A Multi-Vehicle Control Framework with Application to Automated Valet Parking

Maximilian Kneissl, Anil K. Madhusudhanan, Adam Molin, Hasan Esen, and Sandra Hirche, *Fellow, IEEE*

***Abstract*—We introduce a distributed control method for coordinating multiple vehicles in the framework of an automated valet parking (AVP) system. The control functionality is distributed between an infrastructure server, called parking area management (PAM) system, and local autonomous vehicle control units. Via a vehicle-to-infrastructure (V2I) communication interface, model predictive control (MPC) decisions of the vehicles are shared with the coordination unit in the PAM. This unit in turn computes a coupling feedback which is shared with the vehicles. The control system is integrated in an automated test-system to cope with the high test requirements and short development cycles of highly automated systems. Evaluations conducted with the test-system show the functionality of the proposed distributed control method for multi-vehicle coordination. Results indicate safe coordination, and an efficiency increase compared to an uncoordinated method in an AVP simulation environment.**

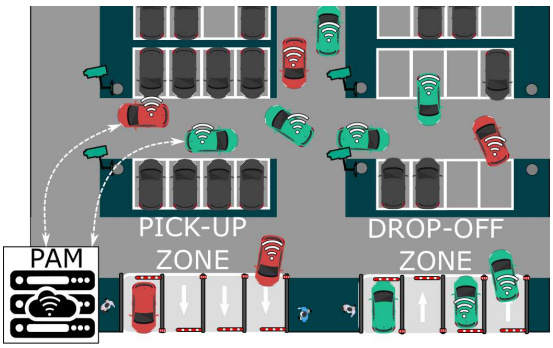


Fig. 1: Automated valet parking scenario with a parking area management (PAM) server and multiple coordinated autonomous vehicles with V2I communication capabilities.

I. Introduction

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) system to take control in order to autonomously guide the vehicle through the parking area and into a designated parking bay [1]. AVP is expected to be one of the first scenarios of automated driving satisfying the SAE (Society of Automotive Engineers) Level 4 requirements [2]. Level 4 autonomous driving defines cases where vehicles are able to move without any driver interaction in specific scenarios. Such systems will be available at airports, shopping malls, and at large parking areas next to places of interest.

Related works on AVP have covered the field of parking space optimization based on k-stacks [3], [4], and the security of user information [5]. Another important problem is the localization and mapping task in GPS-denied parking environments [6], and the motion planning problem. The survey in [7] presents an overview of motion planning for AVP systems considering global and local planning tasks, whereby important contributions are provided by the V-CHARGE research project [8]. However, to the best of our knowledge, none of the approaches in the literature discusses motion planning for several vehicles moving simultaneously in an AVP environment.

In order to guarantee safe and efficient movements for AVP, we developed a distributed control method for multi-vehicle coordination on predefined routes. This article focuses on the coordination phase after a target parking slot [3], [4], and a respective path [7] have been computed. The coordination task is decomposed into a local vehicle control and a global coordination task. Both units - the local vehicle control and the control coordination in the PAM infrastructure - are connected via vehicle-to-infrastructure (V2I) communication. The coordination unit determines dependencies for the exchange of control data between vehicles by constructing appropriate safety constraints for local vehicle problems, whereas the local vehicle control is responsible for efficient movements, path tracking, and the fulfillment of the safety constraints. For longitudinal and lateral control, we apply distributed model predictive control (MPC) and gain-scheduled linear-quadratic-regulators (LQR), respectively. Longitudinal predictions, resulting from each vehicle's local MPC problem, are shared with the coordination unit. The latter collects the predictions, and a data set is forwarded to the respective neighboring vehicles which will process it in the next planning step. Fig. 1 shows an exemplary AVP scenario for multi-vehicle coordination.

Due to its environmental setup, the problem of coordinating several vehicles within parking areas is strongly related to the multi-intersection coordination scenario. The challenge of guiding vehicles autonomously through intersections has been investigated in [9] by using a reservation scheme, and an extension to the multi-intersection problem is proposed in [10]. A focus on finding optimal or quasi-optimal solutions for intersection crossing from a control perspective is, for example, proposed in [11]–[13]. Therein, vehicles negotiate a solution while approaching the intersection, and consecutively cross the intersection-zone safely and efficiently. A drawback, however, is that only a single vehicle at a time

This work has been 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.

can be in the intersection zone. The studies described in [14], [15] suggest distinguishing between several geometric regions within a single intersection, thus reducing the conservatism. It focuses on the verification of a deadlock-free intersection management algorithm. Similarly, the authors of [16] define points in the intersection area where vehicles' paths intersect and compute an MPC-based control solution. We extended the distinction of such geometrical regions to a multi-intersection formulation. In [17] trajectories are computed for crossing two consecutive intersections using optimal control with a varying finite time of the local vehicle optimization problems. However, to guarantee a feasible solving time for embedded implementations, it is beneficial to know the optimization effort (horizon length) prior to the problem implementation. MPC is a widely used method for vehicle coordination problems, as the finite optimization problem finds close-to-optimal solutions, and constraints can be considered in the problem formulation [13], [16], [18]–[23]. However, a bottleneck of some intersection coordination problems is the requirement that the MPC horizon length has to cover the intersection area, when negotiating the solution [16], [24]. Thus, such approaches are not scalable for a multi-intersection scenario.

To overcome these challenges, we propose a distance-based formulation of the coordination problem that enables a smooth extension from a single intersection coordination problem to a complete network of intersections. Therefore, in our distributed formulation, the concept of virtual platooning [25]–[27] is adapted to the MPC problem, which combines inter-vehicle distance keeping with the intersection crossing scenario. Another distinct feature of the proposed approach is that the coordination procedure is not limited to the prediction horizon of the local MPC controllers, while it automatically reduces the number of coupling constraints between vehicles to a minimal set of safety conditions for the local MPC problem.

Guaranteeing a safe operation of highly automated vehicles is one of the major challenges, and is thus subject of projects such as PEGASUS [28] and ENABLE-S3 [29]. An unfeasible amount of test kilometers is required to ensure the safety of such systems. To overcome this challenge, one trend is to move a large portion of the required testing into high-fidelity virtual simulation environments. This not only enables testing specifically challenging corner cases, but also provides the possibility of automating the testing process. Therefore, it is essential to provide a framework for distributed control applications that is suitable for being embedded in an automated test-system. We propose such an implementation, which can be seamlessly integration into an automated test-system, where agents can be added to or removed from the distributed system the in a plug-and-play manner. Furthermore, changes in the environmental setup can be considered in a flexible way.

The remainder of the paper is organized as follows. Section II introduces the three layers of the complete automated test-system for validating AVP. The system under test (SUT) in the lowest layer of the test-system is the focus of this paper. The control design within the SUT is discussed in detail in Section III, where the infrastructure and autonomous vehicle functions are explained. The implementation of the control system into the automated test-system is discussed in Section IV and an evaluation is conducted in an automotive simulation environment in Section V. The paper is concluded in Section VI.

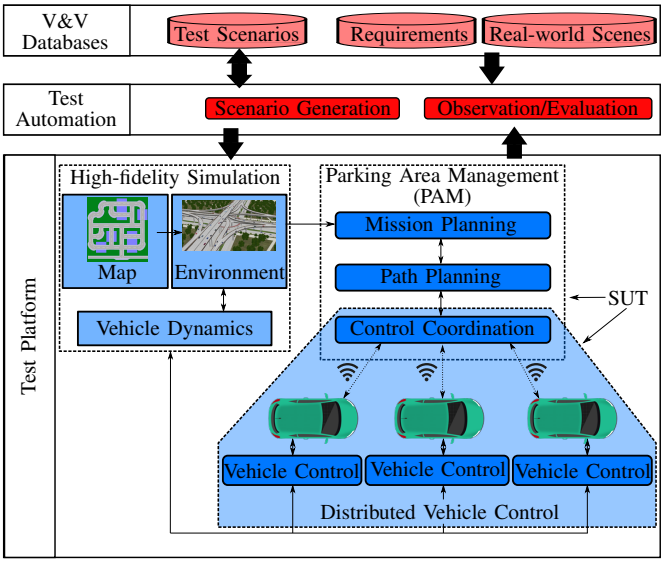


Fig. 2: Architecture of automated test-system.

II. Valet Parking Test System Framework

This section presents a simplified overview of the test architecture used in the ENABLE-S3 project, illustrated in Fig. 2.

The top level verification and validation (V&V) databases contain scenarios to be tested, requirements which should be fulfilled during the trials, and a real-world scenes database.

The test automation layer is responsible for running the tests 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. 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, and the test is passed if all the requirements were met by the SUT. A final evaluation classifies the conducted test run according to its degree of reality by comparing it with real-world scenes.

The bottom layer defines the test platform, which is responsible for simulating the parking processes for the provided scenarios. It consists of a high-fidelity simulation environment and the SUT, which is a combination of the PAM system and the distributed autonomous vehicle control systems. The high-fidelity virtual environment simulates the behavior of all vehicles and possibly other moving objects based on their respective dynamic models. The simulation uses an environment model, which is provided by the scenario generation tool, and consists of map data as well as a visual database. Furthermore, the virtual simulator provides a visualization of the tested scenario, and the option of modeling sensor behaviors, e.g. LiDAR sensors, in the environment. This paper focuses on the

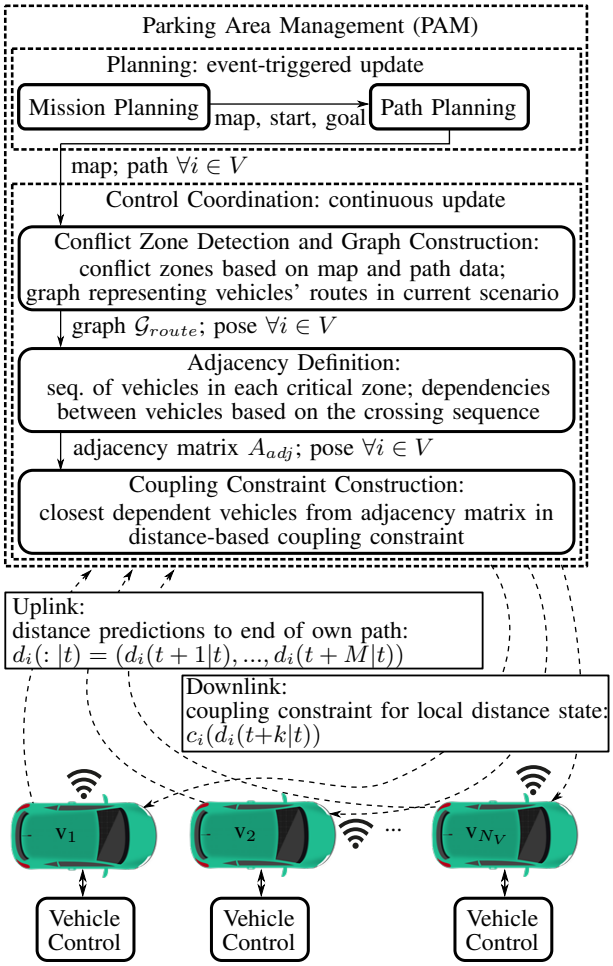


Fig. 3: Distributed control system architecture and signal flow of the automated valet parking system.

test platform layer and especially on the control components in the SUT, which is described in the following section.

III. Distributed Control System Design

This section describes the distributed control system of the AVP system. Fig. 3 summarizes the functionality of the sub-components and illustrates the signal flow between them. A detailed description of the PAM functions is given in Section III-A, whilst the autonomous vehicle control functions as well as their couplings are presented in Section III-B.

A. PAM Infrastructure Functions

Similar to [30], we assume separation of the planning phase into *mission planning*, *path planning*, and *trajectory planning*. The outputs of the Mission Planning and Path Planning are assumed to be predefined. The former organizes the parking lot situation in the parking environment and determines the destination positions for each vehicle, while the latter computes a path, i.e. geometric waypoints, provided with the map data, and each vehicles' start and target position. The path and map data are then shared with the control coordination unit. This unit cooperatively determines longitudinal trajectories, which will be tracked by the respective vehicle's low level

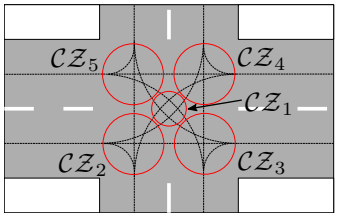


Fig. 4: Intersection area with all possible paths and *conflict zones* $\{\mathcal{CZ}_1, ..., \mathcal{CZ}_5\}$.

controller. A crucial property is the feasibility of a vehicle's trajectory because safety of a scenario can only be ensured if a vehicle is able to track its computed trajectory. To achieve this requirement, we propose a decomposition of the trajectory generation process between the PAM infrastructure and the local vehicle controller. This subsection describes the control coordination procedure computed in the PAM. Based on predefined vehicle paths and trajectory predictions from local vehicle control units a vehicle adjacency will be determined with the help 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:

1) Conflict Zone Detection: Intersections, i.e. points where vehicle paths cross, are a bottleneck in coordinating vehicles within the given parking environment. This is because not only a control decision has to be computed between consecutive vehicles, but also a combinatorial decision on the vehicle order. By making use of a central coordination unit, we separate the combinatorial decision, which we refer to as scheduling, from the MPC problem in the vehicles.

Fig. 4 shows possible paths in an exemplary intersection area. 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 named *conflict zones* $\mathcal{CZ}_i$ where $i \in \mathbb{I}_{1:5}$. Notation $\mathbb{I}_{a:b}$ describes a set of integers $a, b \in \mathbb{N}$ such that the integers in the set $\{a, a+1, a+2, ..., b\}$ are obtained. The conflict zones of all intersections are gathered in the set $\mathcal{I}_{inter} = \{\mathcal{CZ}_j, ..., \mathcal{CZ}_{5N_I}\}$, where N_I is the total number of intersections in the parking environment, and $j \in \mathbb{N}$ a unique ID label for each conflict zone. 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 $|\mathcal{I}_{end}| = N_V$ and N_V indicates the total number of moving vehicles in a coordinated parking scenario. Note that $|\mathcal{I}_{end}|$ means the cardinality of its argument set $\mathcal{I}_{end}$. Finally, we gather all these conflict zones in the set

$$\mathcal{I} = \mathcal{I}_{inter} \cup \mathcal{I}_{end}. \tag{1}$$

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. Furthermore, we assume that paths are tracked accurately such that a vehicle's bounding 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 1 (Conflict zones): The set in (1) can be extended by further zones, $\mathcal{CZ}_i$s, at arbitrary positions where collisions between vehicles can occur. This is for example the case if a vehicle crosses an accommodating lane to navigate into a parking bay. The coordination procedure is not affected by this action. Therefore, we refrain from explaining this part, for the sake of simplicity of presentation.

2) Graph Representation: The set

$$V = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_{N_V}\} \tag{2}$$

contains all vehicles that are coordinated by the PAM, where each vehicle $\mathrm{v}_i \in V$ 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 $\mathcal{I}, V$, and $\mathcal{G}_{route}$ are continuously updated to cope with changes, such as vehicle poses and picked-up or newly dropped-off vehicles.

3) Adjacency Definition: The resulting $\mathcal{G}_{route}$ is used to determine a set of crossing orders

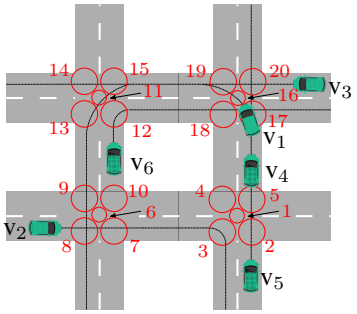
$$\mathcal{O} = \{o_i = (\mathrm{v}_j, \mathrm{v}_k, ...) \mid \forall i \in \mathbb{I}_{1:|\mathcal{I}|}; j, k \in \mathbb{I}_{1:N_V}\}. \tag{3}$$

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 (Subsection III-B). The scheduling orders in (3) are determined on a first-come-first-served basis, where the sequences through the conflict zones in an intersection area are sorted according the vehicles' distances to the respective intersection. Alternative heuristics to generate such scheduling orders on road networks are discussed in [31].

Finally, we determine the adjacency matrix $A_{adj} = (a_{ij})^{N_V \times N_V}$, with $a_{ij} \in \mathcal{T}$, and

$$\mathcal{T} = \{(m_1, ..., m_p, ..., m_q) \cup \emptyset \mid q \in \mathbb{N}, \forall p \in \{1, ..., k\} : m_p \in \mathcal{I}\}. \tag{4}$$

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 $i, j \in \mathbb{I}_{1:N_V}$, 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. $a_{ij} = (\mathcal{CZ}_k, \mathcal{CZ}_l, ...)$ 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.



$o_8 = (\mathrm{v}_2, \mathrm{v}_1)$

$o_{15} = (\mathrm{v}_3, \mathrm{v}_1)$

$o_{17} = (\mathrm{v}_1, \mathrm{v}_4, \mathrm{v}_6, \mathrm{v}_5)$

$o_{19} = (\mathrm{v}_3, \mathrm{v}_1)$

$o_{20} = (\mathrm{v}_3, \mathrm{v}_4, \mathrm{v}_5)$

Fig. 5: Example scenario with six vehicles and pre-computed paths. Labels at circles are the conflict zone IDs. Scheduling orders for affected conflict zones are shown on the right.

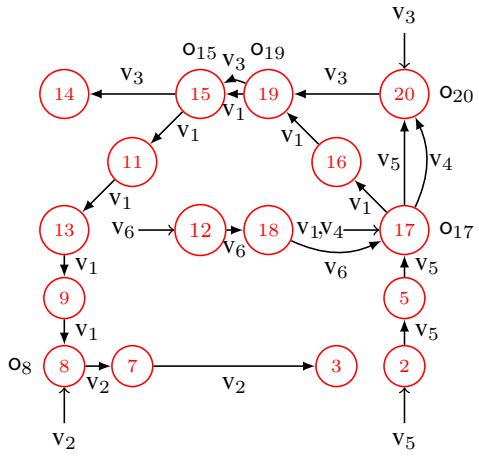


Fig. 6: Graph representation ($\mathcal{G}_{route}$) of the multi-vehicle scenario in Fig. 5.

Illustrative Example 1: In Fig. 5 we present an example coordination scenario with the vehicle set

$$V = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_6\} \tag{5}$$

and four intersections leading to the conflict zones

$$\mathcal{I} = \{\mathcal{CZ}_1, \mathcal{CZ}_2, \ldots, \mathcal{CZ}_{20}\}. \tag{6}$$

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 Fig. 6. 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 Fig. 5 and finally, the scenario's adjacency matrix is given as follows:

$$A_{adj} = \begin{array}{c} \\ \mathrm{v}_1 \\ \mathrm{v}_2 \\ \mathrm{v}_3 \\ \mathrm{v}_4 \\ \mathrm{v}_5 \\ \mathrm{v}_6 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathrm{v}_1 & \mathrm{v}_2 & \mathrm{v}_3 & \mathrm{v}_4 & \mathrm{v}_5 & \mathrm{v}_6 \end{array} \\ \begin{pmatrix} - & \mathcal{CZ}_8 & \mathcal{CZ}_{19,15} & & & \\ & - & & & & \\ & & - & & & \\ \mathcal{CZ}_{17} & & \mathcal{CZ}_{20} & - & & \\ & & & \mathcal{CZ}_{20} & - & \mathcal{CZ}_{17} \\ & & & \mathcal{CZ}_{17} & & - \end{pmatrix} \end{array}.$$

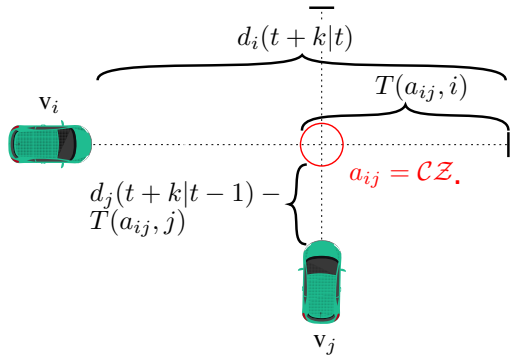


Fig. 7: Illustration of the coupling constraint formulation using an example with one conflict zone $\mathcal{CZ}_i$, and two vehicles approaching form different directions (dotted lines).

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, $\mathrm{v}_i \in V$, shares the prediction vector of its distance state,

$$d_i(:|t) = (d_i(t+1|t), d_i(t+2|t), ..., d_i(t+M|t)) \in \mathbb{R}^M, \tag{7}$$

with the PAM. The state d_i is the vehicle's distance to the end of its path, and M is the prediction length. The notation $d_i(t+k|t)$ means the predicted value of state d_i for time step $t+k$, with $0 \leq k \leq M$, at the discrete time step t. 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 $\mathrm{v}_i \in V$ is calculated by determining the most critical predecessor vehicle with

$$j_k^* = \underset{j \in \mathbb{I}_{1:N_V}}{\operatorname{argmin}} \{d_i(t+k|t-1) - [T(a_{ij},i) + d_j(t+k|t-1) - T(a_{ij},j)]\}, \tag{8}$$

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 (8), the coupling constraint is given by

$$c_i(d_i(t+k|t)) = d_i(t+k|t) - [T(a_{ij_k^*},i) + d_{j_k^*}(t+k|t-1) - T(a_{ij_k^*}, j_k^*)], \tag{9}$$

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, $t-1$. Fig. 7 illustrates the coupling constraint formulation in Equation (8) and (9) for a determined predecessor v_j of vehicle v_i with respect to the conflict zone a_{ij} at one prediction time step $t+k$. Note that the procedure is the same if vehicles approach the conflict zone from the same lane.

B. Autonomous Vehicle Functions

The local vehicle controller is decomposed into a longitudinal control and lateral control. In this subsection, we describe the implementation of the high-level control, whose output will be passed to the low-level control units to ensure the hardware actuation, i.e. acceleration (throttle and brake), and steering.

1) Longitudinal Motion Model and Control Design: A vehicle v_i is modeled using the states d_i, v_i, and a_i. 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:

$$\underbrace{\begin{pmatrix}\dot{d}_i\\ \dot{v}_i\\ \dot{a}_i\end{pmatrix}}_{\dot{x}_{c,i}^{lg}} = \underbrace{\begin{pmatrix}0 & -1 & 0\\ 0 & 0 & 1\\ 0 & 0 & -\frac{1}{\tau_i}\end{pmatrix}}_{A_{c,i}^{lg}} \underbrace{\begin{pmatrix}d_i\\ v_i\\ a_i\end{pmatrix}}_{x_{c,i}^{lg}} + \underbrace{\begin{pmatrix}0\\ 0\\ \frac{1}{\tau_i}\end{pmatrix}}_{B_{c,i}^{lg}} u_i, \tag{10}$$

where τ_i is a time constant for throttle and brake actuation, and u_i is an exogenous control input, which represents the desired acceleration [32]–[34].

The longitudinal MPC law is computed solving the following optimization problem

$$L_i^*\left(x_i^{lg}(t), u_i(t)\right) = \tag{11a}$$

$$\min_{U_i} \sum_{k=1}^{M} \|\Delta_r x_i^{lg}(t+k|t)\|_{Q_i}^2 + \sum_{k=0}^{M-1} \|u_i(t+k|t)\|_{R_i}^2$$

s.t.

$$x_i^{lg}(t+k+1|t) = A_i^{lg} x_i^{lg}(t+k|t) + B_i^{lg} u_i(t+k) \qquad k \in \mathbb{I}_{0:M-1} \tag{11b}$$

$$x_i(t|t) = x_i(t) \tag{11c}$$

$$v_{min} \leq v_i(t+k|t) \leq v_{max} \qquad k \in \mathbb{I}_{1:M} \tag{11d}$$

$$a_{i,min} \leq a_i(t+k|t) \leq a_{i,max} \qquad k \in \mathbb{I}_{1:M} \tag{11e}$$

$$c_i\left(d_i(t+k|t)\right) \geq d_s \qquad k \in \mathbb{I}_{1:M}. \tag{11f}$$

Here, $x_i(t+k|t)$ is the state prediction for time $t+k$ at the current discrete time step t. The model (11b) is attained by discretizing (10) using a sampling time T_s^{lg}. The optimized variables vector is of the form $U_i = (u_i(t), u_i(t+1), ..., u_i(t+M-1)) \in \mathbb{R}^M$ with the horizon length M. After optimizing (11) in each time step t, the control law $U_i(1) = u_i(t)$ determines the desired acceleration input to the vehicle, and the procedure is repeated in consecutive time steps in a receding horizon fashion. The objective function in (11a) consists of a state error cost and a control input cost. The longitudinal reference vector in the state error vector, $\Delta_r x_i^{lg}(t+k|t) = x_i^{lg}(t+k|t) - x_{i,ref}^{lg}$, is of form

$$x_{i,ref}^{lg} = (d_{i,ref}, v_{ref}, 0)^T, \tag{12}$$

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

$$d_{i,ref} = \begin{cases} 0 & \text{if } \mathrm{v}_i \text{ has no predecessor} \\ d_s + d_{slack} & \text{if } \mathrm{v}_i \text{ has a predecessor.} \end{cases} \tag{13}$$

If the reference distance in (13) becomes active, 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. In (11a), we use the notation $\|\alpha\|_P^2 \triangleq \alpha^T P \alpha$ with a vector $\alpha \in$

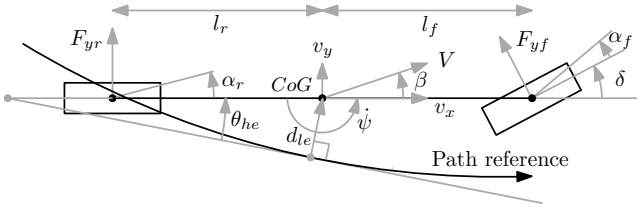


Fig. 8: Single track vehicle model, showing the vehicle model variables [36] and the lateral distance and heading angle errors with respect to the path reference [35].

$\mathbb{R}^n$ and a matrix $P \in \mathbb{R}^{n\times n}$. The weighting matrices in (11a) are

$$Q_i = diag(q_{i,11}, q_{i,22}, q_{i,33}) \in \mathbb{R}^{3\times 3} \text{ and } R_i \in \mathbb{R}. \quad (14)$$

Here $Q_i \succeq 0$ and $R_i \succ 0$ are positive semi-definite and positive definite matrices, respectively.

Among the model-based equality constraint (11b) problem (11) considers inequality box constraints to limit the vehicle's velocity (11d) and acceleration (11e). Finally, (11f) represents the coupling constraint, which ensures that vehicle 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.

2) Lateral Motion Model and Control Design: We use the following continuous time matrix equation to model the lateral dynamics [35], [36]:

$$\dot{x}_c^{lt} = A_c^{lt} x_c^{lt} + B_{1,c}^{lt}\delta + B_{2,c}^{lt}\kappa + B_{3,c}^{lt}\beta, \quad (15)$$

where

$$x_c^{lt} = \left(d_{le}, \theta_{he}, \dot{\psi}\right)^T, \quad (16)$$

$$A_c^{lt} = \begin{pmatrix} 0 & v_x & 0 \\ 0 & 0 & 1 \\ 0 & 0 & \frac{-\left(l_f^2 C_f + l_r^2 C_r\right)}{J_z v_x} \end{pmatrix}, \quad (17)$$

$$B_c^{lt} = \begin{pmatrix} 0 & 0 & v_x \\ 0 & -v_x & 0 \\ \frac{l_f C_f}{J_z} & 0 & \frac{l_r C_r - l_f C_f}{J_z} \end{pmatrix}. \quad (18)$$

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. Fig. 8 illustrates the model parameters of the bicycle model. κ is the path reference curvature, β the vehicle sideslip angle, and δ the steering angle control input. Therefore, the pair $\left(A_c^{lt}, B_{1,c}^{lt}\right)$ 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 the MATLAB command *DLQR* is used to design the Discrete-time Linear Quadratic Regulator (DLQR) gains. The weighting matrices

$$Q^{lt} = diag(q_{11}^{lg}, q_{22}^{lg}, q_{33}^{lg}) \in \mathbb{R}^{3\times 3} \text{ and } R^{lt} \in \mathbb{R} \quad (19)$$

are used to calculate the DLQR gains, which are positive semi-definite and positive definite, i.e. $Q^{lt} \succeq 0$ and $R^{lt} \succ 0$.

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 speed-dependent gain scheduling is used so that the controller is robust to variations in the vehicle speed. The gain scheduling uses convex summation of the DLQR gains, as shown below:

$$G_{dlqr} = \begin{cases} G_1 & \text{if } v_{min} \le v_x \le v_1 \\ (v_x - 1)\, G_1 + (2 - v_x)\, G_2 & \text{if } v_1 < v_x \le v_2 \\ (v_x - 2)\, G_2 + (3 - v_x)\, G_3 & \text{if } v_2 < v_x \le v_{max} \\ G_3 & \text{if } v_x > v_{max} \end{cases}. \quad (20)$$

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

$$\hat{\delta} = G_{dlqr} x_c^{lt} \quad (21)$$

$$\delta = \begin{cases} \hat{\delta}(t) & \text{if } -\delta_{max} \le \hat{\delta} \le \delta_{max} \\ \pm\delta_{max} & \text{if } \pm\hat{\delta} > \delta_{max}. \end{cases} \quad (22)$$

In (22), $-\delta_{max}$ and δ_{max} are the lower and upper limits of the vehicle steering angle. The vehicle yaw rate and speed, which are required to calculate the control input, are measured by vehicle sensors.

IV. Integration of the Distributed Control System into the Test Platform

In order to test the growing complexity of distributed systems, it is crucial to use a systematic way of integrating the system functions. It should be possible to seamlessly test any changes in the distributed system. Changes may for example occur in the test environment or in the system components. To account for this, we propose the implementation method described below, utilizing emerging standards for road network topology descriptions, and a variable amount and type of sub-systems.

The test-system components in Fig. 2 are embedded in a Robotic Operating System (ROS) [37] network. The parking area management (PAM) unit and each vehicle controller are realized as separate ROS-nodes. We use the commercial virtual environment VTD[1] to simulate the parking scenario, and implemented a *VTD-to-ROS* unit to read data from and a *ROS-to-VTD* unit to send data to the virtual environment. The scenario generation unit in Fig. 2 provides a map in the de facto standard format openDRIVE [38], populated with a certain number of vehicles, as well as a scenario file, which contains initial scenario conditions. The generated openDRIVE map is interpreted by the VTD environment, and is at the same time used by the PAM ROS-node. The scenario file specifies the IDs of the vehicles 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

[1] https://vires.com/vtd-vires-virtual-test-drive/

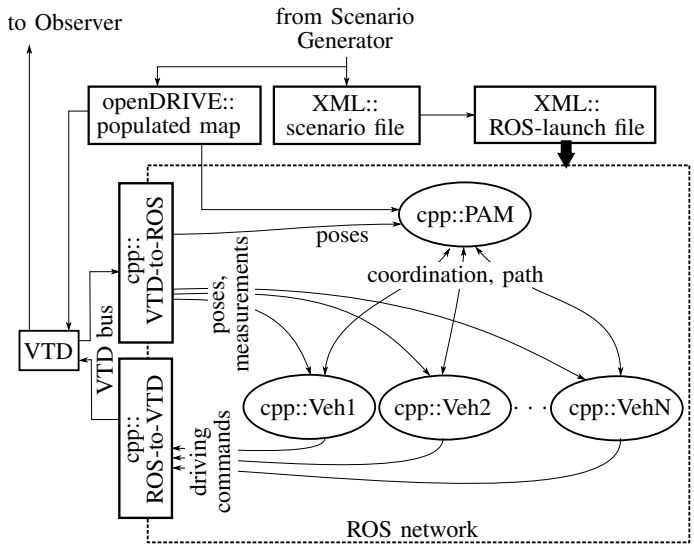


Fig. 9: Simplified overview of the ROS network integration with connection to the simulation environment VTD, and the scenario generator output.

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 programmin (QP) formulation and passed to the integrated qpOASES solver [39]. 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 information included in the scenario-file. This method enables the simulation of a varying number of vehicles in different simulation scenarios. Fig. 9 illustrates the described setup and presents a simplified structure of the ROS network. Table I 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 our MPC problem. 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.

V. Numerical Evaluation

To run the tests, we distribute the test architecture of Fig. 9 on two PCs, as shown in Fig. 10. The *Simulation PC* runs the test management system (see Fig. 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

TABLE I: Explanation of the most important ROS messages in the multi-vehicle control framework.

Signal name in Fig. 9	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 computed by pathplanner in PAM
driving command	ackermann_msgs/ AckermannDrive	steering and acceleration input
coordination	ctrl/Ctrl_msg	distance predictions of predecessor, reference values and constraints for MPC
VTD bus	Runtime Data Bus (RDB)	VTD internal user interaction at run-time

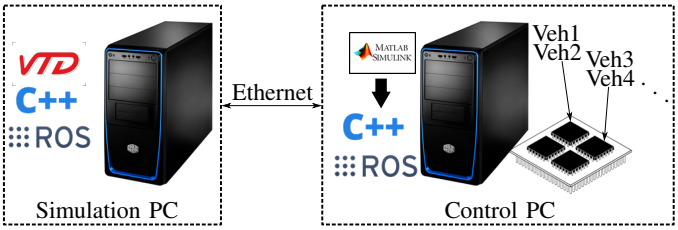


Fig. 10: Distributed implementation of the test-system in our lab.

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.

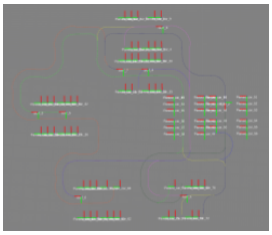
In Fig. 11, we give insights into the test platform visualization during a trial. Figure 11a shows the visualization of the VTD environment. The plot in Fig. 11b visualizes all objects (moving and parked vehicles) in the scenario using the ROS-rviz tool, and displays the paths provided by the PAM pathplanning unit. The bottom plot in Fig. 11c illustrates a randomly generated openDRIVE map of the trial from the scenario generation tool.

Now, we illustrate the functionality of the coordination and control method, and highlight the benefits compared to an uncoordinated scenario by evaluating the simulation results. Table II introduces the control parameters used for the evaluations.

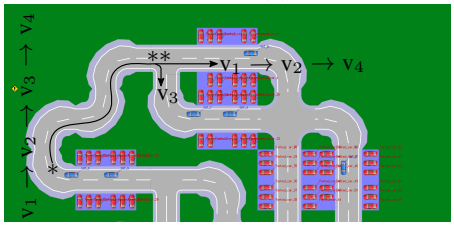
Note that the actual maneuvering into/out of the parking bay may also require backward driving and thus the velocity constraint should also allow $v_{min} < 0$. However, as this paper focuses on the coordination procedure within the parking area, we avoid backward driving.

First, the longitudinal coordination control is investigated. We thus illustrate the step-response of a single vehicle in Fig. 12. The vehicle starts from standstill and receives a reference

(a) VTD scenario visualization



(b) ROS-rviz visualization



(c) openDRIVE excerpt of randomly generated map.

Fig. 11: A randomly generated test scenario.

TABLE II: Control parameters.

Name	Parameter	Value
long. sampling time	T_s^{lg}	$0.1s$
long. time constant	τ_i	$0.1s$
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})	$(-4m/s^2, 1m/s^2)$
long. state weights	$(q_{i,11}, q_{i,22}, q_{i,33})$	$(1, 60, 30)$
long. input weight	R_i	30
lat. sampling time	T_s^{lt}	$0.01s$
lat. state weights	$(q_{11}^{lt}, q_{22}^{lt}, q_{33}^{lt})$	$(100, 10, 1)$
lat. input weight	R^{lt}	20
velocity intervals	(v_1, v_2)	$(1m/s, 2m/s)$
steering constraint	δ_{max}	$0.48rad$

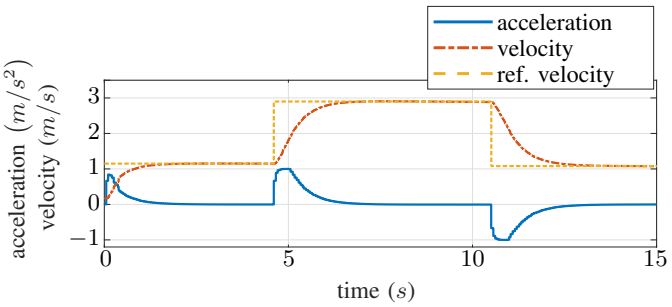


Fig. 12: Step-response of velocity and acceleration from a single vehicle.

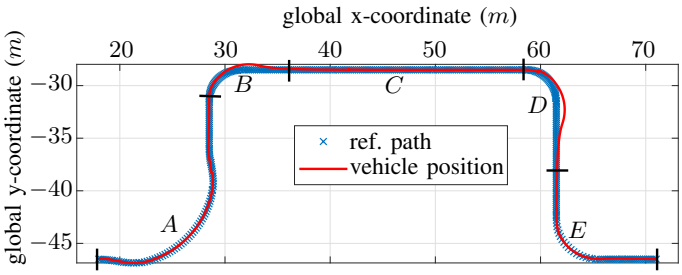


Fig. 13: Measured vehicle position for tracking a given path and the resulting tracking error.

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 (11) are chosen such that we avoid an 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. Fig. 13 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 Fig. 13) and accurate path following for straight segments (segment C). As the path is computed globally in the PAM, there is no kinematic feasibility guarantee for local vehicle tracking. The evaluation, however, shows a stable and adequate tracking behavior for infeasible 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 Fig. 11. Four vehicles $(\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3, \mathrm{v}_4)$ start from standstill in the area marked with a single $*$ in Fig. 11c. The resulting acceleration and velocity profiles of vehicles v_2 and v_4 are plotted in Fig. 14. 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 Fig. 14 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 $V = \{\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_7\}$ are randomly placed in the map as illustrated in Fig. 11c, and the PAM algorithm 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 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

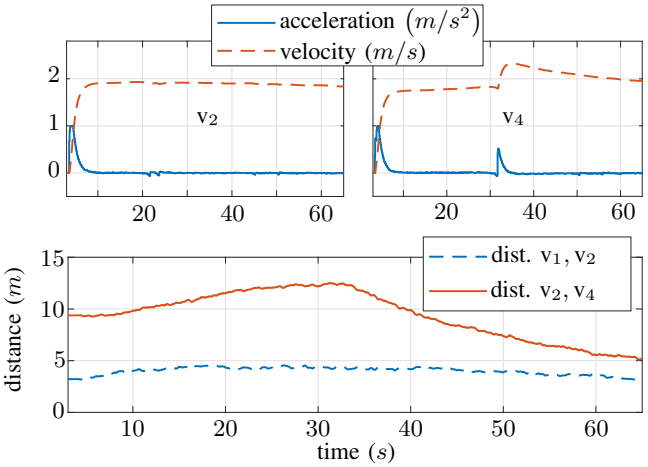


Fig. 14: Coordinated driving in a platoon, where vehicle v_3 leaves the platoon around $t = 27s$.

TABLE III: 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

method is supposed to simulate the behavior of human-driven cars in parking environments. The third method represents the coordinated control method proposed in this paper. 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 ($\Sigma_t a_i(t) > 0$). Furthermore, we analyze the duration of the parking process, referred to as time to park (ttp). Fig. 15 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 better 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 III lists the number of vehicle stops during a simulation scenario. A higher number of stops using the uncoordinated control method requires a higher interaction amount using the coordinated control method.

Table IV 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

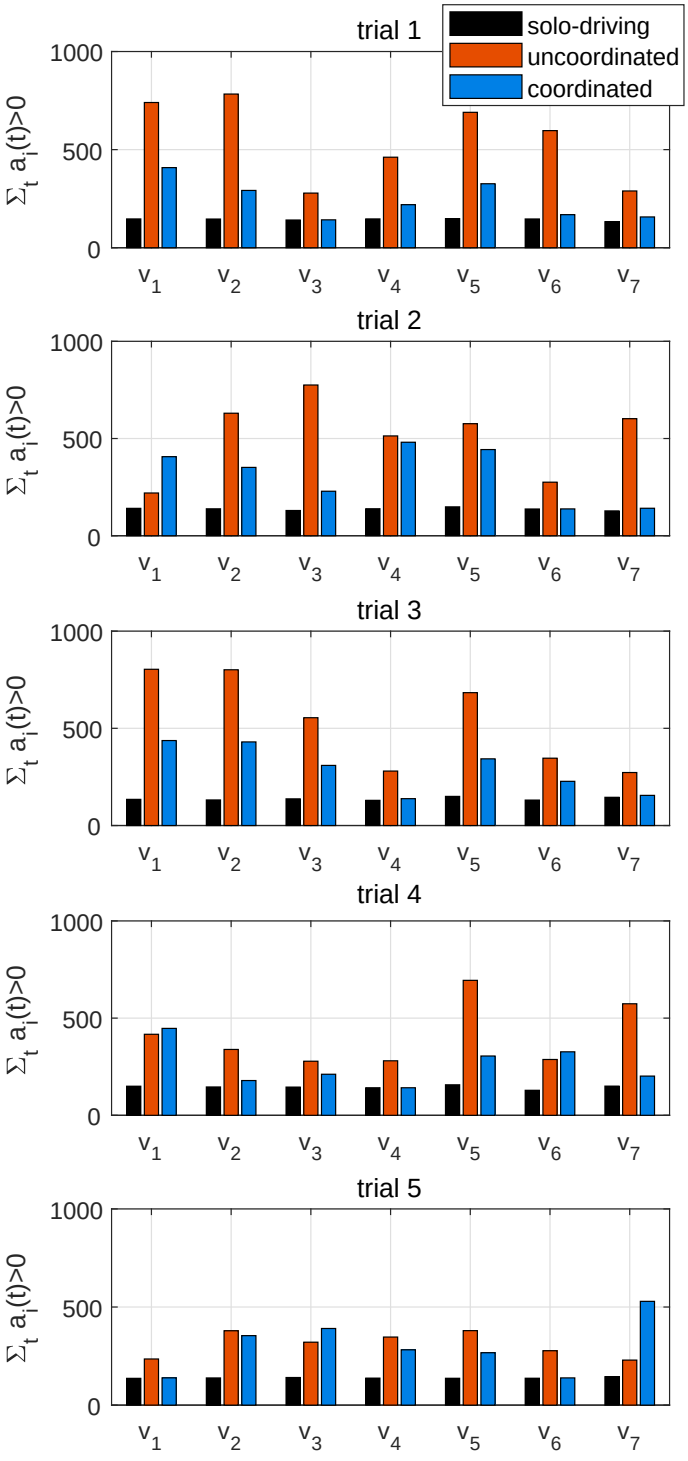


Fig. 15: Performance comparison of different control coordination methods for 7 vehicles and 5 trials with different initial conditions.

TABLE IV: Overall performance evaluation of the parking process with different control methods.

	control method					
	solo-driving		uncoordinated		coordinated	
	$ttp(s)$	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

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 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. Furthermore, 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.

VI. Conclusions

This paper presents a control framework for coordinating autonomous vehicles in parking areas by distributing the trajectory generation between vehicles and an infrastructure. Potential collision areas, called conflict zones, are used for the coordination procedure. We integrate this system into an automated test-system. The integration allows seamless variations of the system under test, such as changing agents or environments - factors which play an important role for testing highly automated systems.

Simulations illustrate the functionality and behavior of the integrated distributed control system. Furthermore, the increase in efficiency of the distributed coordination control is shown and compared with an uncoordinated method.

Future work includes the integration of uncertain behavior into the distributed setup. Uncertainty can result for example from an environmental perspective such as pedestrians in the parking area. Furthermore, among the control coordination, higher layer planning levels, i.e. mission planning and path planning, have a significant impact on the performance of the coordination procedure. Therefore, investigating appropriate algorithms for those layers shall be considered in future work.

References

[1] K.-W. Min and J.-D. Choi, "Design and implementation of autonomous vehicle valet parking system," in *16th Int. IEEE Conf. Intelligent Transportation Systems (ITSC)*, pp. 2082–2087, 2013.

[2] T. SAE, "Definitions for terms related to driving automation systems for on-road motor vehicles," *SAE Standard J3016*, 2016.

[3] J. Timpner, S. Friedrichs, J. van Balen, and L. Wolf, "k-stacks: high-density valet parking for automated vehicles," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pp. 895–900, 2015.

[4] H. Banzhaf, F. Quedenfeld, D. Nienhüser, S. Knoop, and J. M. Zöllner, "High density valet parking using k-deques in driveways," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pp. 1413–1420, 2017.

[5] C. Huang, R. Lu, X. Lin, and X. Shen, "Secure automated valet parking: a privacy-preserving reservation scheme for autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11169–11180, 2018.

[6] S. Klemm *et al.*, "Autonomous multi-story navigation for valet parking," in *19th Int. IEEE Conf. Intelligent Transportation Systems (ITSC)*, pp. 1126–1133, 2016.

[7] 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)*, pp. 1827–1834, 2017.

[8] U. Schwesinger *et al.*, "Automated valet parking and charging for e-mobility," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pp. 157–164, 2016.

[9] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of artificial intelligence research (JAIR)*, vol. 31, pp. 591–656, 2008.

[10] M. Hausknecht, T.-C. Au, and P. Stone, "Autonomous intersection management: multi-intersection optimization," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 4581–4586, 2011.

[11] 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 (CDC)*, pp. 2932–2937, 2014.

[12] L. Makarem and D. Gillet, "Fluent coordination of autonomous vehicles at intersections," in *IEEE Int. Conf. Systems, Man, and Cybernetics (SMC)*, pp. 2557–2562, 2012.

[13] 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. 12008–12014, 2017.

[14] R. Naumann, R. Rasche, J. Tacken, and C. Tahedi, "Validation and simulation of a decentralized intersection collision avoidance algorithm," in *IEEE Conf. Intelligent Transportation System (ITSC)*, pp. 818–823, 1997.

[15] 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.

[16] 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.

[17] 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)*, pp. 6227–6232, 2016.

[18] R. Kianfar *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.

[19] R. Hult *et al.*, "Design and experimental validation of a cooperative driving control architecture for the grand cooperative driving challenge 2016," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1290–1301, 2018.

[20] 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.

[21] 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, 2017.

[22] K.-D. Kim and P. R. Kumar, "An mpc-based approach to provable system-wide safety and liveness of autonomous ground traffic," *IEEE Trans. Automat. Contr.*, vol. 59, no. 12, pp. 3341–3356, 2014.

[23] X. Qian, J. Gregoire, A. De La Fortelle, and F. Moutarde, "Decentralized model predictive control for smooth coordination of automated vehicles at intersection," in *European Control Conference (ECC)*, pp. 3452–3458, 2015.

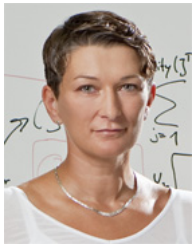
[24] R. Hult, M. Zanon, S. Gros, and P. Falcone, "Primal decomposition of the optimal coordination of vehicles at traffic intersections," in *55th IEEE Conference on Decision and Control (CDC)*, pp. 2567–2573, 2016.
[25] A. Uno, T. Sakaguchi, and S. Tsugawa, "A merging control algorithm based on inter-vehicle communication," in *Proc. Int. IEEE/IEEJ/JSAI Conf. Intelligent Transportation Systems (ITSC)*, pp. 783–787, 1999.
[26] X.-Y. Lu and J. K. Hedrick, "Longitudinal control algorithm for automated vehicle merging," *International Journal of Control*, vol. 76, no. 2, pp. 193–202, 2003.
[27] 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.
[28] "Pegasus project." https://www.pegasusprojekt.de/en/home. Accessed: 2018-10-22.
[29] "Enable-s3 project." https://www.enable-s3.eu/. Accessed: 2018-10-22.
[30] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.
[31] J. Gregoire, S. Bonnabel, and A. De La Fortelle, "Priority-based coordination of robots," 2014.
[32] 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.
[33] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.
[34] 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.
[35] 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.
[36] H. Pacejka, *Tire and vehicle dynamics*. Elsevier, 2005.
[37] M. Quigley *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, 2009.
[38] M. Dupuis and H. Grezlikowski, "Opendrive®-an open standard for the description of roads in driving simulations," in *Proc. Driving Simulation Conference*, pp. 25–36, 2006.
[39] 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.

Maximilian Kneissl received his Bachelor of Science and Master of Science degree in Electrical and Computer Engineering from the Technical University of Munich, Germany, in 2013 and 2016, respectively. He is currently working towards a Doctor of Engineering degree in Electrical and Computer Engineering at DENSO AUTOMOTIVE Deutschland GmbH (Eching, Germany) in cooperation with the chair of Information-Oriented control, Department of Electrical and Computer Engineering, Technical University of Munich, Germany. His research interests include distributed control and planning for cooperative vehicles in the field of autonomous driving, model-based system engineering, and simulation methods for distributed control systems.

Anil Kunnappillil Madhusudhanan is a Research Associate at the Department of Engineering, University of Cambridge. He received Bachelor of Technology in Electronics and Communication Engineering from National Institute of Technology Allahabad, India, in 2006. During the years 2006 to 2009, he worked at STMicroelectronics (Greater Noida, India), Indian Institute of Science (Bangalore, India) and Cranes Software (Bangalore, India) for one year each. He received Master of Science Cum Laude in Systems and Control, and PhD from Delft University of Technology, The Netherlands, in 2011 and 2016 respectively. Before relocating to Cambridge, he worked for about two years at TNO Automotive, The Netherlands.

Adam Molin is currently a research engineer at DENSO AUTOMOTIVE Deutschland GmbH in Eching, Germany. Prior to that, he was a postdoctoral researcher at the Department of Automatic Control, Royal Institute of Technology (KTH), Stockholm, Sweden, from 2014 to 2016. He received his Diplom degree in electrical engineering in 2007 and his Doctor of Engineering degree in 2014, both from the Department of Electrical Engineering and Information Technology, Technical University of Munich (TUM), Germany. His Ph.D. thesis was awarded with the Kurt-Fischer-Prize by the Department of Electrical Engineering and Information Technology, TUM, in 2014. His main research interests include the development of testing and design methods for networked control and cyberphysical systems with applications to automotive systems.

Hasan Esen is a Technical Manager in the Corporate R&D Department at DENSO AUTOMOTIVE Deutschland GmbH. He is responsible for the advanced control and system engineering R&D activities in Europe. He received his PhD in Control Engineering from Technical University of Munich, Germany, M.Sc in Mechatronics from Technical University of Hamburg-Harburg, Germany, and B.Sc in Mechanical Engineering from Technical Universtiy of Istanbul, Turkey.



Sandra Hirche (M03-SM11) received the Diplom-Ingenieur degree in aeronautical engineering from Technical University Berlin, Germany, in 2002 and the Doktor-Ingenieur degree in electrical engineering from Technical University Munich, Germany, in 2005. From 2005 to 2007 she was awarded a Postdoc scholarship from the Japanese Society for the Promotion of Science at the Fujita Laboratory, Tokyo Institute of Technology, Tokyo, Japan. From 2008 to 2012 she has been an associate professor at Technical University Munich. Since 2013 she is TUM Liesel Beckmann Distinguished Professor and heads the Chair of Information-oriented Control in the Department of Electrical and Computer Engineering at Technical University Munich. Her main research interests include cooperative, distributed and networked control with applications in human-robot interaction, multi-robot systems, and general robotics. She has published more than 150 papers in international journals, books and refereed conferences. Dr. Hirche has served on the Editorial Boards of the IEEE Transactions on Control of Network Systems, IEEE Transactions on Control Systems Technology, and the IEEE Transactions on Haptics.