Mixed-Reality Testing of Multi-Vehicle Coordination in an Automated Valet Parking Environment

Maximilian Kneissl [*,***] **Sebastian vom Dorff** [*,****]
Adam Molin [*] **Maxime Denniel** [**] **Tong Duy Son** [**]
Nicolas Ochoa Lleras [*] **Hasan Esen** [*] **Sandra Hirche** [***]

[*] *Corporate R&D, DENSO Automotive Deutschland GmbH, Freisinger Str. 21-23, 85386 Eching, Germany (e-mail: {m.kneissl,s.vomdorff,a.molin,h.esen}@denso-auto.de)*
[**] *Siemens PLM Software, 3001 Leuven, Belgium (e-mail: son.tong@siemens.com)*
[***] *Institute for Information- oriented Control, Technische Universität München, Arcisstr. 21, 80290 München, Germany (e-mail: hirche@tum.de)*
[****] *Department of Computing Science, Carl von Ossietzky Universität Oldenburg, Oldenburg, Germany*

Abstract: The development of highly automated driving functions requires rigorous testing to demonstrate the safety and functionality of the automated vehicle. One open question is how to perform such tests to sufficiently prove the vehicle's capabilities. Designing a proper testing platform is particularly important for multi-vehicle scenarios, because test setups with actual real vehicles are not scalable. This paper proposes a mixed-reality testing framework which seamlessly combines virtual and real testing. The authors conducted a mixed-reality experiment of an automated valet parking (AVP) scenario, where virtual vehicles interact with a real vehicle-in-the-loop system in a simulated world. The experiment was developed to evaluate the algorithmic design of a distributed control scheme for an AVP system. Comparing the mixed-reality results to those of a pure virtual simulation allowed the authors to demonstrate the robustness of the algorithmic design against certain disturbances, while also revealing which parts of the system were sufficiently or inadequately modeled during the development process.

Keywords: Mixed-reality, Simulation, Interactive Vehicle Control, Automated Valet Parking, AVP, Virtual Test

1. INTRODUCTION

The development of automated driving has gained interest in both academia and industry in recent years. Advanced driver assistant systems such as emergency braking or adaptive cruise control are already commercially available, and they are becoming more sophisticated and connected with every new release. However, available systems require a human driver to interact in critical or unknown situations. The next step is to develop systems where no human driver is required. The SAE (Society of Automotive Engineers) levels classify the degree of automation of cars (see SAE (2016)). With respect to the classification Level 4, vehicles can act autonomously in certain operational design domains (ODD), i.e. pre-defined environments and scenarios. Automated valet parking (AVP) is expected to be one of the first commercially available Level 4 automated driving functions (ERTRAC (2017)), as for parking environments a clear ODD can be specified. In AVP a vehicle can be left at a drop-off zone in front of a parking lot, where the system takes the vehicle over from the human driver and automatically guides it to its parking bay. After a user-request the vehicle can navigate back to a pick-up zone, where the driver takes over control again. Figure 1 illustrates an AVP scenario.

A major challenge of such highly automated systems is to guarantee their safe and reliable functionality. It is known that pure test driving is unfeasible to provide this guarantee (Kalra and Paddock (2016)). Therefore, testing during the development process plays an increasingly important role. Stellet et al. (2015) and Huang et al. (2016) provide an overview of different testing methods for autonomous driving. One target is to move a large portion of the required test effort to virtual testing in high-fidelity simulation environments. This enables an increased test speed and the ability to specifically test critical situations. Significant efforts towards virtual test-platforms and the automation of the tests were conducted, for example, in

⋆ 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 receives support from the European Union's HORIZON 2020 research and innovation programme and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.

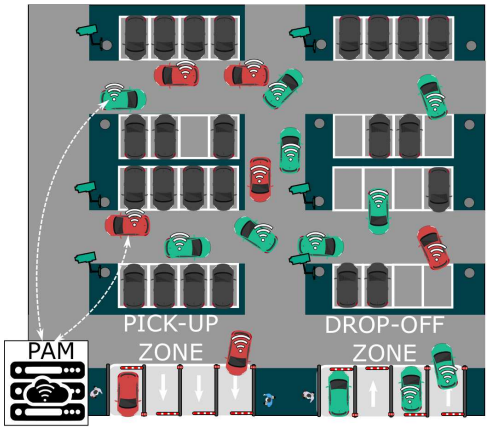

Fig. 1. Automated valet parking scenario.

the PEGASUS and the ENABLE-S3 projects (PEGASUS (2016),ENABLE-S3 (2016)). Son et al. (2019) and Son et al. (2018) describe a setup of high-fidelity simulation environments for automated driving. They use co-simulation methods to test planning and control algorithms.

A benefit of automated vehicles is their possibility to communicate and share intentions with other surrounding vehicles or infrastructure systems. By such cooperative behaviors traffic efficiency can be improved in terms of energy consumption and time wastage. The ability to test the functionality of multi-vehicle scenarios is a further strength of virtual test-platforms. However, the final goal is to additionally validate developed systems on real vehicle hardware. In multi-vehicle scenarios this is too time consuming and costly with actual real vehicles. A first step can be testing in a miniature vehicle setup (Fok et al. (2012)).

To go one step further towards reality we propose the extension of a virtual test-platform for multi-vehicle simulation with a real vehicle, which replaces one of the virtual vehicles. An essential point is the modular integration of the components into the simulation framework. This is important to enable exchange of modules and therefore contribute to future seamless test-platforms. With the resulting mixed-reality tests one can draw conclusions on the correctness of models and algorithmic behavior incorporating a real vehicle. Furthermore, it enables validation of the multi-vehicle behavior with reasonable time and cost effort. A similar idea for a single vehicle is proposed by Zofka et al. (2018) in the field of sensor data augmentation, where real-world lidar data is augmented with virtual objects, such as pedestrians. Furthermore, Quinlan et al. (2010) present a multi-vehicle mixed-reality test to verify their autonomous intersection crossing protocol using one real vehicle and virtual opponents.

The authors developed a distributed and coordinated control methodology for multi-vehicle scenarios integrated into a virtual AVP test environment. The applied distributed model predictive control (MPC) algorithms were extended to obtain any-time guaranteed feasible and safe longitudinal coordinated solutions (Kneissl et al. (2019)). In this paper we additionally propose a path planning strategy integrated in the mixed-reality test-platform, which, together with the control coordination, ensures collision avoidance in the parking area.

Different path planning techniques have been presented in the literature (Paden et al. (2016)). Considering the specific parking implementation challenges in this work, we have developed a local vehicle path planning based on Hybrid A* graph search (Dolgov et al. (2008)).

The paper presents two main contributions. First, the demonstration of a fully integrated mixed-reality platform for testing AVP functions in a multi-vehicle setup. The used methodology allows a transition from purely virtual testing to ViL (vehicle-in-the-loop) testing in a short amount of time. Moreover, it represents a cost and time efficient way of validating multi-vehicle systems. Vehicles are virtually simulated in a high-fidelity simulation environment and controlled by a distributed system. One of the vehicles participating in the AVP is a real vehicle that interacts with the other virtual agents in the environment. Second, we introduce the applied algorithms which ensure a safe-by-design coordination of multiple autonomous vehicles in parking environments, while enabling a distributed and optimized computation of the planning and control functions. Therefore, we integrated multiple components in an AVP system, which are, mapping, sensing and localization, global planning, local vehicle path planning and longitudinal/lateral control.

The remainder of the paper is organized as follows. Section 2 introduces the virtual test architecture and describes the real vehicle's setup and its integration into the test-platform. In Section 3 we introduce the path planning strategy, and distributed control methodology for multi-vehicle coordination, as well as the applied sensing and localization methods for the real vehicle. Experimental tests are presented in Section 4 and the paper is concluded in Section 5.

2. MIXED-REALITY ARCHITECTURE

This section introduces the virtual test-platform for AVP in the following Section 2.1, and the extension of it with the real test vehicle in Section 2.2.

2.1 Virtual Automated Valet Parking Architecture

The AVP test-platform, illustrated in Fig. 2, consists of a high-fidelity simulation environment (left side) and the system-under-test (SUT) on the center and right. The simulation environment uses a map of the respective parking area provided in the OpenDRIVE map format (Dupuis et al. (2010)). The driving simulator Vires VTD is used to simulate the dynamic behavior and visualize the movements in the given map environment. The SUT is the complete AVP system. It consists of a central unit, the parking area management (PAM) system, and local vehicle planning and control systems. The PAM system is an infrastructure unit located at the parking area and vehicles exchange information with it via vehicle-to-infrastructure (V2I) communication. Tasks of the PAM are distinguished between *mission planning*, *path planning*, and *control coordination*. The mission planner is aware of the parking lot situation and decides on the goal position, i.e. designated parking bay or exit gate, once it receives a

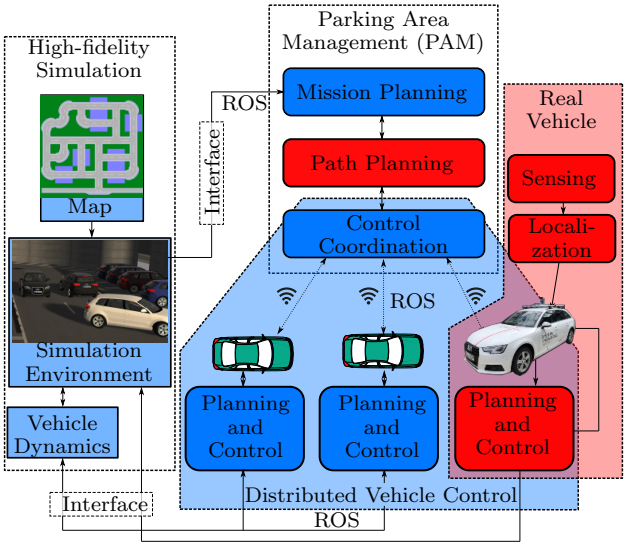

Fig. 2. Mixed-reality test architecture.

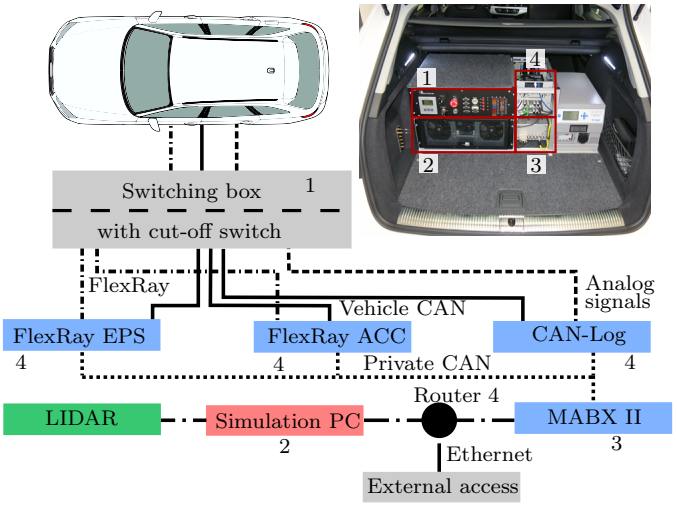

Fig. 3. Schematic network diagram of the SUT and vehicle integration.

vehicle's request to park or return to the pick-up zone. The functionalities of path planning, control coordination, as well as the local planning and control units are introduced in Sections 3.1 and 3.2. Blocks are implemented in C++ and Python and communicate via a ROS middleware with each other. The high-fidelity simulation is also interfaced to the ROS system. Note that the whole system is tested on a simulation PC and thus the V2I communication is subject of the simulation as well, rather than real communication channels.

2.2 Vehicle Demonstrator Integration

In the following section we will provide an overview of the real vehicle integration into the test set-up.

Hardware In order to provide a base platform for the real world vehicle an Audi A4 B9 station wagon was used. The series production car base eases the requirements of complying with road regulations and certification. The vehicle was not intrinsically modified but all modifications have been implemented as piggy-back modules attached to the baseline vehicle. All required low level functions are already available, embodied by electric power steering (EPS), brake actuation by the electronic stability program (ESP), acceleration through the adaptive cruise control (ACC) and body control functions such as indicator- and brake lights by the respective electronic control units (ECUs). Parking brake and drive mode selector are also realized in a "by-wire" implementation, offering a simple hijacking of the corresponding analog signal lines. The vehicle setup is illustrated in Fig. 3.

The communication and handling of signals between the virtual simulation world and the real vehicle is provided by a set-up consisting of two FlexRay (FR) controllers, a CAN-Bus development unit, and a dSPACE MicroAuto-Box II (MABXII), which is again connected to a network router. The router interfaces the planning phase from the virtual reality with the actors in the real world. It also acts as entry point to access the in-vehicle infrastructure from external devices e.g. for monitoring and debugging. The car-PC simulates the virtual environment and executes the planning for the vehicle's trajectory. In order to precisely localize the vehicle in the parking environment a LIDAR delivers positioning data to the car-PC.

The communication topology is split into different levels. Starting from the vehicle, FR, CAN and analog lines are connected to a switching box. These relay controlled connections can be interrupted by an emergency switch which cuts all connections immediately and sends an analog command to set the parking brake as last action. From the switching box, two FR devices are connected to the FR 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 control body and auxiliary functions such as the indicator lights, brake light, gear selection, and parking brake. The FR signals are internally mapped to a private CAN protocol to preserve the secrecy of the FR bus. Serving as next bus level on the topology, the CAN-Log and the MABXII are also connected to this private CAN. The MABXII acts as translator between the vehicle interfaces and the simulation PC via an ethernet connection. Since the access to the vehicle is a mere manipulation of existing ECU signals, it uses the same safety functions as the series production car. 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.

In order to satisfy the increased demands on the power supply of the set-up, an auxiliary battery was built into the car, providing 1.2 kWh of energy on 12 V DC or 230 V AC level. It is coupled to the vehicle's power circuit and can be charged by the alternator, as well as by an external 230 V AC supply. The AC/DC converter system works bi-directionally, therefore also being able to charge a depleted 12V car battery due to frequent start-stop operations. In order to provide sufficient cooling to the hardware an additional air conditioning (A/C) unit has been installed under the trunk floor. It is supplied by the vehicle's A/C coolant circuit.

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 read and converted to the control parameters of the vehicle. Since the control parameters are unique to the specific vehicle, the translation is done as abstract as possible in the toolchain. This enables the ROS bridge to be reused for other test vehicles in the future. As the operation of the simulated vehicles 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 car, e.g. the corresponding drive mode selection. To catch potentially flawed input data, all input values are limited in their range. All signals provide the option to be switched to a manual control option, which enables manipulating values for monitoring and calibration from an external PC.

3. ALGORITHMIC DESIGN

In this section we introduce the applied algorithms of the mixed-reality test-platform. First, the semi-structured path planning for parking environments is introduced. Second, the local vehicle planning and control methods are defined. Third, we describe the sensing and localization method for the real vehicle.

3.1 Semi-structured Pathplanning

Infrastructure-supported Planning After receiving the initial and goal pose of the automated vehicle, the path-planning module at the PAM first computes the shortest path based on the lane network topology (top left Fig. 2) that is stored by the infrastructure. This topology encodes drivable lanes including their length, direction, and their interconnection with each other. The obtained abstract shortest 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 the automated vehicle (black area right Fig. 6). 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 by incorporating the conflict zone (cz) area (cf. Section 3.2) at intersections (1a Fig. 6) and at the vicinity of the parking bay (1b Fig. 6) to facilitate parking maneuvers. The obtained occupancy grid map (OGM) is then forwarded to the local planning module in the vehicle.

Local Vehicle Planning In order to drive the car from its starting position to the goal parking spot, a path linking those two positions needs to be computed. The path needs to be drivable, collision free, and easy to be followed by the car controller.

The conventional A* search algorithm connects two points in an OGM, without colliding and in the shortest way by linking together center of cells. The Hybrid A* algorithm is an extension of A* algorithm to deal with non-holonomic behavior and continuous path generation. The algorithm utilizes a given OGM describing the environment together with start and goal point coordinates, i.e. (x_s, y_s, θ_s) and (x_g, y_g, θ_g), respectively. 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, $cost_i(n)$, and a heuristic, $h(n)$, in order to obtain desirable path quality and fast exploration. The costs include computed cost from the start to the current node and multiple driving objectives such as path length, steering, and directional gear selection. The heuristic is used to drive the search towards the goal. The algorithm takes into account non-holonomic behavior of the vehicle 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

$$\min_n \ (h(n) + \sum_i cost_i(n)) \ , \forall n \in OpenSet, \qquad (1)$$

where $OpenSet$ is the set of candidates nodes in the drivable area of the OGM. Moreover, h and $cost_i$ 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, resulting in the quality and the computation time of the path. The cost parameters can be tuned to promote or prevent some behaviors. For instance, increasing the cost of reverse gear will make the planner more likely to use a forward gear to drive to the goal.

Fig. 4 and Fig. 5 demonstrate validation results of the developed path planning algorithm from 1000 different starting positions to a defined target parking goal. The validation is conducted via a co-simulation of Siemens Simcenter Amesim and Prescan software. Notice that the goal coordinate and parking type (parallel parking) are the same in both cases, however, the orientation of the starting point is opposite: the vehicle starts heading to east/right side in Fig. 4 and to west/left side in Fig. 5. The results show that the success rate is 100% in the first case, and 99.3% in the second case. In case of failure, the vehicle can try to drive forward or reverse a short distance so that it can find another (feasible) starting point to complete the task.

3.2 Vehicle Coordination Control

The vehicles are coordinated within the parking area via a cooperative trajectory generation method. Therefore, tasks are distributed between the control coordination block on the PAM system and the local vehicle control systems. The coordination procedure receives the paths from the path planning units. Thus, each vehicle knows its own path and the PAM stores all paths of vehicles in the parking area. Given this information each vehicle tracks its path. Due to the low speed operation within parking environments, the longitudinal and lateral path-tracking problem can be separated. The longitudinal movement (acceleration) is computed with a model predictive control (MPC) law, while a velocity-based gain-scheduling LQR law is applied for the lateral tracking (steering). Local longitudinal predictions resulting from the MPC are shared with the PAM coordination unit. It decides on the

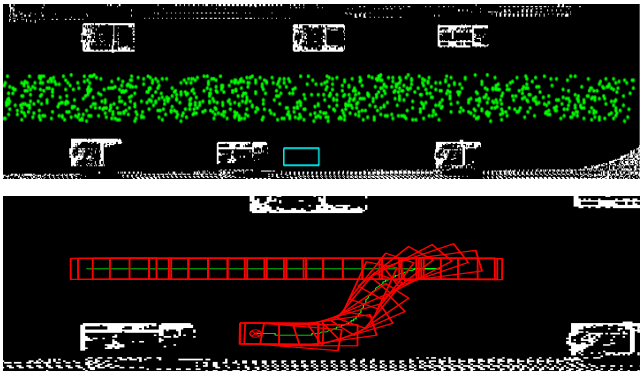

Fig. 4. Validation of the path planner for parallel parking: all starting position are heading east, goal position is heading east (100% success), and an example of the generated path (bottom).

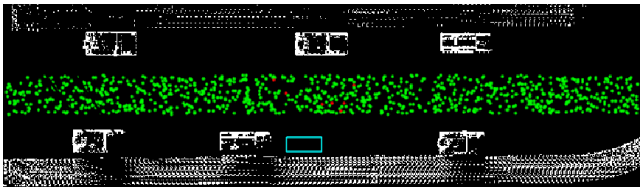

Fig. 5. Validation of the path planner for parallel parking: all starting position are heading west, goal position is heading east (99.3% success - failed cases are the red dots).

crossing order of all vehicles at potential conflict zones (czs), such as intersections and maneuvering zones, in the parking area and forwards the prediction information to the relevant vehicles. Predictions received from other vehicles are incorporated in their own local MPC problems in the form of safety-distance constraints. This procedure results in collision-free local vehicle trajectories and an overall smooth coordination.

Longitudinal Vehicle Control In the following, we state the applied MPC and LQR problems including their models for a single vehicle. By superscripts lg and lt we distinguish between longitudinal and lateral models, respectively. The continuous-time longitudinal motion model is given by

$$\underbrace{\begin{pmatrix}\dot{d}\\ \dot{v}\\ \dot{a}\end{pmatrix}}_{\dot{x}_c^{lg}} = \underbrace{\begin{pmatrix}0 & -1 & 0\\ 0 & 0 & 1\\ 0 & 0 & -\frac{1}{\tau}\end{pmatrix}}_{A_c^{lg}} \underbrace{\begin{pmatrix}d\\ v\\ a\end{pmatrix}}_{x_c^{lg}} + \underbrace{\begin{pmatrix}0\\ 0\\ \frac{1}{\tau}\end{pmatrix}}_{B_c^{lg}} u, \tag{2}$$

where states d is the distance to the vehicle's goal position, v its velocity, and a its acceleration. The control input u represents the vehicle's jerk, and τ is a time constant. For simplicity of notation, we skip the time dependency of states and the input in the above model. (2) is discretized using a *tustin* method with a sampling time T_s^{lg} and the MPC problem is formulated as

$$V^*\left(x^{lg}(t), u(t)\right) = \tag{3a}$$

$$\min_U \sum_{k=1}^{M} \|x^{lg}(t+k|t) - x_{ref}^{lg}\|_{Q^{lg}}^2 + \sum_{k=0}^{M-1} \|u(t+k|t)\|_{R^{lg}}^2$$

s.t.

$$x^{lg}(t+k+1|t) = A^{lg}x^{lg}(t+k|t) + B^{lg}u(t+k) \quad \forall k \in \mathbb{I}_{0:M-1} \tag{3b}$$

$$v_{min} \le v(t+k|t) \le v_{max} \quad \forall k \in \mathbb{I}_{1:M} \tag{3c}$$

$$a_{min} \le a(t+k|t) \le a_{max} \quad \forall k \in \mathbb{I}_{1:M} \tag{3d}$$

$$c\left(d(t+k|t)\right) \ge d_s \quad \forall k \in \mathbb{I}_{1:M}. \tag{3e}$$

Thereby, M is the prediction horizon, notation $(t+k|t)$ refers to a predicted vector value $(t+k)$ computed at a discrete time step t, $x_{ref}^{lg} = (0, v_{ref}, 0)^T$ is a reference vector, which is constant for the prediction horizon, and v_{ref} is a reference velocity, $Q^{lg} \in \mathbb{R}^{3\times3}$ is a positive semi-definite diagonal matrix applied in the weighted $2-$norm of the state tracking, and similar for the input with $R^{lg} \in \mathbb{R}$, $U = (u(t), u(t+1), ..., u(t+M-1)) \in \mathbb{R}^M$ is the optimization vector, $v_{min}, v_{max}, a_{min}$, and a_{max} are constraints on the velocity and acceleration state, respectively, $\mathbb{I}_{1:M}$ defines the set $\{1, ..., M\}$. Finally, $c\left(d(t+k|t)\right) \ge d_s$ describes a safety distance constraint, with $c\left(d(t+k|t)\right)$ being a transformation of the local distance state d w.r.t. a dependent vehicle's position (entrance of the path into a cz). This information is computed in the PAM control coordination procedure and shared with the local vehicle. Problem (3) is a quadratic program (QP) and can be computed efficiently by suitable solvers. After the computation, the trajectory $x^{lg}(t+k|t), k \in \mathbb{I}_{1:M}$, is shared with the PAM and $U(1)$ is applied to the local vehicle. Then, problem (3) is recomputed in the next sampling time step T_s^{lg} in a receding horizon fashion.

Lateral Vehicle Control Next the lateral motion is modeled with

$$\underbrace{\begin{pmatrix}\dot{d}_{le}\\ \dot{\theta}_{he}\\ \ddot{\psi}\end{pmatrix}}_{\dot{x}_c^{lg}} = \underbrace{\begin{pmatrix}0 & v & 0\\ 0 & 0 & 1\\ 0 & 0 & \frac{-(l_f^2 C_f + l_r^2 C_r)}{J_z v}\end{pmatrix}}_{A_c^{lg}} \underbrace{\begin{pmatrix}d_{le}\\ \theta_{he}\\ \dot{\psi}\end{pmatrix}}_{x_c^{lg}} + \underbrace{\begin{pmatrix}0\\ 0\\ \frac{l_f C_f}{J_z}\end{pmatrix}}_{B_c^{lg}} \delta, \tag{4}$$

where d_{le} is the lateral distance error between the vehicle's center of gravity and the reference path, θ_{he} is the heading angle error, $\dot{\psi}$ the yaw rate, C_f and C_r the cornering stiffness of the front and the rear axle, respectively, J_z is the momentum of inertia around the yaw axis, and l_f and l_r are the distances from the center of gravity to the front and rear axle, respectively. The control input δ represents the steering angle. Similar as above, model (2) is discretized using a sampling time T_s^{lt}. This enables us to compute Linear-quadratic-regulator (LQR) gains, $G \in \mathbb{R}^{1\times3}$, with MATLAB's DLQR command, using diagonal and positive semi-definite weighting matrices $Q^{lg} \in \mathbb{R}^{3\times3}$ and $R \in \mathbb{R}$. The resulting control law for the lateral tracking is thus given by

$$\delta = G(v)x_c^{lt}, \tag{5}$$

where $G(v)$ is computed for different values of $v \in \{v_a, v_b\}$.

3.3 Sensing and Localization

While for virtual vehicles the ground truth position is available, the localization algorithm for the real vehicle is based on an Extended Kalman Filter (EKF, Thrun et al. (2005)) 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 produce an estimate of 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 the parking lot. Since the model's differential equations are non-linear, they are locally linearized for each execution of the Kalman filter's model update.

Even if the bicycle model is accurate enough and the odometry data is reliable, the model updates tend to drift as time advances. Therefore, LIDAR data is used to correct the position estimate, a step commonly referred to as *measurement update*. The SICK NAV245 LIDAR sensor performs a 2-D scan and produces a list of detected landmarks, which have been previously mapped, at a rate of 25Hz. The constellation of detected landmarks is compared to the pre-existing map to deduce what position and orientation of the vehicle 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.

The parking lot was fitted with 8 landmarks, ensuring that at least 3 of them are always within the field of view (FOV) of the sensor during the tests (Fig. 6). The landmarks are made of highly reflective material that allows the NAV245 to pick them out among other objects and produce an accurate location estimate for each landmark. The map of landmarks was created by placing the sensor, stationary, at a level position from which all landmarks were visible within its FOV. Repeated measurements of the landmark locations were used to produce the ground truth estimate of their locations in the map.

4. EXPERIMENTAL TESTING

The experiments were conducted using the virtual model of a parking garage (bottom plot of Fig. 6) with three vehicles driving in the parking area, namely v_1, v_2, and v_3. The test scenario and start positions are illustrated in Fig. 6. 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 (*czs*) in the scenario, i.e. two intersections and one maneuvering zone, which are used by the coordination procedure to ensure safe movement of the vehicles.

Table 1 summarizes the model parameters used for the experiments.

Table 1. Control parameters.

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

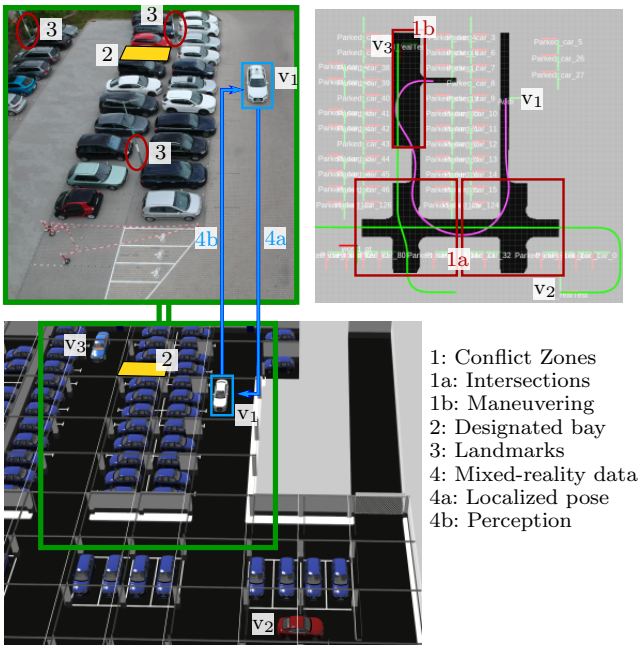

Fig. 6. Mixed-reality test scenario.

4.1 Virtual Testing

In the first experiment we conduct a pure virtual simulation of the above introduced 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 Fig. 6 is first crossed by v_1 and then by v_2, while v_1 crosses the left intersection after v_3. For v_1 we apply the semi-structured path planning algorithm introduced in Section 3.1. Here we restrict the planner to forward parking maneuvers only (without reversing) as the main validation target is the functionality of the coordination procedure. A path planning result for v_1 is illustrated by the purple lane in the top right plot of Fig. 6 and in more detail in the left plot of Fig. 9. Furthermore, the control signals and vehicle measurements of v_1 are plotted in the left plots of Fig. 7 for the acceleration control command and measured velocity, as well as the steering commands. In the top left plot of Fig. 7 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

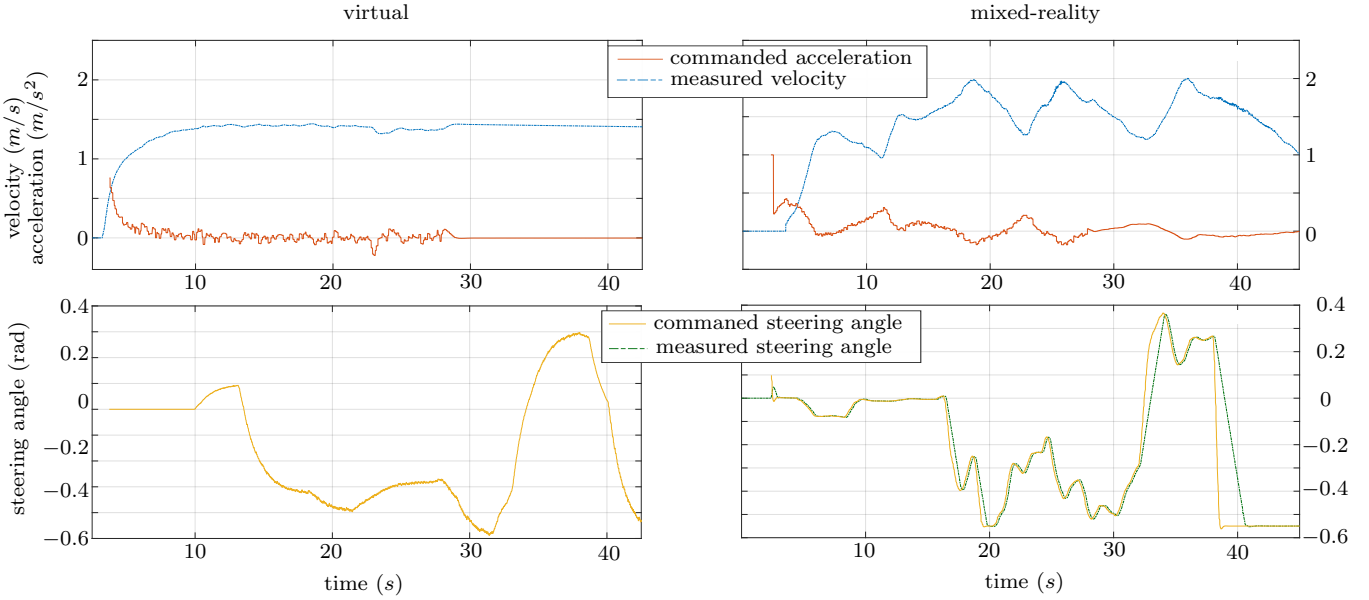

Fig. 7. Longitudinal and lateral vehicle signals of vehicle v_1 for virtual simulation (left) vs. mixed-reality testing (right) in scenario of Fig. 6. Top plots: acceleration control signals and resulting velocity; bottom plots: steering angle control input and actual applied steering signal for the real-vehicle case.

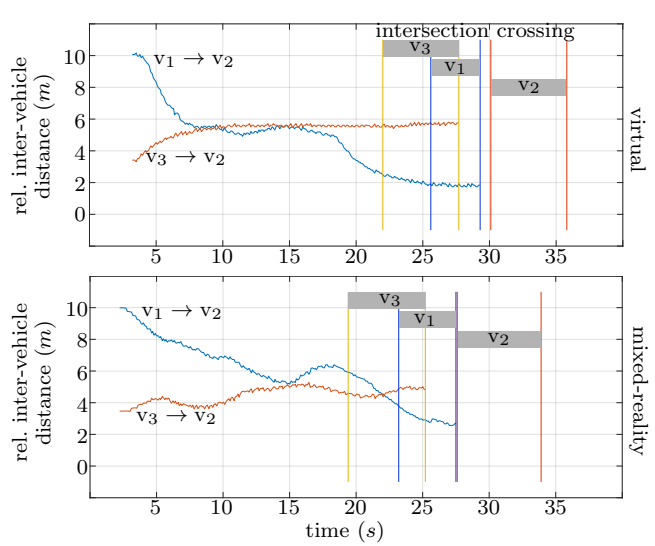

Fig. 8. Relative inter-vehicle distances for virtual simulation (top) and mixed-reality simulation (bottom) w.r.t. the entrance of the left intersection area in Fig. 6, as well as crossing times of vehicles through that intersection area.

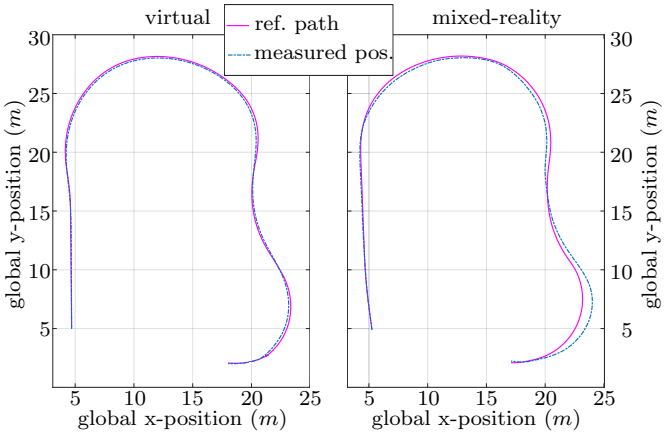

Fig. 9. Planned reference path (solid) and actual tracking position (dashed) for vehicle v_1 of the scenario in Fig. 6. Left: virtual simulation; right: mixed-reality simulation.

noisy inter-distance measurements (compare Fig. 8). The lateral path-tracking performance is shown in the left plot of Fig. 9. Finally, in the top plot of Fig. 8 we investigate the relative vehicle distance of v_1 and v_2 with respect to the entrance of the left intersection area (Fig. 6). Furthermore, 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.

4.2 Mixed-reality Testing

In the second experiment, two vehicles are simulated as purely 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 Fig. 7 for the acceleration control command and measured velocity, as well as the commanded and actual applied steering commands in the bottom right of Fig. 7. The lateral path-tracking performance for the real vehicle is shown on the right of Fig. 9. The bottom plot of Fig.

8 shows the relative vehicle distance of v_1 and v_2 for the mixed-reality test.

We find that the longitudinal acceleration signal in Fig. 7 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. 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. Fig. 8). Additionally, the right plot of Fig. 9 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 Fig. 7.

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.

5. CONCLUSION

In this paper we presented the results from an experiment of a mixed-reality multi-vehicle simulation in automated valet parking (AVP) environment. Thereby, virtual vehicles in a high fidelity simulation environment are cooperatively coordinated together with a real vehicle participating in the scenario by mapping its behavior into the virtual world. The applied algorithms for path planning and vehicle control have been introduced. They are based on hybrid A* planning, as well as MPC and LQR, respectively. Evaluation shows the robustness of the distributed vehicle coordination control and a suitable way for model fidelity checking.

Ongoing work includes consideration of time-delay models for the vehicle controllers. Furthermore, the incorporation of uncertainties in the operational domain, such as pedestrians, will be considered in future work.

REFERENCES

Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2008). Practical search techniques in path planning for autonomous driving. In *First International Symposium on Search Techniques in Artificial Intelligence and Robotics*. AAAI.

Dupuis, M., Strobl, M., and Grezlikowski, H. (2010). Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks. In *Driving Simulation Conference Europe*, 231–242.

ENABLE-S3 (2016). Enable-s3 project. https://www.enable-s3.eu/. Accessed: 2019-08-12.

ERTRAC (2017). Ertrac roadmap automated driving. https://www.ertrac.org/index.php?page=ertrac-roadmap. Accessed: 2019-10-18.

Fok, C.L., Hanna, M., Gee, S., Au, T.C., Stone, P., Julien, C., and Vishwanath, S. (2012). A platform for evaluating autonomous intersection management policies. In *2012 IEEE/ACM Third International Conference on Cyber-Physical Systems*, 87–96.

Huang, W., Wang, K., Lv, Y., and Zhu, F. (2016). Autonomous vehicles testing methods review. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 163–168.

Kalra, N. and Paddock, S.M. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94, 182–193.

Kneissl, M., Molin, A., Esen, H., and Hirche, S. (2019). A one-step feasible negotiation algorithm for distributed trajectory generation of autonomous vehicles. In *58h IEEE Conference on Decision and Control (CDC)*.

Paden, B., Čáp, M., Yong, S.Z., Yershov, D., and Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1), 33–55.

PEGASUS (2016). Pegasus project. https://www.pegasusprojekt.de/en/home. Accessed: 2019-08-12.

Quinlan, M., Au, T.C., Zhu, J., Stiurca, N., and Stone, P. (2010). Bringing simulation to life: A mixed reality autonomous intersection. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 6083–6088.

SAE, T. (2016). Definitions for terms related to driving automation systems for on-road motor vehicles. *SAE Standard J3016*.

Son, T.D., Bhave, A., and Van der Auweraer, H. (2019). Simulation-based testing framework for autonomous driving development. In *2019 IEEE International Conference on Mechatronics (ICM)*, volume 1, 576–583.

Son, T.D., Hubrechts, J., Awatsu, L., Bhave, A., and Van der Auweraer, H. (2018). A simulation-based testing and validation framework for adas development. In *2018 7th Transport Research Arena TRA*.

Stellet, J.E., Zofka, M.R., Schumacher, J., Schamm, T., Niewels, F., and Zöllner, J.M. (2015). Testing of advanced driver assistance towards automated driving: A survey and taxonomy on existing approaches and open questions. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 1455–1462.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Zofka, M.R., Essinger, M., Fleck, T., Kohlhaas, R., and Zöllner, J.M. (2018). The sleepwalker framework: Verification and validation of autonomous vehicles by mixed reality lidar stimulation. In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 151–157.