



TECHNISCHE UNIVERSITÄT MÜNCHEN
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Informationstechnische Regelung

Distributed Optimal Control for Multi-Vehicle Coordination

Maximilian Johannes Kneißl

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr.-Ing. Wolfgang Kellerer

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Sandra Hirche
2. Prof. Sébastien Gros, Ph.D.

Die Dissertation wurde am 13.01.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 14.05.2021 angenommen.

To Christina.

Preamble

This thesis has been conducted in the frame of a collaboration between the Chair of Information-oriented Control at the Technical University of Munich and the Corporate R&D Department of DENSO AUTOMOTIVE Deutschland GmbH. During my research in the past years, I received outstanding support. It has been a privilege to work with and learn from many high-skilled and supportive people. I want to take a moment to acknowledge the most important of them.

First and foremost, I would like to thank my doctoral supervisor Professor Sandra Hirche for her support, valuable advice and discussions, as well as her trust and interest in supervising this industrial-driven Ph.D. project.

Furthermore, I am grateful for the invariable support from my industrial supervisor Dr. Hasan Esen. His deep technical background and his excellent personal and professional support ensured a unique research environment within DENSO. I highly appreciate that I was allowed to learn and develop myself under his supervision.

Dr. Adam Molin deserves distinct recognition for his continuous effort during my research journey. His ability to rapidly grasp any challenging problem no matter how confusing it was explained, his willingness to discuss endlessly whenever needed, and his motivating words and positive attitude were very beneficial. I feel deeply indebted to him for his selfless support.

I want to thank my students Eda Cicek, Christian Wachenheim, and Yuto Takeuchi for their contributions and high motivation to solve important problems for this Ph.D. work. Special thanks go to my colleagues at DENSO, particularly to Sebastian vom Dorff and Feras Fattohi for making the Ph.D. student lab an enjoyable and humorous place to work in.

Also, I would like to acknowledge my collaborators in the ENABLE-S3 project. Sebastian for bringing in his excellent technical expertise and skills for the real-vehicle tests, as well as Dr. Son Tong and Dr. Anil Madhusudhanan for their collaboration in the planning and control group.

I am thankful for my family and friends. Especially, I want to thank my parents for always supporting me in achieving my goals. Lastly, I want to sincerely thank my wife, Christina, for her love, all the patience, her understanding, and for the encouraging and motivating words.

Ascholding, December 2020

Acknowledgments

This work has been partially conducted within the ENABLE-S3 project that has received funding from the ECSEL Joint Undertaking under grant agreement No 692455. This joint undertaking received support from the European Union's HORIZON 2020 research and innovation program and from the states of Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway. Additionally, the work has been partially funded by DENSO AUTOMOTIVE Deutschland GmbH.

We are driven by self-interest, it is necessary to survive. But we need wise self-interest that is generous and co-operative, taking others' interests into account.

Dalai Lama

Abstract

Autonomous Driving counts as one of the current largest technological challenges. Its breakthrough reveals a huge potential to increase the safety, efficiency, and comfort of daily transportation. The main source to achieve these properties is the possibility of sharing driving intentions through wireless inter-vehicle communication. Such a setup is a natural way to share the large computational load for making vehicle-network-wide optimized decisions and thus coordinate vehicles in a distributed manner. Furthermore, it paves the way to find more efficient solutions to manage traffic flow, currently tackled using traffic lights, signs, and rules. Computing control decisions for highly dynamic and uncertain environments in a distributed autonomous car setting is a challenging and unsolved problem. It requires a meaningful distribution of a large-scale control problem to fulfill application-specific requirements, such as safety, real-time compatibility, and privacy. The induced unreliability through the wireless inter-vehicle communication poses an additional challenge for the distributed control system, which must be considered during the algorithm development phase.

This thesis addresses the question of distributed control design in a multi-vehicle context using optimization-based control methods. It covers the control design process from modeling multi-vehicle coordination problems through algorithm design to testing methods for multiple autonomous vehicles. Thereby, requirements from real applications are considered that take into account limited information sharing and real-time computation capabilities. Furthermore, a suitable decomposition between on-board vehicle computations supported by central coordination decisions leading to close-to-optimal results is discussed.

The contributions of the thesis are twofold. First, it presents an iterative distribution method of a large constrained coordination problem. The problem is decomposed into local dynamical control optimizations with any-time feasibility guarantee of the iterations and decoupled integer decisions for vehicle ordering. Thereby, the distribution complies with safety conditions, while limiting the amount and type of information exchanged between vehicles plays a crucial role to fulfill applicability requirements. Second, an implementation approach for the above discussed multi-vehicle problems is presented on the use case automated valet parking. The implementation is embedded in an automated test platform to efficiently validate these problems, where relevant scenarios and corner cases can be tested in simulation with increased speed. The pure virtual tests are extended with mixed-reality testing, which contains a real vehicle, and enables additional validation of the proposed algorithms.

Zusammenfassung

Autonomes Fahren gilt als eine der aktuell größten technologischen Herausforderungen. Sein Durchbruch birgt ein hohes Potenzial, um die Sicherheit, die Effizienz und den Komfort im Straßenverkehr zu erhöhen. Eine wichtige Grundlage dafür ist, Fahrabsichten durch drahtlose Kommunikation zwischen den Fahrzeugen zu übermitteln. Dadurch wird es möglich die hohe Rechenlast von netzwerkweit optimierten Entscheidungen zu verteilen und so die Fahrzeuge zu koordinieren. Darüber hinaus ebnet es den Weg für einen effizienteren Verkehrsfluss, was derzeit durch Ampeln, Verkehrszeichen und -regeln beeinflusst wird. Die Berechnung von Regelungsentscheidungen für hochdynamische und unsichere Umgebungen mit autonomen Fahrzeugen ist ein anspruchsvolles und ungelöstes Problem. Es erfordert eine sinnvolle Aufteilung eines großskaligen Regelungsproblems, um anwendungsspezifische Anforderungen wie Sicherheit, Echtzeitkompatibilität und Datenschutz zu erfüllen. Die aus der drahtlosen Kommunikation zwischen den Fahrzeugen resultierende Unzuverlässigkeit, stellt eine zusätzliche Herausforderung für die verteilte Regelung dar, welche bei der Algorithmenentwicklung berücksichtigt werden muss.

Diese Arbeit befasst sich mit der Fragestellung des verteilten Regelungsentwurfs im Multi-Fahrzeug-Kontext unter Verwendung optimierungsbasierter Regelungsmethoden. Sie deckt den Reglerentwurfsprozess von der Modellierung von Multi-Fahrzeug-Koordinationsproblemen über den Algorithmenentwurf bis hin zu Testverfahren für Multi-Fahrzeugszenarien ab. Dabei werden Anforderungen aus realen Anwendungen berücksichtigt, die den begrenzten Informationsaustausch und die Echtzeit-Berechnungsmöglichkeiten berücksichtigen. Darüber hinaus wird eine geeignete Dekomposition der fahrzeuginternen Berechnungen hergeleitet, welche durch zentrale Koordinationsentscheidungen unterstützt wird und dadurch zu nahezu optimalen Ergebnissen führt.

Die Arbeit liefert zwei wissenschaftliche Beiträge. Erstens wird eine iterative Methode zur Verteilung eines umfangreichen Koordinationsproblems behandelt. Dieses Problem wird in lokale Optimierungsprobleme bezüglich der Fahrzeugdynamiken mit Lösungsgarantie für jeden Iterationsschritt und kombinatorischen Entscheidungen für die Fahrzeugreihenfolge aufgeteilt. Dabei müssen Sicherheitsanforderungen erfüllt werden, während zugleich die Limitierung der zwischen den Fahrzeugen ausgetauschten Informationen eine entscheidende Rolle spielt, um die Anforderungen an die Anwendbarkeit und die Privatsphäre zu erfüllen. Zweitens wird ein Implementierungsansatz für die oben diskutierten Multi-Fahrzeug-Probleme am Anwendungsfall von automatisiertem Valet-Parken vorgestellt. Die Implementierung ist in eine automatisierte Testplattform eingebettet, um diese Probleme effizient zu validieren. Dadurch können relevante Szenarien in der Simulation schneller als in der Realität getestet werden können. Darüber hinaus werden die rein virtuellen Tests um Mixed-Reality-Tests erweitert, die ein reales Fahrzeug enthalten und somit eine zusätzliche und realitätsnahe Validierung der diskutierten Algorithmen ermöglichen.

Contents

Notation	xviii
List of Publications	xxv
1 Introduction	1
1.1 Challenges and Requirements in Cooperative Autonomous Driving	3
1.2 Control for Autonomous Vehicle Coordination	4
1.3 Contributions and Outline	10
I Distributed Coordination Methodology	13
2 Control Architecture and Model Setting	15
2.1 Distributed Optimal Control	16
2.2 Longitudinal Vehicle Motion Modeling	20
2.3 Coordination Model	22
2.3.1 Conflict Zones	23
2.3.2 Coordination Graphs	25
2.4 Distributed Coordination Architecture	26
2.5 Summary and Discussion	27
3 Distributed Trajectory Computation	29
3.1 Collision Avoidance Conditions	30
3.2 Control Model Decomposition	31
3.2.1 Centralized MPC	31
3.2.2 Distributed MPC	34
3.3 Iterative Jacobi Negotiation	35
3.3.1 Distributed Over-Relaxation Algorithm	36
3.3.2 Feasible Initial Guess	37
3.3.3 Performance Compensation	38
3.3.4 Algorithmic Properties	41
3.3.5 General Fulfillment of Coordination Conditions	44
3.4 Uncertainty Handling using Exact Penalty Functions	46
3.4.1 Exact Penalty Functions	47
3.4.2 Integration into Jacobi Negotiation	49
3.5 Numerical Illustration	49
3.5.1 DJOR Evaluation	50
3.5.2 Uncertainty Simulation	52

3.6	Summary and Discussion	54
4	Vehicle Sequence Computation	55
4.1	Integer-Continuous Variables Decomposition	56
4.2	Integer Decision using Scheduling	59
4.2.1	Intersection Scenario	59
4.2.2	Scheduling Problem Formulation	59
4.2.3	Solution of the RCPSP	62
4.3	Scheduling-Control Interaction	64
4.3.1	Connection to the Distributed MPC Problem	64
4.3.2	Feasibility of the Integer Decision	66
4.3.3	Event-triggered Re-Computation of the Integer Decision	67
4.4	Numerical Examples	69
4.4.1	Simulation Setup	69
4.4.2	Results	69
4.5	Summary and Discussion	76
II	Evaluation on the Use Case Automated Valet Parking	79
5	Integration Platform	81
5.1	Use Case Automated Valet Parking	82
5.2	Integration in Virtual Test System	83
5.2.1	V&V and Test Automation	84
5.2.2	Virtual Test Platform Architecture and Implementation	84
5.3	Mixed-Reality Testing	87
5.3.1	Real Vehicle Integration	87
5.3.2	Sensing and Localization	90
5.4	Summary and Discussion	90
6	Experimental Assesment	91
6.1	Distributed Planning and Control for AVP	92
6.1.1	Semi-structured Path Planning	92
6.1.2	Infrastructure Control Coordination	94
6.1.3	Longitudinal Motion Model and Control Design	100
6.1.4	Lateral Motion Model and Control Design	102
6.2	Experimental Evaluation	103
6.2.1	Virtual Test Setup	103
6.2.2	Virtual Results	104
6.2.3	Mixed-Reality Test Setup and Scenario	110
6.2.4	Mixed-Reality Results	110
6.3	Summary and Discussion	114
7	Conclusions and Future Work	117
7.1	Summary	117

7.2 Outlook	120
List of Figures	123
List of Tables	125
Bibliography	127

Notation

Acronyms

ACC	adaptive cruise control
AD	autonomous driving
AVP	automated valet parking
CAN	controller area network
CAV	connected and automated vehicles
CoG	center of gravity
DJOR	distributed Jacobi over-relaxation
DLQR	discrete time linear quadratic regulator
ECU	embedded control unit
EKF	extended Kalman filter
EPS	electronic power steering
FMU/FMI	functional mockup unit/interface
FR	FlexRay
HiL	hardware in the loop
ILP	integer linear program
IM	intersection management
IP	integer program
JOR	Jacobi over-relaxation
MABXII	Micro Autobox II
MIQP	mixed integer quadratic program
MPC	model predictive control
ODD	operational design domain
OGM	occupancy grid map
PAM	parking area management
QP	quadratic program
RDB	runtime data bus
ROS	robot operating system
SAE	society of automotive engineers

SQP	sequential quadratic programming
SUT	system under test
TTP	time to park
V2I	vehicle to infrastructure
V2X	vehicle to everything
ViL	vehicle in the loop

Symbols

Scripts and Accents

$(\cdot)_0$	initial value
$(\cdot)_c$	continuous time system
$(\cdot)_{des}$	desired/commanded value
$(\cdot)_{end}$	(estimated/predicted) finish time of action
$(\cdot)_i$	reference to local sub-system (vehicle/conflict zone)
$(\cdot)_{i,v}$	reference to velocity state of local system
$(\cdot)_{max}$	upper bound value
$(\cdot)_{min}$	lower bound value
$(\cdot)_{ref}, (\cdot)^{ref}$	reference value
$(\cdot)_{start}$	(estimated/predicted) start time of action
$(\cdot)^a, (\cdot)^b, (\cdot)^s, (\cdot)^e$	reference to conflict zone; a: beginning of a zone, b: end of a zone, s: beginning of a series of zones, e: end of a series of zones
$(\cdot)^{des}$	desired value
$(\cdot)^{(l)}$	inter-sampling iteration count
$(\cdot)^{lat}$	variable refers to lateral motion
$(\cdot)^{lg}$	variable refers to longitudinal motion
$(\cdot)^M$	terminal (end of horizon) value
$(\cdot)^{no}$	nominal value
$(\cdot)^+$	variable at next discrete time step $t + 1$
$(\cdot)^r$	value refers to reachable set
$(\cdot)^*$	optimal/optimized value
$\hat{(\cdot)}$	feasible candidate solution
$\overline{(\cdot)}$	prediction vector
$\tilde{(\cdot)}$	approximated value

Variables

$a(t), a^{long}$	longitudinal vehicle acceleration
A	system matrix
A_{adj}	adjacency matrix
\mathcal{A}	matrix with system constraints
\mathcal{A}_{ij}^d	matrix with inter-vehicle constraints w.r.t. ego vehicle i
a^{lat}	lateral vehicle acceleration
α_i	scheduling activity, Chapter 4
α_{ij}	element of \mathcal{A}_m^d , Chapter 3
\mathcal{A}_m^d	matrix with inter-vehicle constraints dependent on integer decision m
B	input matrix
b	vector with system constraints constants
β	side slip angle
b_{ij}^d	vector with inter-vehicle constraints constants
$b_{i,\tau}$	binary decision variable
b_m^d	vector of inter-vehicle constraints dependent on integer decision m
C_f, C_r	cornering stiffness (front, rear axle)
c_i	coupling constraint
\mathcal{C}_{ij}^d	matrix with inter-vehicle constraints w.r.t. neighbor vehicle j
$d(t)$	distance state
D	deadlock configuration
$\Delta_{\mathbb{T}}$	trajectory offset for \mathbb{T}_{ij}
δ	steering angle, Chapter 6
$\hat{\delta}$	DLQR control input (steering angle), Chapter 6
δ_i	penalty weight, Chapter 3
$\Delta_{r,x}$	state error w.r.t. reference vector
d_{inter}	inter-vehicle distance
$d_{i,s}$	safety distance of vehicle i
d_{le}	lateral distance error
d_s	safety distance
d_{slack}	inter-vehicle slack distance
E	matrix for definition of polyhedral constraints of \mathbb{X}_i^A
e_{jk}	edge connecting \mathcal{CZ}_j and \mathcal{CZ}_k
ϵ_i	slack variable vector
η	vector of scheduling activity durations
F_{aero}	aerodynamic drag force
F_{grav}	gravitational forces
F_{roll}	rolling resistance
F_{tire}	longitudinal tire force
G	goal configuration
γ	terminal condition
Γ	matrix with precedence relations
\mathcal{G}_{com}	graph describing the structure of information exchange between vehicles
G_{dlqr}	DLQR gain matrix

\mathcal{G}_{route}	directed multi-graph describing vehicle routes in the coordination scene
J_z	moment of inertia
k	prediction time step
κ	path curvature
k_{brake}	latest possible start of braking within prediction horizon
$L_{i,x}, L_{i,y}$	length of vehicle i in x/y direction
λ_i^*	Lagragian vector of z_i^*
l_f, l_r	distance between CoG and front/rear axle
m_{veh}	vehicle mass
M	length of prediction horizon
m, m_i	integer optimization variable (global, of vehicle i)
M_{sched}	length of scheduling horizon
N	number of agents
n_c	dimension of coupling constraint
N_{CZ}	number of conflict zones in coordination scenario
n_i	dimension of local (vehicle i) optimization vector
N_I	number of intersection in coordination scene
n_{iq}, n'_{iq}	number of inequality constraints
n_{sched}	sampling time relation between scheduling and control systems
N_v	number of vehicles in coordination scenario
N_{W_i}	number of waypoints in set W_i
n_z	dimension of optimization vector z
o_i	tuple representing the crossing order for conflict zone \mathcal{CZ}_i
Ω	matrix of scheduling demands
ω_i	degree of over-relaxation
ω_{ir}	amount of consumed scheduling resources (element of Ω)
$p(t)$	position state
P_i	terminal state weight
$\dot{\Psi}$	yaw rate
p_x^i, p_y^i	x/y coordinate of waypoint i
Q_i	state weight
Q_i	longitudinal state weighting matrix
ρ_i	scheduling resource
R_i	input weight
T	time constant
t	discrete time
\mathcal{T}	tree of crossing order decisions
τ	continuous time
$ \mathcal{T} $	number of branches in \mathcal{T}
Θ_{he}	heading error
T_i^{act}	actuation time constant
t_i^r	discrete time instant w.r.t. backward-reachset
\mathcal{T}_m	branch of decision tree \mathcal{T}
t'	discrete scheduling time

T_s	sampling time
T_{sched}	sampling time of scheduling problem
t_{stop}	discrete time step at which vehicle stops
$u(t)$	input vector
$v(t)$	velocity state
v_i	label of vehicle i
v_x	longitudinal velocity
$x(t)$	state vector
ξ	vector for definition of polyhedral constraints of \mathbb{X}_i^A
\hat{Z}_i	nominal trajectory without inter-vehicle interaction
z, z_i	continuous optimization variable (global, local vehicle i)

Sets

\mathcal{B}	vehicle bounding box
\mathcal{CZ}	conflict zone
\mathcal{E}_{com}	edge set of \mathcal{G}_{com}
\mathcal{E}_{route}	edge set of \mathcal{G}_{route}
$I = [1, N]$	continuous interval between 1 and N
\mathcal{I}	set of all conflict zones in coordination space
\mathcal{I}_{end}	set of conflict zones located at end of vehicle paths
\mathcal{I}_{inter}	set of conflict zones located in an intersection area
K	set of feasible prediction time steps for which a trajectory to full-stop exists
Λ	set of activities
Λ'	set of non-dummy activities
\mathcal{N}	set of neighboring vehicles w.r.t. \mathcal{G}_{com}
\mathcal{O}	set of crossing orders in coordination scenario
$OpenSet$	set of candidate nodes in OGM
\mathcal{P}	set of predecessor vehicles w.r.t. \mathcal{G}_{com}
\mathcal{S}	set of successor vehicles w.r.t. \mathcal{G}_{com}
\mathbb{T}_{ij}	trigger set w.r.t. vehicle i and j
Υ	set of renewable resources
$\mathcal{V} = \{v_1, v_2, \dots\}$	set of vehicle IDs in coordination scenario used notations referring to set elements: $v_i \in \mathcal{V}$ or $i \in \mathcal{V}$
\mathcal{V}_{route}	vertex set of \mathcal{G}_{route}
\mathcal{V}_{route}^i	vertex set w.r.t. vehicle i
W_i	set of waypoints
\mathcal{Z}	set of conflict zones that two vehicles pass commonly

Numerical Spaces

\mathbb{C}	constraint set
\mathbb{N}	natural numbers
$\mathbb{I}_{a:b}$	set of integers $\{a, a + 1, \dots, b\}$, with $a, b \in \mathbb{N}$
\mathbb{R}	real numbers
\mathbb{R}^n	n -dimensional Euclidean space
$\mathbb{R}^{n \times m}$	space of $n \times m$ -dimensional matrices
\mathbb{R}^+	set of non-negative real numbers
$\mathbb{S}^{n \times n}$	class of $n \times n$ -dimensional positive-definite weighting matrices
$\mathbb{S}_0^{n \times n}$	class of $n \times n$ -dimensional positive-semi-definite weighting matrices
\mathbb{U}	polyhedral input constraints set
\mathbb{X}	polyhedral state constraints set
\mathbb{X}_i^A	admissible state set/backward-reachset
\mathbb{Z}	set of integers
\mathbb{Z}^+	set of non-negative integers
$[a..b]$	discrete interval $[a, a + 1, \dots, b]$, with $a, b \in \mathbb{N}$
$[a, b]$	continuous interval between a and b , with $a, b \in \mathbb{R}$

Functions

$f(\cdot)$	dynamics function
$g_i(\cdot)$	inter-vehicle coupling constraints
$h(\cdot)$	heuristic for pathplanner
$\pi_i(\cdot)$	C^1 curve connecting waypoints
$V(\cdot)$	global objective function of an optimization problem
$V_i(\cdot)$	local objective function of an optimization problem

Operators

A^T	transpose of matrix A
$\dot{x} = \frac{dx}{d\tau}$	time derivative
$\{x_i\}_{i \in \mathcal{S}}$	collection of trajectories x_i with indices i contained in set \mathcal{S}
$v(\cdot)$	all elements of vector v
$[v]_n$	n -th element of vector v
$diag(v_1, v_2) = \begin{pmatrix} v_1 & & \\ & v_2 & \\ & & \end{pmatrix}$	diagonal operator with $v_1, v_2 \in \mathbb{R}$
$v_i \overset{o_k}{>} v_j$	element v_i appears before element v_j in an order tuple o_k
$i \xrightarrow{\pi_k} j$	point i is located before point j following curve π_k in a specified direction

Norms

$\ \cdot\ _1$	1-norm
$\ \cdot\ _2$	2-norm
$\ \cdot\ _\infty$	infinity-norm
$\ x - \hat{x}\ _Q^2 = (x - \hat{x})^T Q (x - \hat{x})$	weighted 2-norm with $x, \hat{x} \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n \times n}$

List of Publications

This thesis is based on results presented in the following publications:

Maximilian Kneissl, Adam Molin, Hasan Esen, and Sandra Hirche, “A Feasible MPC-Based Negotiation Algorithm for Automated Intersection Crossing”, in *Proceedings of the 2018 European Control Conference (ECC)*, 2018.

Maximilian Kneissl, Adam Molin, Hasan Esen, and Sandra Hirche, “A One-Step Feasible Negotiation Algorithm for Distributed Trajectory Generation of Autonomous Vehicles”, in *Proceedings of the 2019 58th IEEE Conference on Decision and Control (CDC)*, 2019.

Maximilian Kneissl, Anil K. Madhusudhanan, Adam Molin, Hasan Esen, and Sandra Hirche, “A Multi-Vehicle Control Framework with Application to Automated Valet Parking”, *IEEE Transactions on Intelligent Transportation Systems*, 2020.

Maximilian Kneissl, Sebastian Kehr, Adam Molin, Hasan Esen, and Sandra Hirche, “Combined Scheduling and Control Design for Coordination of Automated Vehicles at Intersections”, in *Proceedings of the 21st IFAC World Congress*, 2020.

Maximilian Kneissl, Sebastian vom Dorff, Maxime Denniel, Tong Duy Son, Nicolas Ochoa Lleras, Adam Molin, Hasan Esen, and Sandra Hirche, “Mixed-reality Testing of Multi-vehicle Coordination in an Automated Valet Parking Environment”, in *Proceedings of the 21st IFAC World Congress*, 2020.

Maximilian Kneissl, Adam Molin, Hasan Esen, and Sandra Hirche, “Any-time Feasible Trajectory Computation for Distributed Multi-vehicle Coordination”, submitted to *IEEE Transactions on Control Systems Technology*, 2020.

Other Publications

Additional publications with the author's contribution in the context of this thesis:

Hasan Esen, **Maximilian Kneissl**, Adam Molin, Sebastian vom Dorff, Bert Böddeker, Eike Möhlmann, Udo Brockmeyer, Tino Teige, Gustavo Garcia Padilla, and Sytze Kalisvaart. “Validation of Automated Valet Parking”, in *Validation and Verification of Automated Systems*, pp. 207-220, Springer, Cham, 2020.

Sebastian vom Dorff, Bert Böddeker, **Maximilian Kneissl**, Martin Fränzle, “A Fail-safe Architecture for Automated Driving”, in *IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE'20)*, 2020.

Introduction

Autonomous driving is considered to be among the most challenging technical developments after the first crewed moon landing. Its promising advantages prompt all major automakers, suppliers, and tech-giants to invest in this field. Autonomous vehicles are expected to significantly reduce the amount and severity of accidents and save many lives from around 3,000 road fatalities in Germany, 5,000 in Japan, and almost 40,000 in the US in 2016 [Wor18]. Besides the safety argument, the increased efficiency of automated vehicles will contribute to smoother traffic flows, reducing congestion in dense metropolitan areas and leading to reduced energy consumption. Finally, increased passenger comfort and stress relief are important boosters for advancements in the automated vehicle segment [SKBB+18].

Currently available advanced driver assistant systems (ADAS) in serial production vehicles are the first step toward fully autonomous driving. However, the human driver is still fully accountable and has to take over the system as a fall-back option at any time. Known examples are adaptive cruise control, emergency braking, lane-keeping or change assistance, automatic parking, and pedestrian protection systems, to name a few from a long list. These systems are classified as Level 1 and Level 2¹ systems, according to the SAE (society of automotive engineers) classification of autonomous driving levels [Soc16]. First systems with conditional and high automation (Level 3 and Level 4), such as traffic jam assist, highway assist, and automated valet parking, are expected to enter the market shortly [ERT19].

Automated vehicles' connectivity is an essential feature to contribute to the safety, efficiency, and comfort. Data exchange through vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communication supports vehicles in their decision-making such that

¹The SAE levels indicate the degree of automation ranging from Level 0 'only passive driver support' to Level 5 'fully autonomous driving without a required driver interaction in all possible scenarios'.

smarter actions can be performed compared to isolated vehicles. Potential content of shared data ranges from static information such as semantic maps, dynamically updated maps, through cooperative perception systems, to cooperative driving maneuvers. The latter describes the case where vehicles share real-time trajectory information to enable efficient coordination of a vehicle cluster. The infrastructure support levels for automated driving (ISAD) suggest a possible classification of shared information content [ERT19].

While current serial production vehicles' connectivity mainly enhances comfort features such as dynamic congestion avoidance or free parking spot information, recently also safety systems start using out-of-vehicle information. These will warn the driver in unexpected safety-critical situations such as accidents on a highway or behind a curve. The used communication technology builds on either the dedicated short-range communication (DSRC) based on the IEEE 802.11p protocol [JD08] or the 3GPP LTE standard [MSMCP+09]. A major drawback of these communication technologies is the limited bandwidth, which becomes a bottleneck if many vehicles interact. The mentioned protocols are thus mainly suitable for particular and seldom safety warning broadcasts. However, the recently established 5G communication technology is a promising candidate for increased vehicle interaction [IEE17]. Its high throughput, long-range, high reliability, and low latency might pave the way to exchange real-time trajectory information in V2V or V2I scenarios.

Given this background, there is a need for new network control methods that counteract a significant decrease in its effectiveness and, most of all, avoid safety incidents if communication dropouts occur in cooperative vehicle maneuvers [EKM19]. All in all, the development of such control systems will play a key role in future cooperative autonomous vehicle scenarios.

This thesis aims at deriving control strategies suitable for distributed cooperative driving scenarios. Figure 1.1 illustrates an exemplary cooperative scenario with V2V and V2I communication.

Methods and developments in this thesis are presented in the light of autonomous cars. Thus, the term multi-vehicle coordination refers to connected and automated automobiles and their coordination on roads, i.e., intersections, parking areas, lane merging situations, etc. While the presented strategies are applicable to any transportation or logistic application, the functional requirements and assumptions throughout the thesis are tailored to the field of connected and autonomous cars.

Similar problem settings, such as in cooperative autonomous driving, can be applied to logistic warehouse robots, often referred to as automated guided vehicles (AGVs) [LADK06]. Here, multiple robot platforms solve designated tasks, such as transporting goods, while relying on each other's cooperative decisions to avoid collisions and allocate shared resources. An alternative application could be coordinated aircraft surface taxiing, which describes the airplanes' process of moving between their park position and the runway [BJ07; JHM+10]. This process requires safe and efficient coordination in a multi-vehicle setup. Each stop-and-go action of an aircraft poses a significant energy consumption. Certainly, safety violations can lead to dangerous and costly damages. Also, the coordination of vessels suggests a similar problem setup where collision free distributed trajectory computation methods are investigated [Zhe16; FNKAM18].

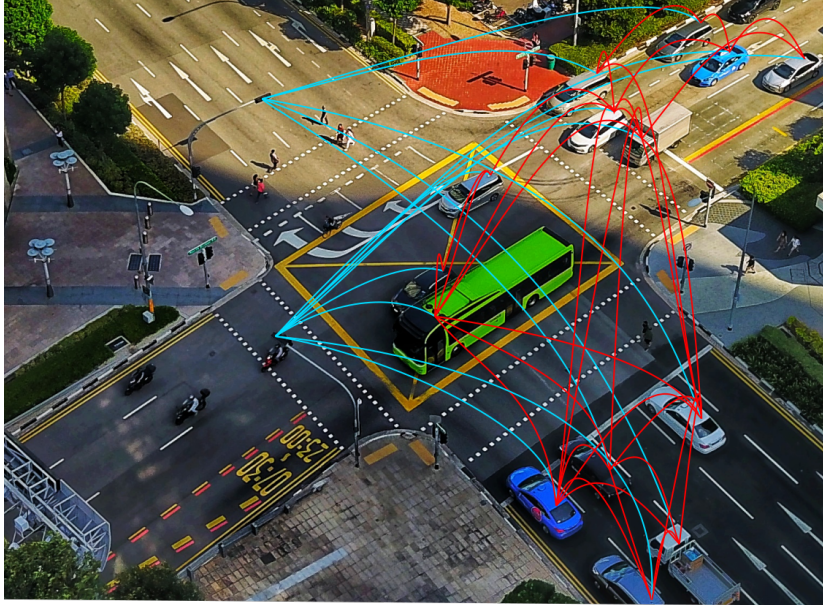


Figure 1.1: Exemplary autonomous and cooperative driving scenario².

1.1 Challenges and Requirements in Cooperative Autonomous Driving

This section discusses important challenges in the software development of autonomous driving functions and resulting requirements for distributed control design. The introduced categories will be used throughout the thesis to classify and discuss related work and presented methodologies.

Safety The foremost goal - and maybe the biggest challenge - in developing autonomous vehicles is to guarantee their safe functioning with sufficient confidence. Safety must be reflected in all levels of the autonomous driving algorithms. In this thesis, we focus on the challenge of finding *feasible* solutions in the space of involved decision-makers for distributed multi-vehicle scenarios. In this context, feasibility refers to safe solutions such that inter-vehicle collisions are avoided while traffic flow is ensured.

Scalability A distributed algorithmic implementation, with safety and privacy-related decisions computed in each vehicle locally, is preferred over centralized architectures. This requires control laws, which are computable on automotive hardware, and scalable such that a large number of vehicles can be considered in coordination scenarios.

Validation and Implementation The challenge of validating a safe and reliable functionality of conventional automobiles is mainly ensured through extensive software-in-

²Figure adapted from https://upload.wikimedia.org/wikipedia/commons/a/a5/The_intersection_of_North_Bridge_Road_and_Rochor_Road._February_2019.jpg, accessed Dec, 19 2020.

the-loop (SiL), hardware-in-the-loop (HiL), and real-world vehicle testing. However, the amount of required test kilometers to guarantee autonomous vehicles' safety is far beyond the practicability with state of the art validation methods. Therefore, the development of suitable virtual test systems and test scenarios is under investigation to accomplish the safety argumentation within a feasible time horizon. Thus, proposing suitable implementation strategies becomes relevant, particularly if a fleet of connected and automated vehicles running a distributed coordination algorithm is considered. For an efficient validation process, it is required to ensure tested scenarios' reproducibility and a seamless adjustment of implemented algorithms and scenarios.

Efficiency Autonomous vehicles are expected to increase transportation systems' efficiency through smoother driving maneuvers with better predictions than conventional human-driven cars. This counts in particular for connected automated vehicles as they can share intentions and react with efficient plans to other vehicles' decisions. Different vehicles will have different interests, and local vehicle interests might differ from global traffic coordination objectives. The challenge is to find solutions, which result in a meaningful trade-off between actors in a coordination scenario. Efficiency has to be considered for several coordination layers. Particularly, a reduced energy consumption for individual vehicles, and a high vehicle throughput from a traffic network perspective.

Cooperation Autonomous vehicles have to be able to resolve conflicts such that deadlocks can be avoided. In traffic situations, a deadlock means that two or more vehicles cannot move any further without exchanging information. The challenge is to define inter-vehicle communication such that deadlocks can be avoided in a cooperative way. In distributed control, the definition of cooperative systems commonly refers to systems with coupled objective functions (cf. Section 2.1). For cooperative autonomous vehicles, this should be understood in a wider context, meaning that vehicles are cooperative if they i) follow commanded instructions from upper hierarchies and ii) share information with other vehicles while making efforts to adjust to their decisions.

Privacy Privacy can be defined as: "Assurance that the confidentiality of, and access to, certain information about an entity is protected" [BSBC13, p. 108]. While an entity often refers to persons, the definition also covers vehicle-related data confidentiality, such as algorithmic or model information. There is a confined acceptance to disclose certain, e.g., technologically relevant, data. Considering that, in general, cooperative autonomous driving contains vehicles from different manufacturers interacting with each other. To achieve a cooperative behavior, it will be necessary to share information between different vehicles. The challenge is to design coordination algorithms, which keep the amount and type of shared information limited and appropriate.

1.2 Control for Autonomous Vehicle Coordination

This thesis deals with control strategies based on numerical optimization. The optimization problems are characterized by a cost function that describes the systems' interests. Ad-

ditionally, these problems consider constraints representing system dynamics and environmental limitations. Note that the formulation of constraints makes it generally impossible to solve these problems analytically, and thus, numerical optimization is applied.

A widely used optimal control method is model predictive control (MPC). MPC predicts the future system behavior for a defined prediction horizon and is formulated as a finite horizon optimization problem. It is repeatedly computed in each time step with updated initial conditions to account for changing environmental conditions. The success of MPC comes from its ability to abstract the optimization problem sufficiently to be real-time applicable while leading to (close-to-)optimal and robust solutions with constraint satisfaction.

In the following, we review literature dealing with control for multi-vehicle coordination – in particular in the field of optimization-based control:

1. Related publications for the major autonomous driving use cases, i.e., vehicle platooning, merging, and intersection crossing are discussed, and their challenges are highlighted.
2. Literature in the field of distributed MPC, which can solve multi-vehicle coordination problems, is reviewed.
3. The state of the art of automated valet parking (AVP) systems, which is the application use case of this thesis, is introduced, and validation strategies using virtual testing for autonomous driving are discussed.

The related literature is assessed according to challenges and requirements formulated in Section 1.1.

Vehicle Coordination Scenarios and Approaches

First attempts at controlling multiple autonomous vehicles were proposed in the frame of automated highway systems [Fen70]. Thereby, forming platoons is a widely studied automation strategy, which requires longitudinal control of involved vehicles [KC11]. This can be achieved without vehicle interaction, i.e., through decentralized control architectures, using adaptive cruise control (ACC) systems [VE03], or with vehicle-to-vehicle communication, in a distributed fashion, called cooperative adaptive cruise control (CACC). Both methods reveal a large potential for increasing traffic efficiency through higher vehicle throughput and lower fuel consumption than standard highway traffic. This has been investigated for heavy-duty vehicles using ACC [AAGJ], and highway experiments with CACC showed the potential of doubling the vehicle throughput [RS01]. However, authors of [VAVDV06] argue that around 60% of vehicles have to be equipped with CACC systems to achieve traffic flow benefits and improved efficiency. The main control goal of platooning is to maintain a constant inter-vehicle distance with certain robustness [XG10] and a stable platoon behavior, referred to as string stability [DC12]. To maximize the efficiency of platooning while ensuring safety, the minimum safe inter-vehicle distance has to be computed. Reachable set theory is a candidate to compute the latter, which has been proposed in [GAAJT11] through the formulation of a pursuer-evader game. To achieve string

stability, a PD control method is proposed for controlling a constant time gap [MSS+14]. Alternatively, distributed MPC can be utilized for an intuitive constraint formulation that guarantees string stability [KAE+12], while [ZLL+16] derives a terminal constraint set, ensuring stability for heterogeneous vehicle models and an unknown setpoint. Research on platooning reveals important aspects of control architectures and concepts for vehicle interaction of multi-vehicle coordination. However, in platooning problems, inter-vehicle relations are clearly defined, which is not the case for general coordination scenarios.

Therefore, an extension is a merging scenario, where a moving platoon opens a gap in which one or several vehicles from a neighboring lane can merge into. Commonly, inter-vehicle communication between an existing platoon and the merging vehicle(s) is utilized [UST99; LH03; LTSH04]. Besides designing a safe and stabilizing control law, the merging maneuvers require a decision about the position of the gap to be opened. To make this decision in an effective and optimized way, authors of [Ath68] formulate a cost function and design a linear quadratic regulator (LQR). Alternatively, a cooperative MPC formulation has been proposed by [CMK+15]. These approaches suggest solutions to determine required sequence decision between vehicles while taking control objectives into account for efficient solutions. However, these solutions cannot ensure applicability and scalability for large multi-vehicle coordination problems.

The coordination of vehicles through intersections is considered one of the most challenging tasks of autonomous vehicle coordination from a control perspective. In addition to ensuring safe inter-vehicle coordination considering vehicle dynamics, a crossing sequence decision must be determined. Part of the intersection problem can be treated as an extension to the merging scenario. Instead of considering one vehicle merging to one lane, multiple merging lanes assigned to multiple vehicles need to be considered for the intersection case together with additional constraints for collision avoidance at intersecting lanes. Comprehensive summaries on intersection coordination with different control architectures and strategies are given in [RTM16; CE15]. Dresner and Stone [DS08] presented an early proposal to solve the intersection crossing problem. They introduce a protocol where approaching vehicles send a reservation request to an intersection manager. If the access is granted, they cross according to the plan while resubmitting a request else. This protocol ensures deadlock-free crossing, but vehicle dynamics are not considered directly in the reservation scheme. However, the latter is important to ensure the Safety requirement.

Optimal control is a frequently applied control strategy for intersection problems. It can consider dynamic feasibility and safety guarantees through constraint formulations and efficiency objectives through cost functions. An LQR-related control law in [MG12] ensures a deadlock-free crossing order. The work is extended in [MG13] with collision avoidance constraints formulated within an MPC law with softened constraints. Solving the MPC problem with soft constraints ensures an efficient computation, which is also achieved through analytic solutions of Hamiltonian functions in [ZMC16]. Nevertheless, the Safety requirement cannot be guaranteed by these approaches. Hard constraints in optimal control problems, to formulate, for example, collision avoidance, require numerical optimization strategies to solve the problem. However, hard constraints can cause feasibility issues in a distributed setup. Campos et al. [CFW+14] derive a distributed MPC (DMPC) setup for conflict resolution at intersections, which they solve sequentially between vehicles

to achieve network-wide feasibility. Using primal decomposition, Hult et al. [HZGF16] compute optimal intersection occupancy time-slots with a similar DMPC formulation. As an extension, a feasibility guarantee for asynchronous information exchange is derived with a distributed sequential quadratic programming (SQP) formulation in [ZGWF17]. These approaches rely on iterative information exchange between vehicles, limiting the Safety guarantees if convergence cannot be achieved timely.

To increase the throughput of an intersection area and reduce the conservatism, it has been proposed to distinguish between several geometrical regions when designing the coordination law [NRTT97; NRT98]. This is also taken into account in [KKJ17], where the points of crossing vehicle paths serve as coordination reference. Resulting non-convex distance safety constraints are solved through semi-definite relaxations (SDR), which enable an iteration-free and parallel computation of DMPC decisions. An introduced non-zero velocity constraint, however, prevents the fulfillment of the Safety requirement.

The vehicle crossing sequence through the intersection area is often assumed to be predefined, or heuristic solutions are used to keep optimal control problems computationally feasible. Conversely, scheduling algorithms have been investigated to compute a sequence decision in an upper intersection-wide management layer. For intersection scenarios, it has been proposed to minimize the problem's makespan, i.e., the time it takes for coordinating all considered vehicles through the intersection area [WATEM12; BVDEG16]. In order to maintain the computational feasibility of such central formulations, only high-level traffic dynamics or simplified vehicle dynamics can be used for the problem formulation [LZ17; ADV16].

For general coordination problems, extending a single intersection coordination problem to several connected intersections is required. Some approaches can be directly extended, such as the reservation protocol in [DS08], which is applied to a multi-intersection scenario in [HAS11]. Alternatively, in [TBS14], a two-level approach is suggested for an intersection network. The first level controls a local single intersection, and in the second level, the flow between intersections in the network is optimized using information from the first level. Furthermore, scheduling approaches can be applied to coordinate the vehicle flow in traffic networks. Authors of [YDM11] suggest a machine scheduling formulation with high-level traffic dynamics models. Sadraddini and Belta [SB16] also use high-level traffic link models combined with a finite horizon MPC law to guarantee a safe coordination behavior through the design of appropriate control invariant sets.

To conclude, it remains an open problem to propose control strategies for multi-vehicle coordination problems that can: i) capture sufficiently detailed vehicle dynamics to ensure Safety, ii) take the combinatoric nature of coordination problems into account to ensure Efficiency, and iii) ensure Scalability considering large-scale coordination spaces.

Distributed MPC

Besides the above-discussed literature for coordination scenarios, we review distributed control literature that supports the fulfillment of requirements from Section 1.1.

MPC and in particular DMPC is a widely used control method for multi-vehicle coordination [ZLL+16; KK14; DM06; KSSP19; ZMC16; HZG+20; QGDLM15; KAE+12; WLZL15; MG13]. The success of MPC is related to its trade-off between computational

effort and (close-to-)optimal solution and the ability to consider constraints in the optimization formulation, which is particularly important for ensuring the safety of vehicle coordination problems. The predictive nature of MPC plays an important role in multi-vehicle coordination.

A possible method of using MPC for multi-vehicle coordination is to couple cost functions between agents to achieve a desired formation [DM06]. Pure coupling in the cost functions, however, cannot guarantee inter-vehicle collision avoidance. Therefore, a brake-safe distance can be formulated as constraints in the MPC formulation with decoupled local control problems [QGDLFM15]. While such decentralized setups ensure Safety, they do not fulfill the Cooperation requirement.

Therefore, it is beneficial to introduce inter-vehicle communication, i.e., a distributed control setup. Still, it is challenging for constraint coupled distributed systems to ensure network-wide feasibility of optimization problems [KK14]. A commonly used method to solve this issue is robust MPC (RMPC), which takes the worst-case actions of neighboring agents into account when computing a local control solution. This enables the use of RMPC both in decentralized (without inter-vehicle communication) and in distributed (with inter-vehicle communication) setups. Richards and How [RH04] ensure vehicle collision avoidance with feasible optimization problems through sequential computations. Their decentralized RMPC problem models decoupled vehicle dynamics with coupling constraints. Coupled dynamics and coupling constraints are assumed in [FS12], where an RMPC law ensures feasibility with neighbor-to-neighbor communication, i.e., distributed RMPC. Schildbach et al. [SSB16] suggest an RMPC law for collision avoidance of an autonomous vehicle, which accounts for possible maneuvers of manually driven vehicles in an intersection scenario. The main drawback of RMPC solutions is that they tend to deliver conservative solutions as they assume worst-case maneuvers of other participants, resulting in a significant loss of Efficiency.

Different from RMPC, many distributed algorithms rely on an iterative process to determine a solution. This is commonly implemented through inter-sampling iterations or, in other words, nested iterations. It means that information is exchanged several times between two consecutive sampling time steps. Iterative methods have the ability to find closer-to-optimal, i.e., less conservative solutions compared to non-iterative approaches. For multi-vehicle scenarios, where inter-sampling iterations mean an information exchange via a wireless communication channel, it is essential to limit the number of iterations to ensure real-time applicability. Reasons are that the nature of wireless communication induces transmission delays and package drop-outs to the networked control system. Doan [DKDS11] proposes an iterative constraint tightening method to ensure feasibility for constraint-coupled DMPC within a finite number of iteration. The method is applicable in hierarchical control architectures. For a similar architecture, Steward et al. [SVR+10] discuss a Jacobi overrelaxation (JOR) algorithm for input constraint DMPC problems, which is feasible after each iteration and converges to an optimal solution. This approach has been extended in [DDKDS17] to general constraint optimization problems with optimality guarantee if constraints remain inactive. While these are promising control candidates, they are not directly applicable to trajectory computation of multi-vehicle problems. Furthermore, they require a centralized update step, which limits their Scalability.

In summary, DMPC solutions have a large potential for multi-vehicle coordination problems, particularly in ensuring the Safety and Cooperation requirements. However, the discussed approaches make either too conservative decisions to ensure Safety or too few restrictions on the vehicle interaction, which can pose a Safety risk, lead to a loss of Privacy or Scalability.

Automated Valet Parking and Validation Methodologies

Automated valet parking (AVP) is expected to be among the first commercially available Level 4 autonomous driving application. Reasons are that i) vehicles move at low speed, which helps to provide safety guarantees, ii) vehicles operate in a limited area (parking lots), which can be well observed by infrastructure sensors, and iii) the uncertainty of the environment can be mitigated, e.g., by blocking out pedestrians.

A survey of AVP is provided in [BNKZ17]. Within the project V-Charge, AVP strategies with close-to-market sensors (camera and ultrasonic) were developed [Sch+16], and [MC13] proposes a software and control architecture. In both studies, the single-vehicle case is considered. [Kle+16] presents a fully integrated AVP demonstrator with an electric vehicle navigating in a multi-story parking garage and automatically docking to a charging station. A major benefit of AVP is the reduction of consumed parking space, which has been investigated in [TFBW15] using k-stack models for space optimization, and k-deques with an upper-bounded number of shunting operations in [BQN+17]. The main focus of AVP literature is the implementation of a single vehicle or the parking spot organization. The consideration of multi-vehicle coordination in parking environments is still a rarely studied topic. Exceptions are discussed in the following.

Strategies for the coordination of multiple vehicles have been suggested in [KK17]. The authors propose a trajectory planning algorithm based on mixed-integer-linear-programming (MILP), where the optimization problem chooses between a set of trajectories taking other vehicles' decisions into account. Real-time computability and, therefore, the requirement of Scalability cannot be guaranteed. Differently, in [LJM19], a Jacobi-based DMPC with cost coupling considering lateral motions between neighboring vehicles is discussed. The cost coupling enables an efficient computation, and mixed-traffic can be considered through an observer unit in an upper hierarchical layer. Pure cost coupling, however, cannot ensure safe vehicle coordination. Moreover, a hierarchical coordination method is discussed in [SZB20]. Collisions and deadlocks are avoided through a parallel and decentralized control law, in which each vehicle reserves its reachable area in a shared occupancy grid map. While the control laws can be computed in parallel, the decentralized implementation lacks the guarantee of Efficiency, and the update of a shared map limits the Scalability of the coordination procedure.

To pave the way for autonomous driving toward application, providing appropriate virtual testing strategies is essential due to the unfeasible amount of required test kilometers [KP16; Wac17; SSSS17]. Virtual testing outperforms field testing for early development phases as it has the potential to deliver scalable, safe, low-cost, and reproducible test results. The survey on virtual testing in [SZS+15] highlights the difference between testing for advanced driver assistance systems (ADAS) and autonomous driving (AD) and the taxonomy of virtual testing. Additionally, [HWLZ16] surveys different simulators,

testing methods, and test system architectures. An essential part of test architectures is high fidelity simulation environments, which can be implemented as co-simulation platforms connecting an environment simulation with vehicle dynamics simulation modules [SHA+18; SBA19]. Most efforts for AD simulations consider the case with a single vehicle in the loop (ViL). An open challenge is to enable a seamless Implementation and Validation by such simulation platforms once they are extended toward multi-ViL. This argumentation excludes large-scale traffic simulators, which apply too coarse vehicle models for AVP scenarios.

To bridge the gap from purely virtual testing toward real-world testing, mixed testing strategies have been investigated. Some parts are tested with real hardware, while the remaining setup is tested in simulators. This enables reproducibility of specific test scenarios. For example, testing a real communication system and real radar sensors together with remaining system parts in a virtual environment is exploited in [LHX+18]. It ensures safe testing, as discussed by Zofka et al. [ZEF+18]. They augment a virtual lidar scan with real obstacle information such as pedestrians, which move in reality in an empty environment. Considering multiple vehicles, mixed-reality testing enables the Scalability of the control algorithm validation. Authors of [QAZ+10] and [GDGK14] implemented a real test vehicle moving on a proving-ground and is projected as an avatar in a virtual scenario where it potentially interacts with other simulated vehicles. An open problem remains to ensure the tested scenarios' reproducibility and seamless integration to different test scenarios and environments.

To summarize, both the application of multi-vehicle approaches for AVP and validation methodologies for multi-vehicle setups are rarely investigated fields. In AVP applications, proposed coordination methods either do not comply with the Safety requirement or cannot ensure Scalability. It is an open problem how to efficiently and seamlessly validate multi-vehicle scenarios using virtual validation.

1.3 Contributions and Outline

This thesis aims at developing optimization-based control strategies for vehicle coordination, considering a multi-vehicle setup with communication capabilities. The algorithms shall be real-time applicable and fulfill requirements discussed in Section 1.1 and, therefore, close important gaps in the literature discussed in the previous section.

The main contributions can be summarized as follows. The presented cooperative trajectory computation method results in feasible, and thus safe, coordination scenarios at anytime of the inter-vehicle communication process. The coordination problem is decomposed into a trajectory computation method and a vehicle sequence computation. Both layers are coupled to account for the problems' feasibility and the trade-off between each layer's efficiency interests. An implementation strategy for an efficient validation of multi-vehicle scenarios is presented and evaluated.

The thesis is split into two parts. Part I (Chapter 2 - Chapter 4) discusses the methodological design of distributed coordination algorithms that ensure a safe and cooperative vehicle interaction to compute privacy-preserving, efficient, and scalable solutions. To test such methods appropriately, Part II (Chapter 5 - Chapter 6) suggests a implementation

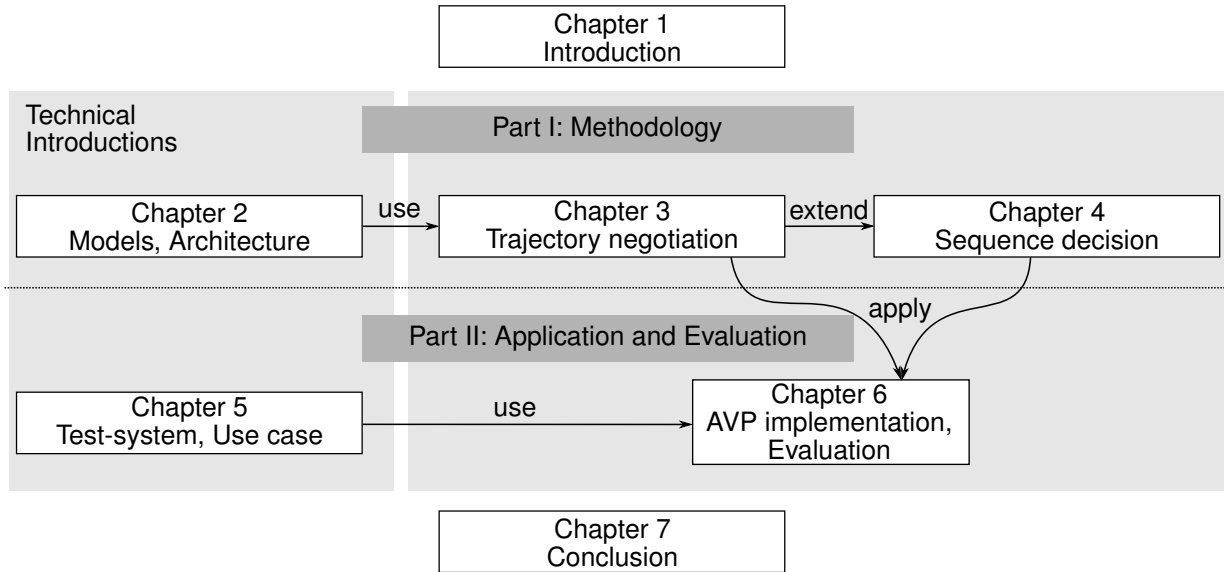


Figure 1.2: Relation between chapters.

and validation setup for multi-vehicle algorithms in the context of automated valet parking. In the following, we summarize each chapter’s content and contribution. Figure 1.2 illustrates the relation between the chapters.

Chapter 2: Model and Architecture Setting

In this chapter, control strategies and a coordination model suitable for multi-vehicle scenarios are introduced. Model predictive control (MPC) and distributed control classifications are discussed. The coordination model relies on defining conflict zones, representing areas in the coordination space where vehicles can enter from different directions and thus are potential collision regions. We specify a hierarchical architecture for computing the presented vehicle coordination problem. An upper-level is responsible for global control decisions, while on a lower-level, optimization problems are distributed between local vehicle units.

Models in this chapter have been partially published in [KMM+20] and have been submitted in [KMEH20]. The coordination and vehicle motion models, as well as the system architecture, have been partially published in [KMEH18; KMEH19; KDM+20; KMK+20].

Chapter 3: Distributed Trajectory Computation

This chapter discusses an algorithm for distributed trajectory negotiation. Each vehicle locally computes its desired trajectory taking decisions from surrounding vehicles into account. The proposed algorithm presents a holistic multi-vehicle coordination approach with rear-end and side collision avoidance for vehicles on a road network. The iterative negotiation process based on distributed model predictive control (DMPC) with Jacobi updates, which we refer to as distributed Jacobi over-relaxation (DJOR), is any-time feasible, i.e., can be interrupted after each iteration with a guaranteed network-wide feasible

solution. It scales independently of the number of participating vehicles and iterates fully distributed. At the same time, it is independent of a vehicle’s prediction horizon length, and safe maneuvers are prioritized over the coordination process.

The contribution of this chapter is based on publications [KMEH19; KMEH20].

Chapter 4: Vehicle Sequence Computation

This chapter presents a methodology to compute the vehicle sequence at conflicting areas where a combinatoric sequence decision is required. To this end, the overall multi-vehicle coordination problem is decomposed in quadratic programs (QP) (discussed in Chapter 3), and an integer program (IP). A resource-constrained project scheduling problem (RCPSP) is discussed as a concrete instance of the IP. It allows a seamless consideration of multi-vehicle coordination constraints and finds deadlock-free precedence relations. The methodology in this chapter is discussed with application to the use case of automated intersection crossing. Simulations show the combined performance of the distributed control negotiation from Chapter 3 and the proposed scheduling strategy.

The results presented in this chapter have been partly published in [KMK+20] and have been submitted for publication [KMEH20].

Chapter 5: Integration Platform

This chapter provides an architecture and the integration of a virtual validation framework for distributed control applications on the example of automated valet parking (AVP). It can be seamlessly integrated into an automated test system. The modular integration of the components into the simulation framework with defacto standard formats and interfaces enables the exchange of modules and contributes to future seamless test platforms.

We propose extending the virtual test-platform for multi-vehicle simulation with a real vehicle in the loop (ViL), which replaces one of the virtual vehicles. With the resulting mixed-reality tests, one can conclude the correctness of models and algorithmic behavior incorporating a real vehicle. Thus, it enables the validation of the multi-vehicle behavior with reasonable time and cost effort.

This chapter is based on methods from [KMM+20; KDM+20].

Chapter 6: Experimental Assessment

This chapter illustrates the evaluation results of the integrated test platform introduced in Chapter 5 with applied algorithmic methodologies introduced in Chapters 3 and 4. The additional introduction of layered path planning and lateral steering control laws completes the control procedure and the distributed coordination control strategies from previous chapters. Consequently, safe-by-design coordination of multiple autonomous vehicles in parking environments is achieved, and a distributed computation of the planning and control functions is enabled. The chapter outlines the modularity of the proposed coordination concept and models. The algorithmic performance is evaluated and discussed in comparison with different coordination strategies.

This chapter is based on results from [KMM+20; KDM+20].

Part I

**Distributed Coordination
Methodology**

Control Architecture and Model Setting

The field of distributed optimal control suggests a natural solution for the problem of coordinating multiple vehicles. This is because, in a multi-vehicle setup, several vehicles have computation and communication capabilities, which enables the formation of a global inter-vehicle network. Within such a network, safe and efficient vehicle trajectories can be found by solving constrained optimization problems.

An essential challenge to apply such distributed control strategies to multi-vehicle coordination is the formulation of appropriate models. In particular, vehicle models, coordination models, and architectures must be defined in a modular way and a suitable level of details to balance applicability and validity for real systems.

Modeling multi-vehicle coordination can be tackled on different levels. A common distinction is macroscopic and microscopic coordination problems. High-level traffic control deals traditionally with macroscopic properties such as traffic density, average speed, and traffic flow. Traffic control aims to improve the flow or throughput in a considered traffic network, such as a city district or highway network. Thus, the goal is optimizing global (traffic) interests in large areas while possibly neglecting single vehicles' interests in the network. Frequently applied models are the Lighthill-Whitham-Richards (LWR), [LW55; Ric56], and the Payne-Whitham, [Pay71; Whi11], models.

In contrast, single vehicles are considered in microscopic models, aiming for smaller areas such as a particular vehicle platoon. These models take each vehicle's local interests into account and often model the states *position*, *velocity*, and *acceleration* [PSSF19].

This thesis proposes a connection of the two often separately investigated fields by deriving distributed control methods for microscopic models while taking macroscopic-driven decisions into account.

The models in this chapter have been introduced in [KMM+20] and [KMEH20]. Furthermore, the coordination model, the vehicle motion models, and the system architecture have been applied in [KMEH18; KMEH19; KDM+20; KMK+20; EKM+20].

Outline

Section 2.1 introduces the concept of distributed optimal control in general and the model predictive control (MPC) framework, which is used throughout the thesis. We introduce longitudinal vehicle motion models for local vehicle control in Section 2.2. Those models are also a part of the multi-vehicle coordination strategy. Section 2.3 presents an introduction of the coordination model concept, which serves as a baseline for the multi-vehicle coordination scenarios. Finally, Section 2.4 discusses a control architecture description that combines local vehicle controllers with a global control that seeks to optimize traffic properties.

2.1 Distributed Optimal Control

This section briefly describes an general optimal control problem, the class of model predictive controllers, and classifications for the distribution of these problems. Optimal control and model predictive control notations are aligned with [RM09, Ch. 2], and the distributed control classifications with [NNC19]. In this section, we make no claim to completeness, but the goal is to provide the reader an overview of the used concepts and terms in the further course of this thesis.

Optimal Control

Let an infinite horizon constrained optimal control problem be described in continuous time τ by

$$\min_{u(\cdot)} \int_0^{\infty} V(x(\tau), u(\tau)) d\tau \quad (2.1a)$$

$$\text{s.t.} \quad \dot{x} = f_c(x, u), \quad (2.1b)$$

$$x(0) = x_0 \quad (2.1c)$$

$$(x(\tau), u(\tau)) \in \mathbb{X} \times \mathbb{U}, \tau \in [0, \infty). \quad (2.1d)$$

Here, $V : \mathbb{R}^{n_x+n_u} \rightarrow \mathbb{R}$ describes a scalar objective function, which depends on the system states $x \in \mathbb{R}^{n_x}$ and the control input vector $u \in \mathbb{R}^{n_u}$ with $u(\cdot)$ denoting its trajectory, x_0 is the initial condition, \mathbb{X} and \mathbb{U} describe state and input constraints sets, respectively. Ideally, one wants to determine a closed-loop control law for Problem (2.1). This can be for example achieved with dynamic programming. However, it is usually impractical for real applications given large state dimensions, an infinite horizon, as well as state and input constraints in (2.1).

A commonly used method to overcome the above discussed drawbacks are model predictive controllers (MPC), which make several simplifications to determine the optimal feedback control law of (2.1).

Model Predictive Control

First and foremost, MPC solves an open-loop optimization problem in every sampling time step with an updated initial state for each solution. Only the first element of the open-loop trajectory is then applied to the plant. This procedure results in an implicit control law, considering only current measurements, rather than an offline computed closed-loop control law for an arbitrary state. The determination of an implicit control law significantly reduces the computational load and is one of the main drivers to make MPC practically appealing. Additionally, an MPC optimization problem usually approximates the time continuous differential equation (2.1b) with a discrete time difference equation. Furthermore, the semi-infinite time interval $[0, \infty)$ is replaced by a finite interval $[0..M]$, with M called horizon, and a terminal condition to mimic the cut interval remainder (M, ∞) . These simplifications lead to a discrete time finite horizon optimization problem of the following form:

$$\min_{u(\cdot)} \sum_{t=0}^{M-1} V((x(t), u(t)) + V^M(x(M)) \quad (2.2a)$$

$$\text{s.t.} \quad x(t+1) = f(x(t), u(t)), \quad t \in [0..M-1] \quad (2.2b)$$

$$x(0) = x_0 \quad (2.2c)$$

$$(x(t), u(t)) \in \mathbb{X} \times \mathbb{U}, \quad t \in [0..M-1] \quad (2.2d)$$

$$x(M) \in \mathbb{X}^M, \quad (2.2e)$$

where t is the discrete time, the time difference equation (2.2b) represents the dynamical system model. $V^M(\cdot)$ is a terminal cost term, and \mathbb{X}^M describes a constraint set for the terminal system states. Remaining parts are defined as in (2.1). The MPC optimization (2.2) is then applied at each time step t in a receding horizon fashion according to Algorithm 1.

Algorithm 1 Model Predictive Control Procedure

- 1: clock $\leftarrow t$
 - 2: **Initialization:** $x(0) \leftarrow x_0$ ▷ measure current system state
 - 3: **Computation:** solve (2.2) ▷ numerically solve optimization problem
 - 4: **Application:** apply $u(0)$ to plant ▷ apply only first control action
 - 5: clock $\leftarrow t + 1$
-

Figure 2.1 conceptually illustrates the predictive nature of an MPC strategy and the applied input at the current time step.

Note that for a reasonable choice of M (dependent on computation platform) Step 3 of Algorithm 1 can be solved fast enough for real-time applications even for nonlinear systems. Moreover, the introduction of an implicit control law in Step 4 (and the following re-computation) induces a certain robustness against uncertainties by-design.

For a detailed discussion about MPC and its properties the interested reader is referred to [RM09].

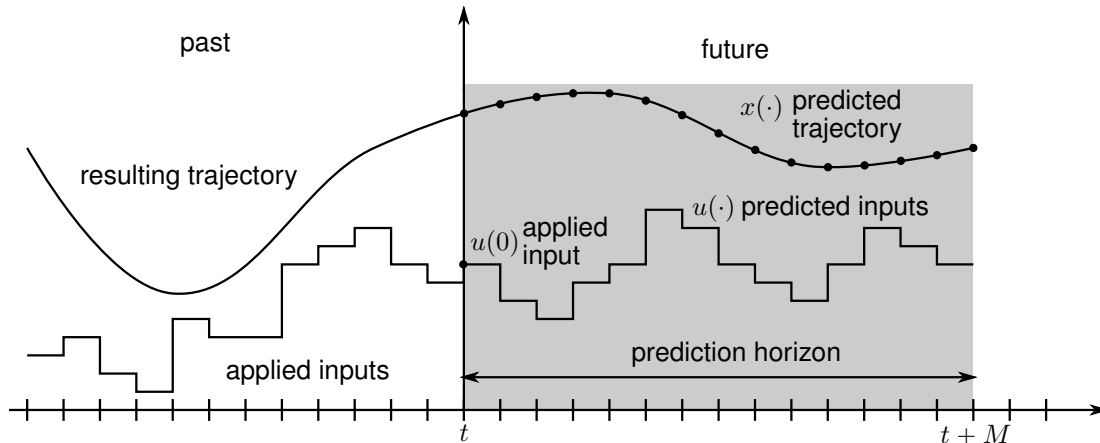


Figure 2.1: Illustration of model predictive control strategy.

Remark 2.1 (Condensed MPC). The formulation of (2.2) is referred to as condensed formulation. This implies that the decision variable solely depends on the control input u , and state x is eliminated through expressing it as dependency of u given the dynamics equation (2.2b). This leads to a reduced size of the optimization problem's decision variable but a dense Hessian matrix of the optimization problem. Alternatively, in a non-condensed formulation, the decision variable contains both state and input vectors, i.e., $z = (x, u)$. This leads to a larger decision variable but a sparse Hessian matrix, which can be exploited by certain numerical solvers and consequently lead to a computation speed compensation compared to a condensed formulation.

Remark 2.2 (Terminal condition). A terminal condition in the MPC formulation is required to provide stability guarantees of the finite horizon problem. It can be achieved by choosing the terminal constraint set to be control-invariant, i.e., a set for which a stabilizing control law exists that keeps the system within this set. The hard constraint set (2.2e) can either be replaced with terminal cost term V^M , or co-exist. Also, in certain cases, system stability can be guaranteed without terminal conditions if the prediction horizon is chosen sufficiently long [PN00].

Distributed Optimization in Control

In several applications it is of interest to solve an optimal control problem in a distributed manner. Driving reasons for this are, for example, that the control system is distributed in nature, i.e., the system consists of entities with different locations, or the problem is too large and requires a decomposition to ensure computational feasibility. A *distributed* control problem consists of N subsystems, often referred to as agents, processors, or units, which cooperatively solve a complex and large control task. Each control agent is assumed to have local computation capability and is able to communicate with other entities. Thereby, the term 'cooperative' means that each agent contributes through its local computation and the communication-based information exchange to the solution of the overall control task. In contrast, in a *decentralized* setup the different agents are not capable of

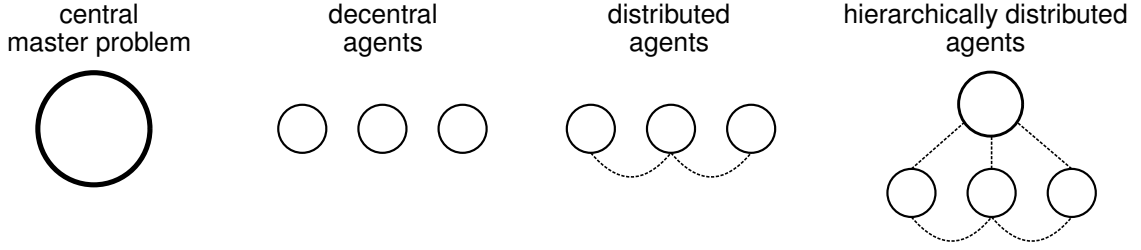


Figure 2.2: Decomposition architectures for large scale control problems. Dashed lines indicate communication flows.

communicating with other entities but solely solve a sub-problem of the overall control task in an isolated manner. A *centralized* problem consists of a single master unit, which would solve the overall global control problem. Note that each distributed and decentral control problem could theoretically be formulated as a central problem, which is in general not desirable or not even realizable for computational, safety, and privacy reasons. Lastly, a *hierarchical* structure can be defined to be a combination of a distributed and centralized architecture in which parts of the overall decision are commanded from a master unit to respective distributed agents. Figure 2.2 illustrates the above discussed control architectures.

In the sequel, we classify different problem structures, which commonly appear in large optimization problems. Assume an optimization problem of the following form, which can describe an overall large-scale control problem:

$$\min_z V(z) \quad (2.3a)$$

$$\text{s.t. } z \in \mathbb{C}. \quad (2.3b)$$

Note that an optimal control problem, such as described in (2.1) or (2.2), fit into this formulation. Thereby, $z \in \mathbb{R}^{n_z}$ is the optimization variable with a large dimension n_z , $V(z) : \mathbb{R}^{n_z} \rightarrow \mathbb{R}$ describes the problem's scalar objective function, and z is subject to constraints $\mathbb{C} \subset \mathbb{R}^{n_z}$. If an optimization problem models a naturally distributed system, the relation between subsystems can be reflected in the problem's structure. Following problem structures are commonly distinguished [NNC19]:

- **cost coupling**

The overall objective (2.3a) reads as a sum of local costs but all depend on the global optimization variable, i.e. $V(z) = \sum_{i=1}^N V_i(z)$, and the constraint set (2.3b) is shared by all agents.

- **common cost**

Objective (2.3a) is the same for all agents and the constraint set is a composition of local constraints, i.e. (2.3b) becomes $z \in \bigcap_{i=1}^N \mathbb{C}_i$, where \mathbb{C}_i is a constraint set that refers only to the variables of a local subsystem i .

- **constraint coupling**

The objective (2.3a) becomes a decoupled sum of local objectives, such that $V(z) =$

$\sum_{i=1}^N V_i(z_i)$ holds, with local optimization variables $z_i \in \mathbb{R}^{n_i}$ and dimension n_i . The constraint set (2.3b) is composed from local constraints, i.e., $z \in \bigcap_{i=1}^N \mathbb{C}_i$, but the constraints are coupled through a coupling function described by $\sum_{i=1}^N g_i(z_i) \leq 0$, with $g_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_c}$ and a number of n_c coupling constraints.

The problem of multi-vehicle coordination can be interpreted as an constraint-coupled distributed optimal control problem, because each vehicle is interested in optimizing its local behavior, e.g., fuel consumption or arrival time, but is subjected to constraints, which are coupled with other vehicles through sharing a common resource (road). Therefore, this problem structure will be applied and discussed in more detail in the remainder of the thesis. Before that, we will introduce modeling schemes for multi-vehicle coordination.

Remark 2.3 (Cooperative agents). While we use the term *cooperative* in a general understanding to indicate that multiple agents contribute through local decisions to the solution of an overall complex and distributed problem setting, in the literature, it is sometimes also aligned to the notation of game theory. This means that agents are cooperative if they solve a task in a common cost setting and non-cooperative if decoupled local objectives are modeled.

2.2 Longitudinal Vehicle Motion Modeling

Autonomous vehicles often drive on defined lane structures, e.g. on a road network, in an intersection, or in parking areas. Lateral adjustments might not be possible or not desirable in many of such scenarios. Therefore, the multi-vehicle coordination problem becomes a matter of coordinating the vehicles by adjusting their longitudinal trajectories.

Dynamic Model

Longitudinal vehicle dynamics can be modeled using the following differential equation representation [Raj11, Ch. 4]:

$$\dot{p}(\tau) = v(\tau) \tag{2.4a}$$

$$\dot{v}(\tau) = \frac{1}{m_{veh}} (F_{tire}(\tau) - F_{aero}(\tau) - F_{roll}(\tau) - F_{grav}(\tau)), \tag{2.4b}$$

where τ is the continuous time, $p(\tau)$ and $v(\tau)$ are the vehicle's position and velocity, respectively, m_{veh} the vehicle mass, $F_{tyre}(\tau)$ represents the longitudinal tire forces, $F_{aero}(\tau)$ the aerodynamic drag forces, $F_{roll}(\tau)$ the rolling resistance, and $F_{grav}(\tau)$ gravitational forces. $F_{tire}(\tau)$ moves the vehicle forward while the remaining forces are resistances. Thus, $F_{tire}(\tau)$ is influenced by the control input, which depends on the vehicle's powertrain dynamics. Powertrains can be modeled as a chain of engine dynamics, torque converter dynamics, transmission dynamics, and lastly wheel dynamics.

It is common to distinguish between low-level and high-level controllers to overcome the non-linearity in (2.4b) and ensure scalability. A high-level controller computes the target velocity or acceleration of a vehicle. The low-level controllers track these targets by computing respective powertrain commands.

Extended Kinematic Model

For high-level controllers, powertrain and vehicle dynamics (2.4) can be simplified in a kinematic model of the form

$$\dot{p}(\tau) = v(\tau) \quad (2.5a)$$

$$\dot{v}(\tau) = a(\tau) \quad (2.5b)$$

$$\dot{a}(\tau) = \frac{1}{T_{low}} (a_{des}(\tau) - a(\tau)), \quad (2.5c)$$

where $a(\tau)$ is the actual longitudinal vehicle acceleration state and $a_{des}(\tau)$ is the desired or commanded acceleration from a higher level and represents the control input. Thus, (2.5c) accounts for low-level powertrain and vehicle dynamics with the time constant T_{low} modeling a time constant of low-level control. Note that (2.5) is a linear model which can be handled in a computationally scalable way. Examples for low-level control units in vehicles are adaptive cruise control (ACC) embedded control units (ECU) or electronic power steering (EPS) ECUs. Details on the integration of high- and low-level controllers are discussed in Chapter 5.

Kinematic Model

Often, model (2.5) is further simplified to a double-integrator model,

$$\dot{p}(\tau) = v(\tau) \quad (2.6a)$$

$$\dot{v}(\tau) = a(\tau), \quad (2.6b)$$

where the longitudinal acceleration $a(\tau)$ is the control input. As this model is in the same model class as (2.5) – a linear model – we use (2.6) to derive the coordination methodologies in the following chapters. For implementation in Part II we will then switch to the more realistic model (2.5), what requires only minor adjustments.

Lateral Influence

In general, longitudinal vehicle motions cannot simply be separated from lateral influences as both are coupled to each other. Different strategies take this into consideration. First, for low speed maneuvers a longitudinal-lateral decomposition becomes valid. It is shown in [Pol18, Ch. 4] that kinematic bicycle model for steering, i.e., lateral control with constant speed assumption, is sufficiently accurate compared to high-fidelity driving models if

$$v(\tau) \leq \sqrt{\frac{a_{max}^{lat}}{\kappa(\tau)}}, \quad (2.7)$$

where $v(\tau)$ is the longitudinal velocity, a_{max}^{lat} is the maximum lateral acceleration, and $\kappa(\tau)$ is the driven curvature. Thereby, a_{max}^{lat} is the boundary value before the vehicle would lose traction.

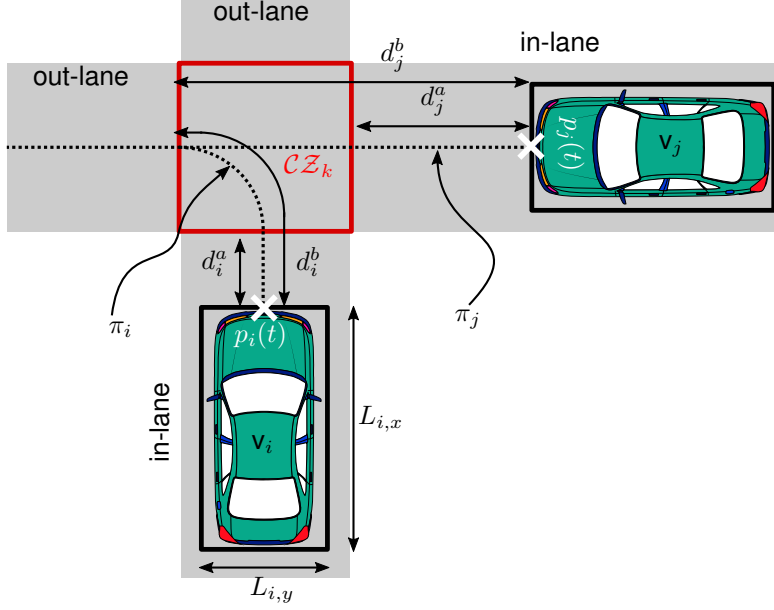


Figure 2.3: Coordination Scenario.

For higher velocities the consistency of above proposed models can be ensured through limiting the longitudinal velocity with respect to lateral forces considering a so called friction ellipse. This enables to relate longitudinal and lateral acceleration according to

$$\left(\frac{a(\tau)}{a_{max}^{long}}\right)^2 + \left(\frac{\kappa(\tau)v(\tau)^2}{a_{max}^{lat}}\right)^2 \leq 1, \quad (2.8)$$

with an upper bound on the longitudinal acceleration a_{max}^{long} and remaining parameters as introduced above [HZGF19].

In summary, the lateral influence on longitudinal dynamics can be neglected for either low speed or low curvature scenarios, given by (2.7). For other scenarios longitudinal acceleration and velocity can be constrained according to (2.8). Applying this within linear models requires appropriate linearization strategies.

2.3 Coordination Model

Assume a set of vehicles

$$\mathcal{V} = \{v_1, v_2, \dots, v_{N_v}\}, \quad (2.9)$$

with N_v vehicles moving along pre-defined paths on a road network. The path of vehicle $v_i \in \mathcal{V}$ is described by a set of N_{W_i} discrete waypoints

$$W_i = \{(p_x^1, p_y^1)^T, \dots, (p_x^{N_{W_i}}, p_y^{N_{W_i}})^T\}, \quad (2.10)$$

with positions in the 2D $x - y$ -plane. The reference point $p_i(t)$ of the vehicle is assumed to be located at the center front of the vehicle. Let $\pi_i : I \rightarrow \mathbb{R}^2$, with interval $I = [1, N_{W_i}]$, describe a \mathcal{C}^1 -curve which connects all waypoints W_i . Furthermore, assume the following:

Assumption 2.1. *Suppose that $p_i(t) \in W_i, \forall t$ hold, which ensures that a path can be perfectly tracked by a vehicle.*

Figure 2.3 illustrates the coordination scenario for two vehicles, as introduced in the following. We represent a vehicle by a rectangular bounding-box $\mathcal{B}_i(p_i(t), L_{i,x}, L_{i,y})$ with parameters $p_i(t) \subset \pi_i$, which is the vehicle's time-dependent position on its path, as well as $L_{i,x}$ and $L_{i,y}$ are the length and width of the box, respectively. By variable $t \in \mathbb{Z}^+$ we denote the discrete time scale. Given the path and vehicle representation, we define vehicle v_i 's distance state $d_i^n(t)$ with the arc-length from a curve segment of π_i , such that

$$d_i^n(t) = \begin{cases} \int_m^n \left\| \frac{\pi_i(s)}{ds} \right\|_2 ds, & \text{if } m \xrightarrow{\pi_i} n \\ - \int_m^n \left\| \frac{\pi_i(s)}{ds} \right\|_2 ds, & \text{if } n \xrightarrow{\pi_i} m, \end{cases} \quad (2.11)$$

with $\pi_i(m) = p_i(t)$, and $\pi_i(n) \in W_i$ a target reference waypoint on the curve. Notation $m \xrightarrow{\pi_i} n$ means that $\pi_i(m)$ is located before $\pi_i(n)$ following the curve in driving direction and vice versa for $n \xrightarrow{\pi_i} m$.

2.3.1 Conflict Zones

In general, collisions between vehicles are avoided if

$$\mathcal{B}_i(t) \cap \mathcal{B}_j(t) = \emptyset, \quad \forall t, \quad i, j \in \mathcal{V}, \quad i \neq j. \quad (2.12)$$

For coordinating vehicles on a road network areas where vehicle paths intersect, merge, and diverge are essential. This is the case at intersections, where condition (2.12) leads to areas which can be accessed only exclusively by a single vehicle at a time. We refer to these zones as conflict zones $\mathcal{CZ}_i \subset \mathbb{R}^2$, for $i \in \mathbb{I}_{1:N_{cz}}$ with a total number of conflict zones N_{cz} in the coordination space. These conflict zones will serve as reference areas to guarantee a safe multi-vehicle coordination. Let vehicle v_i cross \mathcal{CZ}_k before vehicle v_j , then a collision avoidance condition with regard to the vehicles' distance states is

$$d_i^b(t) + L_{i,x} + d_{j,s} \leq d_j^a(t). \quad (2.13)$$

Here, $d_{j,s}$ is a safety distance for vehicle v_j , $d_i^b(t)$ is the distance of vehicle v_i to the exit of \mathcal{CZ}_k , defined according to (2.11), with b describing the first waypoint of W_i outside of \mathcal{CZ}_k in driving direction, i.e.,

$$\begin{aligned} b &= \min_{q \in \mathbb{I}_{1:N_{w_i}}} q & (2.14) \\ \text{s.t. } \hat{q} &\leq q \leq N_{W_i} \\ (p_x^q, p_y^q)^T &\notin \mathcal{CZ}_k \\ (p_x^{\hat{q}}, p_y^{\hat{q}})^T &\in \mathcal{CZ}_k. \end{aligned}$$

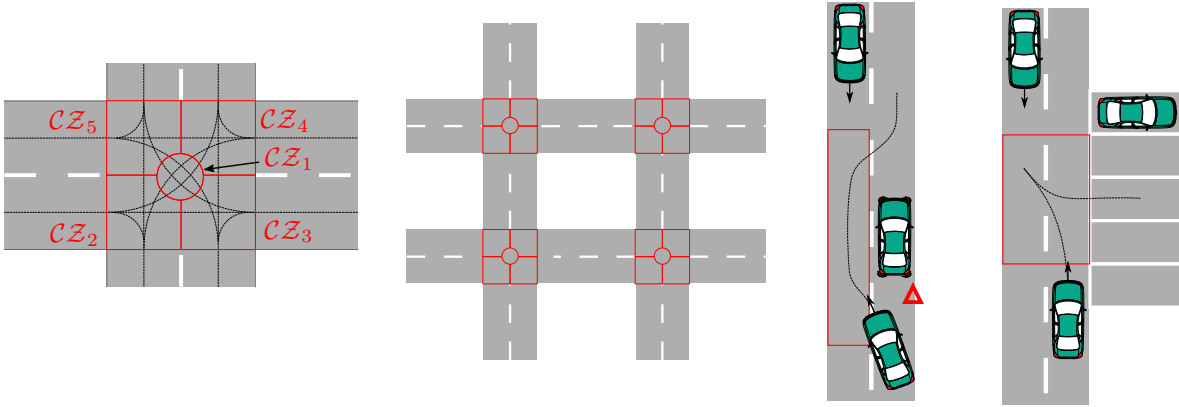


Figure 2.4: Conflict zone example cases.

Similarly, $d_j^a(t)$ is the distance of vehicle v_j to the entrance of \mathcal{CZ}_k , i.e.,

$$\begin{aligned}
 a &= \max_{q \in \mathbb{1}_{1:N_{w_i}}} q & (2.15) \\
 \text{s.t. } & 1 \leq q \leq \hat{q} \\
 & (p_x^q, p_y^q)^T \notin \mathcal{CZ}_k \\
 & (p_x^{\hat{q}}, p_y^{\hat{q}})^T \in \mathcal{CZ}_k.
 \end{aligned}$$

We assume the following:

Assumption 2.2. *Suppose the conditions*

$$d_i^b(t) + L_{i,x} \leq 0 \Rightarrow \mathcal{B}_i(t) \cap \mathcal{CZ}_k = \emptyset$$

and

$$d_i^a \geq 0 \Rightarrow \mathcal{B}_i(t) \cap \mathcal{CZ}_k = \emptyset$$

hold.

In this notation, we neglect the possibility that a vehicle enters or leaves a zone skewed. The simplification can be compensated by adding a buffer to the length $L_{i,x}$ of each vehicle v_i , or by imposing to have an appropriate minimal size of \mathcal{CZ}_k .

Conflict Zone Examples

Figure 2.4 illustrates different cases in which the above defined critical zone concept can be applied for multi-vehicle coordination. Starting from the left, the first plot shows an exemplary intersection scenario with five critical zones and possible paths leading through the intersection area. The second plot shows an extension to a road network scenario where the single intersection area is repeated several times. The third plot sketches a scenario in which two vehicles are coordinated with respect to a conflict zone, as one vehicle has to leave its own lane due to an obstacle ahead. Lastly, the right most plot shows a parking maneuver scenario. One of the vehicles wants to park backwards. Therefore, the zone in front of the parking bay is reserved for it.

Remark 2.4 (Conflict zones). Conflict zones are a commonly used concept for ensuring collision avoidance between robots [GBDLF14]. They are a powerful modeling tool for coordinating vehicles through intersections, especially when applied with optimal control methods, e.g., [ZMC16; HZGF16]. To increase efficiency, the distinction of several zones in one intersection area has been proposed by [ADV16; HZGF18]. Similar to this chapter, a microscopic vehicle modeling approach with conflict zones and collision avoidance conditions for intersection crossing scenarios are discussed in [Hul19, Ch. 3].

2.3.2 Coordination Graphs

In distributed control it is common to model the relation between distributed entities by means of graph theory. In the frame of multi-vehicle coordination it is convenient to distinguish two different types of graphs for describing the coordination scenario. The physical route of vehicles in the coordination scene is described by \mathcal{G}_{route} and is mainly invoked to determine the crossing order of vehicles for a given set of critical zones. Communication between vehicles is encoded in the graph \mathcal{G}_{com} , which determines the structure of information exchange for the distributed control laws computing the vehicles' trajectories. Both, \mathcal{G}_{route} and \mathcal{G}_{com} , are time-dependent graphs, as multi-vehicle coordination is a time-varying problem with moving vehicles which can enter or leave the coordination scene. For reasons of readability, we skip the notation of time dependency as the investigation of time transient behavior is not subject of this thesis.

We define the route graph as

$$\mathcal{G}_{route} := (\mathcal{V}_{route}, \mathcal{E}_{route}), \quad (2.16)$$

with a set of (possibly time-varying) vertices $\mathcal{V}_{route} = \{\mathcal{CZ}_1, \mathcal{CZ}_2, \dots\}$, which contains the conflict zones in the coordination scenario. Directed edges $\mathcal{E}_{route} \subseteq \mathcal{V}_{route} \times \mathcal{V}_{route}$ describe the route for each vehicle in the scenario. Each edge is labeled with the vehicle ID whose route is modeled with the respective edge. We allow several edges to connect the same set of vertices, which makes \mathcal{G}_{route} a directed multigraph.

The inter-vehicle communication structure is represented by

$$\mathcal{G}_{com} := (\mathcal{V}, \mathcal{E}_{com}), \quad (2.17)$$

with a vertex set \mathcal{V} , which is the vehicle set (2.9), and directed edges $\mathcal{E}_{com} \subseteq \mathcal{V} \times \mathcal{V}$ labeled with the conflict zone ID they are referring to. Given \mathcal{G}_{com} , a set of neighbors $\mathcal{N}_i = \{\mathcal{P}_i, \mathcal{S}_i\}$, $i \in \mathcal{V}$ is known. Neighbors can be classified as predecessor vehicles \mathcal{P}_i and successor vehicles \mathcal{S}_i through the direction of edges.

Finally, the relation between \mathcal{G}_{route} and \mathcal{G}_{com} is given through the adjacency matrix $A_{adj} = (a_{ij})^{|\mathcal{V}| \times |\mathcal{V}|}$, with a_{ij} describing n-tuples with passed conflict zones relative to vehicle IDs. Figure 2.5 illustrates the relation between above introduced graphs and Chapter 6 discusses an exemplary setup of them in a multi-vehicle coordination scenario.

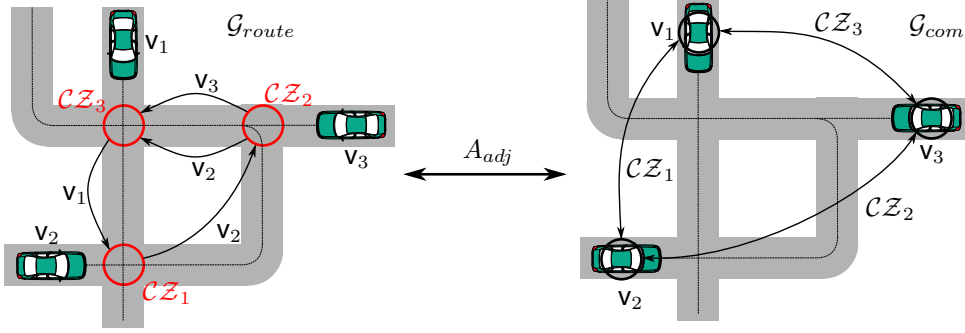


Figure 2.5: Exemplary graph representations and relation between them.

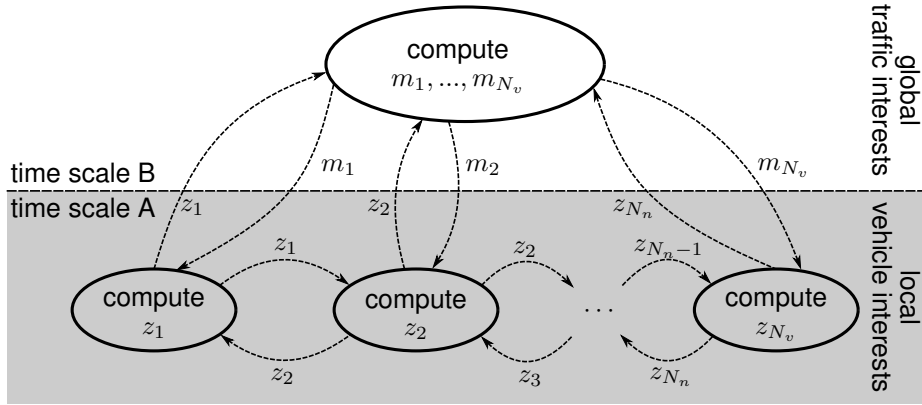


Figure 2.6: Distribution of the coordination problem.

2.4 Distributed Coordination Architecture

We describe a multi-vehicle coordination problem by the following constrained optimization problem:

$$\begin{aligned}
 \min_{\substack{z_1, \dots, z_{N_v} \\ m_1, \dots, m_{N_v}}} & \sum_{i=1}^{N_v} V_i(z_i) \\
 \text{s.t.} & z_i \in \mathbb{X}_i \times \mathbb{U}_i, \quad i \in \mathcal{V} \\
 & \sum_{i=1}^{N_v} g_i(z_i, m_i) \leq 0, \quad i \in \mathcal{V} \\
 & m_i \in \mathbb{Z}^+, \quad i \in \mathcal{V}.
 \end{aligned} \tag{2.18}$$

Thereby, z_i describes a continuous optimization variable referring to a local vehicle $i \in \mathcal{V}$, which is constrained by the set $\mathbb{X}_i \times \mathbb{U}_i \subset \mathbb{R}^{n_i}$ with a variable dimension n_i . The function $g_i: \mathbb{R}^{n_i} \times \mathbb{Z} \rightarrow \mathbb{R}^{n_c}$ models inter-vehicle coupling constraints, which depends on z_i and integer variables m_i . The dimension of coupling constraints is defined by n_c . Continuous variables z_i refer to local vehicle dynamics, while the integer variables m_i mirror the combinatoric nature of a multi-vehicle coordination problem. The sum of objective functions $V_i: \mathbb{R}^{n_i} \rightarrow \mathbb{R}$, which shall be minimized, represents the respective vehicles' interests.

Figure 2.6 illustrates how the overall optimization problem (2.18) is decomposed into

two different distributed computation problems. The hierarchical architecture contains local optimization problems - one for each vehicle - in the lower layer. They are computed in a distributed fashion while local vehicle interests and safety constraints are considered for a provided integer decision. This layer refers to microscopic models and is covered in Chapter 3.

In contrast, the upper layer is responsible for computing the integer solution of problem (2.18) distributed from the remaining problem in the lower layer. It represents the interests from a global traffic perspective, i.e., from a macroscopic point of view. A solution of it is introduced in Chapter 4. Note that both layers may operate on different time scales, as problems have different complexity, sizes, and necessity of updating results.

2.5 Summary and Discussion

In this chapter, control methods and models that are suitable for multi-vehicle coordination scenarios are formulated. The proposed models enable to use longitudinal vehicle dynamics, which can be controlled by linear methods. This contributes to a scalable implementation of the distributed coordination problem. The introduction of the conflict zone framework to model the coordination space enables the derivation of safety definitions for inter-vehicle coordination scenarios. Lastly, the proposed hierarchically distributed control architecture serves as the baseline for developed control concepts, which bridge global traffic interests with local vehicle interests.

The validity of linear longitudinal vehicle dynamics is limited to low vehicle speeds. A (non-linear) coupling to the lateral dynamics has to be considered to apply to more general scenarios. This is taken into account by the constraint (2.8), limiting the longitudinal acceleration and velocity with respect to lateral dynamics. Alternatively, it is possible to constrain the lateral steering capability, as in (2.7), which is proposed in [PANLF18] for an MPC framework. Using these relations in the proposed control setting requires a linearization method.

Problem (2.18) describes the multi-vehicle coordination problem as an optimization problem with coupling constraints between locally separated sub-units. The following chapters introduce distributed solution approaches to this problem according to the architecture given in Figure 2.6.

Distributed Trajectory Computation

The trajectory of an automated vehicle shall be computed in real-time and it must ensure a safe driving decision. At the same time, it shall provide efficient and comfortable maneuvers. In the case of multiple automated vehicles, the decision of a single-vehicle depends on the other vehicles' decisions. This interdependency makes it challenging to ensure safe trajectory decisions for each local vehicle, and to still ensure efficient and comfortable maneuvers. The fact that vehicles have to communicate via a wireless channel to exchange information, e.g., with vehicle-to-everything (V2X) communication, poses an additional challenge to the system design. A crucial property of multi-vehicle coordination for real-world implementations is to guarantee safety not only for nominal driving scenarios but also in the case of unforeseen events. The challenge is that even if a brake-safe distance is considered, a violation of a coupling-constraint due to an unforeseen event can lead to infeasibility of the optimization problems.

Safety is commonly ensured through the formulation of constraint optimization problems, while predictive control methods target achieving efficient maneuvers. Model predictive control (MPC) is a powerful trade-off methodology for computing trajectories in real-time. It enables the combined consideration of dynamical vehicle and safety constraints, as well as global coordination performance interests, and has therefore been widely applied in the literature, e.g., [ZLL+16; KK14; DM06].

In a multi-vehicle setting, MPC can either be used in a decentralized manner [QGDLFM15], i.e., without neighbor communication, or in a distributed setup using robust control formulations [FS12]. While these two approaches consume little or no communication effort, they tend to result in conservative solutions. Iterative negotiation approaches, which exchange information between vehicles several times during one sampling time step, can be applied to achieve increased efficiency. It can be realized by sequential computations [CFW+14; RH04] or parallel iterations [ZGWF17; KMEH18]. This, in turn, can lead to high utilization of the communication channel with the risk of losing real-time guarantees.

In this chapter, trajectories are computed in an iterative and distributed MPC setup while solutions are guaranteed to fulfill constraints after each iteration step. This is referred to as *any-time feasibility*. It means that iterations can be stopped after each iteration with a resulting safe solution. This preserves real-time properties while in case of enough communication resources, further iterations lead to improved, i.e., more efficient, trajectories. The chapter covers the lower negotiation of local vehicle interests, as illustrated in Figure 2.6. The methodology is extended such that it can deal with emerging unplanned events. This enables the consideration of long prediction horizons, leading to efficient solutions. Moreover, the vehicles can safely react to uncertainties that have not been considered in the nominal planning phase. Overall, the approach prioritizes safety reactions, such as emergency braking, over the nominal driving behavior.

The chapter is based on publications [KMEH19; KMEH20].

Outline

At the beginning of the chapter, collision avoidance conditions are stated, which must be met during the distributed trajectory computation process in Section 3.1. This is followed by a decomposition of the central coordination problem into local optimization problems in Section 3.2. Section 3.3 introduces the distributed algorithm and its properties, which is extended toward uncertainty consideration in Section 3.4. Lastly, the chapter provides numerical results in Section 3.5 and concluding remarks in Section 3.6.

3.1 Collision Avoidance Conditions

Recapitulating the collision avoidance condition from Section 2 leads to areas, which can only be accessed by a single vehicle. They are labeled \mathcal{CZ}_i with a unique identifier i . In relation to these zones, we introduced the inequality condition (2.13), which guarantees collision avoidance between vehicles sharing a common \mathcal{CZ}_i .

Given the fact that vehicles move on pre-defined lanes on a road network, the validity of (2.13) can be specified in more details if we are able to distinguish between different maneuvers. Therefore, we define the following cases, distinguishing the directions in which vehicles v_i and v_j approach \mathcal{CZ}_k (in-lane) and in which they leave it (out-lane):

- **c1:** from **same** in-lane to **same** out-lane,
- **c2:** from **same** in-lane to **different** out-lane,
- **c3:** from **different** in-lane to **same** out-lane,
- **c4:** from **different** in-lane to **different** out-lane.

Figure 2.3 illustrates Case **c3** as an example. Let t_i^b be the time-instant just after vehicle v_i leaves \mathcal{CZ}_k (exit-time), such that

$$d_i^b(\tau_i^b) = -L_{i,x} \text{ and } t_i^b = \left\lceil \frac{\tau_i^b}{T_s} \right\rceil \quad (3.1)$$

holds, where $\tau_i^b \in \mathbb{R}_0^+$ is the exact solution in continuous time and T_s the discrete sampling time. This enables a time-dependent validity definition according to cases **c1-c4** such that (2.13) holds for

$$\mathbf{c1:} \quad \forall t \tag{3.2a}$$

$$\mathbf{c2:} \quad t < t_i^b \tag{3.2b}$$

$$\mathbf{c3:} \quad t \geq t_i^b \quad \vee \quad d_j^a(t) > d_{j,s} \text{ for } t < t_i^b \tag{3.2c}$$

$$\mathbf{c4:} \quad d_j^a(t) > d_{j,s} \text{ for } t < t_i^b. \tag{3.2d}$$

3.2 Control Model Decomposition

In this section, the centralized optimization problem is introduced as a mixed integer quadratic program (MIQP). Thereafter, the centralized problem will be decomposed into a distributed MPC setup with local quadratic programming (QP) problems and a given vehicle crossing sequence. For a single vehicle, we assume the time-discrete dynamics to be:

$$\underbrace{\begin{pmatrix} d_i \\ v_i \end{pmatrix}^+}_{x_i(t+1)} = \underbrace{\begin{pmatrix} 1 & -T_s \\ 0 & 1 \end{pmatrix}}_{A_i \in \mathbb{R}^{2 \times 2}} \underbrace{\begin{pmatrix} d_i \\ v_i \end{pmatrix}}_{x_i(t)} + \underbrace{\begin{pmatrix} -T_s^2 \\ T_s \end{pmatrix}}_{B_i \in \mathbb{R}^{2 \times 1}} \underbrace{a_i}_{u_i(t)}, \tag{3.3}$$

with distance state $d_i = d_i^{N_{W_i}}$ (according to (2.11)) representing the distance to the end of vehicle v_i 's path with a number of N_{W_i} discrete waypoints, v_i its velocity, and a_i its acceleration input.

3.2.1 Centralized MPC

The centralized system is achieved by concatenating individual vehicle models, such that we achieve the system and input matrix

$$A = \text{diag}(A_1, A_2, \dots, A_{N_v}), \quad B = \text{diag}(B_1, B_2, \dots, B_{N_v}), \tag{3.4}$$

and the state and input vectors

$$x(t) = (x_1^T(t), \dots, x_{N_v}^T(t))^T \quad \text{and} \quad u(t) = (u_1^T(t), \dots, u_{N_v}^T(t))^T, \tag{3.5}$$

respectively, with N_v vehicles in the scenario. Now, we formulate the centralized coordination task as a finite horizon optimization problem

$$V^* = \min_{\bar{x}, \bar{u}, m} \sum_{i=1}^{N_v} V_i(x_i(t), u_i(t)) \quad (3.6a)$$

s.t.

$$x(k+1|t) = Ax(k|t) + Bu(k|t) \quad k \in \mathbb{I}_{t:t+M-1} \quad (3.6b)$$

$$x(t|t) = x(t) \quad (3.6c)$$

$$x(k|t) \in \mathbb{X} \quad k \in \mathbb{I}_{t+1:t+M} \quad (3.6d)$$

$$u(k|t) \in \mathbb{U} \quad k \in \mathbb{I}_{t:t+M-1} \quad (3.6e)$$

$$d_j^a(k|t) - d_{j,s} \geq 0 \quad \begin{cases} \mathbf{c3}, \mathbf{c4} \\ (i, j) \in \mathcal{T}_m \end{cases} \quad (3.6f)$$

$$d_i^b(k|t) + L_{i,x} + d_{j,s} - d_j^a(k|t) \leq 0 \quad \begin{cases} \mathbf{c1-c3} \\ (i, j) \in \mathcal{T}_m \end{cases} \quad (3.6g)$$

$$m \in \mathbb{I}_{1:|\mathcal{T}|}, \quad (3.6h)$$

where the optimization variables are defined by

$$\bar{x} = (x(t+1|t)^T, \dots, x(t+M|t)^T), \quad (3.7)$$

and

$$\bar{u} = (u(t|t)^T, \dots, u(t+M-1|t)^T), \quad (3.8)$$

for prediction horizon M . The objective function (3.6a) is the sum of the local vehicle objectives

$$V_i(x_i(t), u_i(t)) = \|x_i(M|t) - x_i^{ref}\|_{P_i}^2 + \sum_{k=t}^{t+M-1} \left(\|x_i(k|t) - x_i^{ref}\|_{Q_i(k)}^2 + \|u_i(k|t)\|_{R_i(k)}^2 \right), \quad (3.9)$$

with constant state references x_i^{ref} and positive semi-definite weighting matrices $P_i \in \mathbb{S}_0$ and $Q_i(k) \in \mathbb{S}_0^{2 \times 2}$, as well as a positive definite input weighting matrix $R_i(k) \in \mathbb{S}$. Notation $x(k|t)$ describes the prediction of the state vector for time step k computed at time t , and similar for the control input u . Constraint (3.6b) is the central model containing all individual vehicle dynamics, (3.6c) is the initial state, \mathbb{X} and \mathbb{U} are polyhedral constraint sets. Let \mathcal{T} be a given tree in which a node is defined by a tuple (i, j) describing the order of two vehicles (v_i before v_j) and each path from the root to a leaf node, \mathcal{T}_m , describes a feasible crossing order through a conflict zone, i.e.,

$$\mathcal{T}_m = ((i, j), (j, k), (k, l), \dots), \quad i, j, k, l \in \mathcal{V}. \quad (3.10)$$

Thus, optimizing over the integer variable m determines the best tree path with respect to (3.6a). Let, with a slight abuse of notation, $|\mathcal{T}|$ be the number of paths (feasible crossing orders) of \mathcal{T} for a given scenario.

Distance constraints (3.6f)–(3.6g) are valid for certain time intervals which are defined according to Cases **c1-c4** and listed in Table 3.1. Figure 3.1 illustrates the feasible $d_i - d_j$ configuration space according to these distance constraints for a crossing tuple (i, j) .

Table 3.1: Time intervals for distance constraints.

$k \in \dots$	(3.6g),(3.13i)	(3.13j)	(3.6f)	(3.13h)
c1:	$\mathbb{I}_{t+1:t+M}$	$\mathbb{I}_{t+1:t+M}$	-	-
c2:	$\mathbb{I}_{t+1:t_i^b}$	$\mathbb{I}_{t+1:t_j^b}$	-	-
c3:	$\mathbb{I}_{t_i^b:t+M}$	$\mathbb{I}_{t_j^b:t+M}$	$\mathbb{I}_{t+1:t_i^b}$	$\mathbb{I}_{t+1:t_j^b}$
c4:	-	-	$\mathbb{I}_{t+1:t_i^b}$	$\mathbb{I}_{t+1:t_j^b}$

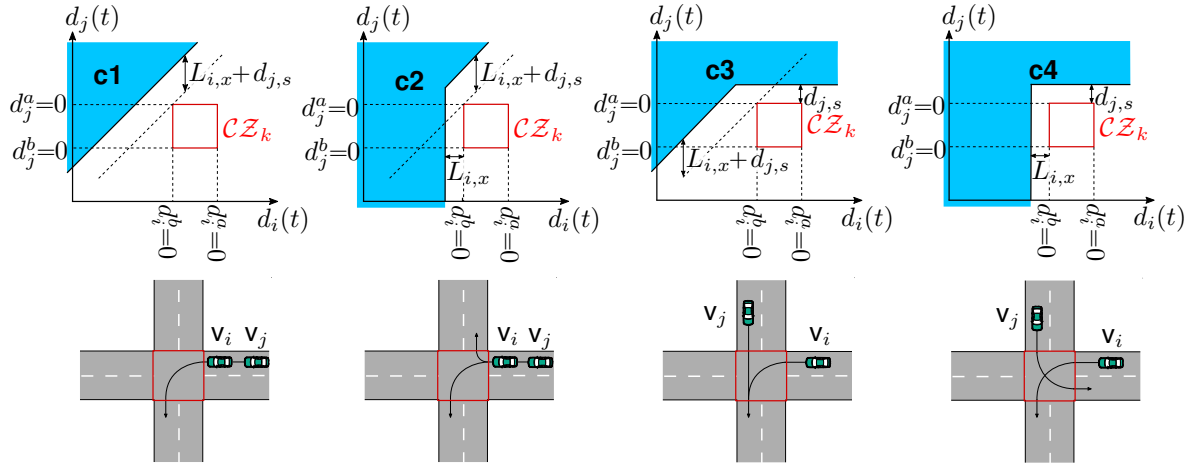


Figure 3.1: Top row: feasible $d_i - d_j$ configuration space (blue shaded area) of coordination problem (3.6) for crossing order (i, j) . Bottom row: exemplary vehicle constellation at an intersection.

Note that if a case distinction is not possible, application of Case **c1** leads always to a safe vehicle coordination scenario. Moreover, the above optimization problem (3.6) is formulated for a single conflict zone for simplicity of notation. The consideration of several zones, however, can be obtained by extending the problem with several decision trees \mathcal{T} , one for each zone.

As (3.6a) is quadratic, (3.6b)–(3.6g) are linear, and (3.6h) is an integer, (3.6) is a mixed integer quadratic program (MIQP). There are evident reasons why this problem is impractical to be solved online in real applications. First, the problem is computationally hard to solve, as scalability cannot be guaranteed through the central formulation, and the integer decision introduces a further computational burden. Additionally, the time intervals in Table 3.1 depend implicitly on the solution of (3.6) that requires the computation of a multi-level optimization problem. Solving such problems exactly is infeasible on real-time platforms [HZGF18]. Second, (3.6) requires central knowledge of all individual vehicle models, what is undesirable considering the Privacy of vehicle data requirement. Third, trajectories are safety-relevant decisions, which are preferred to be computed on-board of a vehicle in order to guarantee a safe behavior even in the case of communication disruptions.

To overcome the drawbacks discussed above, we propose a decomposition of (3.6) into local optimization problems solved separately by each individual vehicle which is connected, and sharing resulting information with neighboring vehicles. The decomposition is designed

to achieve computationally feasible sub-problems, where vehicle models are kept private and trajectories can be verified locally to fulfill necessary Safety criteria.

3.2.2 Distributed MPC

The diagonal structure of (3.4) immediately suggests a decomposition of (3.6a) and (3.6b) into local sub-problems. However, constraints (3.6f)–(3.6g) contain inter-vehicle relations. To decompose these, we repeat respective coupling constraints in each coupled local sub-system and move the integer decision (3.6h) to the central infrastructure node. A scheduling-based method where m is computed in an approximate way given vehicles' trajectories is introduced in Chapter 4. In the following, we will assume a given integer decision m , i.e., a path from the root to a leaf node of \mathcal{T} and thus the order in which vehicles cross \mathcal{CZ}_k . The crossing order can be represented with a directed graph, as defined in Subsection 2.3.2,

$$\mathcal{G}_{com} = (\mathcal{V}, \mathcal{E}_{com}), \quad (3.11)$$

where the set of vehicles \mathcal{V} are the vertices and directed edges $(i, j) \in \mathcal{E}_{com}$, with $i, j \in \mathcal{V}$, meaning that vehicle v_i crosses \mathcal{CZ}_k before vehicle v_j . Given \mathcal{G}_{com} , we define the set of neighbors of vehicle v_i ,

$$\mathcal{N}_i = \{\mathcal{P}_i, \mathcal{S}_i\}, \quad (3.12)$$

where the set of predecessors \mathcal{P}_i contains nodes connected via incoming edges to node $i \in \mathcal{V}$ and similarly successors \mathcal{S}_i nodes from outgoing edges.

Now, (3.6) can be decomposed into local quadratic programming (QP) problems with coupling constraints. Thus, a local QP problem of vehicle v_i has the form

$$V_i^* = \min_{\bar{x}_i, \bar{u}_i} V_i(x_i(t), u_i(t)) \quad (3.13a)$$

s.t.

$$x_i(k+1|t) = A_i x_i(k|t) + B_i u_i(k|t) \quad k \in \mathbb{I}_{t:t+M-1} \quad (3.13b)$$

$$x_i(t|t) = x_i(t) \quad (3.13c)$$

$$x_i(k|t) \in \mathbb{X}_i \quad k \in \mathbb{I}_{t+1:t+M} \quad (3.13d)$$

$$u_i(k|t) \in \mathbb{U}_i \quad k \in \mathbb{I}_{t:t+M-1} \quad (3.13e)$$

$$x_i(t+M|t) \in \mathbb{X}^M \quad (3.13f)$$

$$u_i(t+M-1|t) \in \mathbb{U}^M, \quad (3.13g)$$

$$d_i^a(k|t) - d_{i,s} \geq 0 \quad \mathbf{c3}, \mathbf{c4}, j \in \mathcal{P}_i \quad (3.13h)$$

$$d_i^b(k|t) + L_{i,x} + d_{j,s} - d_j^a(k|t) \leq 0 \quad \mathbf{c1-c3}, j \in \mathcal{S}_i \quad (3.13i)$$

$$d_j^b(k|t) + L_{j,x} + d_{i,s} - d_i^a(k|t) \leq 0 \quad \mathbf{c1-c3}, j \in \mathcal{P}_i, \quad (3.13j)$$

with local optimization variables

$$\bar{x}_i = (x_i(t+1|t)^T, \dots, x_i(t+M|t)^T), \quad (3.14)$$

$$\bar{u}_i = (u_i(t|t), \dots, u_i(t+M-1|t)), \quad (3.15)$$

and local polyhedral constraint sets (3.13d) and (3.13e). The definition of terminal constraints (3.13f) and (3.13g) will be discussed in Subsection 3.3.2.

coupling constraints and only in bilateral form, i.e., a constraint between system variable of the vehicle v_i and a neighboring vehicle v_j . Similarly, the centralized problem can be written as

$$(z^*, m^*) = \underset{z, m}{\operatorname{argmin}} \sum_{i=1}^{N_v} V_i(z_i) \quad (3.17a)$$

$$\text{s.t. } \mathcal{A}z - b \leq 0 \quad (3.17b)$$

$$\mathcal{A}_m^d z - b_m^d \leq 0 \quad (3.17c)$$

$$m \in \mathbb{I}_{1:|\mathcal{T}|}, \quad (3.17d)$$

for (3.6) with $z = (\bar{x}, \bar{u})$, (3.17b) representing (3.6b)–(3.6e), and (3.17c) containing (3.6f)–(3.6g).

3.3.1 Distributed Over-Relaxation Algorithm

The DJOR algorithm iteratively negotiates between vehicles $v_i \in \mathcal{V}$ towards a solution, while an interruption after each iteration results in a feasible solution. Therefore, let $z_i^{(l)}$ be the solution of the l -th inter-sampling iteration between t and $t + 1$, \hat{z}_i a feasible initial candidate at the beginning of iterations, and z_i^* the result of a local optimization problem (3.16). Algorithm 2 summarizes the DJOR procedure, in which the update variable ω_i

Algorithm 2 Distributed Jacobi Over-Relaxation

- 1: clock $\leftarrow t$
- 2: **Initialization:**
- 3: $l \leftarrow 0$
- 4: $\forall i \in \mathcal{V}$: receive $z_j^{(l)} = \hat{z}_j, j \in \mathcal{N}_i$
- 5: **repeat**
- 6: **Computation:**
- 7: $\forall i \in \mathcal{V}$ in parallel: compute $z_i^* \left(\left\{ z_j^{(l)} \right\}_{j \in \mathcal{N}_i} \right)$
- 8: determine next iterate with $\omega_i \geq 0, \omega_i + \omega_j = 1$:

$$z_i^{(l+1)} = \omega_i z_i^* \left(\left\{ z_j^{(l)} \right\}_{j \in \mathcal{N}_i} \right) + (1 - \omega_i) z_i^{(l)} \quad (3.18)$$

- 9: **Synchronization:**
 - 10: share $z_i^{(l+1)}$ with Vehicles $j \in \mathcal{N}_i$
 - 11: $l \leftarrow l + 1$
 - 12: **until** $l > l_{max} \vee V_i(z_i^{(l-1)}) - V_i(z_i^{(l)}) < \gamma \forall i \in \mathcal{V}$
 - 13: apply $u_i(t|t), \forall i \in \mathcal{V}$ to local vehicle systems
 - 14: clock $\leftarrow t + 1$
-

defines the degree of over-relaxation and γ a termination condition. The signal flow is illustrated in Figure 3.2. Solving the optimization problems and the iterative updates can be conducted fully distributed with trajectory exchange between neighboring vehicles after

each iteration. To illustrate this, compare (3.18) with

$$z^{(l+1)}(\omega_i, z_i^*, z^{(l)}), \quad i \in \mathcal{V}, \quad \omega_i \geq 0, \quad \sum_{i=1}^{N_v} \omega_i = 1, \quad (3.19)$$

which states the structure of the standard Jacobi over-relaxation (JOR) update step. It requires central knowledge of z and depends on the total number of vehicles N_v .

Remark 3.1 (Over-relaxation concept). Let two vehicles, v_i and v_j , negotiate about their inter-vehicle distance $d_{inter} \in \mathbb{R}$ during a platooning scenario. Figure 3.3 illustrates the negotiation process example with a safety distance constraint $d_{j,s}$. Then, d_{inter} is a shared resource between both vehicles since longitudinal control actions from both vehicles influence this resource. This inter-vehicle distance is a convex space as it lies in \mathbb{R} . At iteration step l both vehicles know the other vehicle's state and state predictions. Now, both vehicles make a new control decision in parallel for iteration step $l + 1$ while based on the given state knowledge from step l they decide how much of the shared resource shall be consumed by each vehicle. Due to the parallelism, it cannot be ensured that this limited resource is distributed between the two vehicles without exceeding the resource amount. Thus, the convex update (3.18) between each vehicle's decision from step l and $l + 1$ is applied, respectively, which ensures that the resource cannot be overused with $\omega_i + \omega_j = 1$ and $\omega_i, \omega_j \geq 0$.

An interesting aspect of this concept is that decisions can be made without knowing the other vehicle's dynamics. This supports the Privacy requirement stated in Chapter 1, since there is no need to exchange vehicle models. Moreover, Safety with respect to the shared resource d_{inter} is guaranteed as it can never be overused, Scalability is ensured as (3.18) updates only local variables, and the iterative procedure targets at an increased Efficiency of the resulting trajectory through negotiating about the consumption of the shared resource d_{inter} .

Originally, the iterative Jacobi over-relaxation (JOR) algorithm has been introduced in [BT89, Ch. 2.4], which discusses the important property that each iteration results in a network-wide feasible solution. A JOR decomposition for input-constraint distributed MPC problems is discussed in [SVR+10], which has been extended towards general constraint optimization problems in [DDKDS17]. An essential novelty in this thesis is that the update step can be conducted fully distributed, while in the above-discussed references, a global update step is required.

In the next subsection, we present the construction of initial candidates \hat{z}_j , an essential point to guarantee feasibility. Thereafter, we discuss the properties *feasibility*, *scalability*, and *convergence* of Algorithm 2.

3.3.2 Feasible Initial Guess

At the beginning of a time step, each vehicle suggests a feasible initial candidate \hat{z}_i for the inter-sampling iterations, which will be shared with its neighboring vehicles $v_j \in \mathcal{N}_i$. In general, it is difficult to determine candidates which preserve feasibility among the complete distributed system. This is because, in the given setup, local vehicles do not

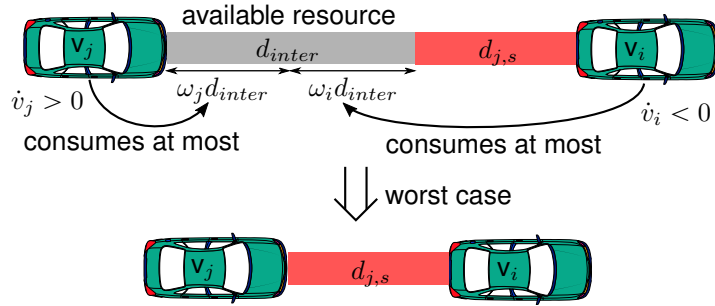


Figure 3.3: Illustration of the over-relaxation concept. Vehicles v_i and v_j negotiate in parallel about the shared distance space d_{inter} . The shared space cannot be overused and a minimum safety distance $d_{j,s}$ has to be ensured.

have information about their neighbors' dynamical capabilities. To resolve this, authors of [SVR+10; DDKDS17] propose using the system's steady-state as a terminal constraint since they handle stabilizing problems such as spring-damper systems. In order to be applicable in a multi-vehicle scenario, this idea requires adjustments. Therefore, we use a stand-still condition as terminal constraints (3.13f) and (3.13g), such that

$$\mathbb{X}^M = \{x_i | v_i = 0\} \subset \mathbb{X}_i \text{ and } \mathbb{U}^M = 0 \subset \mathbb{U}_i. \quad (3.20)$$

Now, we suggest the following initial candidate for the upcoming time step $t + 1$

$$\hat{z}_i(: | t+1) = \left(z_i^{(l)}(t+1:t+M|t), (x_i'^T, u_i'^T) \right), \quad (3.21)$$

with the last negotiation result $z_i^{(l)}$ from time step t and the stand-still extension $(x_i'^T, u_i'^T) = (d_i(t+M|t), 0, 0)$.

However, it is not the actual target to bring all vehicles to stand-still. To this end, the terminal condition will be solely used to guarantee system-wide feasibility, while stand-still is not desired to be applied during a nominal coordination scenario. For that reason, we propose the following method.

3.3.3 Performance Compensation

In order to emulate the behavior of a trajectory which is not required to come to a full stop, time-dependent objective weights can be applied. Therefore, we introduce two planning phases within the horizon length M of a local MPC problem (3.13). The first one is referred to as *nominal-planning*, for $1 \leq k < k_{brake}$, and the second as *planning-to-full-stop*, for $k_{brake} \leq k \leq M$. Accordingly, we define the weights

$$Q_i(k)[R_i(k)] = \begin{cases} Q_i^{no}[R_i^{no}] & \text{for } 1 \leq k < k_{brake} \\ \mathbf{0} & \text{for } k_{brake} \leq k \leq M, \end{cases} \quad (3.22)$$

where $Q_i^{no} \in \mathbb{S}_0^{2 \times 2}$ is a constant weight for the nominal-planning phase and similar for $R_i^{no} \in \mathbb{S}$. $\mathbf{0}$ describes a zero matrix with appropriate dimensions. The point k_{brake} is

the prediction step from which onward the trajectory shall be planned to a full-stop of vehicle v_i , which is achieved by applying (3.22) in the local MPC objectives (3.13a). The trajectories are re-computed in each sampling time step in a receding horizon fashion, and only the first sample will be tracked by the respective vehicle. Thus, the goal is to move the planning-to-full-stop phase as far back in the prediction horizon as possible, and therefore achieve an actual driving behavior which is not affected by the planning-to-full-stop phase.

In what follows, we find the latest possible k_{brake} , i.e.,

$$k_{brake} = \max K, \quad (3.23)$$

with

$$K = \left\{ \tilde{k} \mid \exists u_i(k|t) = u_i^*(k|t), k \in \mathbb{I}_{0:\tilde{k}-1}, \wedge \right. \\ \left. u_i(k|t) \in \mathbb{U}_i, k \in \mathbb{I}_{\tilde{k}:M-1}, \text{ such that (3.13b) - (3.13g), with } \tilde{k} \in \mathbb{I}_{1:M-1} \right\}.$$

Here, $u_i^*(k|t)$ results from the solution of problem (3.13a) - (3.13c).

Let the backward reach-set

$$\mathbb{X}_i^A(t_i^r) = \left\{ x_i(t_i^r|t) \mid \exists z_i(k|t), k \in \mathbb{I}_{r:M}, \text{ such that (3.13b) - (3.13g)} \right\} \quad (3.24)$$

be the set of admissible states at time step t_i^r for which a trajectory exists that fulfills the problem constraints. Now, we apply the following method to find k_{brake} :

Algorithm 3 Brake point k_{brake}

- 1: Compute the desired trajectory z_i^{des} , with respective state values x_i^{des} , by solving (3.13) without (3.13f) and (3.13g). Set $t_i^r \leftarrow M-1$.
 - 2: Compute the backward reach-set $\mathbb{X}_i^A(t_i^r)$. If $x_i^{des}(t_i^r|t) \in \mathbb{X}_i^A(t_i^r)$, then $k_{brake} = t_i^r$ and continue with Step 3, else $t_i^r \leftarrow t_i^r - 1$ and repeat Step 2.
 - 3: Compute (3.13) using (3.22) and including (3.13f) as well as (3.13g). Apply the result in the Jacobi negotiation scheme of Algorithm 2.
-

Figure 3.4 illustrates the process in Algorithm 3 for an exemplary 2-state system.

An essential point is the computation of $\mathbb{X}_i^A(t_i^r)$ in Step 2 of Algorithm 3. To solve this, we apply the set-projection-algorithm presented in [KG87]. The resulting admissible set $\mathbb{X}_i^A(t_i^r)$ is a polyhedral of the form $\mathbb{X}_i^A = \{x_i(t_i^r|t) | E x_i(t_i^r|t) + \xi \leq 0\}$, $E \in \mathbb{R}^{n_{iq} \times 2}$ and $\xi \in \mathbb{R}^{n_{iq}}$, as we assume an LTI model with linear constraints. Most parts of the set-projection-algorithm are computationally simple. However, the resulting matrix E , describing the set \mathbb{X}_i^A with n_{iq} inequality constraints is a redundant representation, i.e., there exists a $\mathbb{X}_i^{A'} = \{x_i(t_i^r|t) | E' x_i(t_i^r|t) + \xi' \leq 0\}$, $E' \in \mathbb{R}^{n_{iq}' \times 2}$ and $\xi' \in \mathbb{R}^{n_{iq}'}$, with $n_{iq}' < n_{iq}$, such that $\mathbb{X}_i^A = \mathbb{X}_i^{A'}$. To keep the computational burden low during the recursive determination of $\mathbb{X}_i^A(t_i^r)$, it is required to find a $\mathbb{X}_i^{A'}$ representing the minimal, or close-to-minimal representation. Efficient methods for that exist, as it is a well studied problem in the field of linear programming [PS10]. A possible method, which we apply, is presented in [BBR+87].

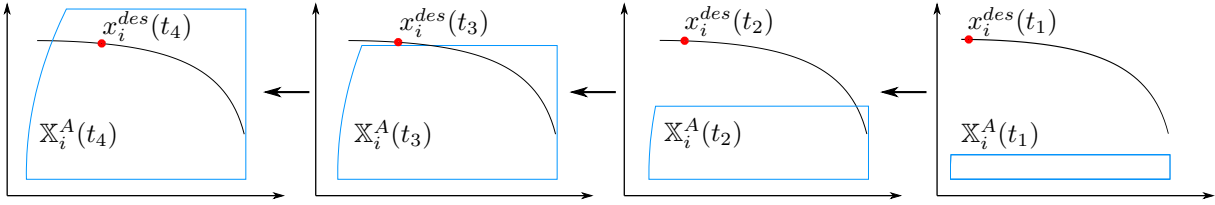


Figure 3.4: Example of a backward reach-set computation for a 2-state system. A reference trajectory is shown by the solid black line with respective states at time steps $t_1 - t_4$. t_4 is the first time step for which the the state lies inside the backward reachable set \mathbb{X}_1^A and thus a braking maneuver has to start at latest at $k_{brake} = t_4$.

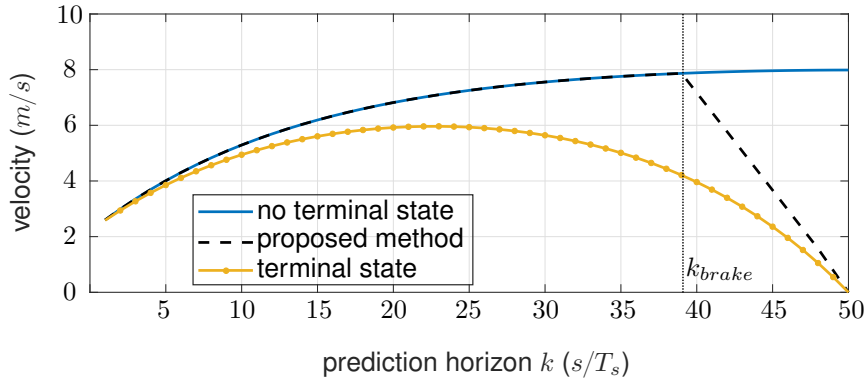


Figure 3.5: Velocity state of example trajectory comparing solutions with no terminal state, proposed method of terminal-state-less emulation, and terminal state without time-varying objective weights.

For the computation of k_{brake} , we assume that a vehicle's trajectory can be planned to full-stop during its horizon length, what results in a solution guarantee for Algorithm 3.

Assumption 3.1. *The horizon M of vehicle v_i 's MPC problem (3.13) is large enough to guarantee $\mathbb{X}_i \subseteq \mathbb{X}_i^A(1)$.*

Lemma 3.1. *Let the system be described by dynamics (3.3) and the applied MPC control problem by (3.13). Let Assumption 3.1 hold. Then, Algorithm 3 terminates for problem (3.13) with a solution for k_{brake} .*

Proof. Termination is guaranteed if Step 3 is reached. Step 2 is always reached after Step 1 and it continues to Step 3 if state $x_i^{des}(r|t) \in \mathbb{X}_i^A(r)$, which is at the latest ensured for $r = 1$ due to Assumption 3.1. \square

Figure 3.5 exemplary illustrates the difference between planning methods by plotting the predicted velocity state of a vehicle v_i . The solid line represents a solution without the

terminal condition of coming to full-stop at the end of the prediction horizon. This would be the case in the centralized solution (3.6) extracted for a single vehicle. The method proposed in Algorithm 3 and plotted with the dashed line, shows a similar behavior to the terminal-state-less solution until k_{brake} . When using a terminal state without the emulation procedure, however, the resulting trajectory shows a significant divergence from the respective solution without terminal constraint (solid-dotted line).

3.3.4 Algorithmic Properties

First, we provide algorithmic properties for Algorithm 2 with respect to Case **c1** where vehicles arrive at a conflict zone \mathcal{CZ}_k from the same in-lane and also leave it on the same out-lane. We will, thereafter, generalize the results to Cases **c2–c4** in Subsection 3.3.5.

Feasibility The following paragraph shows that each iteration in Algorithm 2 results in a recursive feasible solution for local vehicle problems (3.13).

Lemma 3.2. *Let Case **c1** (Section 3.1) hold. Let the system be described by dynamics (3.3) and the applied MPC optimization problem by (3.16). Given a feasible initial solution $z_i^{(0)}$ and feasible candidates $z_j^{(l)}$, the iterations (3.18) are locally feasible for vehicle v_i .*

Proof. To prove this lemma, it is utilized that a convex combination of two vectors in a convex set remains within this set. As (3.16) is a QP problem, there exists a unique solution. $z_j^{(l)}$, $l \geq 0$ are feasible for the problem (by assumption) and thus $z_i^*(z_j^{(l)})$ is a feasible (and optimal) solution. Sets \mathbb{X}_i and \mathbb{U}_i define polyhedral constraints with convex subsets $\mathbb{X}^M \subset \mathbb{X}_i$ and $\mathbb{U}^M \subset \mathbb{U}_i$. Consequently, (3.16b) and (3.16c) are linear constraint sets which are also convex in z_j . A convex combination of $z_i^*(z_j^{(0)})$ and $z_i^{(0)}$, conducted in (3.18), remains in the set defined by (3.16b) and (3.16c) and is thus again a feasible solution to (3.16). The feasibility guarantee for $l > 0$ follows by induction. \square

In Lemma 3.2 we showed that the inter-sampling iterations of the DJOR algorithm lead to feasible solutions for local problems v_i . Next, we derive a guarantee that the solutions are also feasible for neighboring vehicles v_j .

Lemma 3.3. *Let Case **c1** (Section 3.1) hold. Let the system be described by dynamics (3.3) and the applied MPC optimization problem by (3.16). Given feasible candidates $z_i^{(l)}$ and $z_j^{(l)}$ at iteration step l , the solution of (3.18) is feasible for neighbor Vehicles $j \in \mathcal{N}_i$.*

Proof. This proof makes use of the fact that decisions from neighboring subsystems (v_i and v_j) are coupled through the global property $\omega_i + \omega_j = 1$. By assumption, we are given globally feasible solution vectors $z_i^{(l)}$, $i \in \mathbb{I}_{1:N_v}$. Based on these solutions, suppose all subsystems conducted a local optimization and compute the combination in (3.18). Select

any vehicle v_i and any coupled constraint in (3.16c) for $v_j \in \mathcal{N}_i$, then it holds that

$$\begin{aligned} \begin{pmatrix} z_i^{(l+1)} \\ z_j^{(l+1)} \end{pmatrix} &= \begin{pmatrix} \omega_i z_i^* + (1 - \omega_i) z_i^{(l)} \\ \omega_j z_j^* + (1 - \omega_j) z_j^{(l)} \end{pmatrix} \\ &= \begin{pmatrix} \omega_i z_i^* + \omega_j z_i^{(l)} \\ \omega_j z_j^* + \omega_i z_j^{(l)} \end{pmatrix} \\ &= \omega_i \begin{pmatrix} z_i^* \\ z_j^{(l)} \end{pmatrix} + \omega_j \begin{pmatrix} z_i^{(l)} \\ z_j^* \end{pmatrix}. \end{aligned}$$

Both solution vectors in the last line are feasible with respect to the coupled constraint. This is because in the first vector z_i^* is feasible as argued in Lemma 3.2 and $z_j^{(l)}$ by assumption, and similar for the second vector. Due to linearity of the constraints with respect to z_j and z_i , the convex combination with $\omega_i + \omega_j = 1$ is also a feasible solution for (3.16c). This completes the proof. \square

Algorithm 2 iterates in between two sampling time steps. At the beginning of each iteration it requires initial candidates which are feasible solutions for the local optimization problems (3.16). Therefore, the following lemma proves feasibility for a time step transition $t + 1$ by using trajectory candidates (3.21).

Lemma 3.4. *Let Case **c1** (Section 3.1) hold. Let the system be described by dynamics (3.3) and the applied MPC optimization problem by (3.16). The trajectory candidate $\hat{z}_i(\cdot | t + 1)$ from (3.21) is feasible for (3.16) at iteration $l = 0$ and time step $t + 1$.*

Proof. The idea of this proof is that at time t all vehicles plan to reach a full-stop. Thus, in time step $t + 1$ the plan (prediction trajectory) to remain at full-stop will be a feasible candidate solution.

The last iteration of time step t , $z_i^{(l)}(k|t)$ with $k \in \mathbb{I}_{t:t+M}$ was feasible according to Lemma 3.2 and Lemma 3.3. Thus, $z_i^{(l)}(k|t)$ with $k \in \mathbb{I}_{t+1:t+M}$ will be feasible at time step $t + 1$ as model uncertainty is neglected. It holds that $z_i(t + M|t) \in \mathbb{X}^M \times \mathbb{U}^M$. Furthermore, the final element of $\hat{z}_i(\cdot | t + 1)$, $(x_i'^T, u_i'^T) \in \mathbb{X}^M \times \mathbb{U}^M$, extends the stand-still condition. This concludes that (3.21) is feasible for (3.16). \square

For Case **c1** we make the following assumption on the inter-vehicle distance to simplify presentation:

Assumption 3.2. *If vehicle v_j is driving on the same lane behind vehicle v_i , they are traveling with an inter-vehicle distance $d_{inter}(k|t) > d_{j,s} + d_i^b - d_i^a$, $k \in \mathbb{I}_{t:t+M}$.*

Definition 3.1. *An MPC problem computed with (3.13) is recursive feasible if*

$$\begin{aligned} x_i(k|t) \in \mathbb{X}_i, \quad k \in \mathbb{I}_{t:t+M} \wedge x_i(t+M|t) \in \mathbb{X}^M \wedge u_i(k|t) \in \mathbb{U}_i, \quad k \in \mathbb{I}_{t:t+M-1} \\ \Rightarrow \exists u_i(k|t+1) \in \mathbb{U}_i, \quad k \in \mathbb{I}_{t+1:t+M} \text{ such that} \\ x_i(k|t+1) \in \mathbb{X}_i, \quad k \in \mathbb{I}_{t+1:t+M+1} \wedge x_i(t+1+M|t+1) \in \mathbb{X}^M. \end{aligned}$$

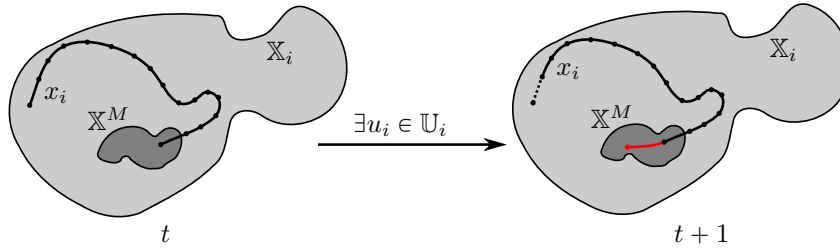


Figure 3.6: Recursive feasibility. There exists a control input u_i that keeps the trajectory inside the state constraint set \mathbb{X}_i and the terminal state inside the terminal set \mathbb{X}^M at the consecutive time step.

Figure 3.6 illustrates the Definition 3.1. Note that the trajectory can have a varying shape as long as the state constraints \mathbb{X}_i and the terminal condition $x_i(t+1+M|t+1) \in \mathbb{X}^M$ are fulfilled.

Theorem 3.1. *Let Case c1 (Section 3.1) hold. Let the system be described by dynamics (3.3) and the applied MPC optimization problems by (3.16). Let Assumption 3.2 hold. Then, each iteration of Algorithm 2 leads to a recursive feasible solution, according to Definition 3.1, for all vehicles v_i in \mathcal{G}_{com} .*

Proof. This proof argues that combining findings from Lemmas 3.2–3.4 results in recursive feasibility.

From Assumption 3.2 we know that there exist feasible solutions $z_i^{(0)}$ and $z_j^{(0)}$ for $v_i, v_j \in \mathcal{V}$. Following inter-sampling iterations remain feasible for local problems according to Lemma 3.2 together with Lemma 3.3. Moreover, following the reasoning in Lemma 3.4, feasible trajectories exist for time step transitions $t \rightarrow t+1$. Consequently, Definition 3.1 is fulfilled for all vehicles $v_i \in \mathcal{V}$ with problems (3.16) interacting according to Algorithm 2. \square

Scalability The standard JOR algorithm requires a centralized update step (3.19). Even though this is computationally not demanding, it is required to gather all trajectories at a central point [SVR+10; DDKDS17]. Moreover, with condition $\sum_{i=1}^{N_v} \omega_i = 1$ the speed of convergence depends on the number of vehicles in network \mathcal{G}_{com} .

Applying (3.13) with Algorithm 2 enables a scalable and fully distributed computation, as guaranteed by the following theorem.

Theorem 3.2. *Let Case c1 (Section 3.1) hold. Let the system be described by dynamics (3.3), the MPC optimization problem by (3.13), and the control law be applied according Algorithm 2. Then, $\omega_i = 0.5, \forall v_i \in \mathcal{V}$ is a valid choice for the step size in (3.18) independent of \mathcal{G}_{com} .*

Proof. To prove this theorem it is utilized that a distance constraint is shared at most by two vehicles.

By construction of the coordination problem, it holds for the central inter-vehicle distance constraints $\mathcal{A}_m^d = (\alpha_{ij}) \in \{-1, 0, 1\}^{p \times q}$ and thus

$$\|\mathcal{A}_m^d\|_\infty = \max_{i \in \mathbb{1}_{1:p}} \sum_{j \in \mathbb{1}_{1:q}} |\alpha_{ij}| = 2,$$

for matrix dimensions p and q . Consequently, each constraint, i.e., each row in (3.17c), is shared between a vehicle $v_i \in \mathcal{V}$ and at most one other vehicle $v_j \in \mathcal{N}_i$ in the distributed setup. For such a setup, we know from Lemma 3.3 that the DJOR update (3.18) between vehicles v_i and v_j is feasible. For any neighbor permutation $v_i, v_j \in \mathcal{V}$ and $v_i \neq v_j$ (where indices may mutually vary), $\omega_i = \omega_j = 0.5$ thus fulfills the convexity condition $\omega_i + \omega_j = 1$. \square

Remark 3.2 (Inter-vehicle relation). Knowing that vehicle v_i is a predecessor of vehicle v_j and at the same time vehicle v_j is a successor of vehicle v_i for all pairs $(v_i, v_j) \in \mathcal{E}_{com}$ and all prediction time steps k , enables the relaxing of condition $\omega_i = \omega_j = 0.5$ to the more general condition $\omega_i + \omega_j = 1, \omega_i > 0, \omega_j > 0$. Compare the illustration in Figure 3.3.

Convergence Convergence for the overall system is guaranteed by the fact that local cost functions are decoupled as well as the following proposition.

Proposition 3.1. *Let the central coordination problem be described by (3.17) and assume a given vehicle sequence decision m . Let the central problem be distributed into local optimization problems (3.13) with local dynamics (3.3), and let the distributed system be applied according to Algorithm 2. Then, the DJOR iterations (3.18) converge for $l \rightarrow \infty$.*

Proof. Convergence is guaranteed since the cost functions are monotonically decreasing between optimization steps and locally decoupled costs.

First, we show monotonicity between two iteration steps of individual decoupled vehicle costs:

$$\begin{aligned} V_i(z_i^{(l+1)}) &= V_i(\omega_i z_i^* + (1 - \omega_i)z_i^{(l)}) \\ &\leq \omega_i V_i(z_i^*) + (1 - \omega_i)V_i(z_i^{(l)}) \\ &\leq \omega_i V_i(z_i^{(l)}) + (1 - \omega_i)V_i(z_i^{(l)}) \\ &= V_i(z_i^{(l)}). \end{aligned}$$

This holds for all vehicles $v_i \in \mathcal{V}$. The first line applies (3.18), the second line follows from convexity of the cost functions, and the third line follows from optimality of local problems (3.16). As all individual quadratic cost functions V_i are bounded below, the overall system cost $V = \sum_{i=1}^{N_v} V_i$ in (3.17a) is also bounded, and thus convergence can be guaranteed system-wide as $l \rightarrow \infty$. \square

3.3.5 General Fulfillment of Coordination Conditions

This subsection discusses the extension of the feasibility results from Algorithm 2 to the coordination Cases **c2-c4**. In the first step, we present an approximate solution to compute the crossing time steps.

Crossing time approximation

To avoid the computation of a multi-level optimization problem, we utilize the receding horizon nature of MPC. This enables an approximation of the exit time defined in (3.1) by referring to the trajectory computed at the previous time step $t - 1$ such that

$$\tilde{t}_i^b := \begin{cases} \infty, & \text{if } d_i^b(t + M|t - 1) + L_{i,x} > 0 \\ \operatorname{argmin}_{d_i^b(k|t-1) \leq -L_{i,x}} d_i^b(k|t - 1), & \text{else.} \end{cases} \quad (3.25)$$

In the following, we substitute $t_i^b = \tilde{t}_i^b$ with its approximation.

Remark 3.3 (Crossing time computation). The exit time, i.e., the time a vehicle predicts to leave the respective conflict zone is approximated in (3.25) using the MPC trajectory computed in the previous time step. The approximation error can be expected to be negligible given that the MPC optimization is recomputed in each sampling time step with updated initial conditions and assuming that only small changes occur between two sampling steps. The latter assumption is also supported by relatively slow vehicle dynamics, which suppress fast changes for a small sampling time step. This simplification is an important contribution to fulfill the Scalability requirement (Chapter 1).

Alternatively, [ZMC16] proposes to compute analytic solutions to vehicles' exit times of intersection areas using Hamiltonian functions. Authors of [HZGF16] compute occupancy time slots using constraint optimal control problems. This means crossing times are solved with a separate optimization problem (separated from the original multi-level optimization problem).

Case distinction

For Case **c2**, where vehicles v_i and v_j approach from the same lane (vehicle v_i driving in front of vehicle v_j) but leave \mathcal{CZ}_k on different lanes, Theorem 3.1 is still valid, as the initial conditions do not change. The difference is solely that constraints (3.13i) and (3.13j) are not formulated for the complete prediction horizon, but only from time step $t + 1$ until time step t_i^b .

In Case **c3** vehicles approach from different lanes and merge onto the same lane in \mathcal{CZ}_k . Thus, coupling constraints are not required until the time of merging, i.e., only from t_i^b until $t + M$ (assuming vehicle v_i crosses before vehicle v_j). Constraint (3.13h) ensures consistency of the vehicle order by guaranteeing that a vehicle remains in front of the conflict zone \mathcal{CZ}_k until its predecessor has predicted to have crossed it, in what we assume to be a feasible solution:

Assumption 3.3. $d_j^b(t_0 + M|t_0) \geq d_{j,s}$ is feasible for vehicle v_j , with t_0 being the time of starting the negotiation according to Algorithm 2 with its neighbor vehicles.

This enables the feasibility guarantee for Case **c3**.

Theorem 3.3. *Let Case **c3** (Section 3.1) hold. Let the system be described by dynamics (3.3), the MPC optimization problems by (3.13), and the control law be applied according to Algorithm 2. Let Lemma 3.4 and Theorem 3.1 be given, and let Approximation (3.25), as well as Assumption 3.3 hold. Then it holds that, a partial coupling along the horizon, $\mathbb{I}_{t_i^b:t+M}$, of constraints (3.13i)–(3.13j) results in feasible solutions for vehicles v_i and v_j .*

Proof. This proof argues that a coupling between two vehicles (potentially a partial coupling along the horizon) becomes only activated if a feasible solution with respect to the coupling constraints exists.

Let vehicle v_i cross \mathcal{CZ}_k before vehicle v_j . For the case $t_i^b = \infty$, i.e., the horizon of vehicle v_i does not yet predict to cross the conflict zone \mathcal{CZ}_k , the set $\mathbb{I}_{t_i^b:t+M} = \emptyset$ is empty and problems (3.13) of vehicles v_i and v_j are not coupled through (3.13i) and (3.13j). Without coupling, it follows feasibility for the local problem (3.13a)–(3.13g) of the first vehicle v_i . Given Assumption 3.3 we can also guarantee feasibility for vehicle v_j with problem (3.13a)–(3.13h).

If the prediction of vehicle v_i crosses \mathcal{CZ}_k at time step $t - 1$, i.e., $t_i^b \leq t + M$ according to (3.25), it follows with Lemma 3.4 that the same solution will be a feasible candidate at time step t . Before v_i and v_j become coupled, (3.13h) was feasible for vehicle v_j . Now, for v_i 's problem we know from (3.13i):

$$\underbrace{d_i^b(t_i^b) + L_{i,x}}_{\leq 0} \leq \underbrace{d_j^a(t_i^b) - d_{j,s}}_{\geq 0}, \quad (3.26)$$

where the left side of the inequality follows from the condition that vehicles become coupled and the right side from v_j 's constraint (3.13h). The same argumentation holds from v_j 's perspective referring to constraint (3.13j). The feasibility of activated coupling constraints for prediction steps $\mathbb{I}_{t_i^b:t+M}$ between vehicles v_i and v_j follows from the reasoning in Theorem 3.1.

The discussion for the scenario where vehicle v_j crosses \mathcal{CZ}_k before vehicle v_i , follows analogously. \square

Feasibility for Case **c4** follows directly from Assumption 3.3.

3.4 Uncertainty Handling using Exact Penalty Functions

In Section 3.3 we proposed a distributed negotiation algorithm for multi-vehicle coordination which guarantees a feasible, and thus safe solution after each iteration step. The algorithmic guarantees are provided in absence of model and environmental uncertainties. Long prediction horizons are desirable for increasing the performance of the coordination procedure as, e.g., the approximation (3.25) will become more accurate with longer horizons. However, predicting the environment of an autonomous vehicle with the required confidence will only be possible for shorter horizons. Reacting to uncertain events in the environment, such as a pedestrian appearing in a vehicle's sensor view, might require deviating from the plan with respect to the long horizon which has been agreed on with

other vehicles in the network. This can cause infeasibility in the network due to a possible violation of the coupling constraints (3.13i)–(3.13j). However, it does not mean that such a violation of the long horizon plan leads to an unsafe behavior, as vehicles travel with a safety distance $d_{i,s}$ between each other and they can still react locally to unforeseen events.

We propose to induce this local reaction behavior by relaxing the coupling constraints in the form of exact penalty functions. If possible, vehicles will fulfill the formulated inter-vehicle coupling constraints and conduct the long horizon plan negotiated with its neighboring vehicles. Whenever unforeseen events occur and no feasible agreement with the neighboring vehicles can be found, vehicles can violate the coupling constraints to conduct, for example, an emergency braking maneuver. Thus, a prioritization of safety over coordination is achieved. After such a local reaction the vehicles automatically recover from the coupling constraint violation and return to a network-wide feasible solution.

In the following, we introduce the concept of exact penalty functions and thereafter describe how they are integrated into the DJOR algorithm.

3.4.1 Exact Penalty Functions

First, we recast the local QP problem (3.16) in an optimization problem with soft constrained inter-vehicle distances:

$$z_i'^* = \underset{z_i}{\operatorname{argmin}} V_i(z_i) + \delta_i \left\| \sum_{j \in \mathcal{N}_i} (\mathcal{A}_{ij}^d z_i + \mathcal{C}_{ij}^d z_j - b_{ij}^d)^+ \right\|_1 \quad (3.27a)$$

$$\text{s.t.} \quad \mathcal{A}_i z_i - b_i \leq 0, \quad (3.27b)$$

with penalty weight $\delta_i \in \mathbb{R}$ and

$$\left[(\mathcal{A}_{ij}^d z_i + \mathcal{C}_{ij}^d z_j - b_{ij}^d)^+ \right]_n := \max \left([\mathcal{A}_{ij}^d z_i + \mathcal{C}_{ij}^d z_j - b_{ij}^d]_n, 0 \right), \quad (3.28)$$

where index n indicates the n -th element of vector $\mathcal{A}_{ij}^d z_i + \mathcal{C}_{ij}^d z_j - b_{ij}^d$. Let λ_i^* be the Lagrangian vector corresponding to the feasible and optimal solution z_i^* of (3.16).

Theorem 3.4. *If $\delta_i > \|\lambda_i^*\|_\infty$, then the minimizers z_i^* and $z_i'^*$ are identical.*

Proof. See [Fle87, Theorem 14.3.1] □

How to compute appropriate values of δ_i in an MPC setting is discussed in [KM00; Hov11].

The non-smoothness of the 1-norm in Problem (3.27) ensures its exactness with respect to the original Problem (3.16), but a non-smooth optimization problem cannot be computed using standard algorithms. Therefore, an equivalent problem is formulated using the slack variables vector ϵ_i to achieve a QP problem again which can be efficiently solved [OB94; SR99]:

$$z_i'^* = \underset{z_i, \epsilon_i}{\operatorname{argmin}} V_i(z_i) + \delta_i \|\epsilon_i\|_1 \quad (3.29a)$$

$$\text{s.t.} \quad \mathcal{A}_i z_i - b_i \leq 0 \quad (3.29b)$$

$$\mathcal{A}_{ij}^d z_i + \mathcal{C}_{ij}^d z_j - b_{ij}^d \leq [\epsilon_i]_l, \quad j \in \mathcal{N}_i, \quad l \in \mathbb{I}_{1:|\mathcal{N}_i|} \quad (3.29c)$$

$$-\epsilon_i \leq 0. \quad (3.29d)$$

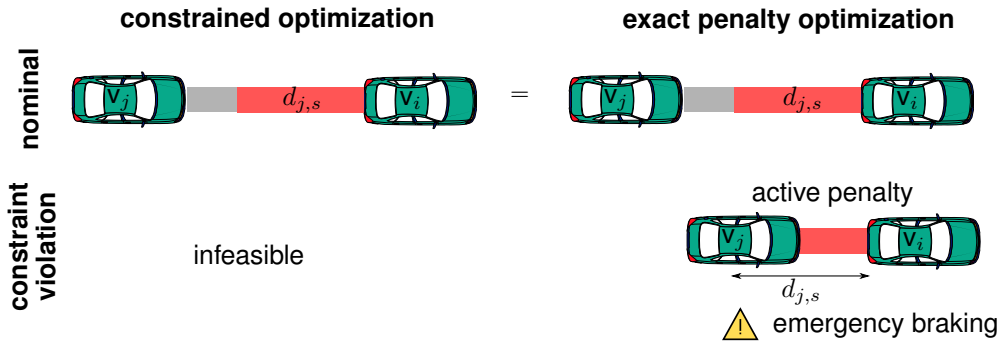


Figure 3.7: Exact penalty example. In an nominal scenario (constraints can be fulfilled) a constrained optimization delivers the same result an exact penalty optimization formulation. If a constraint is violated (here: inter-vehicle distance $d_{j,s}$) only the exact penalty method finds a solution.

Remark 3.4 (Exact penalty concept). Exact penalty functions are one form of constraint softening. The basic idea behind constraint softening is to move (hard) constraints of an optimization problem to its objective function, i.e., softening them. A high penalty weight on the softened constraints in the objective function shall ensure that the resulting optimized solution fulfills the original constraint formulations. The benefits of this procedure are that i) the resulting unconstrained optimization problem can be solved efficiently, for example, by analytic solutions, and ii) the optimization problem cannot become infeasible, e.g., due to numerical errors or system noise; thus, it always delivers a solution. Note that the latter is an important property for real-world implementations of optimization-based control systems.

Rather than achieving an increase in computational efficiency, the proposed constraint softening targets using exact penalties to avoid infeasibilities of the networked optimization problem due to changing environmental conditions. An essential benefit of the discussed exact penalty functions is that the result coincides with the respective hard constraint formulation if a solution exists. If no solution to the hard constraint problem exists, i.e., it is infeasible, the penalty softens the problem and results in a close-to-feasible solution. We utilized this property to compute control solutions that fulfill the desired inter-vehicle safety distance constraint $d_{j,s}$ in nominal scenarios (whenever possible). However, the problem also delivers a solution if the constraint (distance $d_{j,s}$) cannot be satisfied, e.g., in emergency braking maneuvers. Figure 3.7 illustrates the described setup. The standard constraint optimization problem loses feasibility if the inter-vehicle distance $d_{j,s}$ is violated, e.g., due to unexpected environmental changes. However, the exact penalty method results in a braking maneuver.

In summary, the method prioritizes the reaction to uncertain maneuvers over a nominal driving behavior, i.e., keeping the safety distance, while the latter is performed whenever possible. This addresses the Safety requirement (Chapter 1).

The idea is applicable to a wide range of scenarios, as, e.g., shown in [FBEG16], where inter-vehicle collision avoidance is prioritized over vehicle stabilization control through the application of exact penalty functions.

3.4.2 Integration into Jacobi Negotiation

If a solution to (3.16) exists, then this solution will be (exactly) found by (3.29). If no solution exists, (3.29) will compute the closest possible solution, i.e., the solution with the least possible constraint violation. This means it will attempt to keep the coupling distances to its neighbor vehicles, represented by (3.29c), whenever possible and violates it only if necessary due to, e.g., environmental uncertainties.

Given the responsibility of a vehicle to avoid a collision with its preceding vehicle and the fact that the predecessor's dynamics are not known locally by the following vehicle, a collision can be avoided if the inter-vehicle distance is larger than the required stopping distance. This means, we can compute the safety distance by finding the required stopping distance of vehicle v_i , such that it holds

$$d_{i,s} = \min_{z_i, t_{stop}} d_i(t_{stop}) \quad (3.30a)$$

$$\text{s.t. (3.13b), (3.13d), (3.13e)} \quad (3.30b)$$

$$t_{stop} \in \mathbb{I}_{0:M} \quad (3.30c)$$

$$v_i(\bar{t}) = 0 \quad \bar{t} \in \mathbb{I}_{t_{stop}:M} \quad (3.30d)$$

$$v_i(0) = v_i(t) \quad (3.30e)$$

$$d_i(0) = 0, \quad (3.30f)$$

where t_{stop} is the time step at which the vehicle comes to full stop, and we assume that the horizon is large enough, such that the optimization problem has a solution. In a simplified but more conservative way $d_{i,s}$ can be determined offline by considering a maximum velocity such that $v_i(0) = v_{max}$ and choosing z_i with $u_i(t) = \min_{u \in \mathbb{U}_i} u$, until the vehicle stops.

A potential constraint violation needs to be considered in the DJOR negotiation to guarantee collision avoidance between vehicles. A slack variable unequal to zero, $\epsilon_i > 0$, after the local optimization (Line 7 in Algorithm 2) indicates that the coupling constraints need to be violated. If this is the case, the negotiation step (3.18) has to be modified by choosing $\omega_i = 1$ to ensure the validity of the stopping distance $d_{i,s}$.

After an unexpected event, (3.29) will automatically recover by computing a solution which fulfills the coupling constraints again as soon as possible. Once $\epsilon_i = 0$ holds, the recovery is completed and the negotiation can be switched back to $\omega_i = 0.5$.

Remark 3.5 (Constraint softening). Constraint softening, such as exact penalty functions, is commonly applied to all constraints and used to avoid infeasibilities of the optimization problem triggered through model uncertainties or sensor noise. Here, however, we soften the coupling constraints solely to emphasize the methodology of avoiding coordination infeasibilities.

3.5 Numerical Illustration

Numerical simulations were conducted to illustrate the functionality of the methodologies introduced in this chapter. Methods of this chapter will be extended with sequence

Table 3.2: Simulation Parameters in Subsec. 3.5.1

Param.	Vehicle ID	
	1	2
$x_{i,v}^{ref}$	$7m/s$	$8.5m/s$
Q_i^n	$\text{diag}(0, 5)$	$\text{diag}(0, 5)$
R_i^n	1	1
U_i	$[-7m/s^2, 4m/s^2]$	$[-7m/s^2, 4m/s^2]$
$\mathbb{X}_{i,v}$	$[0m/s, 9m/s]$	$[0m/s, 9m/s]$
\mathbb{X}_i^M, U_i^M	(3.20)	(3.20)
$d_{i,s}$	$2m$	$2m$
ω_i	0.5	0.5
l_{max}	4	4

decisions in Chapter 4. Therefore, Section 4.4 discusses numerical simulations for the combined methodologies and compares the results with alternative approaches. The used PC contains an Intel Core i5 double core processor with $2.5GHz$ and $8GB$ RAM memory. Simulations were implemented with MATLAB and its *quadprog()* solver to compute the QP problems.

3.5.1 DJOR Evaluation

In the first step, we evaluate the influence of the prediction horizon length M on the coordination procedure. Therefore, two vehicles are simulated to cross a common \mathcal{CZ}_k in a scenario setup as shown in Figure 2.3 with local MPC laws (3.13) and the negotiation according to Algorithm 2.

Table 3.2 summarizes the applied simulation parameters for this example, where $x_{i,v}^{ref}$ is the state reference value referring to the velocity state, and similar for the constraint set $\mathbb{X}_{i,v}$. The sampling time is $T_s = 0.1s$.

Figure 3.8 illustrates the resulting vehicle distance state in the $d_1 - d_2$ configuration space for Cases **c3** and **c4** with horizon lengths $M = 30, 50, 100$ and vehicle v_1 crossing \mathcal{CZ}_k before vehicle v_2 . For each horizon length the distributed optimization results in a feasible and safe coordination. However, increasing M leads to a smoother coordination result, since the exit time t_i^b in (3.25) can be evaluated earlier than that with shorter horizons. This becomes visible through the dots in Figure 3.8 which show the terminal states for respective planning steps.

Next, we simulate a negotiation process of $N_v = 6$ vehicles stimulated by a changing reference value. Figure 3.9 illustrates the influence of the inter-sampling iterations in the same simulation setup as above by varying the parameter l_{max} . Given a reference change from $v_{i,ref} = 4m/s \rightarrow 9m/s$, the trajectory approaches the centralized solution (dashed line) with increasing inter-sampling iterations l_{max} .

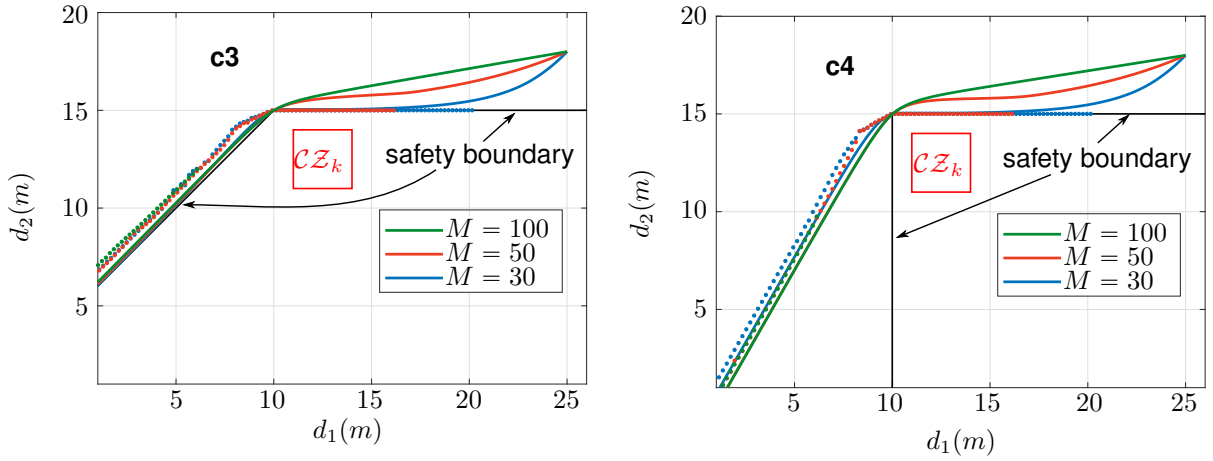


Figure 3.8: Trajectories for varying horizon lengths M in the $d_1 - d_2$ configuration space for vehicles v_1 and v_2 crossing \mathcal{CZ}_k in cases **c3** (left) and **c4** (right), with safety boundaries according to Figure 3.1. Vehicle v_1 crosses before vehicle v_2 . Solid lines are the actual distance states, and dots show the terminal states for each planning step.

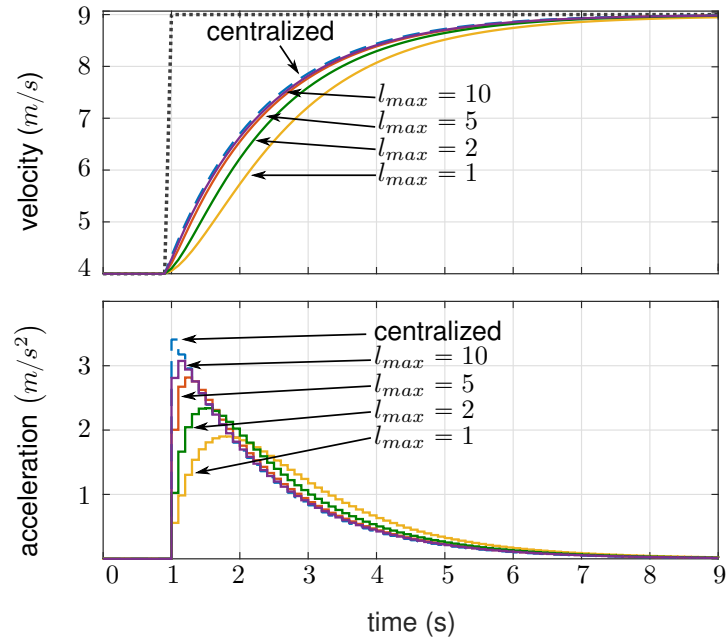


Figure 3.9: Varying number of inter-sampling iterations of the proposed emulation method compared to the centralized solution (dashed line).

Table 3.3: Simulation Parameters in Subsec. 3.5.2

Param.	Vehicle ID		
	1	2	3
M	50	50	50
$x_{i,v}^{ref}$	7m/s	8m/s	9m/s
Q_i^n	diag(0, 5)	diag(0, 10)	diag(0, 50)
R_i^n	10	10	10
\mathbb{U}_i	$[-7m/s^2, 4m/s^2]$	$[-7m/s^2, 4m/s^2]$	$[-5m/s^2, 4m/s^2]$
$\mathbb{X}_{i,v}$	$[0m/s, 10m/s]$	$[0m/s, 10m/s]$	$[0m/s, 10m/s]$
$\mathbb{X}_i^M, \mathbb{U}_i^M$	(3.20)	(3.20)	(3.20)
$d_{i,s}$	2m	2m	2m
ω_i	0.5	0.5	0.5
l_{max}	4	4	4
δ_i	4e3	4e3	4e3

Remark 3.6 (Optimality). In general, convergence to the centralized solution is not guaranteed for a dynamical-decoupled decomposition, as applied in this case. Due to the homogeneous vehicle models, however, the iterations approach to the centralized solution. We use this effect to illustrate the best possible solution for the distributed Jacobi negotiation algorithm.

3.5.2 Uncertainty Simulation

This subsection illustrates simulation results with exact penalty functions on the inter-vehicle coupling constraint.

Table 3.3 summarizes the applied simulation parameter, where $x_{i,v}^{ref}$ is the state reference value referring to the velocity state, and similar for the constraint set $\mathbb{X}_{i,v}$. The sampling time is $T_s = 0.1s$. We model three vehicles moving in a platoon with the order: $v_1 \rightarrow v_2 \rightarrow v_3$. At time $t = 5s$ vehicle v_2 conducts an emergency braking maneuver with $a_2 = a_{2,min} = -7m/s^2$, which is shown in the top middle plot of Figure 3.10. Before this maneuver, vehicles v_2 and v_3 drive with minimum inter-vehicle distance $d_{3,s} = 2m$ (lower plot of Figure 3.10). Lower braking capability of vehicle v_3 , i.e., $a_{3,min} = -5m/s^2$, would lead to infeasible solutions of the distributed optimization without softened constraints due to an inter-vehicle constraint violation. Figure 3.11 shows how the respective slack variable ϵ_3 becomes active during the braking phase and the automatic recovery after $t = 7.9s$. Once the distributed system has recovered into a feasible area, the negotiation continues as it would in the nominal case (with hard coupling constraints) due to the exactness of the penalty function.

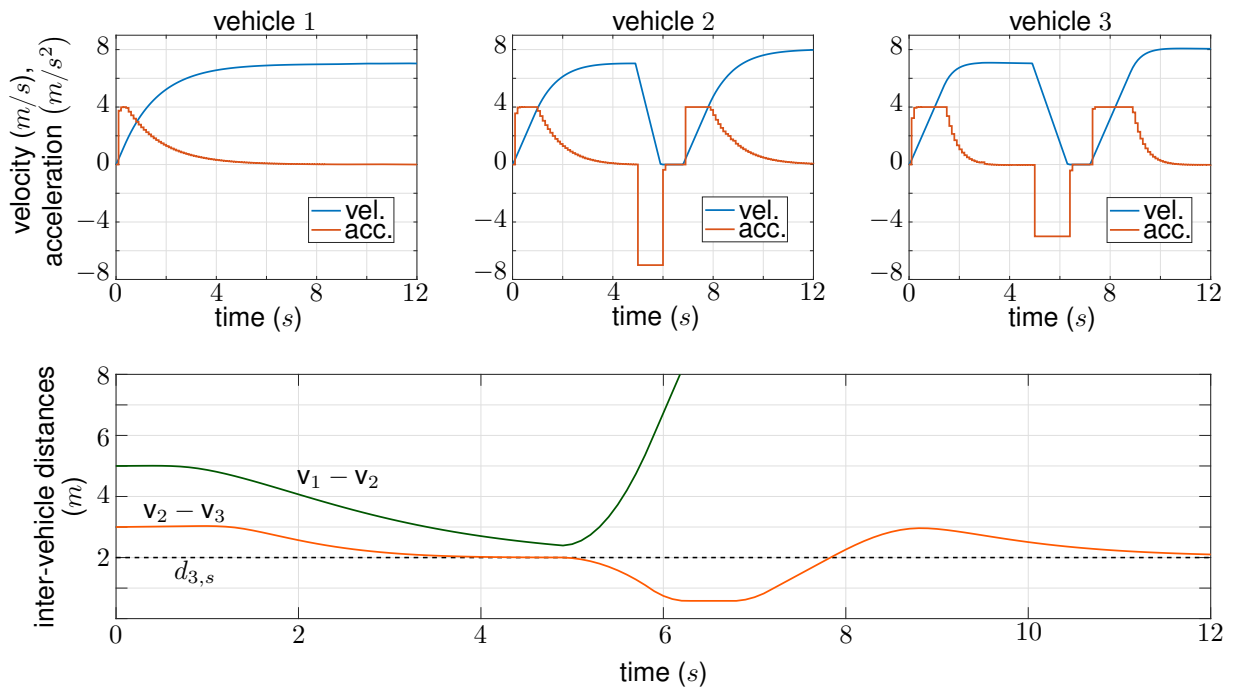


Figure 3.10: Emergency braking maneuver in distributed vehicle setup. Top row: vehicles' acceleration and velocity profiles. Bottom row: inter-vehicle distances with minimum distance $d_{3,s}$ between vehicles v_2 and v_3 .

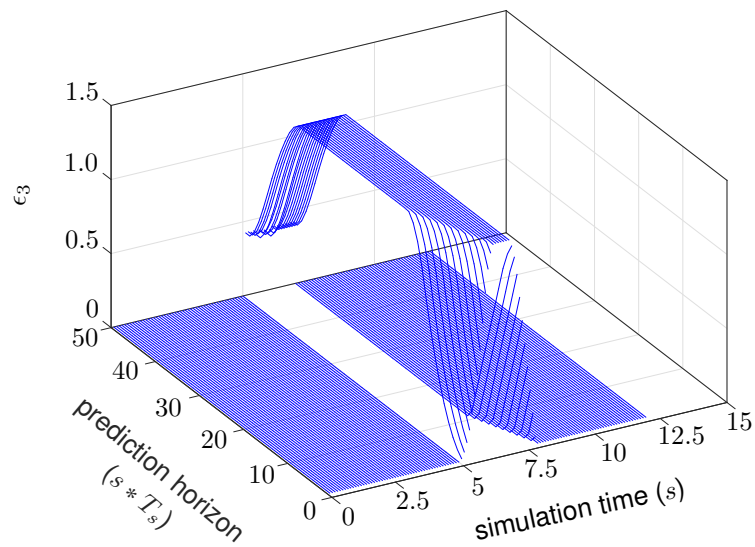


Figure 3.11: Slack variable ϵ_3 of constraint (3.13j) during emergency braking maneuver.

3.6 Summary and Discussion

This chapter proposes a trajectory computation methodology for a multi-vehicle setup, in which vehicles are capable of exchanging information between each other through V2X communication. With the proposed approach, conservative solutions can be avoided through iterative inter-vehicle negotiations. However, if communication resources become limited, a single exchange of trajectories with neighboring vehicles is sufficient for a solution that fulfills the optimization constraints network-wide what is consequently a safe trajectory solution. Thus, the approach exploits both the iterative nature to find close-to-optimal solutions whenever possible and any-time feasibility to cope with the Safety requirement.

The methodology uses a primal decomposition method based on Jacobi over-relaxations and ensures Scalability of the implementation. To enable a safe trajectory planning and privacy of vehicle models, each vehicle locally computes a model predictive control (MPC) law to determine its own trajectories. The resulting optimization solutions are shared with relevant vehicles in the coordination network. Moreover, the distributed optimization method is extended with the concept of exact penalty functions that enable considering unforeseen events in the coordination scenario while the above-discussed properties remain valid. In summary, this chapter addresses the requirements Safety, Efficiency, Privacy, Cooperation, and Scalability (introduced in Chapter 1).

For the baseline Jacobi over-relaxation (JOR) algorithm, it has been shown that it iterates to an optimal solution if no coupling state constraints are modeled [LJM19], or if coupling constraints remain inactive [DDKDS17]. In scenarios where coupling constraints become active, only a sub-optimal solution can be found. The degree of sub-optimality did not show a relevant impact for applications considered in this thesis. However, it might be a relevant criterion for other applications.

The advantage of iterative computation in this chapter is that the initially found solution can be improved towards its optimum. On the other hand, considering iterative solutions introduces challenges when implementing them and an additional load to the communication network. If these aspects are crucial, iterations free concepts are a valid alternative, for example, proposed for distributed MPC approaches in [KKJ17], and [KSSP19] in the case of multi-vehicle coordination scenarios.

Several approaches can consider safety by-design – in the sense of inter-vehicle collision avoidance. These include decentralized approaches where a brake-safe distance is sufficient to ensure safety and feasibility of the optimization problems [QGDLFM15]. Additionally, robust control methods, such as robust MPC [SSB16], can be used to account for worst-case actions of other vehicles. Distributed systems, in turn, require additional considerations. In this chapter, inter-vehicle safety (and feasibility of optimization problems) is ensured through an exact penalty formulation of coupling constraints. However, a frequent activation and deactivation of the exact penalty constraint in highly uncertain environments potentially leads to non-smooth driving behaviors. The derivation of a sufficiently large penalty weight generates further computational effort [KM00].

This chapter derives a trajectory computation strategy for a fixed inter-vehicle sequence graph. The vehicle sequence is a challenging decision for general multi-vehicle coordination problems. This challenge is addressed in the following chapter.

Vehicle Sequence Computation

In the previous chapter, a distributed trajectory computation strategy has been derived assuming a given crossing sequence of vehicles. The proposed method is based on distributed MPC optimization problems. However, a general multi-vehicle coordination problem includes among such dynamical optimization problems a combinatorial decision to determine the vehicle sequence at intersecting paths. This decision induces non-convexity into the optimal control problems and makes them prohibitively difficult to solve to optimality when real-time requirements are given. Therefore, it is desirable to decompose the overall problem rather than solving it in a centralized way. Considering requirements from Chapter 1 raises the need for a meaningful decomposition that takes properties such as Safety, Scalability, Efficiency, and Privacy into account.

A commonly used strategy to simplify computations is heuristic solutions, which approximate the optimal combinatorial decision [ZMC16; KK14]. However, heuristics that are too simple risk leading to an overall significantly sub-optimal solution to the intersection coordination problem, e.g., if the dynamic differences between vehicles are not considered. Alternatively, scheduling problems are candidates to determine sequences required in multi-vehicle coordination problems [LZ17; YDM11; WATEM12; BVDEG16]. Nevertheless, they rely on simplistic dynamics models if problems have to be computable on a fast time scale, which can be insufficient for highly dynamic tasks such as automated driving. Unlike the above-discussed approximation strategies, the overall coordination problem can be formulated as a mixed-integer program (MIP). This can be distributed using decomposition methods such as optimal branch-and-bound and dynamic programming (DP) [GK95], or genetic algorithms, simulated annealing, and greedy randomized search, which are heuristic methods [PPMR95]. These parallelized methods lack the capability to ensure the Privacy of vehicle models and fulfilling Safety requirements.

Counteracting the drawbacks, this chapter proposes a hierarchical decomposition approach aligned with the multi-vehicle coordination architecture in Figure 2.6. The central

coordination problem is decomposed such that the integer decision (vehicle sequence) can be computed in an upper hierarchical layer and the remaining dynamics optimization locally in the vehicles. This bridges safety assurance for local vehicle decisions with sufficiently accurate dynamics models (discussed in Chapter 3) and efficient sequence decisions using a scheduling method in the upper layer (discussed in this chapter).

The chapter is based on publications [KMK+20; KMEH20].

Outline

Section 4.1 presents how to detach the integer optimization problem from a central MIQP coordination problem. It is followed by a resource-constrained scheduling problem (RCPSP) setup derived on an intersection crossing example and its solution in Section 4.2. Section 4.3 relates the scheduling solution to the lower level distributed optimization problems for trajectory computation. Feasibility of the vehicle sequence scheduling decision for the multi-vehicle coordination problem is discussed, and an event-triggered re-computation law is presented. Section 4.4 illustrates the numerical simulation results, and a discussion in Section 4.5 concludes the chapter.

4.1 Integer-Continuous Variables Decomposition

Be reminded that we formulated the central and optimal solution to the multi-vehicle coordination problem as an MIQP of the form:

$$(z^*, m^*) = \underset{z, m}{\operatorname{argmin}} \sum_{i=1}^{N_v} V_i(z_i) \quad (4.1a)$$

$$\text{s.t. } \mathcal{A}z - b \leq 0 \quad (4.1b)$$

$$\mathcal{A}_m^d z - b_m^d \leq 0 \quad (4.1c)$$

$$m \in \mathbb{I}_{1:|\mathcal{T}|}, \quad (4.1d)$$

where the objective (4.1a) is the sum of local vehicle costs, (4.1b) contains vehicle dynamics constraints, (4.1c) inter-vehicle distance constraints, and (4.1d) is an integer decision representing the vehicle crossing order.

As previously discussed, solving (4.1) is in general not desirable. This is because the problem does not fulfill Scalability with respect to growing number of vehicles which makes it computationally infeasible, Privacy of vehicle information cannot be ensured, and Safety requirements might not be fulfilled through a central implementation, to name a few reasons.

In order to comply with requirements stated in Section 1.1, we propose a hierarchical decomposition of (4.1) such that the central problem is separated in a lower layer distributed QP optimization network and an upper layer integer optimization problem (compare Figure 2.6).

This results in solving distributed problems for a given vehicle order decision, i.e., a

pre-defined neighbor set \mathcal{N}_i :

$$z_i^* = \underset{z_i}{\operatorname{argmin}} V_i(z_i) \quad (4.2a)$$

$$\text{s.t. } \mathcal{A}_i z_i - b_i \leq 0 \quad (4.2b)$$

$$\mathcal{A}_{ij}^d z_i + \mathcal{C}_{ij}^d z_j - b_{ij}^d \leq 0, \quad j \in \mathcal{N}_i, \quad (4.2c)$$

which has been discussed in Chapter 3. In the upper layer it remains to determine the vehicle sequence through an integer program (IP) of the form:

$$m^* = \underset{m}{\operatorname{argmin}} \tilde{V}(\tilde{z}, m) \quad (4.3a)$$

$$\text{s.t. } \mathcal{T}_m \in \mathcal{T} \quad (4.3b)$$

$$m \in \mathbb{I}_{1:|\mathcal{T}|}, \quad (4.3c)$$

with an objective function $\tilde{V}(\tilde{z}, m)$ discussed in the paragraph below, and a tree \mathcal{T} defining possible crossing decisions. The tree \mathcal{T} consists of $|\mathcal{T}|$ paths from the root to leaf nodes such that each individual path \mathcal{T}_m represents a feasible crossing order through a conflict zone in a coordination scenario (see Remark 4.1 and Figure 4.1 for an example). Thus, a path is defined as an ordered set of tuples, $\mathcal{T}_m = ((i, j), (j, k), (k, l), \dots)$, $i, j, k, l \in \mathcal{V}$, where a tuple (i, j) describes that v_i crosses the respective conflict zone before v_j . Consequently, a decision m^* allows an unambiguous conclusion about the crossing order $o_k = (v_i, v_j, \dots)$, $v_i, v_j \in \mathcal{V}$ through a conflict zone \mathcal{CZ}_k and consequently the neighbor sets \mathcal{N}_i , $v_i \in \mathcal{V}$, can be determined.

Remark 4.1 (Decision tree concept). In this thesis, a decision tree defines a tree structure such that each path starting at the root represents a feasible decision of a vehicle sequence. The challenge in multi-vehicle coordination problems lies in the fact that not all vehicle sequence permutations result in feasible sequences. This is because the lane structure of a road network and the vehicle positions therein play a role in the sequence decision. Additionally, certain vehicle types, e.g., emergency vehicles, need to be prioritized over others. Figure 4.1 illustrates a simplistic example in which three vehicles v_1, v_2 , and v_3 cross a common conflict zone \mathcal{CZ} . All cases where v_3 crosses before v_2 are infeasible.

The decision space increases exponentially with growing number of vehicles and conflict zones. Therefore, to ensure Scalability, seamless methodologies to determine feasible sequences in a coordination scenario are required. Section 4.2 discusses such a methodology, which is based on scheduling theory.

The separation of continuous and integer variables in the optimization problem (4.1) can result in a loss of optimality. This is because the interdependent decision variables z and m are solved separately in (4.2) and (4.3) by assuming a given (fixed) decision of the respective other variable. However, the optimality gap is expected to remain small through a meaningful choice of the objective function $\tilde{V}(\tilde{z}, m)$ in (4.3) for the considered multi-vehicle coordination scenarios. Note that the objective $V_i(z_i)$ in (4.2) can be extracted directly from (4.1), but $\tilde{V}(\tilde{z}, m)$ needs to be designed to mimic the implicit

decision tree \mathcal{T} , which is a nontrivial problem itself.

4.2 Integer Decision using Scheduling

This section discusses a method to hierarchically decompose an IP as described in the previous section. We apply a scheduling method to determine a feasible integer decision based on dynamics approximation received from a distributed MPC implementation in the lower layer. The use of scheduling methods in this architecture comes with several benefits: i) scheduling theory is a well studied and understood problem in computer science, ii) constraints, which ensure feasibility of the integer decision, can be seamlessly considered, iii) there exist efficient solution strategies for scheduling problems.

4.2.1 Intersection Scenario

We derive the scheduling decision for multiple vehicles in an intersection crossing use case. This use case contains essential aspects of multi-vehicle coordination what makes its generalization straight forward. An important aspect is the fact that several vehicles argue about shared conflict zones, while rash decisions can lead to deadlock situations.

We model the intersection scenario consisting of a set of N_v connected automated vehicles (CAV) v_i , $i \in \mathbb{I}_{1:N_v}$, with local control units and an intersection management (IM) unit with which the CAVs exchange information via vehicle-to-infrastructure (V2I) communication. Figure 4.2 illustrates the introduced intersection setup. We distinguish between a scheduling area and an intersection area. Vehicles approaching the intersection enter the scheduling area in which, based on their local control computations, the IM determines a crossing order for the inner intersection area and shares this information with the vehicles. Assume that the vehicles cross the intersection along pre-defined paths with accurate tracking, where all possible paths are drawn in the figure. *Conflict zones*, \mathcal{CZ}_j , $j \in \mathbb{I}_{1:6}$, divide the intersection area into zones for potential rear-end, front and side collisions ($\mathcal{CZ}_1, \dots, \mathcal{CZ}_5$), and rear-end collisions on approaching lanes (\mathcal{CZ}_6). Note that, in general, the concept of conflict zones and the negotiation process presented in this thesis are applicable to arbitrary scenarios in which several vehicles share a common area and where collision conflicts can occur, e.g., obstacle avoidance scenarios.

Finally, we define a multi-graph $\mathcal{G}_{route} = (\mathcal{V}_{route}, \mathcal{E}_{route})$ describing the given vehicles' routes through the intersection. The set of vertices $\mathcal{V}_{route} = \{\mathcal{CZ}_1, \dots, \mathcal{CZ}_6\}$ contains all conflict zones and $\mathcal{V}_{route,i} \subset \mathcal{V}_{route}$ all conflict zones that vehicle v_i passes. Directed edges $e_{jk} \in \mathcal{E}_{route}$ are connections between consecutive conflict zones, which indicates that a vehicle crosses these zones and the driving direction, i.e. $e_{jk} = (\mathcal{CZ}_j \rightarrow \mathcal{CZ}_k)$, $j, k \in \mathbb{I}_{1:6}$.

4.2.2 Scheduling Problem Formulation

To formulate the scheduling problem, we first introduce the standard notation of a resource-constrained project scheduling problem (RCPSP) as classified in [BDM+99]. Following this, we map this notation to the intersection problem. Finally, we solve it by formulating an integer linear problem (ILP).

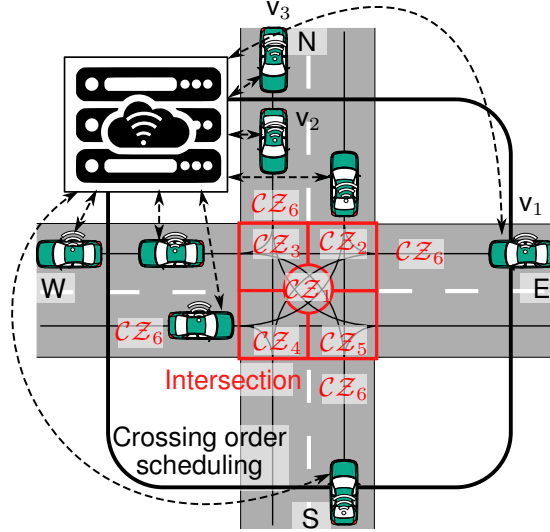


Figure 4.2: Intersection crossing scenario.

General RCPSP Definition

The RCPSP is defined by the tuple

$$(\Lambda, \eta, \Gamma, \Upsilon, \sigma, \Omega), \quad (4.5)$$

where $\Lambda = \{\alpha_0, \dots, \alpha_{n+1}\}$ is a set of activities and the sub-set $\Lambda' = \{\alpha_1, \dots, \alpha_n\} \subset \Lambda$ with n non-dummy activities; $\eta \in \mathbb{N}^{n+2}$ is a vector describing the duration of each activity and we set the dummy activities' duration to $[\eta]_0 = [\eta]_{n+1} = 0$; matrix $\Gamma \in \mathbb{N}^{l \times 2}$ contains $l \in \mathbb{N}$ precedence relations where each row in Γ with elements $(\alpha_i, \alpha_j) \in \Lambda'$, $i \neq j$, means that activity α_i precedes activity α_j ; $\Upsilon = \{\rho_1, \dots, \rho_m\}$ is the set of renewable resources with $m \in \mathbb{N}$; $\sigma \in \mathbb{N}^m$ is a vector describing the amount of available resources with the respective identifier; finally the matrix of demands is given by $\Omega = (\omega_{ir})^{(n+2) \times m}$, $\omega_{ir} \in \mathbb{N}$, with elements describing the amount of consumed resources $\rho_r \in \Upsilon$ for each activity $\alpha_i \in \Lambda$.

Formulation for Multi-vehicle Coordination

The mapping into the intersection framework is proposed as follows. Each non-dummy activity $\alpha_i \in \Lambda'$, $i \in \mathbb{I}_{1:n}$, indicates a vehicle's route through the scheduling and intersection zone, i.e.,

$$\alpha_i = (e_{jk}, \dots, e_{lm}), \text{ where } e_{jk}, e_{lm} \in \mathcal{E}_{route}. \quad (4.6)$$

We distinguish two types of activities. The first one, *drive to*, models the vehicle driving in the scheduling zone towards the beginning of the intersection. The second one, *cross*, passing through the intersection. For clarification, consider the following demonstrative example.

Example 4.1. Assume vehicle v_1 in Figure 4.2 driving from E to S , then its route is $CZ_6 \rightarrow CZ_2 \rightarrow CZ_1 \rightarrow CZ_4$. Vehicle v_1 's *drive to* activity corresponds to "driving in CZ_6 " and its *cross* activity corresponds to "driving through CZ_2 , CZ_1 , and CZ_4 ".

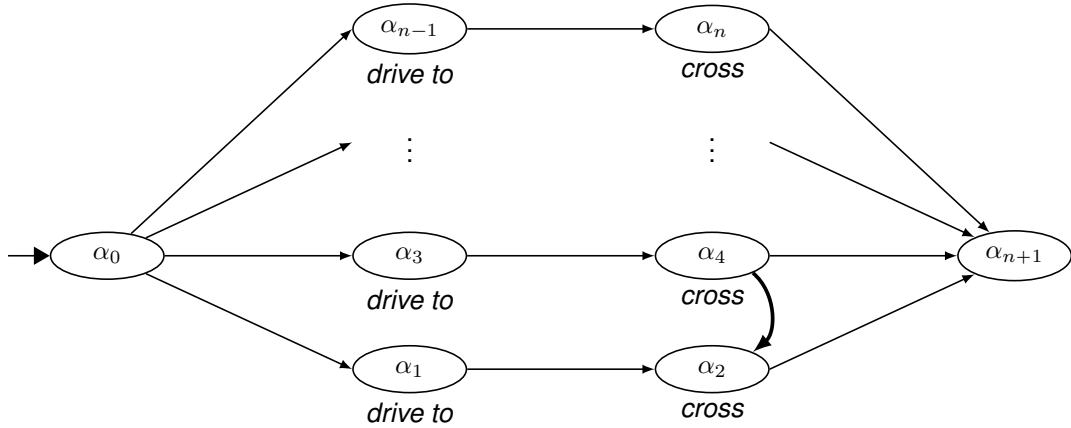


Figure 4.3: Precedence graph representing the vehicles' routes through the intersection by distinguishing the activity types *drive to* and *cross*, as well as a priori known inter-vehicle relations as illustrated between α_4 and α_2 .

Duration η_i of an activity $\alpha_i \in \Lambda'$ models the expected time a vehicle consumes to perform the respective activity, i.e., to drive to the intersection or to cross it. The activity duration vector η represents the interface to the local vehicle control problems. Here several candidates for suitable heuristics exist. We propose an MPC-related duration measure in order to achieve a close link to the local vehicles' control decisions. Therefore, let us define

$$\eta_i = f_j(\hat{Z}_j), \quad (4.7)$$

with activity α_i 's duration η_i given by vehicle v_j 's MPC optimization \hat{Z}_j (see Section 4.3). We introduce a precedence relation in Γ for the routes of a vehicle, i.e., the *drive to* activity precedes the *cross* activity. A precedence relation is also added if we a priori know a certain crossing order, e.g., if two vehicles approaching the intersection on the same lane and the first cannot be taken over by its follower (rear-end collision avoidance) such as illustrated in the following example.

Example 4.2. Assume vehicle v_2 in Figure 4.2 driving from N to S and vehicle v_3 from N to E . Let $\alpha_i = (\mathcal{CZ}_6 \rightarrow \mathcal{CZ}_3, \mathcal{CZ}_3 \rightarrow \mathcal{CZ}_4)$ be the intersection crossing activity of vehicle v_2 , and $\alpha_j = (\mathcal{CZ}_6 \rightarrow \mathcal{CZ}_3, \mathcal{CZ}_3 \rightarrow \mathcal{CZ}_1, \mathcal{CZ}_1 \rightarrow \mathcal{CZ}_5)$ of vehicle v_3 . Then a pair (α_i, α_j) in Γ ensures that v_2 enters the intersection before v_3 .

The resources represent the set of conflict zones in the scenario, i.e., $\Upsilon = \{\mathcal{CZ}_1, \dots, \mathcal{CZ}_6\}$ in our intersection setup. The availability of the respective resources at a certain time-instant is defined by $\sigma = (1, 1, 1, 1, 1, N_v)^\top$ and elements of the demand matrix Ω are

$$\omega_{ir} = \begin{cases} 1 & \text{if } \mathcal{CZ}_r \in \alpha_i \\ 0 & \text{else} \end{cases}. \quad (4.8)$$

Figure 4.3 illustrates the construction of a precedence graph according to the definition of Γ and Υ , while Table 4.1 summarizes the scheduling taxonomy related to the intersection model.

Table 4.1: Intersection scheduling taxonomy.

Scheduling meaning	param.	model param.	Intersection meaning
non-dummy activity	α_i	$= (e_{jk}, \dots, e_{lm}),$ $e_{jk}, e_{lm} \in \mathcal{E}_{route}$	route through intersection
duration	η_i	$= f_j(z_j)$	crossing duration prediction
precedence relations	$\epsilon_{k,:}$	$= (\alpha_i, \alpha_j)$	vehicles on same lane
resources	Υ	$= \{\mathcal{CZ}_1, \dots, \mathcal{CZ}_6\}$	conflict zones
availabilities	σ	$= (1, \dots, 1, N_v)^T$	# respective conflict zones
demands	ω_{ir}	$= \begin{cases} 1 & \text{if } \mathcal{CZ}_r \in \alpha_i \\ 0 & \text{else} \end{cases}$	passed \mathcal{CZ} s

4.2.3 Solution of the RCPSP

Problem (4.5) can be solved by formulating it as ILP. We suggest a modified version of the discrete-time ILP formulation introduced by [PWW69].

Let $\beta_{i,t'}$ be a binary decision variable with $\beta_{i,t'} = 1$ if activity i starts at time t' and $\beta_{i,t'} = 0$ otherwise, then (4.5) can be formulated as the following ILP:

$$\beta^* = \underset{\beta}{\operatorname{argmin}} \sum_{i \in \mathbb{I}_{1:|\Lambda|}} \sum_{t' \in \mathbb{I}_{0:M_{sched}}} t' \beta_{i,t'} \quad (4.9a)$$

s.t.

$$\sum_{t' \in \mathbb{I}_{0:M_{sched}}} t' \beta_{j,t'} \geq \sum_{t' \in \mathbb{I}_{0:M_{sched}}} t' \beta_{i,t'} + \eta_i \quad (i, j) \in \Gamma \quad (4.9b)$$

$$\sum_{i=1}^n \left(\omega_{ik} \sum_{m=t'-\eta_i+1}^{t'} \beta_{i,m} \right) \leq \Omega_k \quad t' \in \mathbb{I}_{0:M_{sched}}, k \in \Upsilon \quad (4.9c)$$

$$\sum_{t' \in \mathbb{I}_{0:M_{sched}}} \beta_{i,t'} = 1 \quad i \in \Lambda \quad (4.9d)$$

$$\beta_{i,t'} \in \{0, 1\} \quad i \in \Lambda, t' \in \mathbb{I}_{0:M_{sched}}, \quad (4.9e)$$

with scheduling horizon M_{sched} , scheduling sampling time

$$T_{sched} = n_{sched} T_s, \quad (4.10)$$

with $n_{sched} \in \mathbb{Z}^+ \setminus \{0\}$, such that it holds

$$M_{sched} T_{sched} > M T_s, \quad (4.11)$$

and the optimization variable $\beta = (\beta_{1,1}, \dots, \beta_{i,t'}, \dots, \beta_{n+2, T_{sched}})$, $i \in \mathbb{I}_{1:|\Lambda|}, t' \in \mathbb{I}_{0:M_{sched}}$. Note that (4.9a) is formulated such that it minimizes the problem's makespan as well as each activity's makespan, i.e., finds the solution with the shortest overall and individual vehicles' time consumption. This objective is chosen because the infrastructure's goal is to maximize the vehicle throughput in the intersection area.

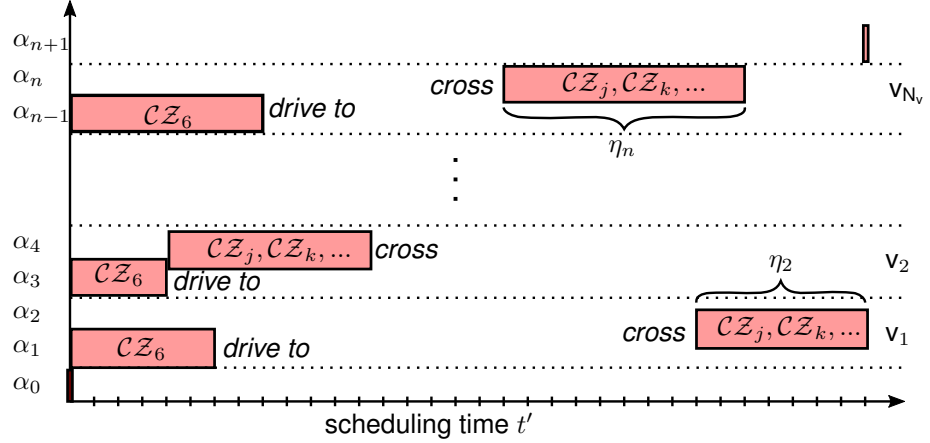


Figure 4.4: Exemplary scheduling result indicating time and duration of execution for each modeled activity, while for vehicles v_i activities *drive to* the intersection area and *cross* the intersection are distinguished. For each activity the consumed resources are illustrated with $j, k \in \{1, \dots, 5\}$.

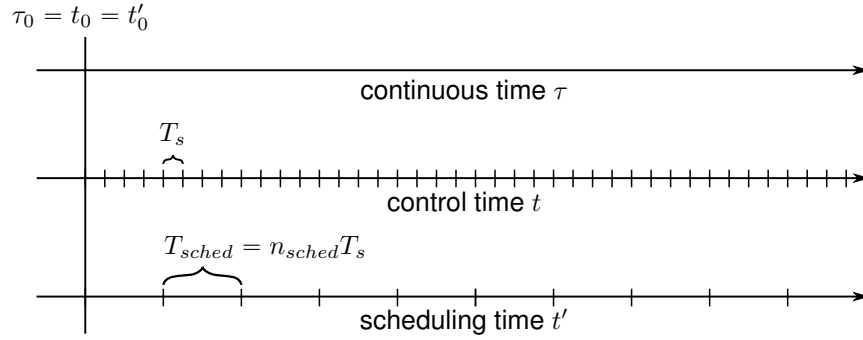


Figure 4.5: Time-scale relation between control and scheduling problems.

The scheduling result can be conveniently illustrated by a Gantt-chart as exemplary shown in Figure 4.4.

Given the result of (4.9) and the respective consumed resources of each activity $\alpha_j \in \Lambda'$, we are able to extract a crossing order o_i for each critical zone \mathcal{CZ}_i , $i \in \mathbb{I}_{1:5}$. This is achieved by neglecting the actual scheduling time t' and solely extract the order in which vehicles v_j , $j \in \mathbb{I}_{1:N_v}$ are scheduled to cross a certain conflict zone \mathcal{CZ}_i . Consequently, we achieve a set of orders

$$\mathcal{O} = \{o_i = (v_j, v_k, \dots) \mid i \in \mathbb{I}_{1:5}; j, k \in \mathbb{I}_{1:N_v}\}. \quad (4.12)$$

Note that (4.10) and (4.11) enable the allocation of different time-scales between the scheduling problem (4.9) and the control problem (3.13), as illustrated in Figure 4.5.

Remark 4.2 (Crossing order). Modeling the scheduling such that activities reserve the complete intersection area (*cross*) does not mean that in the end, only a single vehicle can be in the intersection, as we only extract the order decision and pass this information to the local control units, as discussed in the following section.

4.3 Scheduling-Control Interaction

In this section, we first discuss the distributed control law and its connection to the scheduling decision. Thereafter, we argue why the proposed RCPSP results in a deadlock free decision and thus enables a guaranteed feasible control negotiation.

4.3.1 Connection to the Distributed MPC Problem

Given the orders (4.12), computed with the scheduling law, the local vehicles can receive a set of neighbors. Based on this information the vehicle trajectories will be computed using distributed MPC laws with respective neighbor predictions. After formulating this distributed MPC setup we introduce how to predict the vehicles' activity duration (4.7).

The MPC laws, solved locally in each vehicle $v_i, i \in \mathbb{I}_{1:N_v}$, are given by the problem definition from Section 3.2.2:

$$Z_i^* = \underset{Z_i=(\bar{x}_i, \bar{u}_i)}{\operatorname{argmin}} V_i(x_i(t), u_i(t)) \quad (4.13a)$$

s.t.

$$x_i(k+1|t) = A_i x_i(k|t) + B_i u_i(k|t) \quad k \in \mathbb{I}_{t:t+M-1} \quad (4.13b)$$

$$x_i(t|t) = x_i(t) \quad (4.13c)$$

$$x_i(k|t) \in \mathbb{X}_i \quad k \in \mathbb{I}_{t+1:t+M} \quad (4.13d)$$

$$u_i(k|t) \in \mathbb{U}_i \quad k \in \mathbb{I}_{t:t+M-1} \quad (4.13e)$$

$$x_i(t+M|t) \in \mathbb{X}_i^M \quad (4.13f)$$

$$u_i(t+M-1|t) \in \mathbb{U}_i^M, \quad (4.13g)$$

$$d_i^a(k|t) - d_{i,s} \geq 0 \quad \mathbf{c3, c4}, j \in \mathcal{P}_i \quad (4.13h)$$

$$d_i^b(k|t) + L_{i,x} + d_{j,s} - d_j^a(k|t) \leq 0 \quad \mathbf{c1-c3}, j \in \mathcal{S}_i \quad (4.13i)$$

$$d_j^b(k|t) + L_{j,x} + d_{i,s} - d_i^a(k|t) \leq 0 \quad \mathbf{c1-c3}, j \in \mathcal{P}_i, \quad (4.13j)$$

Thereby, Z_i is the local optimization vector which defines the trajectory of vehicle v_i , (4.13b), (4.13d), and (4.13e) describe dynamics, state, and input constraints, respectively, (4.13c) is the initial condition, and the terminal constraints (4.13f) and (4.13g) contribute to the recursive feasibility guarantee of the distributed computations (see Section 3.3.4). For a given scheduling decision (4.12) we are able to formulate the distance constraints (4.13h) - (4.13j) distinguishing between cases **c1-c4** according to the vehicle configuration at a conflict area. Note that all feasible combinations $\mathcal{T}_m \in \mathcal{T}$ are replaced by these local constraints for a single combinatoric decision. This reduces the computational effort significantly because the integer decision vanishes. The set of predecessors,

$$\mathcal{P}_i = \{ v_j \mid j \in \mathbb{I}_{1:N_v} \wedge v_j \stackrel{ok}{>} v_i, \mathcal{CZ}_k \in \mathcal{V}_{route}^i \setminus \mathcal{CZ}_6 \}, \quad (4.14)$$

contains all vehicles crossing before vehicle v_i on its route through the intersection.

Similar, we achieve the set of successors,

$$\mathcal{S}_i = \{ v_j \mid j \in \mathbb{I}_{1:N_v} \wedge v_j \stackrel{ok}{<} v_i, \mathcal{CZ}_k \in \mathcal{V}_{route}^i \setminus \mathcal{CZ}_6 \}. \quad (4.15)$$

We find that local problems (4.13) are convex as (4.13a) is quadratic and (4.13b) - (4.13j) are linear constraints. These problems are thus QPs and can be solved efficiently.

Above, we described how the global scheduling decisions are incorporated in the local control problems. Now, it remains to discuss the reverse link between local control decisions and the scheduling problem. This link is represented by (4.7). Solving (3.13) is conducted in a distributed and iterative manner where neighbor intentions are shared in each iteration step as discussed in Chapter 3. During iterations in the procedure each vehicle computes a nominal trajectory \tilde{Z}_i , obtained by neglecting (4.13f) - (4.13j) from (4.13), and Z_i^* , i.e., the full problem (4.13) to provide the feasibility guarantee. In what follows we will refer to the states from the nominal optimization vector \tilde{Z}_i .

For a given vehicle v_i 's activity α_j , we extract $\mathcal{CZ}_s \in \mathcal{V}_{route}$ and $\mathcal{CZ}_e \in \mathcal{V}_{route}$ which are the first zone in the intersection of v_i 's route and the zone after leaving the intersection area, respectively. We estimate the duration of activity α_j by

$$\eta_j = \lfloor t_{end} - t_{start} \rfloor, \quad (4.16)$$

which is computed distinguishing the following cases:

$$t_{end} = \begin{cases} \min \left((t + M)T_s + \frac{\tilde{d}_i^{cz_e}(t+M|t)}{v_i(t+M|t)}, tT_s + M_{sched}T_{sched} \right) & \text{if } \tilde{d}_i^{cz_e}(t + M|t) \geq 0 \\ k_{end}T_s, & \text{if } \tilde{d}_i^{cz_e}(t + M|t) < 0, \end{cases} \quad (4.17)$$

with

$$\begin{aligned} k_{end} &= \underset{k}{\operatorname{argmin}} |\tilde{d}_i^{cz_e}(k|t)| \\ &\text{s.t. } k \in \mathbb{I}_{t:t+M}. \end{aligned} \quad (4.18)$$

Similarly, t_{start} is computed by substituting k_{end} with k_{start} and $\tilde{d}_i^{cz_e}$ with $\tilde{d}_i^{cz_s}$ in (4.17) and (4.18). Here, $\tilde{d}_i^{cz_e}$ refers to the distance state achieved through the computation of \tilde{Z}_i with the superscript cz_e describing the distance to the waypoint where the last zone of the intersection is left, and cz_s to the last waypoint before the intersection is entered (cf. Section 2.3)

Finally, Algorithm 4 gives a summary of the combined scheduling-control coordination procedure.

Algorithm 4 Combined Scheduling-Control Procedure

- 1: **Vehicles:**
 - 2: compute distributed MPC problems (4.13) without (4.13f) - (4.13j)
 - 3: **IM:**
 - 4: compute activity duration estimation (4.16)
 - 5: solve (4.9) and determine (4.12)
 - 6: **Vehicles:**
 - 7: negotiate distributed intersection crossing using (3.13) according to Algorithm 2
-

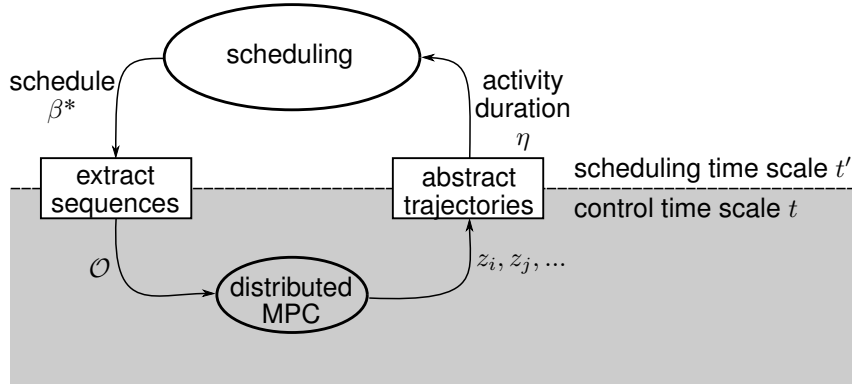


Figure 4.6: Control-scheduling interaction.

Remark 4.3 (Layer interaction concept). A key idea of the separation between the upper layer sequence decision (scheduling method) and the lower layer trajectory control (distributed MPC from Chapter 3) is separating their time-scales. The upper layer computes the scheduling problem on a coarse time-scale while the lower layer computes safety-related trajectory decisions using short sampling intervals. This improves the Scalability of the upper layer decision. The expected loss of optimality through the time abstraction is compensated because only the vehicle order is extracted and passed to the lower trajectory control layer from the scheduling solution. Consider, for example, an intersection crossing scenario. Then, switching the vehicle order is a rather “coarse” decision compared to fine changes of the acceleration profile to adjust the desired trajectory, e.g., to increase the inter-vehicle distance at the intersection. Figure 4.6 illustrates that the received vehicle trajectories are mapped onto the coarse scheduling time-scale. After computing the scheduling solution, the timed information is discarded to extract the sequence information solely.

Similarly, authors of [SDJ16] apply a hierarchical decomposition to a power system optimal control problem, where a binary integer decision, which determines the generator on-off status, can be solved on a different (slower) time scale as the lower layer (fast) generator dynamics.

4.3.2 Feasibility of the Integer Decision

This section discusses the deadlock free scheduling solution using an example and how it relates to dynamic feasibility of local control decisions.

Assume two vehicles, v_1 and v_2 , conduct a left turn in an intersection, described in Section 4.2.1, from opposite directions. Then, in the $d_1(t)$ - $d_2(t)$ -configuration-space there are unfeasible areas due to commonly passed \mathcal{CZ}_i s, as illustrated by the red boxes in Figure 4.7. As the *cross* activity of the RCPS (4.5) groups all passed \mathcal{CZ} s of a vehicle, a consistent order solution is computed. Compare the blue shaded area in the left plot of Figure 4.7, which indicates the possible $d_1(t)$ - $d_2(t)$ -trajectory space if v_1 has to cross before v_2 . This space is a connected set and thus there exists a homotopy class of $d_1(t)$ - $d_2(t)$ -trajectories [GBDLF14]. On the contrary, the right plot in Figure 4.7 shows a

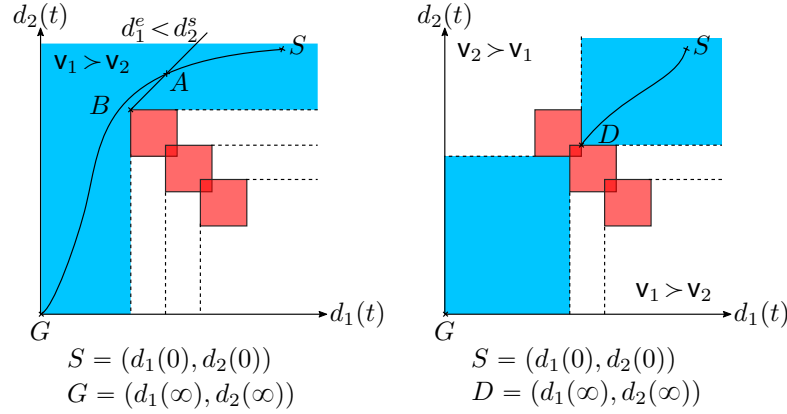


Figure 4.7: Feasible trajectory space (blue shaded area) in the d_1 - d_2 -configuration-space for two vehicles on a left turn from opposite directions. Left: consistent scheduling decision with $v_1 > v_2$ for all commonly passed CZ s; Right: non-consistent scheduling decision ($v_1 > v_2$ and $v_1 < v_2$) leading to a deadlock situation.

non-consistent order decision where the trajectory ends in deadlock configuration D and thus the goal configuration G cannot be reached.

In summary, ensuring the existence of a connected homotopy class in the trajectory configuration space results in the existence of a dead-lock free solution of the coordination problem.

In general, Theorem 3.1 guarantees feasibility of local problems which ensure feasible dynamics decisions through the formulation of constraints. To be valid, this theorem requires that i) Assumption 3.2 is fulfilled, which specifies the inter-vehicle distance for vehicles traveling on the same lane, and ii) that a vehicle which joins the negotiation process has a distance to the respective conflict zone greater than its braking distance, i.e., $d_i^a \geq d_{i,s}$. By implication, a sequence decision for vehicle v_i can be changed as long as the previous inequality is met. If the inequality is violated, the previously found sequence needs to be fixed for the following sequence optimization problem (4.9), which can be ensured through the formulation of additional precedence constraints $\gamma \in \Gamma$. Thus, this procedure leads to sequence decisions for which a dynamical feasible solutions, in the context of the distributed problems (4.13), exist.

4.3.3 Event-triggered Re-Computation of the Integer Decision

Algorithm 4 has to be repeated continuously in order to account for dynamic changes in the scenarios such as newly entering vehicles. While the underlying trajectory negotiation (Line 7 of Algorithm 4) has to be re-computed on a frequent basis since its decision can be safety critical, the upper level integer decision (Lines 1–5 of Algorithm 4) might act on a different re-computation law (compare also Figure 2.6 and Figure 4.5). In the following, we introduce such a re-computation law which acts according to an event-based fashion rather than commonly used time-triggered updates.

The re-scheduling strategy triggers a new computation of the vehicle order only if vehicles deviate to a certain extend from their previous plan. Thus, computation resources are

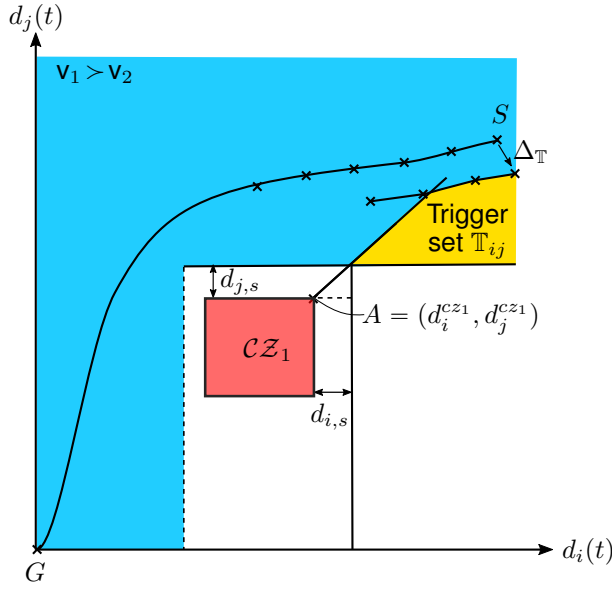


Figure 4.8: Illustration of trigger set (yellow region).

spared as in nominal scenarios no re-computation is required. Moreover, communication resources are less demanded compared to a time-triggered realization. To this end, assume that a first set of trajectories has been found at time step t_1 following the vehicle sequence $v_i \stackrel{oi}{>} v_j$, i.e., v_i crosses \mathcal{CZ}_1 before v_j . Then, we define the trigger set as

$$\mathbb{T}_{ij} = \left\{ (d_i, d_j) \in \mathbb{R}^2 \mid d_j - d_j^{\mathcal{CZ}_1} \leq d_i - d_i^{\mathcal{CZ}_1} \wedge d_i \geq d_{i,s} \wedge d_j \geq d_{j,s} \wedge \bigwedge_{k \in \mathbb{I}_{t_1, t_1+M-2}} d_j - (d_j(k|t_1) - \Delta_{\mathbb{T}}) \leq \frac{d_j(k|t_1) - d_j(k+1|t_1)}{d_i(k|t_1) - d_i(k+1|t_1)} (d_i - (d_i(k|t_1) + \Delta_{\mathbb{T}})) \right\}, \quad (4.19)$$

with $\Delta_{\mathbb{T}}$ being the offset to the initially predicted trajectory in both d_i , and $-d_j$ direction. Variable $\Delta_{\mathbb{T}}$ is a design parameter of the triggering mechanism and can be potentially individual for each vehicle. It needs to balance how quickly a re-scheduling should be triggered and needs to incorporate the amount of uncertainty in the system behavior.

The set \mathbb{T}_{ij} must be checked at each time step k against all predicted trajectory points $(d_i(k|t), d_j(k|t))$, $k \in \mathbb{I}_{t, t+M-1}$, i.e., if there exists

$$(d_i(k|t), d_j(k|t)) \in \mathbb{T}_{ij} \quad (4.20)$$

for some $k \in \mathbb{I}_{t, t+M-1}$, then re-scheduling is triggered. For vehicle v_i , to evaluate if re-scheduling is to be triggered, the above check needs to be done considering respective neighbor vehicles in the sets \mathcal{P}_i and \mathcal{S}_i .

An illustration of the trigger set (4.19) is given in Figure 4.8 through the yellow region on the right side.

Remark 4.4 (Event-triggering concept). Given predictive driving trajectories, it is not required to adjust the vehicle sequence in a coordination scenario if vehicles can accu-

rately follow their planned trajectories. Therefore, the event triggering scheme in this section causes a re-computation of the vehicle sequence only if any of the vehicles deviate from its plan to a certain extent. Consider an intersection crossing scenario where a vehicle has to brake due to a crossing pedestrian that was not considered in its previous predictions. It might be desirable to re-compute the crossing sequence as other vehicles, which are not affected by the crossing pedestrian, can cross the intersection first. This can significantly save computational resources in nominal driving behaviors and support the Scalability challenge (Chapter 1). The proposed triggering set can also exclude safety-critical sequence adjustments, e.g. if a vehicle is already too close to a conflict zone and cannot stop anymore before crossing it. The latter is an important contribution to the Safety requirement.

An alternative event-triggered approach focusing on safe decisions for CAVs is presented in [SHZ+18]. It is proposed to design an infrastructure unit that schedules vehicles to submit their trajectories considering entering a bad set, which would lead to an inter-vehicle collision.

4.4 Numerical Examples

This subsection discusses numerical simulation results of Algorithm 4 and, therefore, a combined implementation of the proposed scheduling approach from this chapter and the DJOR trajectory negotiation (Algorithm 2) from Chapter 3.

The simulation examples are conducted using the intersection scenario described in Subsection 4.2.1 with a simulation setup introduced in the following. Thereafter, we demonstrate an example trial in which the use of the RCPSP method leads to a significantly coordination performance increase compared to a first-come-first-served (FCFS) decision strategy. This is followed by a simulative illustration of the event-triggered recomputation strategy from Section 4.3.3. Lastly, a comprehensive simulation and evaluation of randomly generated intersection scenarios is presented.

4.4.1 Simulation Setup

The simulation were conducted on a PC equipped with an Intel Core i5 double core processor with $2.5GHz$ and $8GB$ RAM memory. QP problems were implemented in MATLAB and solved with *quadprog()*, and the scheduling problem was solved with *intlinprog()*. Table 4.2 lists the applied parameters.

4.4.2 Results

Scheduling Performance

The following evaluation illustrates an example where the proposed RCPSP clearly outperforms a rule-based sequence decision in the intersection example. The rule-based decision is made using a FCFS principle which sorts the crossing sequence of approaching vehicles according to their distance to the intersection area. That means the closest vehicle crosses

Table 4.2: Simulation parameters

Param.	Vehicle ID		
	1, 4	2, 5	3, 6
M	50	50	50
$x_{i,v}^r$	$5m/s$	$6m/s$	$7m/s$
Q_i^n	diag(0, 5)	diag(0, 5)	diag(0, 5)
R_i^n	12	12	12
\mathbb{U}_i	$[-7m/s^2, 4m/s^2]$	$[-7m/s^2, 4m/s^2]$	$[-5m/s^2, 4m/s^2]$
$\mathbb{X}_{i,v}$	$[0m/s, 9m/s]$	$[0m/s, 9m/s]$	$[0m/s, 9m/s]$
$\mathbb{X}_i^M, \mathbb{U}_i^M$	(3.20)	(3.20)	(3.20)
$d_{i,s}$	$2m$	$2m$	$2m$
ω_i	0.5	0.5	0.5

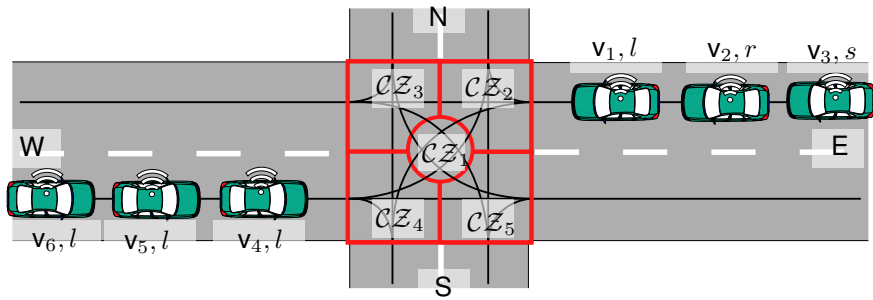


Figure 4.9: Intersection simulation scenario with 6 vehicles approaching the intersection from two different lanes.

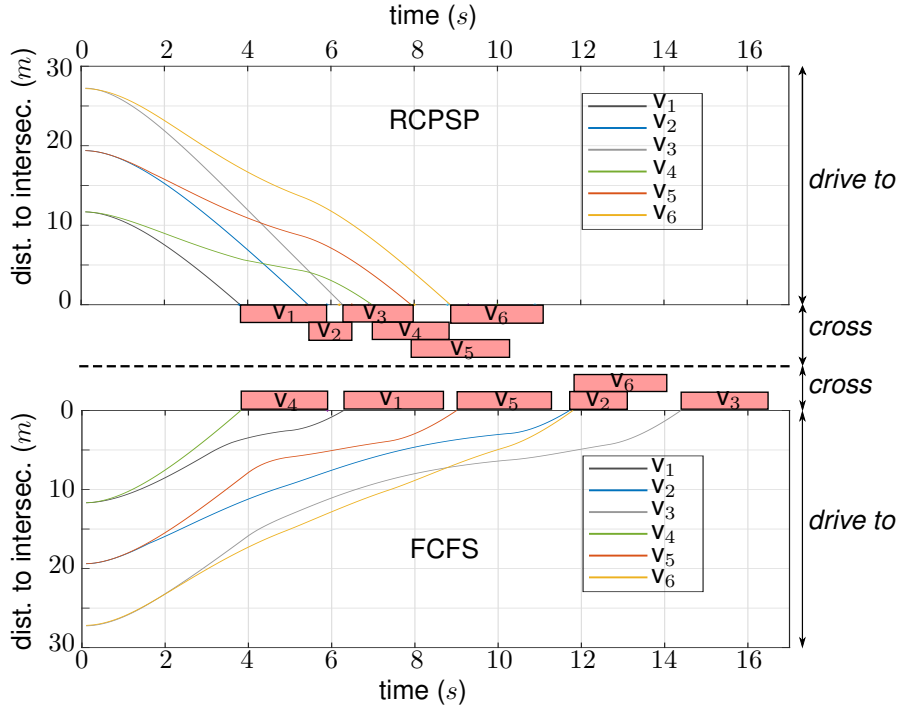


Figure 4.10: Example trial comparing the proposed scheduling method with a first-come-first-served decision strategy.

first, second closest second, etc. Figure 4.9 introduces the vehicle setups of the simulated scenario. The plot shows the vehicle IDs, $v_i, i \in \mathbb{I}_{1:6}$ and their respective maneuvers in the intersection, with right turn r , left turn l , and straight crossing s .

The result of Algorithm 4 applied in the previously introduced scenario is plotted in Figure 4.10. The top plot shows the resulting trajectories from the *drive to* activities. The distributed MPC laws (DJOR) are applied given a computed crossing order from the RCPSP solution. The lower plot represents the implemented FCFS strategy, again with the DJOR trajectory method. The bars in the middle part indicate the actual *cross* duration, i.e., the time spent in the intersection area, of the vehicles for RCPSP and FCFS, respectively.

Snapshots of the scenario in Figure 4.11 support the illustration by showing the coordination progress at time instances $t = 4.0s$, $t = 7.5s$, and $t = 10.0s$ for the RCPSP and FCFS methods, respectively.

Now, we evaluate the scenario using two performance criteria, which are i) the time to complete the scenario (until all vehicles have crossed the intersection) and ii) the acceleration effort. The later is computed by summing all applied absolute acceleration values during the scenario for all vehicles. Figure 4.12 illustrates the performance increase through the RCPSP method compared to a FCFS decision for the given scenario. The completion time of intersection crossing scenario reduces 43.2% through the application of the scheduling strategy and the acceleration effort is 49.2% lower as with the FCFS decision.

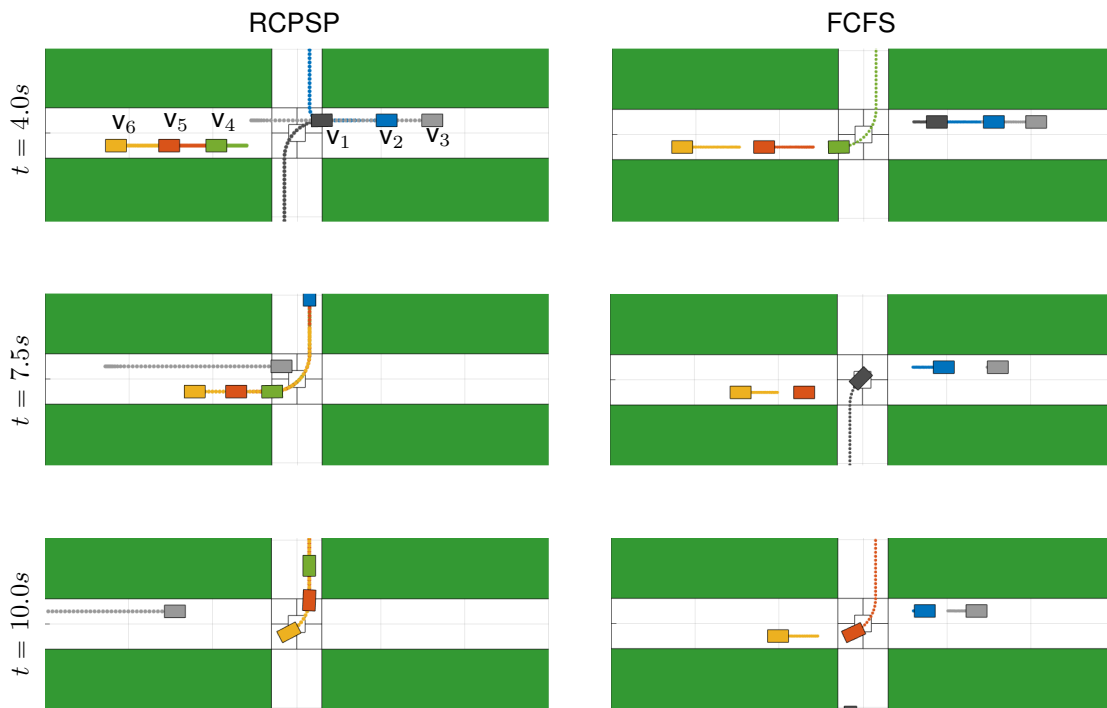


Figure 4.11: Snapshots of simulation scenario.

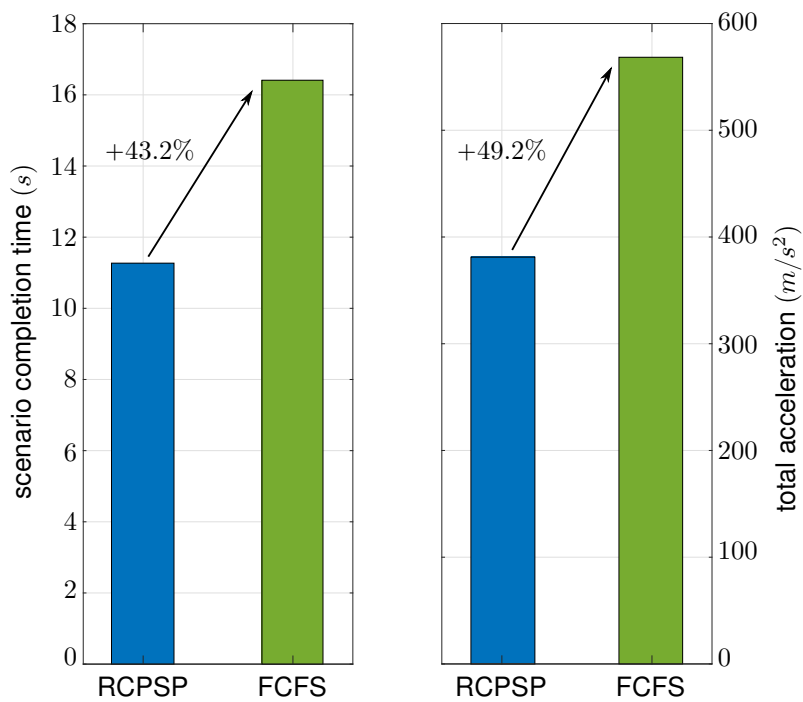


Figure 4.12: Evaluation of the scheduling performance in an example scenario.

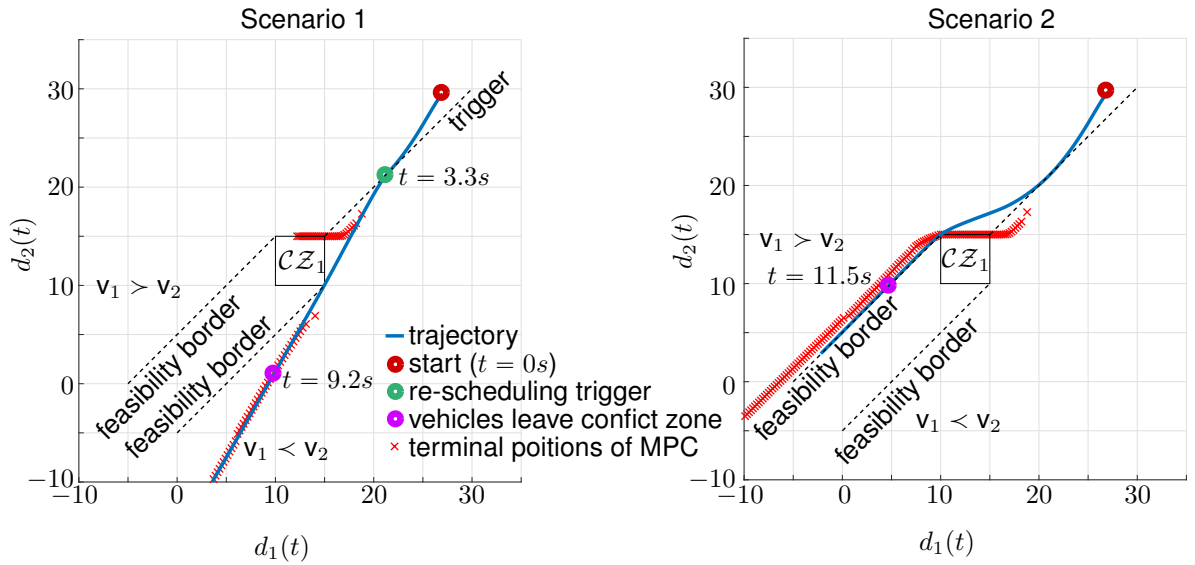


Figure 4.13: Illustration of the resulting vehicle coordination in the d_1 - d_2 -configuration-space for both vehicles for Scenario 1 (re-scheduling) and Scenario 2 (no rescheduling). It shows the feasible configurations (positions of both vehicles), and the resulting joint distance profile from start to end.

Re-Scheduling Illustration

The following numerical example illustrates the functionality and benefit of the re-scheduling law presented in Subsection 4.3.3. We simulate two vehicles, v_1 and v_2 , which approach an intersection from different directions, pass the intersection, and merge towards a common lane. In this example the intersection consists of a single conflict zone \mathcal{CZ}_1 . For illustrative reasons, we apply a simplified triggering set of the form

$$\tilde{\mathbb{T}}_{ij} = \{(d_1, d_2) \in \mathbb{R}^2 \mid d_j - d_j^{\mathcal{CZ}_1} \leq d_i - d_i^{\mathcal{CZ}_1}\}, \quad (4.21)$$

and re-scheduling is triggered if

$$(d_i(t|t), d_j(t|t)) \in \tilde{\mathbb{T}}_{ij}. \quad (4.22)$$

The setting is depicted in Figure 4.13 by the conflict zone \mathcal{CZ}_1 , the triggering manifold, and feasibility borders which depend on the scheduling decision (keep safety distance between the vehicles). We consider two differing scenarios:

- Re-scheduling: v_1 precedes v_2 until the event-trigger is activated to re-schedule the vehicles. In this case, v_2 shall precede v_1 at the intersection. (**Scenario 1**)
- No re-scheduling: v_1 precedes v_2 during the complete run. (**Scenario 2**)

Both vehicles have different characteristics in that v_1 has a reference velocity of $2.0m/s$, and v_2 has a reference velocity of $3.5m/s$, see also Figure 4.14.

The performance of Scenario 1 and Scenario 2 is evaluated according to its time duration and cost performance in Figure 4.15. The time duration of both vehicles to pass the

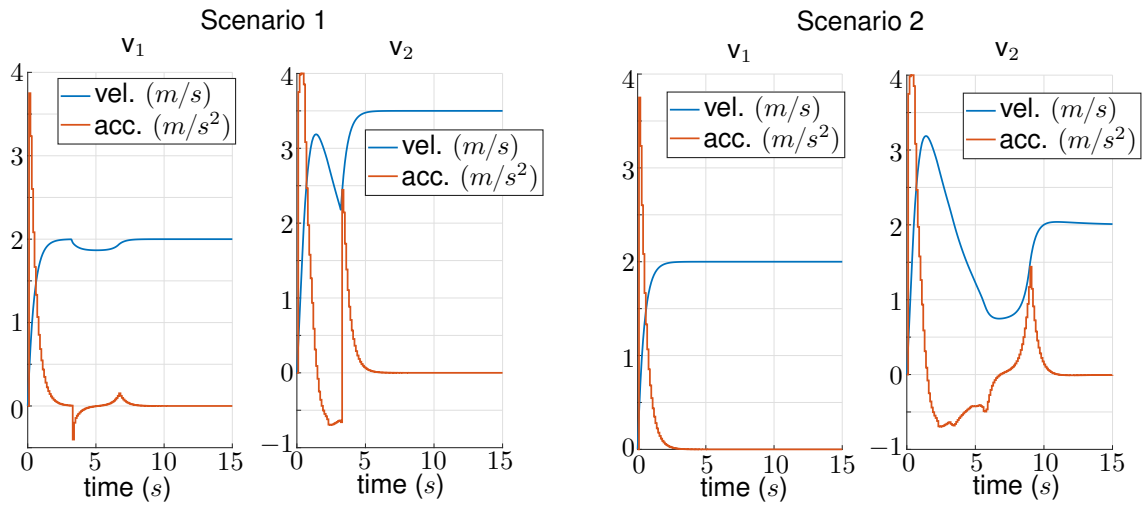


Figure 4.14: Evaluation of velocity and acceleration of vehicles v_1 and v_2 for Scenario 1 and 2. v_1 has a reference velocity of $3.5m/s$, v_2 has a reference velocity of $2.0m/s$. The rescheduling trigger appears at $t = 3.3s$.

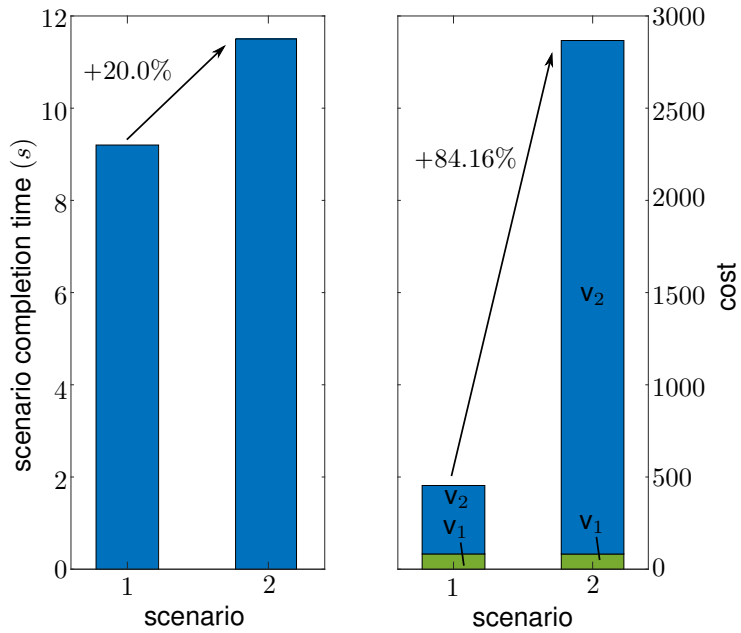


Figure 4.15: Evaluation of time duration and overall costs.

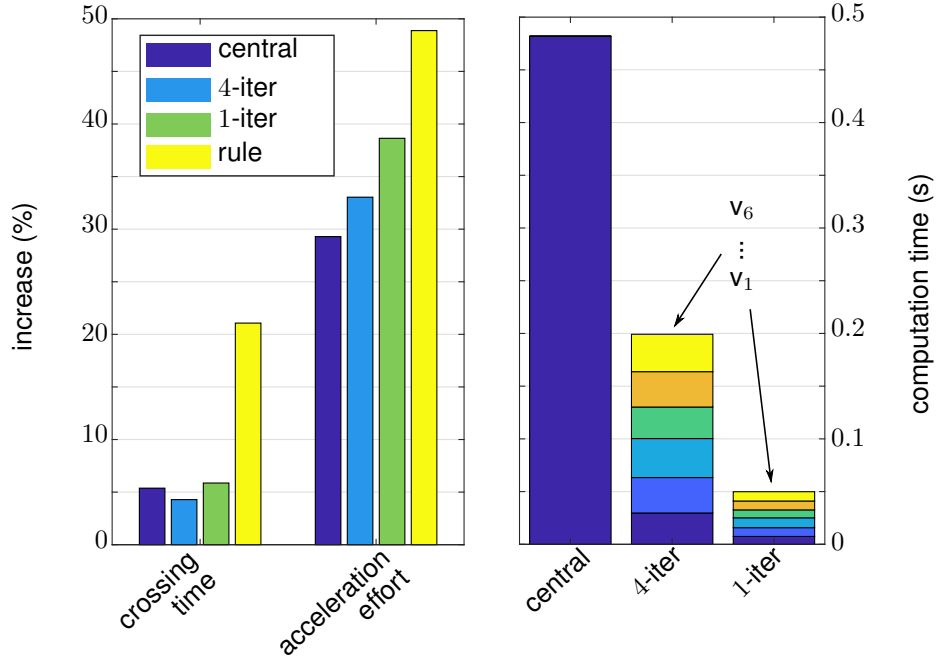


Figure 4.16: Evaluation of the intersection performance simulation.

intersection results in an improvement of 20% in Scenario 1. Moreover, the cumulative costs decrease by 80%. This is mainly due to the inability of vehicle v_2 to attain its reference speed of $3.5m/s$, which is also reflected in its cost function.

Overall, the improvement depicted in Figure 4.15 reveals clearly the benefits of the event-based approach.

Control-Scheduling Evaluation

To make a more general statement about the performance, we simulate 200 randomly generated scenarios in an intersection scenario according to Figure 4.2. In each scenario 6 vehicles are placed on incoming lanes E and W towards the intersection (vehicles v_1, v_2, v_3 on lane E, vehicles v_4, v_5, v_6 on lane W) with random initial distances, between $15m$ and $65m$, to the intersection zones and a minimum initial inter-vehicle distance of $5m$, as well as stand-still initial conditions. The maneuvers, i.e., *right turn*, *straight*, *left turn*, are determined randomly for each vehicle and scenario. For each of the 200 test cases five different control methods are simulated: (i) overpass, where no interaction between vehicles is required; (ii) centralized: the result of (3.6) with a given decision m ; (iii) Jacobi 4-iter, applying problems (3.13) with Algorithm 2 and $l_{max} = 4$; (iv) Jacobi 1-iter, the same as (iii) with $l_{max} = 1$; and (v) traffic rules, where vehicles have to yield the right-of-way which in these scenarios is the case for left turns (if two vehicles want to turn left at the same time lane E is prioritized). The crossing order sequence in (ii), (iii), and (iv) is computed according the scheduling method described in this chapter.

The overall evaluation of the 200 test cases is summarized in Figure 4.16. The left plot shows the average percentage increase of the control methods (ii)-(v) (bars from left to right) compared with the overpass method (i), which is the lower bound for the performance

measures. As performance measures the average crossing time, i.e., the time it takes until all vehicles have crossed the intersection area, and the average cumulative acceleration effort, i.e., the sum of all absolute acceleration values from all vehicles, are evaluated. The right plot illustrates the average computation time of the centralized computation (ii) and methods (iii) and (iv), where each sub-stack (in 4-iter and 1-iter) corresponds to the mean computation time of a single vehicle.

The crossing time evaluation shows similar results for methods (ii)–(iv). The crossing time itself is not the foremost optimization objective, which becomes visible as the average central methods (ii) finds longer results than (iii), the 4-iter method. However, methods (ii)–(iv) result in significantly shorter crossing times than the traffic rule method (v). The acceleration effort is the lowest for the central computation result (ii) which is about 20% less than in the traffic rule simulation (v). Also, a performance improvement effect of increased inter-sampling iterations of the Jacobi approach becomes visible by comparing method (iii) and (iv). Investigating the computation time reveals a significant benefit from using distributed optimization. While a centralized computation consumes on average $480ms$, 4 inter-sampling iterations require between $30ms$ and $40ms$, and a single iteration between $8ms$ and $9ms$ for each vehicle. Remember that the solution for each vehicle in methods (iii) and (iv) can be computed in parallel. The low computation times of the distributed implementations leave significant room for the inter-vehicle communication within a sampling time interval of $T_s = 100ms$.

4.5 Summary and Discussion

This chapter discusses a decomposition of the central MIQP coordination problem into an integer optimization problem (upper layer), which determines the vehicle sequence, and optimization problems with continuous variables (lower layer) that have been solved in Chapter 3. Moreover, a concrete solution instantiation is presented. It is based on the formulation of a resource-constrained project scheduling problem (RCPSP), which is derived aligned to the example of an intersection crossing scenario.

The decomposition ensures a scalable implementation of the non-convex overall coordination problem. This argument is further supported by introducing event-triggered decision updates, which reduce computation and communication effort, while the computation effort can be adjusted through a time-scale abstraction mechanism. The scheduling-based sequence decision’s interaction with the underlying distributed control scheme is discussed and shown to be feasible for the lower control layer. Thus, it contributes to the Safety requirement. Efficiency is targeted through the objective of a high vehicle throughput representing a traffic coordination unit’s interest. The Privacy requirement is considered since pure trajectory data needs to be exchanged but no detailed model information.

A notable strength of scheduling strategies is the seamless consideration of precedence constraints, which is often challenging in classical optimal control formulations [ZGWF17; KMEH18]. The central formulation in the upper layer requires approximative adjustments to remain scalable for large coordination problems. Therefore, the proposed time-scale separation enables to adjust the problem size of the central upper layer sequence decision. Nevertheless, choosing the upper layer time-scale too coarse leads to the risk of losing

coordination performance. Additionally, changing upper layer decisions can lead to non-smooth scenarios in the lower control layer, which correlates with the loose dynamical coupling between the layers. Tighter coupling of layers is proposed in [HZGF18], where authors approximate the sequence decision of the original non-linear problem with an MIQP. This can lead to smoother coordination scenarios, but the MIQP limits the problem scalability.

Part II

Evaluation on the Use Case Automated Valet Parking

Integration Platform

Giving guarantees for autonomous vehicles' safe operation is not feasible through real-world testing, as millions of test kilometers would be required. To overcome this, a large portion of the testing effort has to be moved to virtual test environments, where system safety and performance can be evaluated using scenario-based approaches. An important goal – and at the same time a major challenge – is to ensure reproducibility of the test scenarios, automation of the test process, as well as modularity such that a seamless integration to different test scenarios and environments is enabled. It is an open problem in the field of autonomous driving (AD) testing to extend single-vehicle testing to the validation of multi-vehicle scenarios, including distributed coordination control with vehicle-to-vehicle or vehicle-to-infrastructure (V2X) interaction (cp. Validation and Implementation challenge from Chapter 1).

A detailed co-simulation setup to test autonomous driving algorithms in high-fidelity simulation environments is presented in [SBA19; SHA+18]. Such virtual environments were used to investigate safety guarantees in the projects PEGASUS [WLFM19] and ENABLE-S3 [LWIG19], which tackled the unification of scenario-based testing and the automation of the testing process, respectively. To make predictions about the real-world behavior of functions tested in simulation, the first step can be algorithms in miniature vehicles [FHG+12]. However, finally, it will still be necessary to conduct real vehicle tests. This, in turn, is too expensive and time consuming for a complete multi-vehicle setup. Therefore, mixed-reality test frameworks have been proposed in [QAZ+10] to verify an autonomous intersection crossing protocol using one real vehicle and virtual opponents. This strategy represents a scalable testing method to make conclusions about the control strategies' real-world behavior given a complete vehicle fleet under test. It remains an open problem to ensure reusability for other use cases and the modularity of the integration.

In this chapter, we present a platform for automated high-fidelity simulation of cooperative driving scenarios. Rather than being tailored to a single autonomous vehicle, it can

test distributed control algorithms for multi-vehicle setups. Thereby, agents can be added or removed from the distributed system in a dynamic and plug-and-play manner. Furthermore, changes in the environmental setup can be considered flexibly. We extend the virtual test platform for multi-vehicle simulation with a real-vehicle, i.e., a mixed-reality vehicle-in-the-loop (ViL) test. Essential is the modular integration, which contributes to future seamless test-platforms using (defacto) standards for interfacing and scenario description. It is, thus, a cost and time-efficient way for validating multi-vehicle systems.

Control strategies developed in this thesis will be applied and evaluated in Chapter 6 using the presented platform from this chapter. The application will be demonstrated on the use case automated valet parking (AVP), which is introduced and discussed regarding its significance for multi-vehicle coordination in the following section.

This chapter is based on results published in [KMM+20; KDM+20; EKM+20].

Outline

Section 5.1 describes the AVP use case. Section 5.2 introduces the test architecture of the virtual test platform and its implementation. This architecture is extended in Section 5.3 toward mixed-reality testing. The chapter is concluded in Section 5.4.

5.1 Use Case Automated Valet Parking

An automated valet parking (AVP) scenario describes the use case where a driver drops off his/her car in front of a parking area or drop-off zone and authorizes a parking area management (PAM) infrastructure system to take control in order to autonomously guide the vehicle through the parking area and into a designated parking bay. Similarly, the vehicle is coordinated back to a pick-up zone after a respective user request. Figure 5.1 illustrates an AVP scenario. The system ensures safe and efficient vehicle coordination and storing through collaborative maneuvers, computed in a distributed fashion in vehicles and the infrastructure unit. Such implementations are expected to be available at airports, shopping malls, and large parking areas next to places of interest. As the driver exited the vehicle already before starting the process, vehicles can be parked closer to each other, which increases the vehicle capacity of parking areas. Additionally, the user saves the time of finding an empty parking bay and maneuvering into it.

From a control perspective, the size of a typical coordination space in an AVP scenario makes it relevant to apply modeling strategies that consider both local vehicle-level and traffic coordination aspects. Algorithms applied in AVP systems need to fulfill the Safety, Privacy, Cooperation, and Scalability requirement from Chapter 1. Safety to avoid accidents in parking areas, Privacy as vehicles from different manufacturers will interact, Cooperation to avoid deadlock situations in the parking area, and Scalability to take large parking areas into account.

Autonomous vehicles without a driver on board have to fulfill the SAE (society of automotive engineers) Level 4 safety requirements, which means they can operate under certain environmental conditions, referred to as the operational design domain (ODD). AVP is a strong and representative use case for current AD development since it is possible to de-

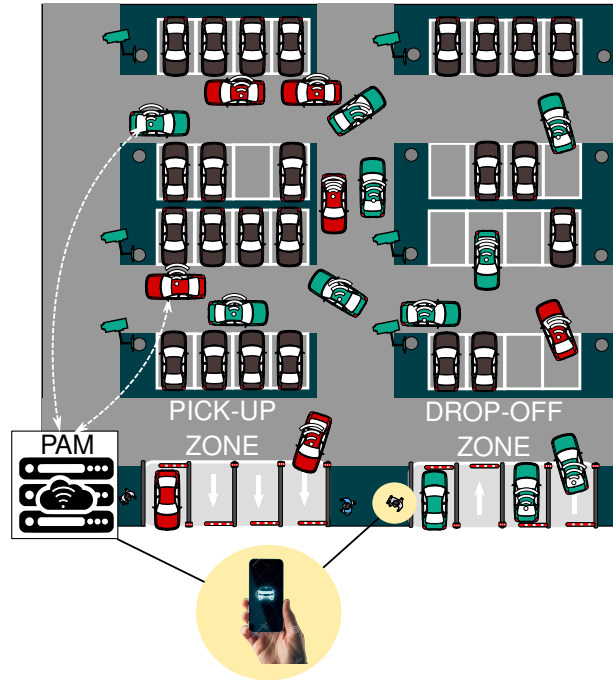


Figure 5.1: Automated valet parking (AVP) scenario with parking area management (PAM) infrastructure and distributed multi-vehicle coordination.

fine and influence a clear ODD definition, which is difficult for general AD scenarios. For example, in an AVP environment, pedestrians can be allowed or restricted to enter. Manual driven vehicles can be considered to interact with autonomous parking vehicles or be restricted to separated parking areas. Alternatively, manual driven vehicles can be transported by robotized platforms that interact with other automated vehicles. Environmental details can be known a priori, such as a detailed map of the parking lot or constant weather conditions in indoor lots. Moreover, the low-speed restriction in parking environments supports providing safety guarantees and will contribute to a faster homologation procedure than high-speed AD scenarios. Finally, there is a close relation between AVP scenarios and other applications, such as automated guided vehicles (AGVs) in warehouses or airplane taxiing at airports. This makes the developed methodologies applicable to a wide range of industry-relevant use cases.

5.2 Integration in Virtual Test System

The multi-vehicle coordination algorithms are integrated into a test system to tackle the Validation and Implementation challenge from Chapter 1. This virtual test system is capable of automatically conducting and evaluating virtual experiments of AVP scenarios. It represents the baseline architecture used in the ENABLE-S3 project [LWIG19]. In the following, the test architecture and its components are introduced. Figure 5.2 illustrates the architecture setup.

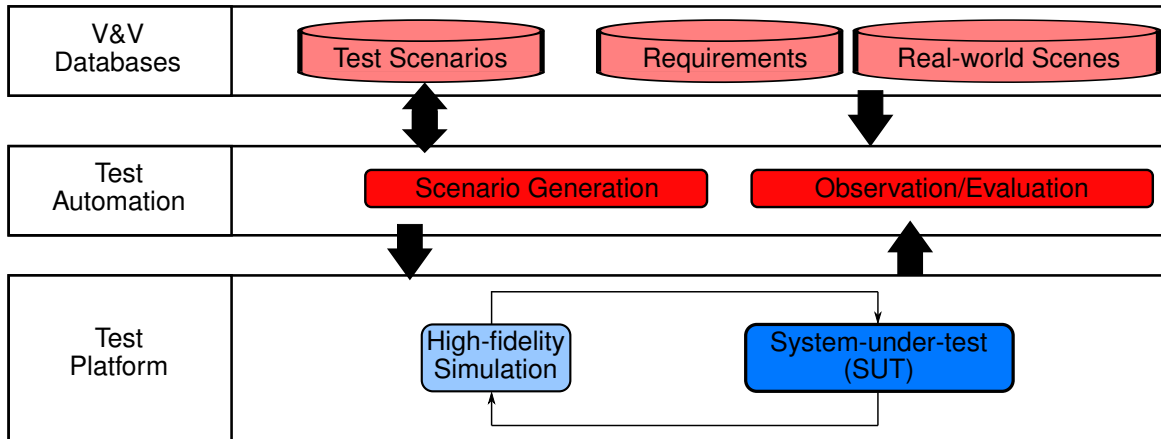


Figure 5.2: Architecture of automated test system.

5.2.1 V&V and Test Automation

The topmost level, i.e., the verification and validation (V&V) management, contains databases with test scenarios, requirements that have to be fulfilled, and a set of real-world scenes.

These databases are accessed and potentially modified by the underlying test automation layer responsible for conducting the test in an automated manner. It generates synthetic scenarios by using the scenario database, then it passes the scenarios one by one to the test platform and triggers its start. The resulting scenario data contains information about the generated valet parking environment and a set of initial conditions. During the tests, an observer records the scenario, and after completion, the traces are evaluated in the scenario evaluation module. This evaluation checks the fulfillment of defined safety requirements. The test is passed if all the requirements were met by the system-under-test (SUT), i.e., the implemented algorithms in the test platform layer. A final evaluation classifies the conducted test run according to its degree of reality by comparing it with real-world scenes. Note that the non-passed test cases can be stored and used to evaluate, re-develop, or adjust the SUT algorithms. They can also be used for an adaptation of the scenario requirements. The main target of this procedure is to accelerate the development process.

Remark 5.1 (Test system capabilities). An essential goal of testing strategies described in this section (cp. Figure 5.2) is to conduct scenario-based testing to achieve a required test case coverage and an increased test speed. This procedure shall support the safety argumentation for AD functions. This thesis's focus lies in the derivation and integration of multi-vehicle coordination strategies such that they are applicable in such automated test systems.

5.2.2 Virtual Test Platform Architecture and Implementation

The bottom layer of the test system defines the test platform, which is responsible for simulating the parking processes for the provided scenarios. Its detailed architecture is

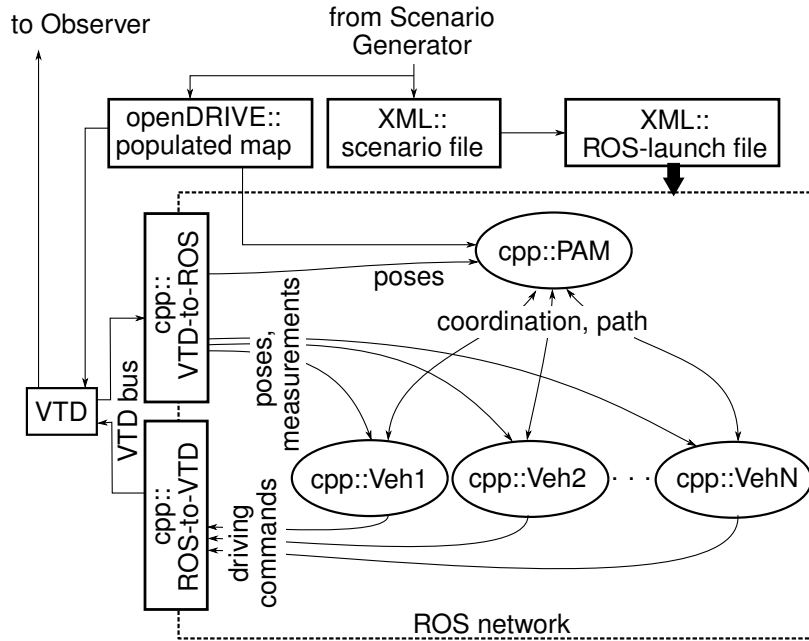


Figure 5.4: Simplified overview of the ROS network integration with connection to the simulation environment VTD, and the scenario generator output.

is interpreted by the VTD environment and is at the same time used by the PAM ROS-node. The scenario file specifies the vehicles' IDs and distinguishes between each vehicle's status, i.e., *dropped-off*, *parked*, or *pick-up-requested*. If a vehicle has the status *parked*, it is not handled by the PAM, whereas *dropped-off* and *pick-up-requested* vehicles will be controlled by the SUT during the test scenario. The PAM ROS-node is programmed in C++ such that it can handle an arbitrary number of vehicles according to the scenario file. We used MATLAB and Simulink to design and develop the vehicle control unit, i.e., the lateral control and the longitudinal MPC. The MPC law is converted into a quadratic programming (QP) formulation and passed to the integrated qpOASES solver [FKP+14]. The controller setup is then exported as a standalone C++ ROS-package using MATLAB's Robotics System Toolbox and Embedded Coder. In order to achieve the simulation of several vehicles, we use the ROS-launch concept. Thereby, several ROS-nodes with varying names and parameters can be launched using the same ROS-package. The necessary extensible markup language (XML) based launch-file is automatically generated based on the scenario-file information. This method enables the simulation of a varying number of vehicles in different simulation scenarios. Figure 5.4 illustrates the described setup and presents a simplified structure of the ROS network, as well as its connection to the simulator and the test management layer. Table 5.1 lists the used message types and explains details for the most important messages transferred within the test platform. The first four messages in the table are defined using standard ROS-message types, whereas the coordination message is a user-defined ROS message tailored to the proposed MPC problems. For communication with VTD (VTD bus), internal runtime data bus (RDB) messages are used. Note that all ROS messages in the table are published separately in the namespace of each vehicle.

Table 5.1: Explanation of the most important ROS messages in the multi-vehicle control framework.

Signal name in Figure 5.4	Message type	Explanation
pose	geometry_msgs/Pose	position and heading
measurements	std_msgs/Float64 std_msgs/Float64	velocity heading rate
path	nav_msgs/Path	waypoints from path planner
driving command	ackermann_msgs/ AckermannDrive	steering and acceleration input
coordination	ctrl/Ctrl_msg	dist. predictions of predecessor, refs. and constraints for MPC
VTD bus	Runtime Data Bus (RDB)	VTD interaction at run-time

5.3 Mixed-Reality Testing

This section extends virtual test system from the previous section to a mixed-reality system. This means that a real test vehicle, driving on a testing-ground, will interact with the remaining virtual simulated vehicles. This testing method enables conclusions of the algorithmic behavior, including real (vehicle) hardware, without the necessity to implement the complete multi-vehicle setup in a real test environment. To this end, the Scalability of the implementation can be preserved.

5.3.1 Real Vehicle Integration

The virtual test platform is modified by removing one of the virtual vehicles in the scenario and integrating a real test vehicle, that becomes part of the SUT control loop, instead. Figure 5.5 demonstrates the extension with a real vehicle. This real vehicle will operate on an empty test ground while its environment is modeled in the high-fidelity simulator. Therefore, the position of the real vehicle will be projected into the virtual environment. This requires a sensing and localization module, which we explain in Section 5.3.2. The planning and control unit corresponds to the virtual opponents' implementation, which implies a collaborative interaction between real and virtual vehicles.

In the following, we provide an overview of the real vehicle integration into the test setup.

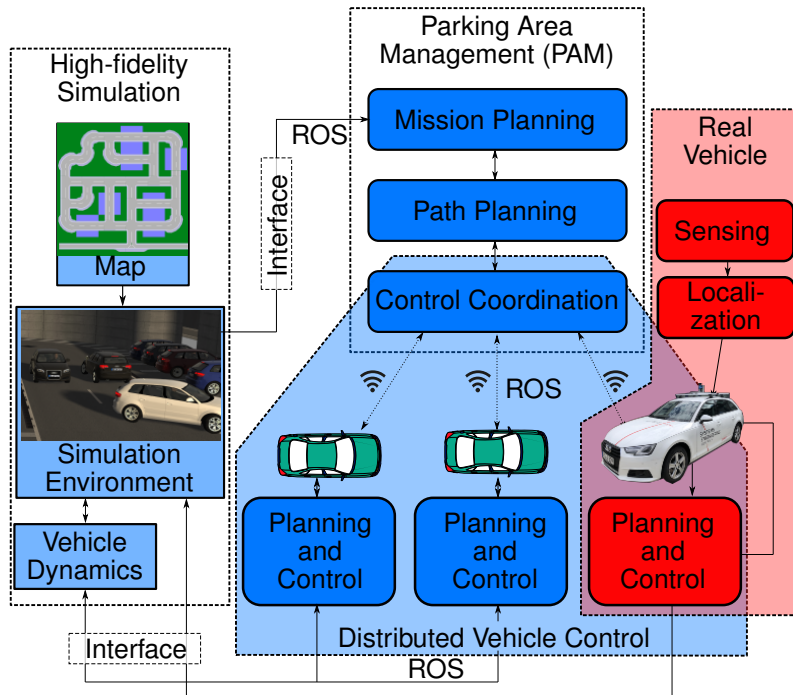


Figure 5.5: Mixed-reality test architecture.

Test-Vehicle Hardware

An Audi A4 car is used to provide a base platform for the real vehicle. All modifications have been implemented as modules attached to the baseline series production vehicle. Required low-level control functions are already available, embodied by electric power steering (EPS), brake actuation by the electronic stability program (ESP), and acceleration through the adaptive cruise control (ACC) electronic control unit (ECU). Parking brake and drive mode selector are also realized in a “by-wire” implementation. The vehicle setup is illustrated in Figure 5.6.

The communication and handling of signals between the virtual simulation and the real vehicle are enabled via a dSPACE MicroAutoBox II (MABXII), which is connected to the vehicle’s FlexRay and controller area network (CAN bus). A Simulation PC, running the virtual simulation and executing the planning for the vehicle’s trajectory, is interfaced through a network router with the MABXII to access the vehicle hardware. To precisely localize the vehicle in the parking environment, a LIDAR delivers positioning data to the PC.

The communication topology is split into different levels. Starting from the vehicle, FlexRay, CAN, and analog lines are connected to a switching box. These connections can be interrupted by an emergency switch, which cuts all connections and sends an analog command to activate an emergency braking maneuver. From the switching box, two FlexRay devices are connected to the FlexRay bus and the CAN bus, one for controlling the lateral movement via the electric power steering (EPS) and one to control the longitudinal movement via the adaptive cruise control (ACC). Furthermore, a CAN-Log device is attached to the vehicle. It can access the CAN bus and provides D/A-converters to con-

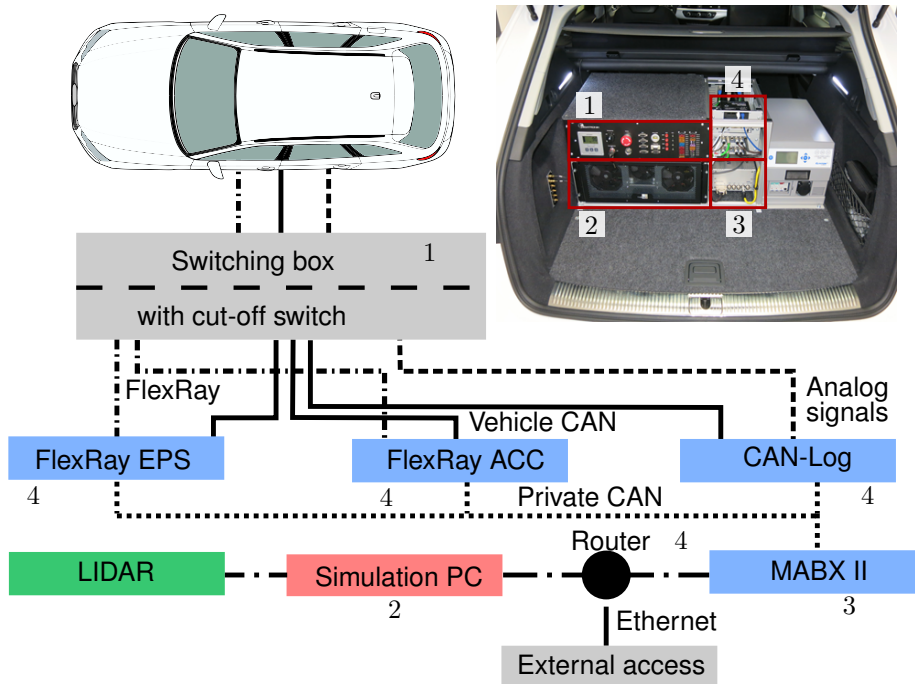


Figure 5.6: Schematic network diagram of the vehicle integration.

trol body and auxiliary functions such as the indicator lights, brake light, gear selection, and parking brake. The MABXII acts as a translator between the vehicle interfaces and the simulation PC via an ethernet connection. The same safety functions as in the series production car are used since the access to the vehicle is a manipulation of existing ECU signals. Every signal can be overridden by the driver's commands in the same way as a driver would counter an unwanted behavior of the factory-installed assistance systems.

Test-Vehicle Software

The software to operate the vehicle is built to mimic a ROS node as a simulated car in the virtual world. Therefore, a ROS bridge handles the communication between the simulation PC and the MABXII. This is performed by sending UDP packages to exchange data. Within the MABXII, these values are converted to the control parameters of the vehicle. Since the control parameters are unique to the specific vehicle, the translation is implemented as abstract as possible in the toolchain. This enables the ROS bridge to be reused for other test vehicles. As the simulated vehicles' operation is simplified in comparison to a real car, the MABXII takes over several macro functions. While simulated vehicles react directly to certain signals such as the acceleration, the MABXII has to conduct more complex actions for the real vehicle, e.g., the corresponding drive mode selection or switching between the throttle and brake actuation.

5.3.2 Sensing and Localization

For virtual vehicles, the ground truth position is available from the simulator. In contrast, for the real vehicle, the position has to be measured by on-board sensors to be projected into the virtual environment. We briefly sketch the applied set-up and implementation to provide a complete picture of the test system integration.

The real vehicle's localization algorithm is based on an Extended Kalman Filter (EKF, [TBF05]) that fuses odometry measurements with LIDAR data. The odometry information consists of vehicle speed and steering angle, captured from the real vehicle's CAN bus. These values are used as inputs for a kinematic bicycle model, which is integrated numerically to estimate the vehicle's position and orientation on the driving plane at 40Hz. This step is usually called *model update*. The kinematic bicycle is considered accurate for low-velocity applications, which is the case in parking lots.

The model updates tend to drift as time advances. Therefore, LIDAR data is used to correct the position estimate, commonly referred to as *measurement update*. A SICK NAV245 LIDAR sensor performs a 2-D scan and produces a list of detected landmarks, which have been mounted at the parking area and have been previously mapped. The constellation of detected landmarks is compared to the pre-existing map to deduce the vehicle's position and orientation, which would produce the observed constellation. Finally, the EKF uses a weighted average between the two estimates (model-based and landmark-based) to produce the corrected localization estimate of the vehicle in the parking lot.

5.4 Summary and Discussion

To automatically test multi-vehicle coordination scenarios, this chapter introduces a multi-level test architecture for virtual scenario simulation and evaluation. An upper-level management layer can automatically generate AVP test scenarios based on several databases and simulate them in a lower-level test platform containing the AD algorithms. Each unit of this test platform is implemented as a separate node to obtain the modularity required for multi-vehicle problems. A ROS middleware is suggested to connect these nodes in the test platform. The platform can be seamlessly extended with a real vehicle to achieve mixed-reality testing. This enables a scalable and cost as well as time-efficient safety and performance evaluation of control algorithms for multi-vehicle coordination applications.

Modularity is an essential feature for the reusability of test platforms in other use cases and entities. Moreover, the reproducibility of results is important for a test case evaluation. The robot operating system (ROS) middleware concept is a flagship example for algorithmic interfacing between several (control) units. It lacks, however, reliability what might lead to a loss of reproducibility. Extensions toward middlewares with reliable component communication, e.g., ROS2, has to be considered for next-generation platforms. Modularity can be improved by standard interfaces such as functional mockup units/interfaces (FMUs/FMIs) [BOA+11; BOA+12], e.g., for dynamics and simulator modules. To support the standardization and, consequently, modularity, open-source efforts such as openADx² will play a key role in future AD platform development.

²wiki.eclipse.org/OpenADx

Experimental Assessment

Test platforms, such as introduced in the previous chapter, play an essential role for validating multi-vehicle algorithms. Additionally, developed algorithms need to be applicable to such platforms to reach the state of production (cp. Validation and Implementation challenge from Chapter 1). Therefore, this chapter presents the algorithmic implementation and experimental evaluation using the test-platforms introduced in the previous Chapter 5, both for virtual and mixed-reality testing. We discuss the distributed multi-vehicle coordination control design applied to the automated valet parking (AVP) use case. Concepts for longitudinal control have been derived in Chapters 3 and 4 by assuming perfect lateral path tracking capabilities. To enable the implementation into high-fidelity simulation platforms and real test vehicles, extensions and adaptations to previously discussed concepts are required. First, it remains to discuss path planning and lateral control strategies to cope with the perfect tracking assumptions (cp. Assumption 2.1 and 2.2). Second, limitations on inter-module communication for platforms described in Chapter 5 have to be considered.

For parking environments, Dolgov et al. [DTMD08; DTMD10] introduce graph-guided (semi-structured) path planning, which rewards the planner staying close to the given structure in the parking area. Additionally, planning and control for AVP have been proposed hierarchically in [KMM+20; LJM19]. We extend these ideas toward a layered semi-structured path planner, where an infrastructure-located layer determines drivable areas and a vehicle-located layer determines a feasible path within respective areas and control signals for lateral path tracking. Moreover, we discuss the implementation of the distributed control concept – with central control coordination and local vehicle control functions – on the example of AVP. It contains a seamless extension of the conflict zone coordination concept from Chapters 2 – 4 towards a multi-intersection scenario in the AVP environment.

This chapter is based on results published in [KMM+20; KDM+20].

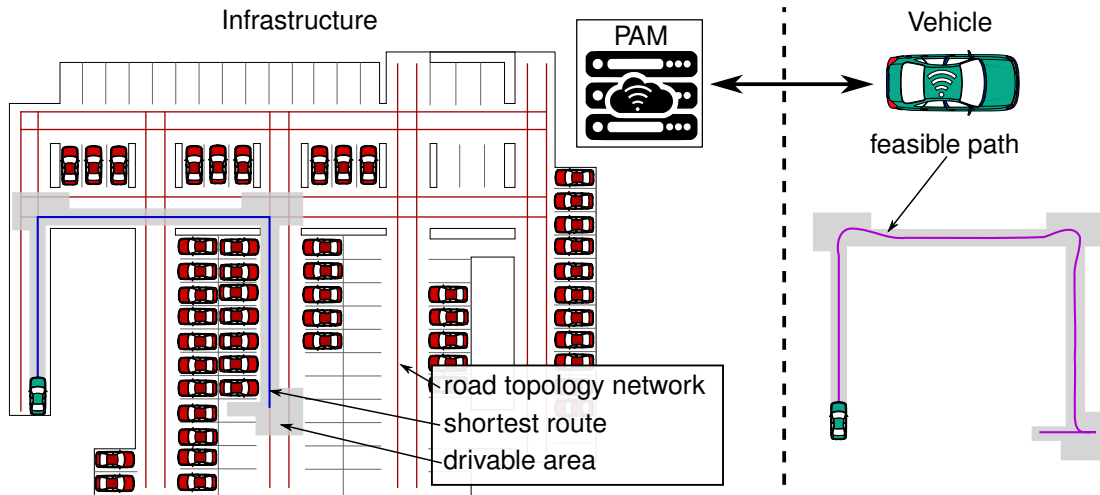


Figure 6.1: Semi-structured path planning for parking environments.

Outline

Section 6.1 introduces the applied algorithms for the AVP experiments. These are path planning, distributed coordination control, and lateral tracking control. In Section 6.2, we discuss the experimental setups for virtual and mixed-reality testing, as well as the results, respectively. Section 6.3 discusses concluding remarks.

6.1 Distributed Planning and Control for AVP

Referring to system-under-test (SUT) components from Figure 5.3, mission planning is not within this thesis’s scope. Its decisions can be computed independently of the proposed control strategies. Therefore, in what follows, we assume a given mission decision, which is - from a control perspective - reducible to the target positions for each coordinated vehicle. In the hierarchical design, the path planning decision is also decoupled from the longitudinal control coordination and could, in that sense, be an assumed component. However, its result is relevant for enabling the collision avoidance guarantees derived in the methodological part of the thesis. Therefore, we introduce the applied path planning conceptually in the following.

6.1.1 Semi-structured Path Planning

The layered semi-structured approach uses a routing algorithm to extract a drivable corridor from the openDRIVE defacto map standard [DSG10]. Next, it computes a feasible path within this corridor, applying a path planner that takes the non-holonomic vehicle model into account. The guarantee that a vehicle operates solely in its allocated corridor will be utilized by the parking area management (PAM) infrastructure to ensure the validity of longitudinal safety guarantees. Figure 6.1 illustrates the hierarchical path planning concept.

Infrastructure-supported Planning

After receiving the automated vehicle’s initial and goal pose, the path planning module of the PAM first computes the shortest path based on the lane network topology stored by the infrastructure. This topology encodes drivable lanes, including their length, direction, and their interconnection with each other. Due to the limited size of parking lots, we apply a Dijkstra algorithm to find the optimal solution (with respect to the route length) in this planning step [Dij+59]. The obtained shortest abstract path results in a sequence of lanes and intersections. These are then refined into an occupancy grip map that encodes the assigned drivable area of an automated vehicle (cp. Figure 6.1). It is computed by taking into account the geometric data of the lanes and intersections. The drivable area for the automated vehicle is potentially enlarged at intersections and in the parking bay’s vicinity to facilitate parking maneuvers. This means that, wherever the vehicle potentially leaves its own driving lane, conflict zones \mathcal{CZ} s are defined. Thus, longitudinal coordination can be applied for these areas, and derived safety guarantees become valid. The obtained occupancy grid map (OGM) is then forwarded to the local planning module.

Local Vehicle Planning

In the second planning phase, an unstructured path planner computes a feasible reference path within the provided corridor (in the OGM). As a vehicle model is utilized for this, it is desirable to compute this locally in the vehicle.

The commonly used conventional A* search algorithm connects two points in an OGM without colliding and in the shortest way by linking together the center of cells [DTMD08]. Here, we apply the hybrid A* algorithm, an extension of the A* algorithm to deal with non-holonomic behavior and continuous path generation [DTMD10]. The algorithm utilizes the provided OGM together with start and goal coordinates, i.e., (x_s, y_s, θ_s) and (x_g, y_g, θ_g) , respectively, where x and y refer to the vehicle’s global position and θ its heading. First, the planner inflates the provided OGM (i.e., shrinks the drivable corridor) to take the vehicle’s dimension into account. Next, it searches through the map by extending the current node using possible steering angles until the final point is reached. The algorithm is guided by multiple costs, represented by $V(n)$, and a heuristic, $h(n)$, to obtain desirable path quality and fast exploration. The costs include computed cost from the start to the current node and the objectives path length and steering effort. The heuristic is used to drive the search towards the goal. The algorithm considers the vehicle’s non-holonomic behavior so that the vehicle steers to the goal with a proper heading. Consequently, the hybrid A* algorithm selects the next node to expand by solving the following problem

$$\min_{n \in \text{OpenSet}} h(n) + V(n), \quad (6.1)$$

where OpenSet is the set of feasible candidate nodes in the drivable area of the OGM. Moreover, h and V are their heuristic and costs values, respectively. As can be seen, there is a compromise between the heuristic and the cost in expanding the node, influencing the path’s quality and computation time.

Remark 6.1 (Support for safe coordination). Section 3.3.4 and Section 3.3.5 discuss a safe coordination guarantee for vehicles crossing shared conflict zones using the coordination model from Section 2.3 and Algorithm 2. Collision avoidance is ensured for perfect path tracking capabilities and negligence of the vehicle skew when entering a conflict zone, stated in Assumption 2.1 and Assumption 2.2, respectively. In real systems, both assumptions might be violated. A violation of Assumption 2.2 can be compensated through an enlargement of the considered conflict zone or an increased vehicle length parameter. The semi-structured path planning strategy in this section mitigates the effect of a potential tracking error and therefore preserves the safe coordination guarantees. The infrastructure planner determines drivable tubes. It defines conflict zones where such tubes intersect, and the local vehicle planning modules determines a kinematically feasible path according to the private vehicle model within allocated driving tubes. Thus, it shall be ensured that the vehicle actually remains within the allocated drivable tube and that the tracking error can be kept small, which enables a safe inter-vehicle coordination procedure according to Section 3.3.4 and Section 3.3.5.

6.1.2 Infrastructure Control Coordination

This section describes the control coordination implementation of the AVP system. Figure 6.2 summarizes the functionality of the sub-components and illustrates the signal flow between them.

Based on predefined vehicle paths and trajectory predictions from local vehicle control units the vehicle adjacency will be determined with the support of a route graph. This information is used to construct distance constraints for vehicles which will result in a safe overall coordination procedure. It is achieved by the following steps:

Conflict Zone Detection

An essential class of conflict zones are located at all intersection areas in the parking environment. Figure 6.3 shows possible paths in an intersection area and an exemplary conflict zone distribution. It is divided into five areas by grouping nearby points, where the vehicle paths either cross, merge, or diverge, thus forming potential collision points. These areas are labeled \mathcal{CZ}_i with $i \in \mathbb{I}_{1,5}$. The conflict zones of all intersections are gathered in the set $\mathcal{I}_{inter} = \{\mathcal{CZ}_1, \dots, \mathcal{CZ}_{5N_I}\}$, where N_I is the total number of intersections in the parking environment. In addition to the intersection areas, we define a set of conflict zones \mathcal{I}_{end} at the end of each vehicle's path where vehicles conduct either the final maneuvering into the parking bay or the hand over to the user. Thereby, the cardinality of the set \mathcal{I}_{end} is $|\mathcal{I}_{end}| = N_v$ and N_v indicates the total number of moving vehicles in a coordinated parking scenario. Lastly, we gather all these conflict zones in the set

$$\mathcal{I} = \mathcal{I}_{inter} \cup \mathcal{I}_{end}. \quad (6.2)$$

The previous steps are carried out before the start of the coordination procedure and the locations of the conflict zones in the set \mathcal{I}_{inter} depend on the geometry of the parking environment. We assume that paths are tracked accurately such that a vehicle's bounding

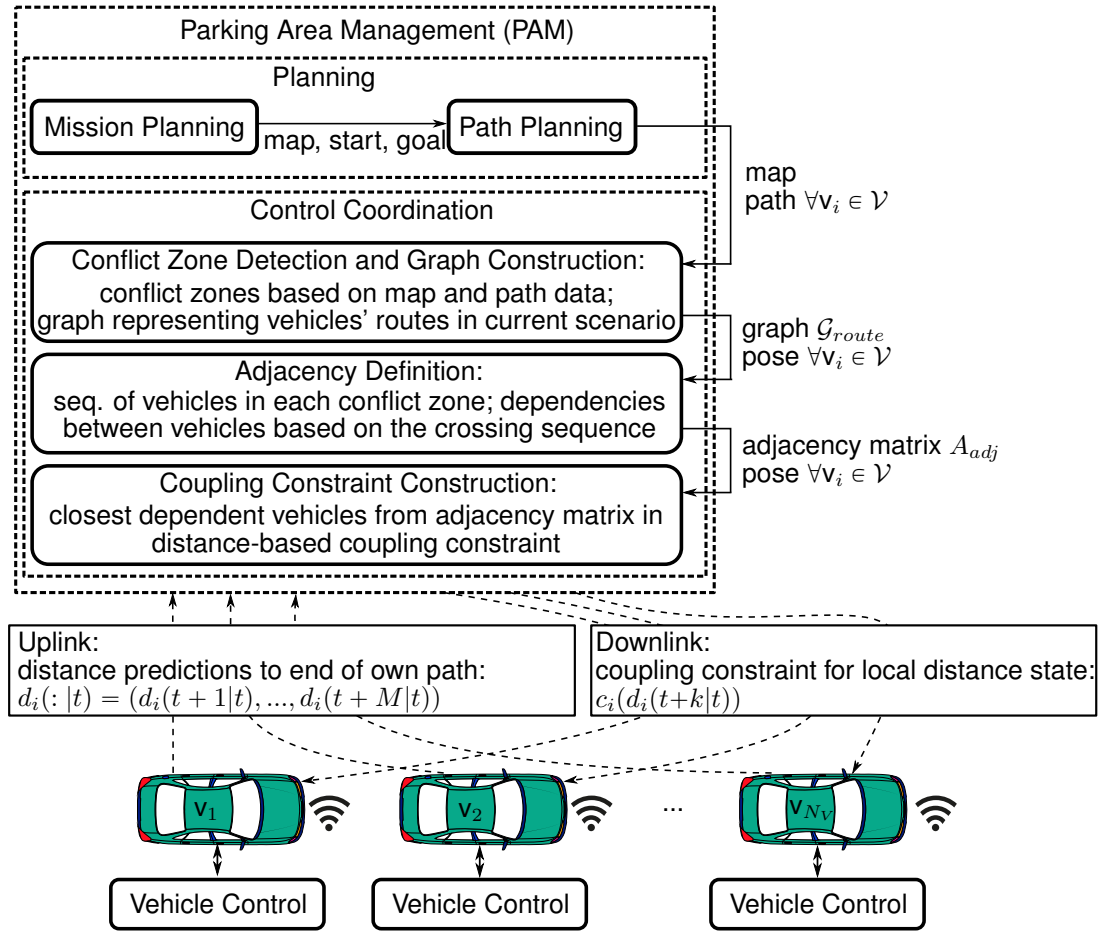


Figure 6.2: Distributed control system architecture and signal flow of the automated valet parking system.

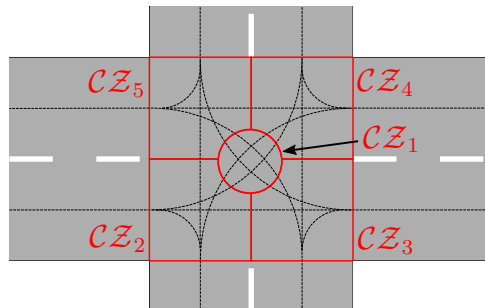


Figure 6.3: Intersection area with all possible paths and conflict zones $\{CZ_1, \dots, CZ_5\}$.

box does not laterally pass a conflict zone other than its allocated path. The elements in \mathcal{I} will serve as reference zones for coordinating the vehicles passing through a zone. This enables coordinating the vehicles approaching from different sides (intersection scenario) and vehicles coming from the same lane (vehicle following scenario) towards a critical zone $\mathcal{CZ}_i \in \mathcal{I}$.

Remark 6.2 (Conflict zones). The set in (6.2) can be extended by further zones, \mathcal{CZ}_i s, at arbitrary positions where collisions between vehicles can occur. For example, this is the case if a vehicle crosses an accommodating lane in tight areas or due to obstacles. The coordination procedure is not affected by this action.

Graph Representation

The set

$$\mathcal{V} = \{v_1, v_2, \dots, v_{N_v}\} \quad (6.3)$$

contains all vehicles that are coordinated by the PAM, where each vehicle $v_i \in \mathcal{V}$ is labeled with a unique ID. Now, we can define a multigraph $\mathcal{G}_{route} = (\mathcal{V}_{route}, \mathcal{E}_{route})$ with vertices $\mathcal{V}_{route} \in \mathcal{I}$ and edges $\mathcal{E}_{route} = \mathcal{V}_{route} \times \mathcal{V}_{route}$, labeled with v_i . An edge-vertices pair in \mathcal{G}_{route} indicates that a vehicle's path connects two conflict zones, while the edge direction represents the driving direction of vehicle v_i between those zones. Every time a vehicle has passed a conflict zone $\mathcal{CZ}_j \in \mathcal{I}$ the corresponding edge is removed from \mathcal{G}_{route} . During execution time, the sets \mathcal{I}, \mathcal{V} , and \mathcal{G}_{route} are continuously updated to cope with changes, such as vehicle poses and picked-up or newly dropped-off vehicles. This implies that the sets are time-varying.

Adjacency Definition

The resulting \mathcal{G}_{route} is used to determine a set of crossing orders

$$\mathcal{O} = \{o_i = (v_j, v_k, \dots) \mid \forall i \in \mathbb{I}_{1:|\mathcal{I}|}; j, k \in \mathbb{I}_{1:N_v}\}. \quad (6.4)$$

It represents the logical sequence in which vehicles must cross the respective conflict zone, while the actual timing, i.e. the resulting trajectory, will be determined by the distributed longitudinal control laws.

Lastly, we determine the adjacency matrix $A_{adj} = (a_{ij})^{N_v \times N_v}$, with $a_{ij} \in \mathcal{Z}$, and

$$\mathcal{Z} = \{(m_1, \dots, m_p, \dots, m_q) \cup \emptyset \mid q \in \mathbb{N}, \forall m_p \in \mathcal{I}\}. \quad (6.5)$$

This implies, a_{ij} defines a dependency between two vehicles v_i and v_j if v_i is a successor of v_j in any scheduling order o_k , where $i, j \in \mathbb{I}_{1:N_v}$, $i \neq j$, and $k \in \mathbb{I}_{1:|\mathcal{I}|}$. Thus, an element $a_{ij} = \mathcal{CZ}_k$ of A_{adj} is labeled with the conflict zone both vehicles will pass through. In those cases where vehicle v_i is a successor of v_j for several conflict zones, for example if v_i 's route partially overlaps with that of v_j , a_{ij} becomes an n -tuple, i.e. $a_{ij} = (\mathcal{CZ}_k, \mathcal{CZ}_l, \dots)$ with $k, l \in \mathbb{I}_{1:|\mathcal{I}|}$. In the following we refer to the first element of a non-empty n -tuple by using the notation a_{ij} , which is the closest conflict zone v_i and v_j have in common.

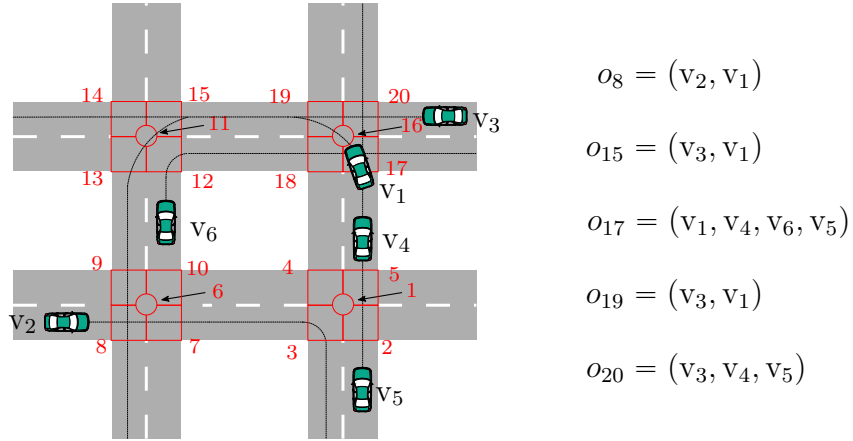


Figure 6.4: Example scenario with six vehicles and pre-computed paths. Labels at intersections are the conflict zone IDs. Scheduling orders for affected conflict zones are shown on the right.

Example 6.1. In Figure 6.4 we present an example of a coordination scenario with the vehicle set

$$\mathcal{V} = \{v_1, v_2, \dots, v_6\} \quad (6.6)$$

and four intersections leading to the conflict zones

$$\mathcal{I} = \{\mathcal{CZ}_1, \mathcal{CZ}_2, \dots, \mathcal{CZ}_{20}\}. \quad (6.7)$$

The example illustrates the states of the coordination process at a certain time instant. The paths, provided by the path planning unit, are marked with black lines and the resulting route graph \mathcal{G}_{route} is shown in Figure 6.5. Each conflict zone $\mathcal{CZ}_i \in \mathcal{I}$ that a vehicle will pass is represented by a vertex and the vehicles' paths are represented by labeled edges. The scheduling unit determines an order o_i for each vertex. The resulting ordering is shown in Figure 6.4 and finally, the scenario's adjacency matrix is given as follows:

$$A_{adj} = \begin{pmatrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ - & \mathcal{CZ}_8 & \mathcal{CZ}_{19,15} & & & \\ & - & & & & \\ \mathcal{CZ}_{17} & & - & & & \\ & & \mathcal{CZ}_{20} & - & & \\ & & & \mathcal{CZ}_{20} & - & \mathcal{CZ}_{17} \\ & & & \mathcal{CZ}_{17} & & - \end{pmatrix} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix}.$$

Remark 6.3 (Vehicle sequence decision). The resulting adjacency matrix is determined based on the crossing order set (6.4). This set results from methods discussed in Chapter 4, where a scheduling formulation is applied to find a close-to-optimal and feasible crossing sequence. It can be considered in parking areas to introduce separated clusters

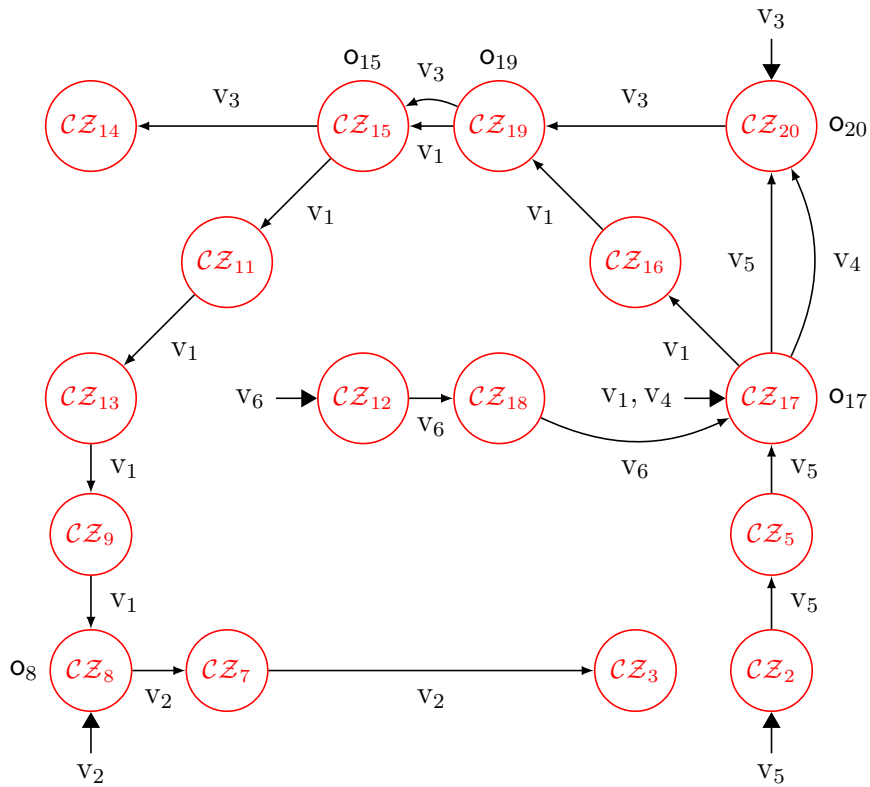


Figure 6.5: Graph representation of the vehicles' routes (\mathcal{G}_{route}) with all conflict zones, that will be passed, represented as vertices in the set \mathcal{V}_{route} and path segments between connecting vertices as edges in \mathcal{E}_{route} labeled with the respective vehicle ID. Scheduled orders o_i are defined in Figure 6.4.

such that one cluster groups several conflict zones. Thus, a sequence decision can be computed in parallel for each cluster. In Figure 6.4, each intersection can be a separate cluster, which ensures that the computation remains scalable also for large areas.

Considering parking areas where vehicles drive with low speed and have similar dynamics, even pure heuristic-based sequence decision can deliver sufficiently performant results. However, it has to be ensured that deadlock situations are avoided, which is ensured by-design in the scheduling solutions from Chapter 4.

Coupling Constraint

Given the adjacency definition A_{adj} , coupling constraints for each vehicle's MPC problem are determined by the PAM. During the coordination, each vehicle, $v_i \in \mathcal{V}$, shares the prediction vector of its distance state,

$$d_i(:, t) = (d_i(t+1|t), d_i(t+2|t), \dots, d_i(t+M|t)) \in \mathbb{R}^M, \quad (6.8)$$

with the PAM. The state d_i is the vehicle's distance to the end of its path, and M is the prediction length. Transformation $T(\mathcal{CZ}_j, i)$ defines the distance from the position where v_i 's path enters a conflict zone \mathcal{CZ}_j to the end of its path. Now, the coupling constraint c_i for each vehicle $v_i \in \mathcal{V}$ is calculated by determining the most critical predecessor vehicle with

$$j_k^* = \operatorname{argmin}_{j \in \mathbb{I}_{1:N_v}} \left\{ d_i(t+k|t-1) - [T(a_{ij}, i) + d_j(t+k|t-1) - T(a_{ij}, j)] \right\}, \quad (6.9)$$

where $k \in \mathbb{I}_{1:M-1}$, and a_{ij} refers to the conflict zone at row i and column j of A_{adj} . Given Equation (6.9), the coupling constraint is given by

$$c_i(d_i(t+k|t)) = d_i(t+k|t) - [T(a_{ij_k^*}, i) + d_{j_k^*}(t+k|t-1) - T(a_{ij_k^*}, j_k^*)], \quad (6.10)$$

with $k \in \mathbb{I}_{1:M-1}$.

For each predicted time step $t+k$, $c_i(d_i(t+k|t))$ is the closest distance of all predecessor vehicles of v_i with respect to their common conflict zone and information from the previous time step, $t-1$. To account for the one-step time shift from $t-1$ to t , $c_i(d_i(t+M|t)) = c_i(d_i(t+M-1|t))$ is kept constant at the end of the horizon. Figure 6.6 illustrates the coupling constraint formulation in (6.9) and (6.10) for a predecessor v_j of vehicle v_i with respect to the conflict zone a_{ij} at time step $t+k$. Note that the procedure is the same if vehicles approach the conflict zone from the same lane.

Remark 6.4 (Privacy of coordination procedure). Assuming that the PAM is a trustworthy unit, Privacy can be ensured by the proposed coordination procedure. Vehicles share their local predictions (6.8) with the PAM and receive a constraint vector (6.10) containing relevant predictions from other vehicles. However, a vehicle does not need to know which neighboring vehicle the information originates from exactly as the necessary transformation is done in the PAM unit. Lastly, (6.10) can be a composition of varying neighbor vehicles limiting the possibility to conclude about a certain vehicle's behavior.

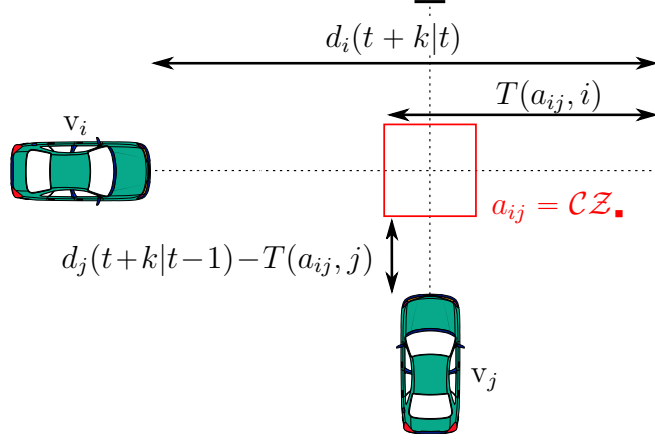


Figure 6.6: Illustration of the coupling constraint formulation using an example with one conflict zone \mathcal{CZ}_i , and two vehicles approaching from different directions.

6.1.3 Longitudinal Motion Model and Control Design

The longitudinal control is in line with derivations from Chapter 3. Remark 6.5 below will discuss adjustments with regard to the DJOR strategy from Chapter 3.

A vehicle v_i is modeled using the states d_i , v_i , and a_i . The distance state d_i is as defined in the previous section, and states v_i and a_i are the velocity and acceleration of vehicle v_i , respectively. The continuous linear-time-invariant (LTI) dynamics of the longitudinal vehicle motion is summarized as:

$$\underbrace{\begin{pmatrix} \dot{d}_i \\ \dot{v}_i \\ \dot{a}_i \end{pmatrix}}_{x_{c,i}^{lg}} = \underbrace{\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\frac{1}{T_i^{act}} \end{pmatrix}}_{A_{c,i}^{lg}} \underbrace{\begin{pmatrix} d_i \\ v_i \\ a_i \end{pmatrix}}_{x_{c,i}^{lg}} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ \frac{1}{T_i^{act}} \end{pmatrix}}_{B_{c,i}^{lg}} u_i, \quad (6.11)$$

where T_i^{act} is a time constant for throttle and brake actuation, and u_i is an exogenous control input, which represents the desired acceleration.

The longitudinal MPC law is computed solving the following optimization problem

$$L_i^* = \min_{\bar{u}_i} \sum_{k=1}^M \|\Delta_{ref} x_i^{lg}(t+k|t)\|_{Q_i}^2 + \sum_{k=0}^{M-1} \|u_i(t+k|t)\|_{R_i}^2 \quad (6.12a)$$

s.t.

$$x_i^{lg}(t+k+1|t) = A_i^{lg} x_i^{lg}(t+k|t) + B_i^{lg} u_i(t+k) \quad k \in \mathbb{I}_{0:M-1} \quad (6.12b)$$

$$x_i(t|t) = x_i(t) \quad (6.12c)$$

$$v_{min} \leq v_i(t+k|t) \leq v_{max} \quad k \in \mathbb{I}_{1:M} \quad (6.12d)$$

$$a_{i,min} \leq a_i(t+k|t) \leq a_{i,max} \quad k \in \mathbb{I}_{1:M} \quad (6.12e)$$

$$c_i(d_i(t+k|t)) \geq d_s \quad k \in \mathbb{I}_{1:M}. \quad (6.12f)$$

Here, $x_i(t+k|t)$ is the state prediction for time $t+k$ at the current discrete time step t . The model (6.12b) is attained by discretizing (6.11) using a sampling time T_s^{lg} . The

optimization variables vector is of the form $\bar{u}_i = (u_i(t), u_i(t+1), \dots, u_i(t+M-1)) \in \mathbb{R}^M$ with the horizon length M . After optimizing (6.12) in each time step t , the control input $u_i(t)$ is applied to the vehicle, and the procedure is repeated in consecutive time steps in a receding horizon fashion. The objective function in (6.12a) consists of a state error cost and a control input cost. The longitudinal reference vector in the state error vector, $\Delta_{ref} x_i^{lg}(t+k|t) = x_i^{lg}(t+k|t) - x_{i,ref}^{lg}$, is of form

$$x_{i,ref}^{lg} = (d_{i,ref}, v_{ref}, 0)^T, \quad (6.13)$$

with the reference velocity v_{ref} and the reference distance

$$d_{i,ref} = \begin{cases} 0 & \text{if } v_i \text{ has no predecessor} \\ d_s + d_{slack} & \text{if } v_i \text{ has a predecessor.} \end{cases} \quad (6.14)$$

If the reference distance in (6.14) is considered in the optimization problem, we add an additional slack value d_{slack} to the inter-vehicle safety distance d_s in order to avoid a reference at the constraint bound. The weighting matrices in (6.12a) are

$$Q_i = \begin{pmatrix} q_{i,11} & 0 & 0 \\ 0 & q_{i,22} & 0 \\ 0 & 0 & q_{i,33} \end{pmatrix} \in \mathbb{S}_0^{3 \times 3}, \text{ and } R_i \in \mathbb{S}, \quad (6.15)$$

with $\mathbb{S}_0^{3 \times 3}$ describing the space of 3-by-3 dimensional positive semi-definite matrices and \mathbb{S} positive definite matrices, accordingly.

Among the model-based equality constraint (6.12b) problem (6.12) considers inequality box constraints to limit the vehicle's velocity (6.12d) and acceleration (6.12e). Lastly, (6.12f) represents the coupling constraint, which ensures that vehicle v_i keeps a minimum safety distance d_s to its predecessor vehicles. It thus forms the interface between the local vehicle control and the global PAM coordination decisions.

Remark 6.5 (Relation to DJOR). The longitudinal control problem (6.12) is implemented in the local vehicle control unit (cp. Figure 6.2) and represents the lower negotiation layer illustrated in Figure 2.6. The negotiation procedure following a distributed Jacobi over-relaxation (DJOR) approach has been discussed in Chapter 3, using the MPC problem (3.13). Because of the real vehicle integration, the following adjustments have been considered: i) the longitudinal motion model has been extended with an integrator state of the acceleration to account for dynamics of real vehicle hardware [SD93; Raj11; SSS00], ii) the distance coupling constraint (6.12f) is only considered for predecessor vehicles and only exchanged once per sampling step, iii) measurement noise and uncertainties in the real implementation are considered through an additional slack distance d_{slack} in (6.14) to keep the optimization problem feasible. Since the extension in i) does not change the model class, it can be seamlessly integrated into the DJOR approach. In general, the communication flow modification in ii) leads to a loss of formal safety guarantees as derived in Chapter 3. This is counteracted by the assumption of homogeneous vehicles (vehicle dynamics), a reasonable assumption for parking environments. The additional slack distance d_{slack} poses an additional buffer to preserve safe

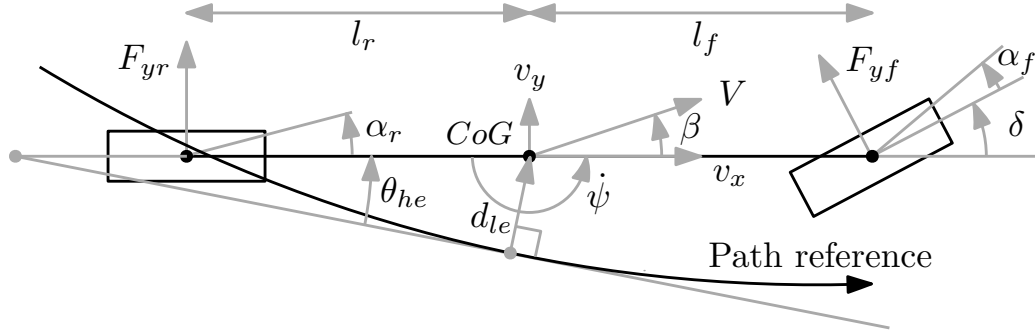


Figure 6.7: Single track vehicle model, showing the vehicle model variables [Pac05] and the lateral distance and heading angle errors with respect to the path reference [KG15].

coordination. Note that problem (6.12) serves to investigate the performance increase through a coordination procedure compared to non-coordinated approaches, as discussed in Section 6.2. An alternative solution for the approach in iii) is the formulation of soft constraints, as discussed in Section 3.4.

6.1.4 Lateral Motion Model and Control Design

We use the following continuous time matrix equation, modeling the lateral vehicle movement [KG15; Pac05]:

$$\dot{x}_c^{lt} = A_c^{lt} x_c^{lt} + B_{1,c}^{lt} \delta + B_{2,c}^{lt} \kappa + B_{3,c}^{lt} \beta, \quad (6.16)$$

where

$$x_c^{lt} = \left(d_{le}, \theta_{he}, \dot{\psi} \right)^T, \quad (6.17)$$

$$A_c^{lt} = \begin{pmatrix} 0 & v_x & 0 \\ 0 & 0 & 1 \\ 0 & 0 & \frac{-(l_f^2 C_f + l_r^2 C_r)}{J_z v_x} \end{pmatrix}, \quad (6.18)$$

$$B_c^{lt} = \begin{pmatrix} 0 & 0 & v_x \\ 0 & -v_x & 0 \\ \frac{l_f C_f}{J_z} & 0 & \frac{l_r C_r - l_f C_f}{J_z} \end{pmatrix}. \quad (6.19)$$

Here, $B_{1,c}^{lt}$, $B_{2,c}^{lt}$, and $B_{3,c}^{lt}$ are the first, second, and third column of B_c^{lt} , respectively, d_{le} is the lateral distance error, θ_{he} is the heading angle error, $\dot{\psi}$ is the yaw rate, v_x is the longitudinal velocity, C_f and C_r are the cornering stiffness of the front and rear axle, respectively, J_z is the moment of inertia around the yaw axis, l_f and l_r are the distances between the center of gravity (CoG) and the front and rear axle, respectively. Figure 6.7 illustrates the model parameters of the bicycle model. The path reference curvature κ , and vehicle sideslip angle β are not control inputs, whereas the steering angle δ is a control input. Therefore, the pair $(A_c^{lt}, B_{1,c}^{lt})$ is used to design the controller. The vehicle sideslip angle is not used as a system state as its estimation results in considerable error.

The simplified lateral model $\dot{x}_c^{lt} = A_c^{lt} x_c^{lt} + B_{1,c}^{lt} \delta$ is discretized using the sampling time T_s^{lt} and then the MATLAB command `dlqr()` is used to design the discrete-time linear

quadratic regulator (DLQR) gains. The weighting matrices

$$Q^{lt} = \begin{pmatrix} q_{11}^{lt} & 0 & 0 \\ 0 & q_{22}^{lt} & 0 \\ 0 & 0 & q_{33}^{lt} \end{pmatrix} \in \mathbb{S}_0^{3 \times 3}, \text{ and } R^{lt} \in \mathbb{S}. \quad (6.20)$$

are used to calculate the DLQR gains.

The model linearization, discretization, and DLQR gain calculations are done at three vehicle speeds v_1 , v_2 , and v_{max} . Using these DLQR gains, a velocity dependent gain scheduling is used so that the controller is robust to variations in the nonlinear velocity state. The gain scheduling uses a convex summation of the DLQR gains, as shown below:

$$G_{dlqr} = \begin{cases} G_1 & \text{if } v_{min} \leq v_x \leq v_1 \\ (v_x - 1) G_1 + (2 - v_x) G_2 & \text{if } v_1 < v_x \leq v_2 \\ (v_x - 2) G_2 + (3 - v_x) G_3 & \text{if } v_2 < v_x \leq v_{max} \\ G_3 & \text{if } v_x > v_{max}. \end{cases} \quad (6.21)$$

The lateral control input δ is generated using the following equation:

$$\hat{\delta} = G_{dlqr} x_c^{lt} \quad (6.22)$$

$$\delta = \begin{cases} \hat{\delta}(t) & \text{if } -\delta_{max} \leq \hat{\delta} \leq \delta_{max} \\ -\delta_{max} & \text{if } \hat{\delta} < -\delta_{max} \\ \delta_{max} & \text{if } \hat{\delta} > \delta_{max}, \end{cases} \quad (6.23)$$

where $-\delta_{max}$ and δ_{max} are the lower and upper limits of the vehicle steering angle, respectively. The vehicle yaw rate and speed, which are required to calculate the control input, are measured by the vehicle sensors.

6.2 Experimental Evaluation

In the following, we demonstrate and discuss results from virtual and mixed-reality experiments, which are conducted on the test platforms introduced in Chapter 5 and using control algorithms summarized at the beginning of this chapter. We introduce the test setup before discussing the evaluation results of the virtual experiments, and similar for the mixed-reality experiments.

6.2.1 Virtual Test Setup

To run the tests, we distribute the test architecture of Figure 5.4 on two PCs, as shown in Figure 6.8. The *Simulation PC* runs the test management system (cp. Figure 5.2), the VTD simulation environment, and the PAM ROS-node. It is equipped with an Intel Core i7 – 6700K 4 core processor and 32 GiB of RAM memory. The vehicle control ROS-nodes are launched on a separate *Control PC* with an Intel Core i7 – 4790K 4 core processor and 32 GiB of RAM memory. We allocate each control process (ROS-node) to a designated core of the CPU, such that two processes share one core. This guarantees the real-time computation of the MPC optimization problem, with an average solving time of $5ms$. The two PCs are connected via an Ethernet connection.

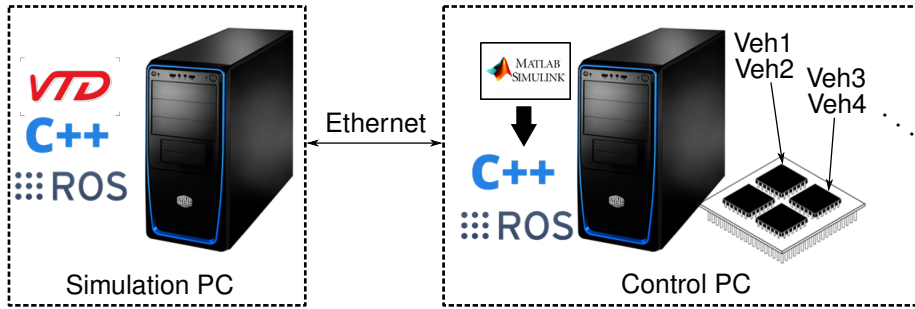
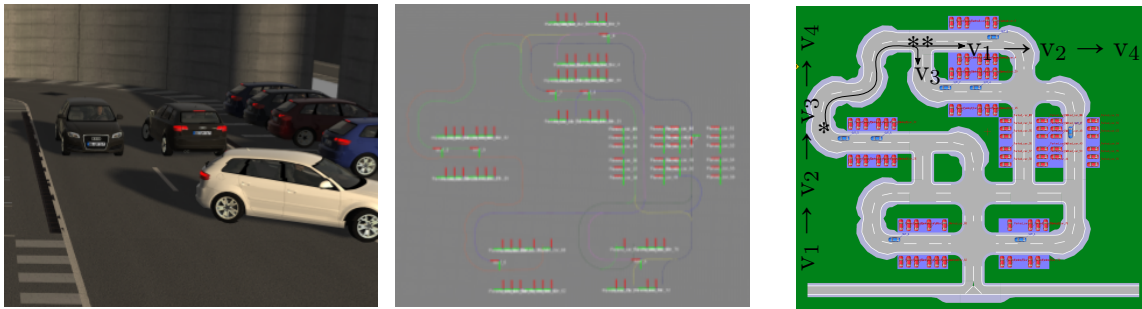


Figure 6.8: Distributed implementation of the test system in our lab.



(a) VTD scenario visualization (b) ROS-rviz visualization (c) openDRIVE excerpt of randomly generated map.

Figure 6.9: A randomly generated test scenario.

In Figure 6.9, we give insights into the test platform visualization during a trial. Figure 6.9a shows the visualization of the VTD environment. The plot in Figure 6.9b visualizes all objects (moving and parked vehicles) in the scenario using the ROS-rviz tool, and displays the paths provided by the PAM path planning unit. Figure 6.9c illustrates a randomly generated openDRIVE map of the trial from the scenario generation tool.

6.2.2 Virtual Results

Now, we illustrate the functionality of the control coordination method, and highlight the benefits compared to an uncoordinated scenario by evaluating the simulation results.

Table 6.1 introduces the control parameters used for the evaluations. Note, for the velocity constraint, that the actual maneuvering into/out of the parking bay may also require backward driving and thus the constraint should also allow $v_{min} < 0$. However, as this evaluation focuses on the coordination procedure within the parking area, we avoid backward driving.

First, the longitudinal coordination control is investigated. We thus illustrate the step-response of a single vehicle in Figure 6.10. The vehicle starts from standstill and receives a reference velocity of $1.1m/s$, which increases to $2.9m/s$ and then steps back to $1.1m/s$. The weights Q_i and R_i of the MPC problem in (6.12) are chosen such that we avoid an

Table 6.1: Control parameters in virtual tests.

Name	Parameter	Value
longitudinal sampling time	T_s^{lg}	0.1s
longitudinal time constant	T_i^{act}	0.1s
MPC horizon length	M	50
reference velocity	v_{ref}	1.8m/s
slack distance	d_{slack}	1m
safety distance	d_s	3m
velocity constraints	(v_{min}, v_{max})	(0m/s, 3m/s)
acceleration constraints	(a_{min}, a_{max})	(-4m/s ² , 1m/s ²)
longitudinal state weights	$(q_{i,11}, q_{i,22}, q_{i,33})$	(1, 60, 30)
longitudinal input weight	R_i	30
lateral sampling time	T_s^{lt}	0.01s
lateral state weights	$(q_{11}^{lt}, q_{22}^{lt}, q_{33}^{lt})$	(100, 10, 1)
lateral input weight	R^{lt}	20
velocity intervals	(v_1, v_2)	(1m/s, 2m/s)
steering constraint	δ_{max}	0.48rad

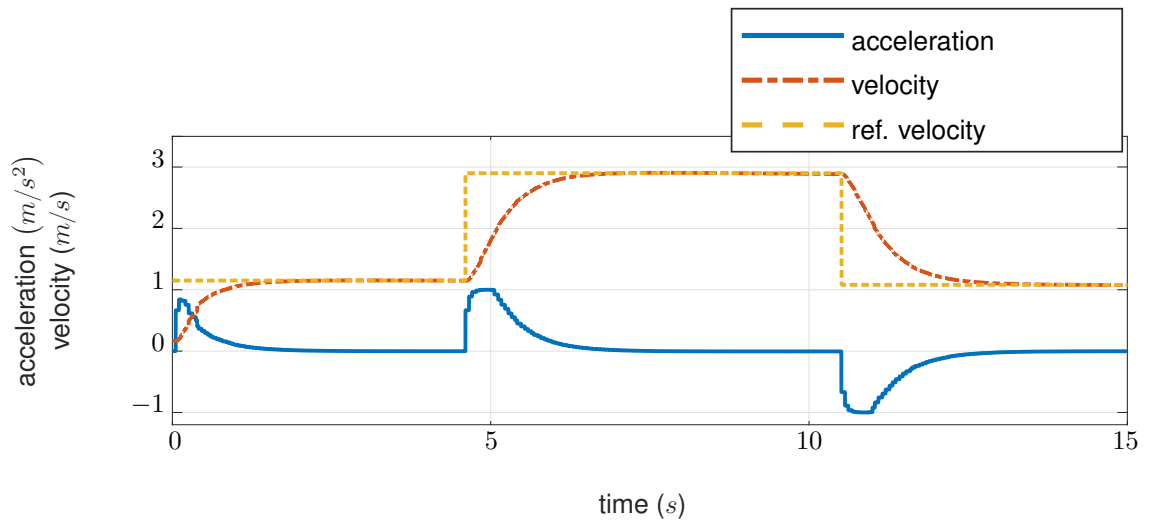


Figure 6.10: Step-response of velocity and acceleration from a single vehicle.

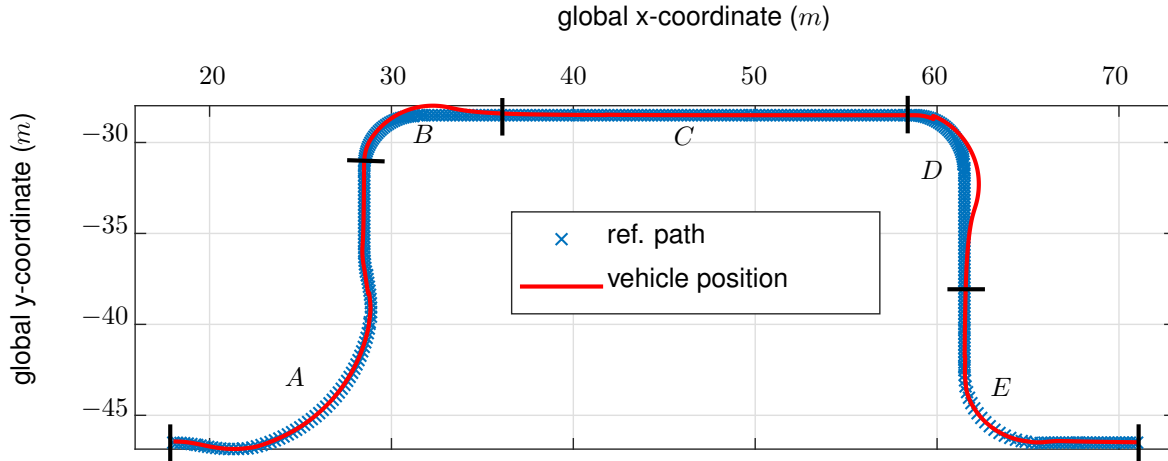


Figure 6.11: Measured vehicle position for tracking a given path and the resulting tracking error.

overshoot in the vehicle's velocity, enabling a safe inter-vehicular distance from possible successor vehicles.

In the next step, we evaluate the lateral tracking control. Figure 6.11 shows a reference path in the global coordinate system and the measured positions of a vehicle, while tracking this path. We find good tracking performance for kinematically feasible curves (segments A and E in Figure 6.11) and accurate path following for straight segments (segment C). To test the tracking control performance, we feed the system with an infeasible reference path. The evaluation shows still a stable and adequate tracking behavior for such references (segments B and D).

In order to illustrate the coordinated behavior of several vehicles in the distributed control framework, we simulate a platoon of homogeneous vehicles in the scenario introduced in Figure 6.9. Four vehicles (v_1, v_2, v_3, v_4) start from standstill in the area marked with a single * in Figure 6.9c. The resulting acceleration and velocity profiles of vehicles v_1, v_2 and v_4 are plotted in Figure 6.12. At the intersection, marked with **, vehicle v_3 leaves the platoon by turning right ($t = 27s$) to possibly park at a different location. In the further course, the vehicle adjacencies change in the PAM algorithm and thus vehicle v_4 has the incentive to reduce the distance to vehicle v_2 . The bottom plot of Figure 6.12 shows an approximately constant inter-vehicle distance of $\approx 4m$ between vehicles v_1 and v_2 . The upper line in the same plot illustrates the inter-vehicle distance between vehicles v_2 and v_4 , which is $\approx 9m$ (chassis dimensions excluded), when vehicle v_3 is in between them, and reduces after v_3 leaves the platoon.

Now, we evaluate the performance of the proposed distributed coordination control by simulating the coordinated parking processes. Vehicles $\mathcal{V} = \{v_1, v_2, \dots, v_7\}$ are randomly placed in the map as illustrated in Figure 6.9c, and the PAM algorithm randomly allocates a free parking spot to each vehicle, as well as a respective path. We simulate a set of five trials with different initial positions. Each trial runs with three different control methods. First, a solo-driving method is tested, where only one vehicle moves at a time. Second, an uncoordinated scenario is simulated. Each vehicle moves in an uncoordinated

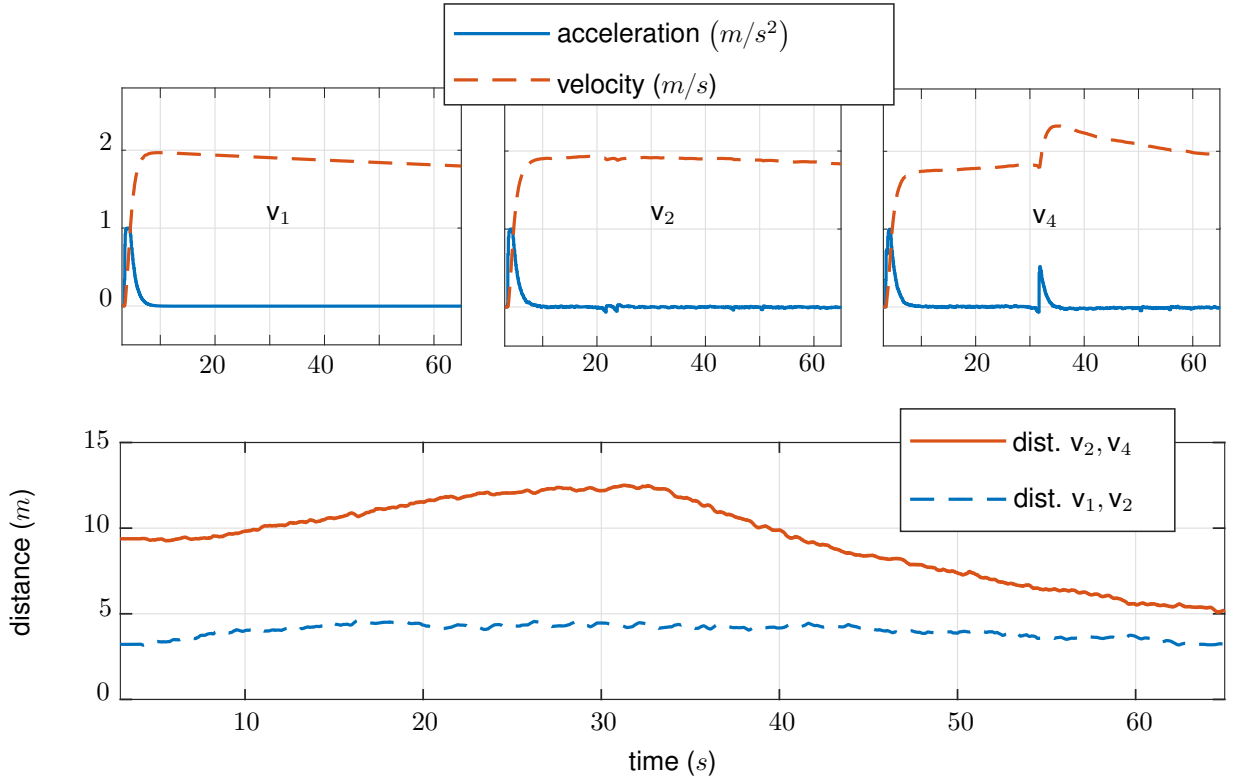


Figure 6.12: Coordinated driving in a platoon, where vehicle v_3 leaves the platoon $t = 27s$.

manner, and if a collision at an intersection would occur, the right-of-way is granted to the vehicle that arrives first. All other vehicles involved in the conflict have to stop. This method is supposed to mimic the behavior of human-driven cars in parking environments. The third method represents the coordinated control method proposed in this chapter. The vehicle sequence decisions (6.4) are computed using a first-come-first served (FCFS) heuristic based on the vehicles' distances to the respective conflict zones. The FCFS law is adapted such that infeasible sequences are detected and sorted out. The implementation of FCFS sequence decision is motivated by the low driving speeds in parking environments and the fact that no uncertainties are modeled in the presented simulations. Coordination messages, (6.8) and (6.10), between vehicles and the PAM unit are shared once in each sampling time step. As performance measures, we evaluate the required acceleration for each vehicle in a trial. To do so, we integrate the positive acceleration values of each vehicle over time ($\sum_t a_i(t) > 0$). Furthermore, we analyze the duration of the parking process, referred to as time to park (TTP). Figure 6.13 shows five trials and compares the acceleration effort of the three methods described above. Vehicles start from standstill at the beginning of each trial. The left bar of a set of three shows the solo-driving results, the middle bar is the uncoordinated scenario, and the right bar is the coordinated method. As the solo-driving method does not require any interaction, it has the lowest acceleration effort for each trial. We find that the coordinated method outperforms the uncoordinated method in most cases. In some cases, the coordinated algorithm is outperformed by the uncoordinated one if vehicles are granted the right-of-way (e.g. v_1 in trial 2 and v_7 in trial 5). However, the overall performance of all vehicles in the respective trial shows bet-

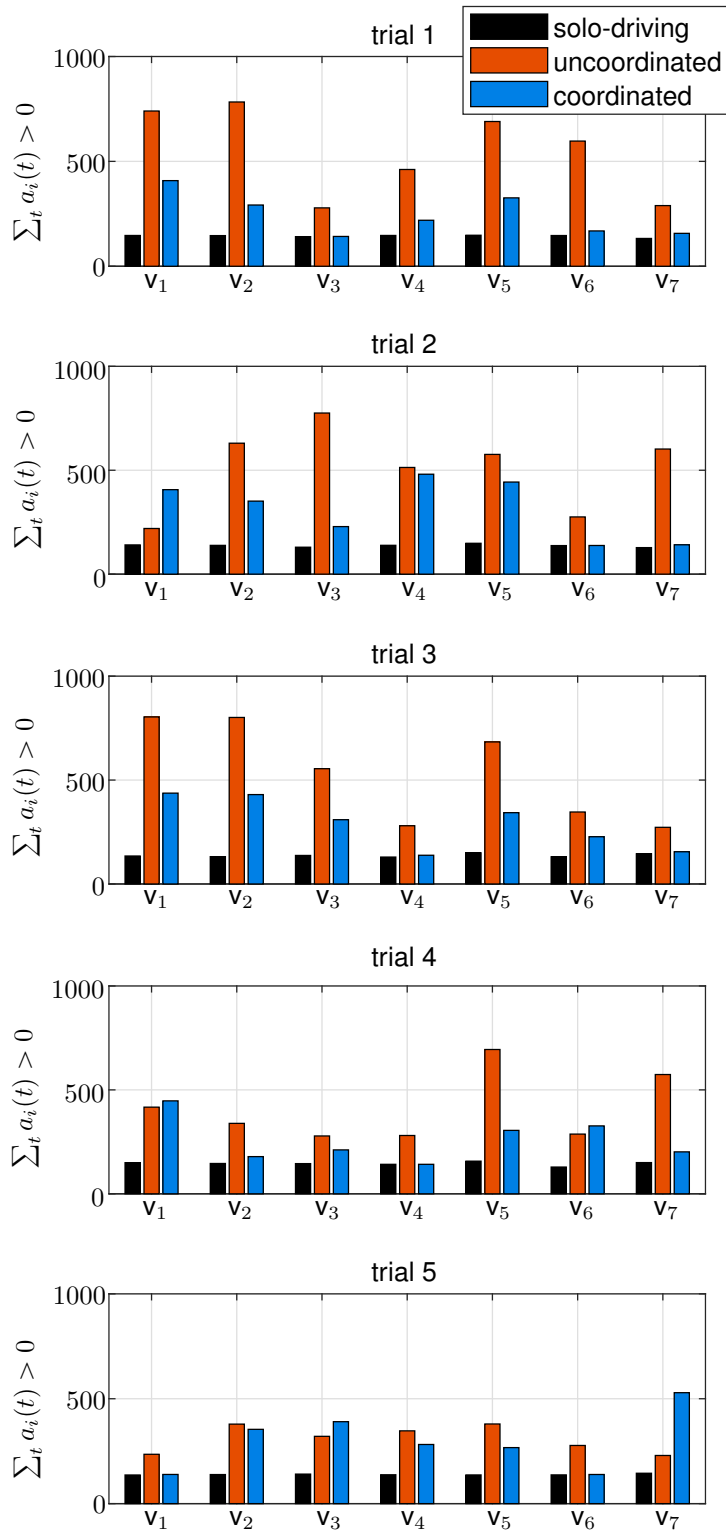


Figure 6.13: Performance comparison of different control coordination methods for 7 vehicles and 5 trials with different initial conditions.

Table 6.2: Number of required vehicle stops during simulation with the uncoordinated control method.

	trial 1	trial 2	trial 3	trial 4	trial 5
# stops	11	9	10	6	3

Table 6.3: Overall performance evaluation of the parking process with different control methods.

	solo-driving		control method			
	$TTP(s)$	Σacc	$TTP(s)$	Σacc	$TTP(s)$	Σacc
			trial 1			
Σ	334.33	1166.1	68.47	4257.5	52.82	2155.5
avg.	41.791	145.76	51.386	532.19	43.656	269.44
			trial 2			
Σ	366.08	1118	61.97	3782.7	61.4	2694.6
avg.	45.76	139.75	47.404	472.84	47.991	336.83
			trial 3			
Σ	385.67	1041.9	66.15	4318.1	62.91	2506.4
avg.	48.209	130.24	52.455	539.76	51.465	313.29
			trial 4			
Σ	391.27	1179.3	70.69	3134.7	62.64	2033.7
avg.	48.909	147.41	48.587	391.84	45.748	254.21
			trial 5			
Σ	280.86	1127.5	57.07	2482.9	51.88	2481
avg.	35.108	140.94	36.48	310.36	35.566	310.12

ter results using the coordinated method. In trial 5, the performance of the coordinated and uncoordinated methods is similar. This is due to the vehicles' initial positions, which only require a small amount of vehicle interaction. To evaluate the amount of interaction, Table 6.2 lists the number of vehicle stops during a simulation of the uncoordinated scenario. A higher number of stops using the uncoordinated control method requires a higher interaction amount using the coordinated control method.

Table 6.3 presents an overall evaluation for each trial, considering both the acceleration effort and the parking duration. Rows labeled Σ sum the acceleration integrals of all vehicles in columns with Σacc , and list the total time of the parking maneuvers in columns with TTP . The total time is the summation of the parking duration of all vehicles in the solo-driving method, as the vehicles are driven one by one. On the contrary, it is

Table 6.4: Control parameters in mixed-reality tests.

Longitudinal		Lateral	
Param.	Value	Param.	Value
T_s^{lg}	0.1s	T_s^{lt}	0.01s
τ	0.8s	C_f	81kN/rad
M	50	C_r	104kN/rad
v_{ref}	1.4m/s	J_z	581kgm ²
d_s	1.8m	l_f	1.1m
(v_{min}, v_{max})	(0m/s, 3m/s)	l_r	1.7m
(a_{min}, a_{max})	(-4m/s ² , 1m/s ²)	$\{v_a, v_b\}$	{1m/s, 2m/s}
Q^{lg}	diag(8, 6, 30)	Q^{lt}	diag(100, 10, 1)
R^{lg}	30	R^{lt}	400

the maximum value of the parking process duration among all simulated vehicles for the uncoordinated and coordinated methods, as the vehicles are driven simultaneously. Rows with *avg.* list the average values of all vehicles in the respective scenario. Due to its extensive total time to park, the solo-driving method is unrealistic for real-world application. Additionally, we find that the uncoordinated method is outperformed by the coordinated method in terms of parking duration and acceleration effort. This underlines the strength of the coordinated method, resulting in an optimized acceleration effort as well as a low time consumption.

6.2.3 Mixed-Reality Test Setup and Scenario

The mixed-reality experiments were conducted using a virtual model of a real parking garage, as well as parking lot with appropriate dimensions. In the scenario, we modeled three vehicles driving in the parking area, namely v_1 , v_2 , and v_3 . The test scenario and start positions are illustrated in Figure 6.14. Vehicle v_1 's goal is to park in the empty bay marked by the yellow box, while the other vehicles' routes are illustrated by the green paths in the figure. We find three resulting conflict zones (\mathcal{CZ} s) in the scenario, i.e., two intersections and one maneuvering zone. The conflict zones are used by the coordination procedure to ensure safe movements of the vehicles.

Table 6.4 summarizes the model parameters used for the experiments.

6.2.4 Mixed-Reality Results

In the first experiment we conduct a pure virtual simulation of the introduced parking scenario. This means the dynamics of all three vehicles are modeled by the virtual simulator. The coordination procedure decides a crossing order such that the right intersection in Figure 6.14 is first crossed by v_1 and then by v_2 , while v_1 crosses the left intersection after v_3 . In the same figure, we illustrate the result of the semi-structured path planning

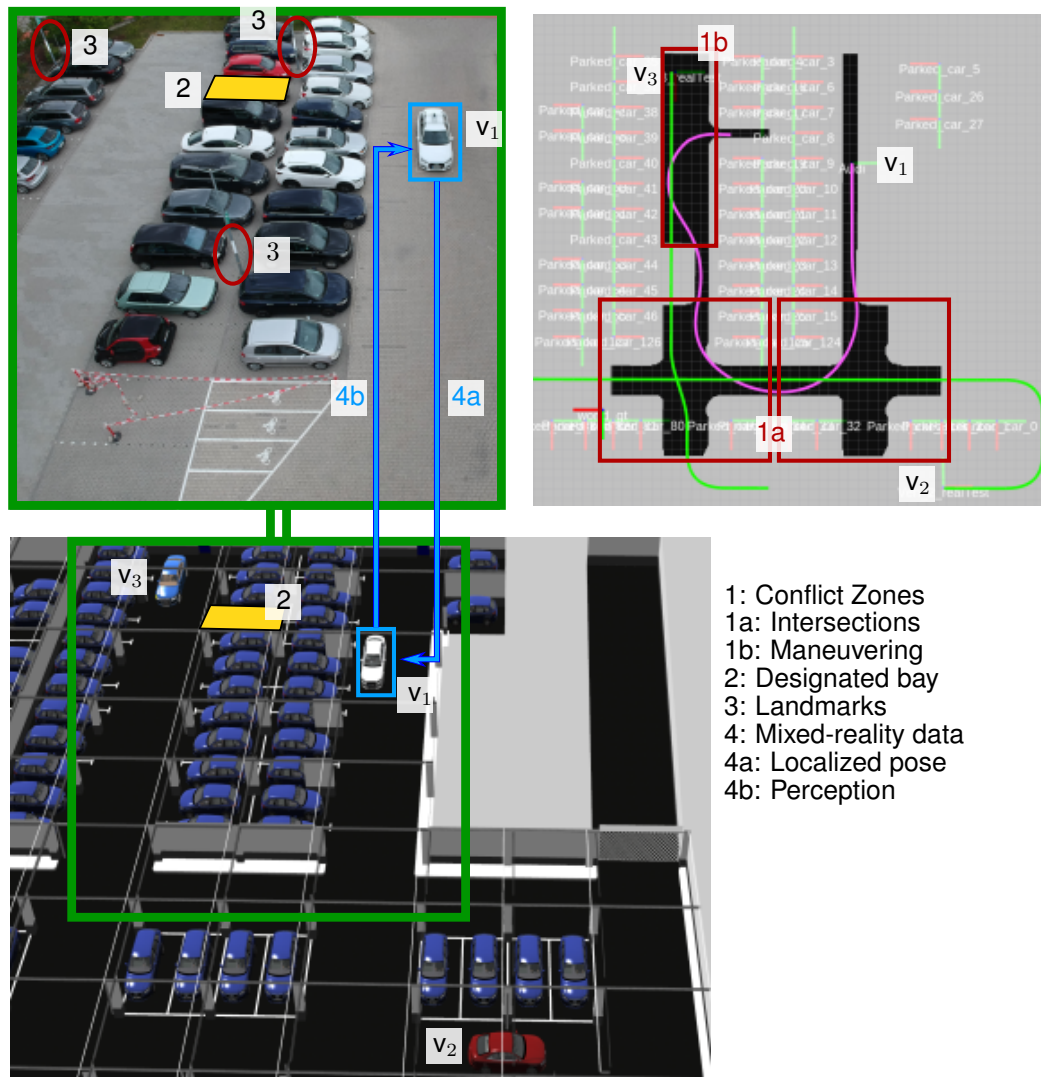


Figure 6.14: Mixed-reality test scenario.

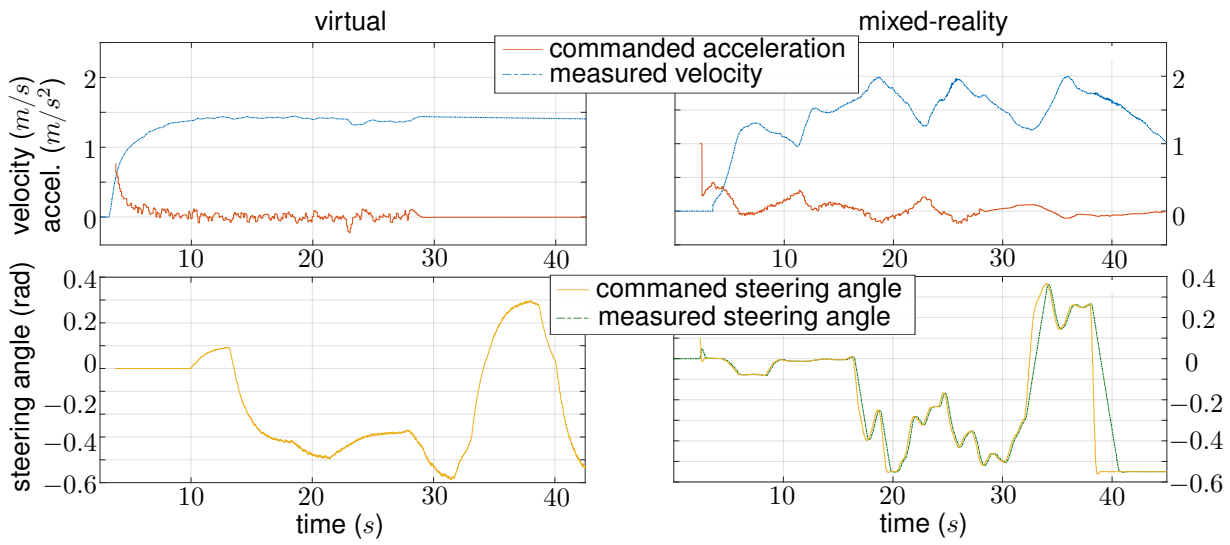


Figure 6.15: Longitudinal and lateral vehicle signals of vehicle v_1 for virtual simulation (left) vs. mixed-reality testing (right) in scenario of Figure 6.14. Top plots: acceleration control signals and resulting velocity; bottom plots: steering angle control input and actual applied steering signal for the real-vehicle case.

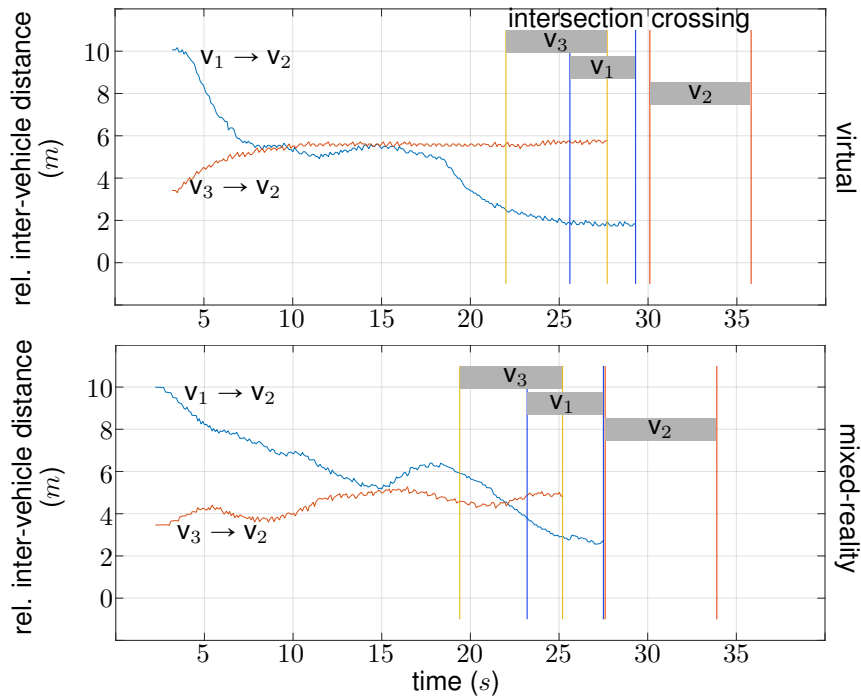


Figure 6.16: Relative inter-vehicle distances for virtual simulation (top) and mixed-reality simulation (bottom) w.r.t. the entrance of the left intersection area in Figure 6.14, as well as crossing times of vehicles through that intersection area.

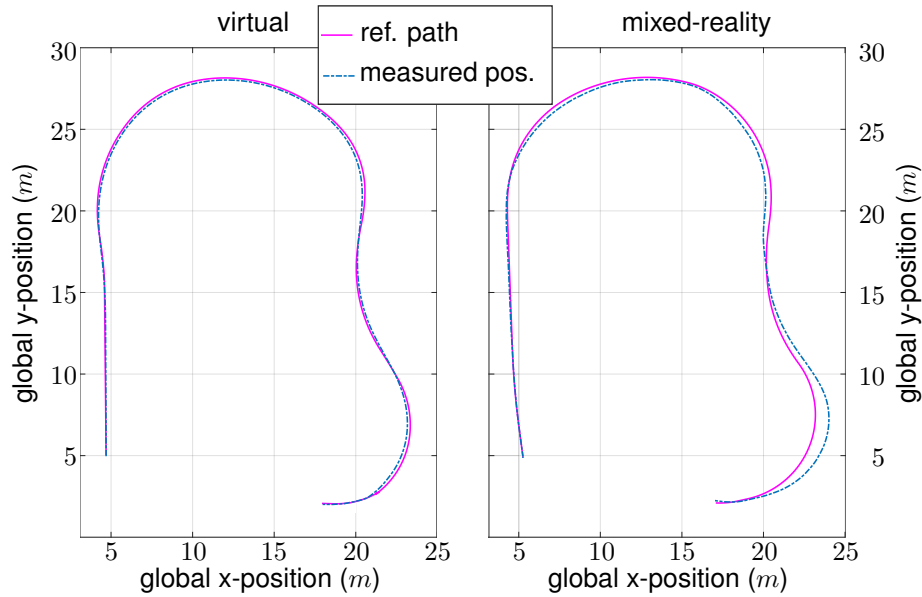


Figure 6.17: Planned reference path (solid) and actual tracking position (dashed) for vehicle v_1 of the scenario in Figure 6.14. Left: virtual simulation; right: mixed-reality simulation.

algorithm introduced in Section 6.1.1 for v_1 . Here we restrict the planner to forward parking maneuvers only (without reversing) as the main validation target is the functionality of the coordination procedure. The unstructured planning result for v_1 is illustrated by the purple lane in the top right plot of Figure 6.14 and in more detail in the left plot of Figure 6.17. The control signals and vehicle measurements of v_1 are plotted in the left plots of Figure 6.15 for the acceleration control command and measured velocity, as well as the steering commands. In the top left plot of Figure 6.15 we recognize a slightly noisy acceleration signal, which vanishes after 28s. This is where the inter-vehicle coupling is disbanded as vehicle v_1 has crossed the last intersection shared with other vehicles on its route. The noisy acceleration signal is expected as a result of the noisy inter-distance measurements (compare Figure 6.16). The lateral path-tracking performance is shown in the left plot of Figure 6.17. Finally, in the top plot of Figure 6.16, we investigate the relative vehicle distance of v_1 and v_2 with respect to the entrance of the left intersection area (Figure 6.14). This figure indicates the crossing intervals of the respective vehicles in the left intersection area. After a vehicle has left a common intersection area, the interaction with its following neighbor vehicle is cut, if vehicles continue on different lanes after the intersection.

In the second experiment, two vehicles are simulated virtual (v_2 and v_3), while one vehicle is a real test vehicle (v_1). This real vehicle is operating in a real-world parking lot which has similar dimensions to parts of the virtual model. By applying a self-localization, the real vehicle's poses can be projected into the virtual model, where the movements are displayed using a virtual twin of the real-vehicle. Start and goal positions are the same as described above. The control signals and vehicle measurements of the real vehicle v_1 are plotted on the right of Figure 6.15 for the acceleration control command and measured

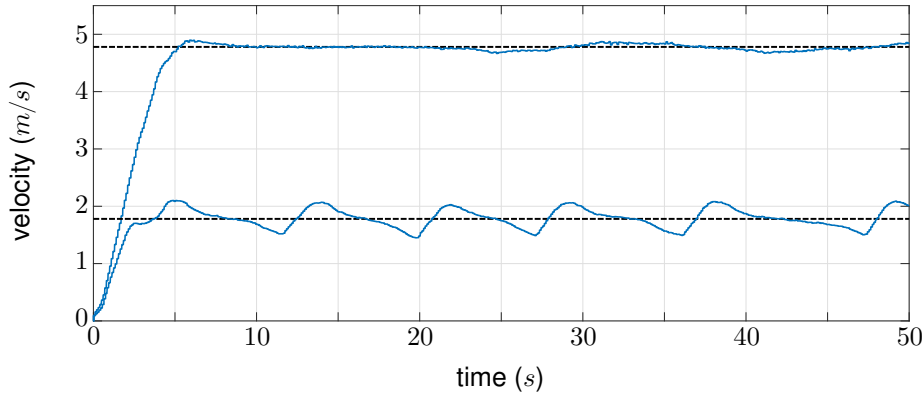


Figure 6.18: Measured velocity tracking performance for reference velocities $v_{ref} = 1.7m/s$ and $v_{ref} = 4.7m/s$.

velocity, as well as the commanded and actual applied steering commands in the bottom right of Figure 6.15. The lateral path-tracking performance for the real vehicle is shown on the right of Figure 6.17. The bottom plot of Figure 6.16 shows the relative vehicle distance of v_1 and v_2 for the mixed-reality test.

We find that the longitudinal acceleration signal in Figure 6.15 has a higher fluctuation compared to its virtual counterpart. The reason is an induced time-delay from opening and closing the vehicle’s clutch, which occur at the tested reference speed and is not considered by the model (6.11). However, this also illustrates the capability of the distributed coordination system to react to such disturbances, as the other vehicles in the scenario react accordingly (cf. Figure 6.16). Figure 6.18 validates the longitudinal MPC (6.12) for tracking the reference velocities $v_{ref} = 1.7m/s$ and $v_{ref} = 4.7m/s$. Increasing the reference velocity show that the clutch effects vanish, and that the proposed longitudinal vehicle model suffices for accurate velocity tracking.

In lateral tracking the right plot of Figure 6.17 shows a gap between the path reference and the actual tracked path, which results from a non modeled heading-rate limitation of the real vehicle’s steering controller. This is visualized in the end of the bottom right plot of Figure 6.15.

In summary, we find that the distributed control system is able to robustly compensate for non-modeled disturbances. At the same time, the mixed-reality testing setup provides a powerful environment to quickly judge the required accuracy of virtual models by comparing them to the real-world results.

6.3 Summary and Discussion

This chapter evaluates virtual and mixed-reality test results by applying distributed coordination methodologies into the test platforms from Chapter 5. A seamless Implementation and Validation process is demonstrated on the automated valet parking (AVP) use case. Results indicate a significant benefit through the coordinated control methods compared to non-coordinated scenarios. We apply a conflict zone coordination concept for the control coordination layer for automated intersection crossing by viewing the AVP scenario as a

multi-intersection scenario. The property that the coordination procedure is not limited to the local MPC controllers' prediction horizon, while it automatically reduces the number of coupling constraints between vehicles to a minimal set of safety conditions, enables its seamless extension from a single to a multi-intersection scenario. The coordination procedure has been implemented for a single information exchange per sampling time step. While in Chapter 3, it has been shown that multiple iterations within one sampling time step can improve the coordination performance, this would require additional synchronization effort by the applied middleware system. This cannot be seamlessly implemented in a ROS system and poses the need for alternative middleware implementations for such test-platforms.

Additionally, algorithms for layered and semi-structured path planning as well as lateral tracking control are introduced. The path planning methodology ensures a feasible realization regarding vehicle kinematics, enabling accurate tracking, and ensuring a safe coordination procedure in parking environments. The safety of the maneuvers, however, relies also on the tracking performance. The applied LQR controller has been tuned to fulfill the safety criteria in the tested scenarios. However, it cannot provide guarantees to meet the required tracking accuracy for all cases. Therefore, the extension to MPC tracking for longitudinal and lateral steering control could be considered in future work. However, the non-linearity of the steering kinematics and dynamics poses a challenge to obtain scalable computation problems.

Conclusions and Future Work

The increasing autonomy of cars toward fully autonomous driving is a tremendous challenge, but at the same time it poses the chance to improve current transportation circumstances significantly. Expected benefits of automated driving are safer systems with improved efficiency and comfort. One essential enabler of this technology will be the collaboration between vehicles. While the collaboration through wireless communication between vehicles is an important feature for efficiency, comfort, and safety, it imposes a non-neglectable risk for malfunctioning. Therefore, distributed control systems must preserve safety and privacy of the multi-vehicle coordination system, while maximizing the benefits of the connectivity.

7.1 Summary

This thesis aims to develop inter-vehicle collaboration strategies using distributed optimal control methods for the safe and efficient coordination of multiple vehicles. Essential emphasis is put on distributing the overall coordination problem to reduce the required inter-vehicle communication while considering further challenges and requirements of multi-agent systems. The challenges considered are Safety, Scalability, Validation and Implementation, Efficiency, Cooperation, and Privacy. In the following, each chapter is summarized.

Chapter 2

The first part discusses distributed control methodologies for the above-mentioned challenges. As a baseline, a modeling framework for multi-vehicle coordination is introduced. It suggests distinguishing different zones on a road network that gather areas of potential

vehicle collisions. These zones are used as system-wide common reference areas to relate optimization decisions of the distributed problems to each other. Vehicle trajectories are modeled concerning their longitudinal motions along pre-defined paths. Consequently, the common geometrical zone understanding can be applied in local optimization decisions to ensure inter-vehicle collision avoidance formally. Besides ensuring safe inter-vehicle decisions, a hierarchical envelope is defined on top of the distributed optimization framework. There are two resulting layers such that the lower bottom layer contains the distributed trajectory computation and the upper hierarchical optimization computes combinatoric decisions of the coordination problem. The combinatoric decision relates to the vehicle crossing order through the system-wide synchronized reference areas.

Chapter 3

The distributed trajectory computation in the lower layer is designed to be suitable for V2X (vehicle-to-everything) applications. A distributed Jacobi algorithm is modified such that it becomes applicable for multi-vehicle coordination problems. In an iterative procedure, primal trajectories resulting from distributed quadratic programs (QP) – one solved in each vehicle – are exchanged between neighboring vehicles. Each QP is formulated as a model predictive control (MPC) problem in which a set of hard constraints ensure the satisfaction of inter-vehicle safety distances. Finding a feasible solution to the local MPC problems results in safe network-wide coordination. A challenge for feasibility in the presented setup is that constraints between sub-systems are coupled. The proposed algorithm ensures feasibility for each sub-problem and each inter-sampling iteration step. The ability to stop the iterative process with a feasible solution after each step is an important property to enable distributed V2X applications. Another distinct feature of the proposed algorithm is that it can be fully distributed without requiring a central update step. To extend the results toward uncertain events, exact penalty functions are introduced on the coupling constraints. On the one hand, this leads to exact distributed trajectory solutions in nominal driving cases, and on the other hand safe and network-wide feasible maneuvers if unexpected events occur.

Chapter 4

The above described lower hierarchical layer ensures the feasibility of trajectories for vehicle networks with coupling constraints. The formulation of these constraints depends on how vehicles access shared zones on the road network and pose non-convexity in the optimization formulation. To overcome non-convexity, this decision is computed in the upper hierarchical layer in a central infrastructure located node. The overall knowledge of the central decision enables us to find deadlock-free sequence solutions. Simultaneously, a bidirectional coupling between layers ensures a feasible sequence decision. Technically, the upper layer decision can be formulated as an integer program (IP) that searches for an optimized solution for the overall problem. Through the bidirectional coupling, the IP's objective can be aligned with local MPC decisions to find close-to-optimal solutions. A resource-constrained project scheduling problem (RCPSP) formulation is proposed to cast multi-vehicle coordination problems into an IP formulation intuitively. To counteract the

scalability degradation through a centralized IP, the problem can be computed at a slower time-scale. Additionally, an event-triggered update rule is discussed, which intervenes only if previously proposed trajectory plans are violated according to a deviation measure.

Chapters 5 and 6

Virtual testing will be one of the key enablers to prove the safety of autonomous driving functions. Given this fact, the second part presents how methodologies from the first part can be seamlessly integrated and validated using virtual and semi-virtual simulation strategies. A modular platform for multi-vehicle scenarios is developed to enable the testing of distributed control algorithms. The components are connected to a high-fidelity virtual simulator. A test-management is utilized by generating random corner cases to achieve a fast failure elimination in the distributed coordination algorithms. Methodologies from the first part are tested on the use case of automated valet parking (AVP). Results show a safe and robust coordination procedure and an efficiency improvement through the proposed coordination algorithms compared to state-of-the-art parking procedures.

Lastly, the virtual system is extended toward a mixed-reality test system. One of the virtual simulated vehicles is synchronized with the movements of an avatar vehicle, which is a real test car driving on an empty test ground. Thus the real vehicle is interacting with simulated neighbor vehicles. This procedure poses an elegant way of testing the algorithmic behavior, including real vehicle dynamics, without automating the complete vehicle fleet.

Challenges and Requirements

In the following, it is summarized how the defined challenges are addressed throughout the thesis.

Safety is considered through the formulation of hard and coupling constraints in distributed optimization problems. The any-time feasibility guarantee of computations and the prioritization of safety over cooperation through exact penalty functions present important aspects for a safe multi-vehicle coordination procedure.

Scalability is guaranteed by the fully distributed DJOR computations, which are small enough to be embedded in local vehicle control units. Additionally, the upper hierarchical central sequence computation allows an adjustment of its sampling time such that an increasing computational burden can be counteracted. This problem is reduced to an IP, which additionally reduces the computational burden. Lastly, the test system development presents a scalable method for algorithmic validation, which is further supported by the idea of mixed-reality testing.

Validation and Implementation are demonstrated in the second part by implementing and evaluating the control algorithms on the use case automated valet parking applied into an automated and modular test platform.

Efficiency is addressed by formulating optimization objectives in both the local MPC problems and the central sequence decision problem. Thus, resulting trajectories present an efficiency trade-off between local vehicle interests and global traffic coordination interests.

Cooperation is considered because vehicles share trajectory intentions, negotiate in the frame of the distributed Jacobi over-relaxation procedure considering solutions of other vehicles, and adhere to sequence decisions from an upper hierarchical layer.

Privacy is ensured because local vehicle models are not required to be shared with other entities and that the proposed approach aims to keep the signal exchange at a low level. To this end, in the coordination architecture, only trajectory data from the vehicles is disclosed. Moreover, these trajectories are primarily shared with a “trustworthy” infrastructure node that enables the anonymization of transferred information.

7.2 Outlook

This thesis suggests distributed optimal control strategies that are in particular suitable for safety-critical scenarios with wireless inter-agent communication. The proposed derivations, e.g., the reduced inter-sampling iteration effort, play a key role in applying real-world multi-vehicle coordination tasks. However, several open research questions shall be addressed in order to extend and generalize presented methodologies.

Uncertainty effects

A strategy of accounting for unexpected events is proposed through exact penalty functions. Nevertheless, a useful extension is to consider uncertainties in a generalized way. Potential candidates for incorporating uncertainties in local control laws are scenario-based and robust control methods. An important direction is the consideration of uncertain prediction models in the distributed framework to consider “non-cooperative” agents such as human-driven vehicles. This requires adaptations in both the local control methods and the negotiation framework.

Improvements towards optimality

Through the Jacobi over-relaxation methodology, any-time feasibility of the inter-agent negotiation can be ensured. Moreover, the convergence of the iterative process can be guaranteed. However, the networked-system might converge toward a local minimum (Nash equilibrium) rather than the global optimal solution. An advantageous extension would be to research abilities to guide the distributed iterative process toward a globally optimal solution. The proposed architecture decomposition, where safety-relevant trajectory decisions in a lower-layer are decoupled from upper hierarchical sequence decisions, offers to apply learning-based methods in the upper-layer. This could enable us to make closer-to-optimal sequence decisions while reducing the online computational effort and preserving safety-guaranteed decisions.

Modeling extensions

In this thesis, linear models for vehicle control and coordination are used. Therefore, longitudinal and lateral models are separated, which is valid for low-velocity scenarios. An extension toward high-velocity coordination requires the consideration of coupled non-linear models. A potential candidate to apply this into the proposed MPC framework while preserving real-time applicability is sequential quadratic programming (SQP). This, in turn, requires an adaptation of the negotiation process and its guarantees. Additionally, time-invariant models have been applied. It would be worth investigating the performance gain through time-variant models within the coordination process.

Standardization efforts

There exists a wide range of different multi-vehicle coordination strategies. To make a step toward real-world applicability, it is required to provide necessary standards for collaborative autonomous driving. On the one hand, the vehicle data exchange formats on a trajectory level for multi-vehicle scenarios have to be standardized to enable the interplay between different entities from different manufacturers. On the other hand, a common understanding of spatial relation must be ensured between entities (conflict zones in this thesis). Who shall provide this information for different locations and settings, and in which format, are important questions to be answered through standardization efforts. Given a clear framework will, in turn, open further research questions from an algorithmic point of view. This thesis hopes to offer stimulating input for such standardization activities.

List of Figures

1.1	Exemplary autonomous and cooperative driving scenario	3
1.2	Relation between chapters	11
2.1	Illustration of the model predictive control strategy	18
2.2	Decomposition architectures for large scale control problems	19
2.3	Coordination scenario	22
2.4	Conflict zone example cases	24
2.5	Exemplary graph representations and relation	26
2.6	Distribution of the overall coordination problem	26
3.1	Feasible configuration space of a coordination problem	33
3.2	Distributed architecture and signal flow	35
3.3	Illustration of the over-relaxation concept	38
3.4	Example of a backward reach-set computation for a 2-state system	40
3.5	Illustration of terminal-state-less emulation method	40
3.6	Recursive feasibility concept	43
3.7	Illustration of an exact penalty example	48
3.8	Illustration of trajectories for varying horizon lengths	51
3.9	Effect of varying number of inter-sampling iterations	51
3.10	Emergency braking maneuver in distributed vehicle setup	53
3.11	Value of slack variable during emergency braking maneuver	53
4.1	Decision tree example showing possible sequences through a conflict zone	58
4.2	Intersection crossing scenario	60
4.3	Precedence graph representing the vehicles' routes through the intersection	61
4.4	Exemplary scheduling result indicating time and duration of execution for each modeled activity	63
4.5	Time-scale relation between control and scheduling problems	63
4.6	Control-scheduling interaction	66
4.7	Feasible trajectory space within the configuration space	67
4.8	Illustration of trigger set	68
4.9	Intersection simulation scenario with 6 vehicles	70
4.10	Example trial comparing the proposed scheduling method with a first-come-first-served decision strategy	71
4.11	Snapshots of simulation scenario	72
4.12	Evaluation of the scheduling performance in an example scenario	72

4.13	Illustration of the resulting vehicle coordination with and without re-scheduling	73
4.14	Evaluation of velocity and acceleration using the event-triggered re-scheduling setup	74
4.15	Evaluation of time duration and overall costs	74
4.16	Evaluation of the intersection performance simulation	75
5.1	Automated valet parking (AVP) scenario	83
5.2	Architecture of automated test system	84
5.3	Virtual test platform	85
5.4	Overview of the ROS network integration with connection to the simulation environment and the scenario generator	86
5.5	Mixed-reality test architecture	88
5.6	Schematic network diagram of the vehicle integration	89
6.1	Path planning concept for parking environments	92
6.2	Distributed control system architecture and signal flow of the automated valet parking system	95
6.3	Intersection area with all possible paths and conflict zones	95
6.4	Example scenario with six vehicles and pre-computed paths	97
6.5	Graph representation of the vehicles' routes	98
6.6	Illustration of the coupling constraint formulation	100
6.7	Single track vehicle model	102
6.8	Distributed implementation of the test system	104
6.9	A randomly generated test scenario	104
6.10	Step-response of velocity and acceleration from a single vehicle	105
6.11	Measured vehicle position for tracking a given path and the resulting tracking error	106
6.12	Coordinated driving in a platoon	107
6.13	Performance comparison of different control coordination methods	108
6.14	Mixed-reality test scenario	111
6.15	Longitudinal and lateral vehicle signals for virtual simulation vs. mixed-reality testing	112
6.16	Relative inter-vehicle distances for virtual simulation and mixed-reality simulations	112
6.17	Planned reference path and actual tracking position for virtual simulation and mixed-reality simulations	113
6.18	Measured velocity tracking performance for reference velocities	114

List of Tables

3.1	Time intervals for distance constraints	33
3.2	Simulation parameters in Subsec. 3.5.1	50
3.3	Simulation parameters in Subsec. 3.5.2	52
4.1	Intersection scheduling taxonomy	62
4.2	Simulation parameters	70
5.1	Most important ROS messages in the multi-vehicle control framework . . .	87
6.1	Control parameters in virtual tests	105
6.2	Number of required vehicle stops during simulation with the uncoordinated control method	109
6.3	Overall performance evaluation of the parking process with different control methods	109
6.4	Control parameters in mixed-reality tests	110

Bibliography

- [ADV16] H. Ahn and D. Del Vecchio, “Semi-autonomous intersection collision avoidance through job-shop scheduling,” in *Proc. 19th Int. Conf. Hybrid Systems: Computation and Control*, 2016, pp. 185–194.
- [AAGJ] A. Al Alam, A. Gattami, and K. H. Johansson, “An experimental study on the fuel reduction potential of heavy duty vehicle platooning,” in *2010 13th Int. IEEE Conf. Intelligent Transportation Systems (ITSC)*, pp. 306–311.
- [Ath68] M. Athans, “A unified approach to the vehicle-merging problem,” Massachusetts Institute of Technology, Electronic Systems Laboratory, Tech. Rep. ESL-P-336, 1968.
- [BJ07] H. Balakrishnan and Y. Jung, “A framework for coordinated surface operations planning at dallas-fort worth international airport,” in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6553.
- [BNKZ17] H. Banzhaf, D. Nienhüser, S. Knoop, and J. M. Zöllner, “The future of parking: A survey on automated valet parking with an outlook on high density parking,” in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1827–1834.
- [BQN+17] H. Banzhaf, F. Quedenfeld, D. Nienhueser, S. Knoop, and J. M. Zöllner, “High density valet parking using k-deques in driveways,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1413–1420.
- [BSBC13] E. Barker, M. Smid, D. Branstad, and S. Chokhani, “A framework for designing cryptographic key management systems,” *NIST Special Publication*, vol. 800, no. 130, pp. 1–112, 2013.
- [BBR+87] H. Berbee, C. Boender, A. R. Ran, C. Scheffer, R. L. Smith, and J. Telgen, “Hit-and-run algorithms for the identification of nonredundant linear inequalities,” *Mathematical Programming*, vol. 37, no. 2, pp. 184–207, 1987.
- [BT89] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

- [BVDEG16] J. J. Besa Vial, W. E. Devanny, D. Eppstein, and M. T. Goodrich, “Scheduling autonomous vehicle platoons through an unregulated intersection,” in *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, 2016.
- [BOA+12] T. Blochwitz, M. Otter, J. Akesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, and D. Neumerkel, “Functional mockup interface 2.0: The standard for tool independent exchange of simulation models,” in *Proc. 9th Int. Modelica Conf.*, 2012, pp. 173–184.
- [BOA+11] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, H. Elmqvist, A. Junghanns, J. Mauß, M. Monteiro, T. Neidhold, and D. Neumerkel, “The functional mockup interface for tool independent exchange of simulation models,” in *Proc. 8th Int. Modelica Conf.*, 2011, pp. 105–114.
- [BDM+99] P. Brucker, A. Drexler, R. Möhring, K. Neumann, and E. Pesch, “Resource-constrained project scheduling: Notation, classification, models, and methods,” *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999.
- [CFW+14] G. R. Campos, P. Falcone, H. Wymeersch, R. Hult, and J. Sjöberg, “Cooperative receding horizon conflict resolution at traffic intersections,” in *53rd IEEE Conf. Decision and Control*, 2014, pp. 2932–2937.
- [CMK+15] W. Cao, M. Mukai, T. Kawabe, H. Nishira, and N. Fujiki, “Cooperative vehicle path generation during merging using model predictive control with real-time optimization,” *Control Engineering Practice*, vol. 34, pp. 98–105, 2015.
- [CE15] L. Chen and C. Englund, “Cooperative intersection management: A survey,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 570–586, 2015.
- [OB94] N. M. de Oliveira and L. T. Biegler, “Constraint handling and stability properties of model-predictive control,” *AIChE journal*, vol. 40, no. 7, pp. 1138–1155, 1994.
- [Dij+59] E. W. Dijkstra *et al.*, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [DDKDS17] M. D. Doan, M. Diehl, T. Keviczky, and B. De Schutter, “A jacobi decomposition algorithm for distributed convex optimization in distributed model predictive control,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4905–4911, 2017.
- [DKDS11] M. D. Doan, T. Keviczky, and B. De Schutter, “A distributed optimization-based approach for hierarchical mpc of large-scale systems with coupled dynamics and constraints,” in *2011 50th IEEE Conf. Decision and Control and European Control Conference*, 2011, pp. 5236–5241.
- [DTMD10] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Path planning for autonomous vehicles in unknown semi-structured environments,” *The Int. Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

-
- [DTMD08] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Practical search techniques in path planning for autonomous driving,” in *1st Int. Symp. Search Techniques in Artificial Intelligence and Robotics*, 2008.
- [DS08] K. Dresner and P. Stone, “A multiagent approach to autonomous intersection management,” *Journal of artificial intelligence research*, vol. 31, pp. 591–656, 2008.
- [DC12] W. B. Dunbar and D. S. Caveney, “Distributed receding horizon control of vehicle platoons: Stability and string stability,” *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 620–633, 2012.
- [DM06] W. B. Dunbar and R. M. Murray, “Distributed receding horizon control for multi-vehicle formation stabilization,” *Automatica*, vol. 42, no. 4, pp. 549–558, 2006.
- [DSG10] M. Dupuis, M. Strobl, and H. Grezlikowski, “Opendrive 2010 and beyond—status and future of the de facto standard for the description of road networks,” in *Driving Simulation Conf. Europe*, 2010, pp. 231–242.
- [EKM19] D. Elliott, W. Keen, and L. Miao, “Recent advances in connected and automated vehicles,” *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 6, no. 2, pp. 109–131, 2019.
- [ERT19] ERTRAC Working Group “Connectivity and Automated Driving”, “Connected automated driving roadmap,” ERTRAC, Tech. Rep., 2019. [Online]. Available: <https://www.ertrac.org/uploads/documentsearch/id57/ERTRAC-CAD-Roadmap-2019.pdf> (visited on 11/30/2020).
- [EKM+20] H. Esen, M. Kneissl, A. Molin, S. vom Dorff, B. Böddeker, E. Möhlmann, U. Brockmeyer, T. Teige, G. G. Padilla, and S. Kalisvaart, “Validation of automated valet parking,” in *Validation and Verification of Automated Systems*, Springer, Cham, 2020, pp. 207–220.
- [FS12] M. Farina and R. Scattolini, “Distributed predictive control: A non-cooperative algorithm with neighbor-to-neighbor communication for linear systems,” *Automatica*, vol. 48, no. 6, pp. 1088–1096, 2012.
- [Fen70] R. E. Fenton, “Automatic vehicle guidance and control—a state of the art survey,” *IEEE Trans. Veh. Technol.*, vol. 19, no. 1, pp. 153–161, 1970.
- [FNKAM18] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, “Coordination of multiple vessels via distributed nonlinear model predictive control,” in *2018 European Control Conference (ECC)*, 2018, pp. 2523–2528.
- [FKP+14] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, “qpOASES: A parametric active-set algorithm for quadratic programming,” *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.
- [Fle87] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 1987.

- [FHG+12] C.-L. Fok, M. Hanna, S. Gee, T.-C. Au, P. Stone, C. Julien, and S. Vishwanath, “A platform for evaluating autonomous intersection management policies,” in *2012 IEEE/ACM 3rd Int. Conf. Cyber-Physical Systems*, 2012, pp. 87–96.
- [FBEG16] J. Funke, M. Brown, S. M. Ertien, and J. C. Gerdes, “Collision avoidance and stabilization for autonomous vehicles in emergency scenarios,” *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1204–1216, 2016.
- [GAAJT11] A. Gattami, A. Al Alam, K. H. Johansson, and C. J. Tomlin, “Establishing safety for heavy duty vehicle platooning: A game theoretical approach,” *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 3818–3823, 2011.
- [GDGK14] F. Gechter, B. Dafflon, P. Gruer, and A. Koukam, “Towards a hybrid real/virtual simulation of autonomous vehicles for critical scenarios,” in *The 6th International Conference on Advances in System Simulation (SIMUL 2014)*, 2014.
- [GK95] A. Grama and V. Kumar, “Parallel search algorithms for discrete optimization problems,” *ORSA Journal on Computing*, vol. 7, no. 4, pp. 365–385, 1995.
- [GBDLF14] J. Gregoire, S. Bonnabel, and A. De La Fortelle, “Priority-based coordination of robots,” 2014. HAL: hal-00828976.
- [HAS11] M. Hausknecht, T.-C. Au, and P. Stone, “Autonomous intersection management: Multi-intersection optimization,” in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, IEEE, 2011, pp. 4581–4586.
- [Hov11] M. Hovd, “Multi-level programming for designing penalty functions for mpc controllers,” *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6098–6103, 2011.
- [HWLZ16] W. Huang, K. Wang, Y. Lv, and F. Zhu, “Autonomous vehicles testing methods review,” in *2016 IEEE 19th Int. Conf. Intelligent Transportation Systems (ITSC)*, 2016, pp. 163–168.
- [Hul19] R. Hult, “Optimization-based coordination strategies for connected and autonomous vehicles,” PhD thesis, Chalmers Tekniska Hogskola, Gothenburg, Sweden, 2019.
- [HZGF16] R. Hult, M. Zanon, S. Gros, and P. Falcone, “Primal decomposition of the optimal coordination of vehicles at traffic intersections,” in *2016 IEEE 55th Conf. Decision and Control (CDC)*, 2016, pp. 2567–2573.
- [HZGF18] R. Hult, M. Zanon, S. Gros, and P. Falcone, “An miqp-based heuristic for optimal coordination of vehicles at intersections,” in *2018 IEEE Conf. Decision and Control (CDC)*, 2018, pp. 2783–2790.
- [HZGF19] R. Hult, M. Zanon, S. Gros, and P. Falcone, “Optimal coordination of automated vehicles at intersections with turns,” in *18th European Control Conference (ECC)*, 2019, pp. 225–230.

-
- [HZG+20] R. Hult, M. Zanon, S. Gros, H. Wymeersch, and P. Falcone, "Optimisation-based coordination of connected, automated vehicles at intersections," *Vehicle System Dynamics*, vol. 58, no. 5, pp. 726–747, 2020.
- [IEE17] IEEE 5G Initiative Technology Roadmap Working Group, "Ieee 5g and beyond technology roadmap," *White Paper*, 2017. [Online]. Available: <https://futurenetworks.ieee.org/roadmap/roadmap-white-paper> (visited on 11/30/2020).
- [JD08] D. Jiang and L. Delgrossi, "Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments," in *VTC Spring 2008-IEEE Vehicular Technology Conf.*, 2008, pp. 2036–2040.
- [JHM+10] Y. Jung, T. Hoang, J. Montoya, G. Gupta, W. Malik, and L. Tobias, "A concept and implementation of optimized operations of airport surface traffic," in *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conf.*, 2010, p. 9213.
- [KP16] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.
- [KG15] N. R. Kapania and J. C. Gerdes, "Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling," *Vehicle System Dynamics*, vol. 53, no. 12, pp. 1687–1704, 2015.
- [KKJ17] A. Katriniok, P. Kleibaum, and M. Joševski, "Distributed model predictive control for intersection automation using a parallelized optimization approach," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5940–5946, 2017.
- [KSSP19] A. Katriniok, P. Sopasakis, M. Schuurmans, and P. Patrinos, "Nonlinear model predictive control for distributed motion planning in road intersections using panoc," in *2019 IEEE 58th Conf. Decision and Control (CDC)*, 2019, pp. 5272–5278.
- [KC11] P. Kavathekar and Y. Chen, "Vehicle platooning: A brief survey and categorization," in *Int. Design Engineering Technical Conf. and Computers and Information in Engineering Conf.*, vol. 54808, 2011, pp. 829–845.
- [KG87] S. Keerthi and E. Gilbert, "Computation of minimum-time feedback control laws for discrete-time systems with state-control constraints," *IEEE Trans. Autom. Control*, vol. 32, no. 5, pp. 432–435, 1987.
- [KM00] E. C. Kerrigan and J. M. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," in *UKACC Int. Conf. (Control 2000)*, Cambridge, 2000.
- [KK17] T. Kessler and A. Knoll, "Multi vehicle trajectory coordination for automated parking," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 661–666.

- [KAE+12] R. Kianfar, B. Augusto, A. Ebadighajari, U. Hakeem, J. Nilsson, A. Raza, R. S. Tabar, N. V. Irukulapati, C. Englund, P. Falcone, *et al.*, “Design and experimental validation of a cooperative driving system in the grand cooperative driving challenge,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 994–1007, 2012.
- [KK14] K.-D. Kim and P. R. Kumar, “An mpc-based approach to provable system-wide safety and liveness of autonomous ground traffic,” *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3341–3356, 2014.
- [Kle+16] S. Klemm *et al.*, “Autonomous multi-story navigation for valet parking,” in *2016 IEEE 19th Int. Conf. Intelligent Transportation Systems (ITSC)*, 2016, pp. 1126–1133.
- [KMEH19] M. Kneissl, A. Molin, H. Esen, and S. Hirche, “A one-step feasible negotiation algorithm for distributed trajectory generation of autonomous vehicles,” in *2019 IEEE 58th Conf. Decision and Control (CDC)*, 2019, pp. 6687–6693.
- [KMK+20] M. Kneissl, A. Molin, S. Kehr, H. Esen, and S. Hirche, “Combined scheduling and control design for the coordination of automated vehicles at intersections,” in *21st IFAC World Congress*, 2020.
- [KMM+20] M. Kneissl, A. K. Madhusudhanan, A. Molin, H. Esen, and S. Hirche, “A multi-vehicle control framework with application to automated valet parking,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 9, pp. 5697–5707, 2020.
- [KMEH18] M. Kneissl, A. Molin, H. Esen, and S. Hirche, “A feasible mpc-based negotiation algorithm for automated intersection crossing,” in *2018 European Control Conference (ECC)*, 2018, pp. 1282–1288.
- [KMEH20] M. Kneissl, A. Molin, H. Esen, and S. Hirche, “Any-time feasible coordination in a distributed multi-vehicle setup,” *IEEE Trans. Control Syst. Technol.*, 2020, submitted.
- [KDM+20] M. Kneissl, S. vom Dorff, A. Molin, M. Denniel, T. Son, N. Ochoa, H. Esen, and S. Hirche, “Mixed-reality testing of multi-vehicle coordination in an automated valet parking environment,” in *21st IFAC World Congress*, 2020.
- [LADK06] T. Le-Anh and M. De Koster, “A review of design and control of automated guided vehicle systems,” *European Journal of Operational Research*, vol. 171, no. 1, pp. 1–23, 2006.
- [LHX+18] J. Lei, Q. Han, Y. Xu, F. Li, R. Chen, and L. Chen, “Semi-virtual test for icvs in automotive emc laboratory,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 43–48.
- [LWIG19] A. Leitner, D. Watzenig, and J. Ibanez-Guzma, *Validation and Verification of Automated Systems*. Springer, 2019.

-
- [LZ17] P. T. Li and X. Zhou, “Recasting and optimizing intersection automation as a connected-and-automated-vehicle (cav) scheduling problem: A sequential branch-and-bound search approach in phase-time-traffic hypernetwork,” *Transportation Research Part B: Methodological*, vol. 105, pp. 479–506, 2017.
- [LJM19] Y. Li, K. H. Johansson, and J. Mårtensson, “A hierarchical control system for smart parking lots with automated vehicles: Improve efficiency by leveraging prediction of human drivers,” in *2019 18th European Control Conference (ECC)*, 2019, pp. 2675–2681.
- [LW55] M. J. Lighthill and G. B. Whitham, “On kinematic waves ii. a theory of traffic flow on long crowded roads,” *Proc. of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 229, no. 1178, pp. 317–345, 1955.
- [LH03] X.-Y. Lu and J. K. Hedrick, “Longitudinal control algorithm for automated vehicle merging,” *Int. Journal of Control*, vol. 76, no. 2, pp. 193–202, 2003.
- [LTSH04] X.-Y. Lu, H.-S. Tan, S. E. Shladover, and J. K. Hedrick, “Automated vehicle merging maneuver implementation for ahs,” *Vehicle System Dynamics*, vol. 41, no. 2, pp. 85–107, 2004.
- [MG13] L. Makarem and D. Gillet, “Model predictive coordination of autonomous vehicles crossing intersections,” in *16th Int. IEEE Conf. Intelligent Transportation Systems (ITSC)*, 2013, pp. 1799–1804.
- [MG12] L. Makarem and D. Gillet, “Fluent coordination of autonomous vehicles at intersections,” in *2012 IEEE Int. Conf. Systems, Man, and Cybernetics (SMC)*, 2012, pp. 2557–2562.
- [MSMCP+09] D. Martín-Sacristán, J. F. Monserrat, J. Cabrejas-Penuelas, D. Calabuig, S. Garrigas, and N. Cardona, “On the way towards fourth-generation mobile: 3gpp lte and lte-advanced,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, no. 1, p. 354 089, 2009.
- [MSS+14] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, “Cooperative adaptive cruise control in real traffic situations,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 296–305, 2014.
- [MC13] K.-W. Min and J.-D. Choi, “Design and implementation of autonomous vehicle valet parking system,” in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, IEEE, 2013, pp. 2082–2087.
- [NRTT97] R. Naumann, R. Rasche, J. Tacke, and C. Tahedi, “Validation and simulation of a decentralized intersection collision avoidance algorithm,” in *Proc. IEEE Conf. Intelligent Transportation Systems (ITSC)*, 1997, pp. 818–823.

- [NRT98] R. Naumann, R. Rasche, and J. Tacken, “Managing autonomous vehicles at intersections,” *IEEE Intelligent Systems and their Applications*, vol. 13, no. 3, pp. 82–86, 1998.
- [NNC19] G. Notarstefano, I. Notarnicola, and A. Camisa, “Distributed optimization for smart cyber-physical networks,” *arXiv preprint arXiv:1906.10760*, 2019.
- [Pac05] H. Pacejka, *Tire and vehicle dynamics*. Elsevier, 2005.
- [PPMR95] P. M. Pardalos, L. Pitsoulis, T Mavridou, and M. G. Resende, “Parallel search for combinatorial optimization: Genetic algorithms, simulated annealing, tabu search and grasp,” in *Int. Workshop Parallel Algorithms for Irregularly Structured Problems*, Springer, 1995, pp. 317–331.
- [PSSF19] C. Pasquale, S. Sacone, S. Siri, and A. Ferrara, “Traffic control for freeway networks with sustainability-related objectives: Review and future challenges,” *Annual Reviews in Control*, vol. 48, pp. 312–324, 2019.
- [PS10] S Paulraj and P Sumathi, “A comparative study of redundant constraints identification methods in linear programming problems,” *Mathematical Problems in Engineering*, 2010.
- [Pay71] H. J. Payne, “Model of freeway traffic and control,” *Mathematical Model of Public System*, pp. 51–61, 1971.
- [Pol18] P. Polack, “Consistency and stability of hierarchical planning and control systems for autonomous driving,” PhD thesis, Paris Institute of Technology, Paris, France, 2018.
- [PANLF18] P. Polack, F. Alché, B. d’Andréa Novel, and A. de La Fortelle, “Guaranteeing consistency in a motion planning and control architecture using a kinematic bicycle model,” in *2018 Annu. American Control Conference (ACC)*, 2018, pp. 3981–3987.
- [PN00] J. A. Primbs and V. Nevistić, “Feasibility and stability of constrained finite receding horizon control,” *Automatica*, vol. 36, no. 7, pp. 965–971, 2000.
- [PWW69] A. A. B. Pritsker, L. J. Waiters, and P. M. Wolfe, “Multiproject scheduling with limited resources: A zero-one programming approach,” *Management science*, vol. 16, no. 1, pp. 93–108, 1969.
- [QGDLFM15] X. Qian, J. Gregoire, A. De La Fortelle, and F. Moutarde, “Decentralized model predictive control for smooth coordination of automated vehicles at intersection,” in *2015 European Control Conference (ECC)*, 2015, pp. 3452–3458.
- [Qui+09] M. Quigley *et al.*, “Ros: An open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, 2009, p. 5.
- [QAZ+10] M. Quinlan, T.-C. Au, J. Zhu, N. Sturca, and P. Stone, “Bringing simulation to life: A mixed reality autonomous intersection,” in *2010 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2010, pp. 6083–6088.

-
- [RS01] R Rajamani and S. E. Shladover, “An experimental comparative study of autonomous and co-operative vehicle-follower control systems,” *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 1, pp. 15–31, 2001.
- [Raj11] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.
- [RM09] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Nob Hill Pub., 2009.
- [RH04] A. Richards and J. How, “A decentralized algorithm for robust constrained model predictive control,” in *Proc. 2004 American control conference (ACC)*, vol. 5, 2004, pp. 4261–4266.
- [Ric56] P. I. Richards, “Shock waves on the highway,” *Operations research*, vol. 4, no. 1, pp. 42–51, 1956.
- [RTM16] J. Rios-Torres and A. A. Malikopoulos, “A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1066–1077, 2016.
- [SB16] S. Sadraddini and C. Belta, “A provably correct mpc approach to safety control of urban traffic networks,” in *American Control Conference (ACC), 2016*, IEEE, 2016, pp. 1679–1684.
- [SSB16] G. Schildbach, M. Soppert, and F. Borrelli, “A collision avoidance system at intersections using robust model predictive control,” in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 233–238.
- [Sch+16] U. Schwesinger *et al.*, “Automated valet parking and charging for e-mobility,” in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 157–164.
- [SR99] P. O. Scokaert and J. B. Rawlings, “Feasibility issues in linear model predictive control,” *AICHE Journal*, vol. 45, no. 8, pp. 1649–1659, 1999.
- [SSSS17] S. Shalev-Shwartz, S. Shammah, and A. Shashua, “On a formal model of safe and scalable self-driving cars,” *arXiv preprint arXiv:1708.06374*, 2017.
- [SD93] S. Sheikholeslam and C. A. Desoer, “Longitudinal control of a platoon of vehicles with no communication of lead vehicle information: A system level study,” *IEEE Trans. Veh. Technol.*, vol. 42, no. 4, pp. 546–554, 1993.
- [SZB20] X. Shen, X. Zhang, and F. Borrelli, “Autonomous parking of vehicle fleet in tight environments,” in *2020 American Control Conference (ACC)*, 2020, pp. 3035–3040.
- [Soc16] Society for Automotive Engineering, “Definitions for terms related to driving automation systems for on-road motor vehicles,” Tech. Rep. J3016, 2016. [Online]. Available: https://www.sae.org/standards/content/j3016_201806/ (visited on 11/30/2020).

- [SDJ16] L. E. Sokoler, P. J. Dinesen, and J. B. Jørgensen, “A hierarchical algorithm for integrated scheduling and control with applications to power systems,” *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 2, pp. 590–599, 2016.
- [SBA19] T. D. Son, A. Bhave, and H. Van der Auweraer, “Simulation-based testing framework for autonomous driving development,” in *IEEE 2019 Int. Conf. Mechatronics (ICM)*, vol. 1, 2019, pp. 576–583.
- [SHA+18] T. D. Son, J. Hubrechts, L. Awatsu, A. Bhave, and H. Van der Auweraer, “A simulation-based testing and validation framework for adas development,” in *7th Transport Research Arena (TRA)*, 2018.
- [SSS00] S. S. Stankovic, M. J. Stanojevic, and D. D. Siljak, “Decentralized overlapping control of a platoon of vehicles,” *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 5, pp. 816–832, 2000.
- [SKBB+18] F. Steck, V. Kolarova, F. Bahamonde-Birke, S. Trommer, and B. Lenz, “How autonomous driving may affect the value of travel time savings for commuting,” *Transportation research record*, vol. 2672, no. 46, pp. 11–20, 2018.
- [SHZ+18] E. Steinmetz, R. Hult, Z. Zou, R. Emardson, F. Brännström, P. Falcone, and H. Wymeersch, “Collision-aware communication for intersection management of automated vehicles,” *IEEE access*, vol. 6, pp. 77 359–77 371, 2018.
- [SZS+15] J. E. Stellet, M. R. Zofka, J. Schumacher, T. Schamm, F. Niewels, and J. M. Zöllner, “Testing of advanced driver assistance towards automated driving: A survey and taxonomy on existing approaches and open questions,” in *2015 IEEE 18th Int. Conf. Intelligent Transportation Systems (ITSC)*, 2015, pp. 1455–1462.
- [SVR+10] B. T. Stewart, A. N. Venkat, J. B. Rawlings, S. J. Wright, and G. Pannocchia, “Cooperative distributed model predictive control,” *Systems & Control Letters*, vol. 59, no. 8, pp. 460–469, 2010.
- [TBF05] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.
- [TFBW15] J. Timpner, S. Friedrichs, J. van Balen, and L. Wolf, “K-stacks: High-density valet parking for automated vehicles,” in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 895–900.
- [TBS14] M. Tlig, O. Buffet, and O. Simonin, “Stop-free strategies for traffic networks: Decentralized on-line optimization,” in *ECAI*, 2014, pp. 1191–1196.
- [UST99] A. Uno, T. Sakaguchi, and S. Tsugawa, “A merging control algorithm based on inter-vehicle communication,” in *Proc. 1999 IEEE/IEEJ/JSAI Int. Conf. Intelligent Transportation Systems (ITSC)*, 1999, pp. 783–787.
- [VE03] A. Vahidi and A. Eskandarian, “Research advances in intelligent collision avoidance and adaptive cruise control,” *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 3, pp. 143–153, 2003.

-
- [VAVDV06] B. Van Arem, C. J. Van Driel, and R. Visser, “The impact of cooperative adaptive cruise control on traffic-flow characteristics,” *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 429–436, 2006.
- [Wac17] W. H. K. Wachenfeld, “How stochastic can help to introduce automated driving,” PhD thesis, Technische Universität Darmstadt, 2017.
- [WLZL15] J.-Q. Wang, S. E. Li, Y. Zheng, and X.-Y. Lu, “Longitudinal collision mitigation via coordinated braking of multiple vehicles using model predictive control,” *Integrated Computer-Aided Engineering*, vol. 22, no. 2, pp. 171–185, 2015.
- [Whi11] G. B. Whitham, *Linear and nonlinear waves*. John Wiley & Sons, 2011, vol. 42.
- [WLFM19] H. Winner, K. Lemmer, T. Form, and J. Mazzega, “Pegasus - first steps for the safe introduction of automated driving,” in *Road Vehicle Automation 5*, Springer, 2019, pp. 185–195.
- [Wor18] World Health Organization, “Global status report on road safety 2018,” Tech. Rep., 2018. [Online]. Available: <https://www.who.int/publications/i/item/9789241565684> (visited on 11/30/2020).
- [WATEM12] J. Wu, A. Abbas-Turki, and A. El Moudni, “Cooperative driving: An ant colony system for autonomous intersection management,” *Applied Intelligence*, vol. 37, no. 2, pp. 207–222, 2012.
- [XG10] L. Xiao and F. Gao, “A comprehensive review of the development of adaptive cruise control systems,” *Vehicle System Dynamics*, vol. 48, no. 10, pp. 1167–1192, 2010.
- [YDM11] F. Yan, M. Dridi, and A. Moudni, “A scheduling approach for autonomous vehicle sequencing problem at multi-intersections,” *International Journal of Operations Research*, vol. 9, no. 1, 2011.
- [ZGWF17] M. Zanon, S. Gros, H. Wymeersch, and P. Falcone, “An asynchronous algorithm for optimal vehicle coordination at traffic intersections,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 008–12 014, 2017.
- [ZMC16] Y. J. Zhang, A. A. Malikopoulos, and C. G. Cassandras, “Optimal control and coordination of connected and automated vehicles at urban traffic intersections,” in *2016 American Control Conference (ACC)*, 2016, pp. 6227–6232.
- [Zhe16] H. Zheng, “Coordination of waterborne agvs,” PhD thesis, Delft University of Technology, Delft, Netherlands, 2016.
- [ZLL+16] Y. Zheng, S. E. Li, K. Li, F. Borrelli, and J. K. Hedrick, “Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies,” *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 899–910, 2016.

- [ZEF+18] M. R. Zofka, M. Essinger, T. Fleck, R. Kohlhaas, and J. M. Zöllner, “The sleepwalker framework: Verification and validation of autonomous vehicles by mixed reality lidar stimulation,” in *2018 IEEE Int. Conf. Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2018, pp. 151–157.