorienbasierte Teleoperation von Straßenfahrzeugen auf Basis eines Shared-Control-Ansatzes (in German)*. PhD thesis, Technical University of Munich, 2015.

[46] Sebastian Gnatzig, Florian Schuller, and Markus Lienkamp. Human-machine interaction as key technology for driverless driving – a trajectory-based shared autonomy control approach. *IEEE International Symposium on Robot and Human Interactive Communication*, pages 913–918, 2012.

[47] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2015.

[48] Gaetano Graf, Yomna Abdelrahman, Hao Xu, Yasmeen Abdrabou, Dmitrij Schitz, Heinrich Hußmann, and Florian Alt. The predictive corridor: A virtual augmented driving assistance system for teleoperated autonomous vehicles. *International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments (ICAT-EVGE)*, pages 61–69, 2020.

[49] Gaetano Graf, Hao Xu, Dmitrij Schitz, and Xiao Xu. Improving the prediction accuracy of predictive displays for teleoperated autonomous vehicles. *IEEE 6th International Conference on Control, Automation and Robotics (ICCAR)*, pages 440–445, 2020.

[50] Alfred Gray, Elsa Abbena, and Simon Salamon. *Modern differential geometry of curves and surfaces with Mathematica®*. Chapman and Hall/CRC, 2017.

[51] Andrew Gray, Mohammad Ali, Yiqi Gao, J Hedrick, and Francesco Borrelli. Semi-autonomous vehicle control for road departure and obstacle avoidance. *IFAC control of transportation systems*, pages 1–6, 2012.

[52] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[53] Sorin Mihai Grigorescu, Bogdan Trasnea, Liviu Marina, Andrei Vasilcoi, and Tiberiu Cocias. Neurotrajectory: A neuroevolutionary approach to local state trajectory learning for autonomous vehicles. *IEEE Robotics and Automation Letters*, 4(4):3441–3448, 2019.

[54] Lars Grüne and Jürgen Pannek. Nonlinear model predictive control. *Springer*, 2017.

[55] Tianyu Gu and John M Dolan. On-road motion planning for autonomous vehicles. *International Conference on Intelligent Robotics and Applications*, pages 588–597, 2012.

[56] Tianyu Gu, Jarrod Snider, John M Dolan, and Jin-woo Lee. Focused trajectory planning for autonomous on-road driving. *IEEE Intelligent Vehicles Symposium (IV)*, pages 547–552, 2013.

[57] Lie Guo, Pingshu Ge, Dachuan Sun, and Yanfu Qiao. Adaptive cruise control based on

model predictive control with constraints softening. *Applied Sciences*, 10(5), 2020.

[58] Benjamin Gutjahr. Recheneffiziente Trajektorienoptimierung für automatisierte Fahreingriffe, PhD thesis (in German). *Karlsruhe Institute of Technology*, Munich, 2018.

[59] Benjamin Gutjahr, Lutz Gröll, and Moritz Werling. Lateral vehicle trajectory optimization using constrained linear time-varying mpc. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1586–1595, 2016.

[60] Benjamin Gutjahr and Moritz Werling. Automatic collision avoidance during parking and maneuvering—an optimal control approach. *IEEE Intelligent Vehicles Symposium (IV)*, pages 636–641, 2014.

[61] Eric A Hansen and Rong Zhou. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28:267–297, 2007.

[62] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[64] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progress in road and lane detection: a survey. *Machine vision and applications*, 25(3):727–745, 2014.

[65] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[66] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[67] Amin Hosseini. Conception of advanced driver assistance systems for precise and safe control of teleoperated road vehicles in urban environments, PhD thesis. *Technical University Munich*, Munich, 2018.

[68] Amin Hosseini, Thomas Wiedemann, and Markus Lienkamp. Interactive path planning for teleoperated road vehicles in urban environments. *IEEE Conference on Intelligent Transportation Systems*, pages 400–405, 2014.

[69] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[70] Rolf Isermann. Digital control systems. *Springer Science & Business Media*, 2013.

[71] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. STOMP: Stochastic trajectory optimization for motion planning. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574, 2011.

[72] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104(2), 2010.

[73] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[74] Jennifer Kay. Stripe: Remote driving using limited image data. Technical report, Carnegie Mellon University - Computer Science Department, 1997.

[75] Soohwan Kim and Jonghyuk Kim. Continuous occupancy maps using overlapping local gaussian processes. *IEEE International Conference on Intelligent Robots and Systems*, pages 4709–4714, 2013.

[76] Sven Koenig and Maxim Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005.

[77] Yoshiaki Kuwata, Justin Teo, Gaston Fiore, Sertac Karaman, Emilio Frazzoli, and Jonathan P How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on control systems technology*, 17(5):1105–1118, 2009.

[78] Lubor Ladickỳ, Paul Sturgess, Chris Russell, Sunando Sengupta, Yalin Bastanlar, William Clocksin, and Philip HS Torr. Joint optimization for object class segmentation and dense stereo reconstruction. *International Journal of Computer Vision*, 100(2):122–133, 2012.

[79] Jean-Claude Latombe. Robot motion planning. *Springer Science & Business Media*, 124, 2012.

[80] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, Department of Computer Science, 1998.

[81] Steven M. LaValle and James J. Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[82] Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos. Optimal control. *John Wiley & Sons*, 2012.

[83] Maxim Likhachev and Dave Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009.

[84] Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz, and Sebastian Thrun. Anytime Dynamic A*: An Anytime, Replanning Algorithm. *ICAPS*, 5:262–271, 2005.

[85] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. *Advances in neural information processing systems*, 16:767–774, 2003.

[86] Xiaohai Lin and Daniel Görges. Robust model predictive control of linear systems with predictable disturbance with application to multiobjective adaptive cruise control. *IEEE Transactions on Control Systems Technology*, 2019.

[87] Yiyan Lin and Sandeep S Kulkarni. Automatic repair for multi-threaded programs with deadlock/livelock using maximum satisfiability. *International Symposium on Software Testing and Analysis*, pages 237–247, 2014.

[88] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *arXiv preprint arXiv:1807.03247*, 2018.

[89] David Luenberger and Yinyu Ye. Linear and nonlinear programming. *Springer Science & Business Media*, 2, 1984.

[90] Jan Lunze and Jan Lunze. Regelungstechnik 1 (in german). *Springer*, 10, 1996.

[91] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[92] Hans Moravec and Alberto Elfes. High resolution maps from wide angle sonar. *IEEE International Conference on Robotics and Automation (ICRA)*, 2:116–121, 1985.

[93] Stefan Neumeier, Nicolas Gay, Clemens Dannheim, and Christian Facchi. On the way to autonomous vehicles teleoperated driving. *Automotive meets Electronics; 9th GMM-Symposium*, pages 1–6, 2018.

[94] Jorge Nocedal and Stephen Wright. Numerical optimization. *Springer Science & Business Media*, 2006.

[95] Society of Automotive Engineers (SAE). Standard J3016: Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. 2014.

[96] So-Ryeok Oh and Jing Sun. Path following of underactuated marine surface vessels using line-of-sight based model predictive control. *Ocean Engineering*, 37(2-3):289–295, 2010.

[97] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[98] Markos Papageorgiou, Marion Leibold, and Martin Buss. Optimierung: Statische, dynamische, stochastische Verfahren (in german). *Springer*, 2012.

[99] Scott Drew Pendleton, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Hong Eng, Daniela Rus, and Marcelo H. Ang. Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1):6, 2017.

[100] Anna Petrovskaya, Mathias Perrollaz, Luciano Oliveira, Luciano Spinello, Rudolph Triebel, Alexandros Makris, John-David Yoder, Christian Laugier, Urbano Nunes, and Pierre Bessière. Awareness of road scene participants for autonomous driving. *Handbook of Intelligent Vehicles*, pages 1383–1432, 2012.

[101] Aurelio Piazzi, Corrado Guarino Lo Bianco, and Massimo Romano. Smooth path generation for wheeled mobile robots using η^3-splines. *IEEE Transactions on Robotics*, 23(5):1089–1095, 2007.

[102] Jochen Pohl, Wolfgang Birk, and Lena Westervall. A driver-distraction-based lane-keeping assistance system. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 221(4):541–552, 2007.

[103] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, Carnegie-Melon University Pittsburgh: Artificial Intelligence and Psychology, 1989.

[104] Helena Pongrac. *Gestaltung und Evaluation von virtuellen und Telepräsenzsystemen anhand von Aufgabenleistung und Präsenzempfinden (in German)*. PhD thesis, Universität der Bundeswehr, 2008.

[105] Jakub Porebski, Krzysztof Kogut, Paweł Markiewicz, and Paweł Skruch. Occupancy grid for static environment perception in series automotive applications. *19th IFAC*

World Congress, 52(8):148–153, 2019.

[106] Sean Quinlan. Real-time modification of collision-free paths, PhD thesis. *Stanford University*, 1995.

[107] Rajesh Rajamani. Vehicle dynamics and control. *Springer Science & Business Media*, 2011.

[108] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. *IEEE International Conference on Robotic and Automation*, pages 489–494, 2009.

[109] James B Rawlings. Tutorial overview of model predictive control. *IEEE control systems magazine*, 20(3):38–52, 2000.

[110] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[111] Johannes Reuter. Mobile robots trajectories with continuously differentiable curvature: an optimal control approach. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1:38–43, 1998.

[112] Eder Santana and George Hotz. Learning a driving simulator. *arXiv preprint arXiv:1608.01230*, 2016.

[113] Dmitrij Schitz and Harald Aschemann. Path optimization for autonomous driving using deep learning. *16th International Conference on Motion and Vibration Control (MoViC)*, 2022, accepted.

[114] Dmitrij Schitz, Shuai Bao, Dominik Rieth, and Harald Aschemann. Shared autonomy for teleoperated driving: A real-time interactive path planning approach. *IEEE International Conference on Robotic and Automation (ICRA)*, pages 999–1004, 2021.

[115] Dmitrij Schitz, Gaetano Graf, Dominik Rieth, and Harald Aschemann. Corridor-based shared autonomy for teleoperated driving. *21st IFAC World Congress*, 53(2):15368–15373, 2020.

[116] Dmitrij Schitz, Gaetano Graf, Dominik Rieth, and Harald Aschemann. Interactive corridor-based path planning for teleoperated driving. *IEEE 7th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, pages 174–179, 2021.

[117] Dmitrij Schitz, Gaetano Graf, Dominik Rieth, and Harald Aschemann. Model-predictive cruise control for direct teleoperated driving tasks. *European Control Conference (ECC)*, pages 1808–1813, 2021.

[118] Dmitrij Schitz, Dominik Rieth, and Harald Aschemann. Corridor-based motion planning for teleoperated driving tasks. *24th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 673–680, 2021.

[119] Thomas B Sheridan. Telerobotics. *Automatica*, pages 487–507, 1989.

[120] Thomas B Sheridan. Space teleoperation through time delay: Review and prognosis. *IEEE Transactions on robotics and Automation*, 9(5):592–606, 1993.

[121] Anthony Stentz. Optimal and efficient path planning for partially-known environments. *International Conference on Robotics and Automation (ICRA)*, 4:3310 – 3317, 1994.

[122] Anthony Stentz. The focussed D^* algorithm for real-time replanning. *IJCAI*, 95:1652–

1659, 1995.

[123] Fredi Tröltzsch. Optimal control of partial differential equations: theory, methods, and applications. *American Mathematical Soc.*, 112, 2010.

[124] Costas S Tzafestas. Virtual and mixed reality in telerobotics: A survey. *Industrial Robotics: Programming, Simulation and Applications*, pages 437–470, 2007.

[125] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[126] Jan BF Van Erp and Pieter Padmos. Image parameters for driving with indirect viewing systems. *Ergonomics*, 46(15):1471–1499, 2003.

[127] Paul Venhovens, Karl Naab, and Bartono Adiprasito. Stop and go cruise control. *International journal of automotive technology*, 1(2):61–69, 2000.

[128] Oskar Von Stryk and Roland Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of operations research*, 37(1):357–373, 1992.

[129] Pengwei Wang, Song Gao, Liang Li, Binbin Sun, and Shuo Cheng. Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm. *Energies*, 12(12):2342, 2019.

[130] Andreas Wedel, Hernán Badino, Clemens Rabe, Heidi Loose, Uwe Franke, and Daniel Cremers. B-spline modeling of road surfaces with an application to free-space estimation. *IEEE transactions on Intelligent transportation systems*, 10(4):572–583, 2009.

[131] Moritz Werling. Optimale Fahreingriffe für Sicherheits- und Komfortsysteme (in german). *DeGruyter Oldenbourg*, 2017.

[132] Moritz Werling and Lutz Groll. Low-level controllers realizing high-level decisions in an autonomous vehicle. *IEEE Intelligent Vehicles Symposium (IV)*, pages 1113–1118, 2008.

[133] Moritz Werling, Sören Kammel, Julius Ziegler, and Lutz Gröll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *The International Journal of Robotics Research*, 31(3):346–359, 2012.

[134] Moritz Werling and Darren Liccardo. Automatic collision avoidance using model-predictive online optimization. *IEEE Conference on Decision and Control (CDC)*, pages 6309–6314, 2012.

[135] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. *IEEE International Conference on Robotics and Automation*, pages 987–993, 2010.

[136] Alan Ft Winfield. Future directions in tele-operated robotics. *Telerobotic applications*, pages 147–163, 2000.

[137] Hermann Winner, Bernd Danner, and Joachim Steinle. Adaptive cruise control (in German). In: Handbuch Fahrerassistenzsysteme. *Wiesbaden: Vieweg + Teubner*, pages 478–521, 2009.

[138] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In

Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2129–2137, 2016.

[139] Jianlong Zhou, Amir H Gandomi, Fang Chen, and Andreas Holzinger. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5):593, 2021.

[140] Adam Ziebinski, Rafal Cupek, Hueseyin Erdogan, and Sonja Waechter. A survey of ADAS technologies for the future perspective of sensor fusion. *International Conference on Computational Collective Intelligence*, pages 135–146, 2016.

[141] Julius Ziegler, Philipp Bender, Thao Dang, and Christoph Stiller. Trajectory planning for Bertha — A local, continuous method. *IEEE Intelligent Vehicles Symposium (IV)*, pages 450–457, 2014.

[142] Matt Zucker, Nathan Ratliff, Anca D. Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. CHOMP: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

Curriculum Vitae

Dmitrij Schitz
Date of Birth: July 14, 1994

Education

04/2019 - 12/2021
University of Rostock, Faculty of Mechanical Engineering and Marine Technology
PhD Mechatronics
Thesis: Conception of Control Paradigms for Teleoperated Driving Tasks in Urban Environments

10/2016 - 03/2019
University of Rostock, Faculty of Mechanical Engineering and Marine Technology
M. Sc. Mechanical Engineering
Thesis: MIMO Control of the Cathode Pressure and Air Massflow of a PEM Fuel Cell System

10/2013 - 09/2016
University of Rostock, Faculty of Mechanical Engineering and Marine Technology
B. Sc. Mechanical Engineering
Thesis: Experimental Validation of Feedback Control and Observer Approaches for a Double Pendulum Test Rig

Professional Experience

01/2022 - ongoing
Research engineer at BMW AG, Department for Integrated Brake Control System

04/2019 - 12/2021
Doctoral student at BMW AG, Department for Autonomous Driving
Development of concepts and computationally efficient motion planning methods for teleoperated driving

04/2017 - 01/2018
Research assistant at the Chair of Mechatronics, University of Rostock
Modeling, simulation and control of mechatronic systems