IEEE Transactions on Field Robotics

Received 7 April 2025; revised 24 June 2025; accepted 11 August 2025.
Date of publication 19 August 2025; date of current version 18 September 2025.
This article was recommended by Executive Editor Sanjiv Singh.

Digital Object Identifier 10.1109/TFR.2025.3600195

A Perception-Based Architecture for Autonomous Convoying in GNSS-Denied Areas

ALEXANDER BIENEMANN (Graduate Student Member, IEEE), LUKAS BEER (Graduate Student Member, IEEE), ANDREAS REICH, THOMAS STEINECKER, BIANCA FORKEL (Member, IEEE), ANTON BACKHAUS (Graduate Student Member, IEEE), PHILIPP BERTHOLD, JUAN DAVID GONZÁLEZ (Graduate Student Member, IEEE), PETER MORTIMER (Graduate Student Member, IEEE), THORSTEN LUETTEL (Member, IEEE), AND MIRKO MAEHLISCH

Institute for Autonomous Driving, University of the Bundeswehr Munich, 85579 Neubiberg, Germany

CORRESPONDING AUTHOR: ALEXANDER BIENEMANN (alexander.bienemann@unibw.de)

This work was supported in part by dtec.bw - Digitalization and Technology Research Center of the Bundeswehr [Project MORE], which is funded by the European Union - NextGenerationEU, and in part by the Federal Office of Bundeswehr Equipment, Information Technology and In-Service Support (BAAINBw).

(Systems Article)

ABSTRACT In this article, we present a perception-based full-stack system for autonomous vehicle following that does not rely on accurate global localization or map data. Our architecture consists of modules for vehicle communication, localization, object tracking, waypoint management, static environment modeling, trajectory planning, and control, which all are covered in the article. To test our system, we conducted several practical experiments in various scenarios on our two autonomous vehicles. Those experiments include the handling of static and dynamic obstacles, driving on- and off-road under different light and weather conditions with distances between the vehicles ranging from 5 to 100 m and with speeds of up to 20 m/s. Furthermore, we showcased our system's performance during the 12th European Land Robot Trial (ELROB) 2024, where our institute participated in the convoying scenario. The tests from the trial and our own experiments showed satisfactory results. Our system archives a high path-following accuracy and is able to cope with various challenging scenarios.

INDEX TERMS Autonomous driving, convoying, off-road, simultaneous localization and mapping (SLAM), tracking, vehicle following.

I. INTRODUCTION

THE term convoy typically refers to a group of vehicles or ships traveling together to a common destination. With an increasing degree of automation of vehicles, the idea of automated driving in convoys has also emerged. The first significant studies in this field were conducted in the nineties, where automated driving of a platoon of trucks was demonstrated as a part of the PATH project in California [1]. The aim of this study was to assess the impact of convoying on energy savings by automating longitudinal movement and distance control. Cooperative driving was achieved by using vehicle-to-vehicle (V2V) communication and radar sensors [2]. Since then, interest from academia and industry has only increased [3]. This is no surprise considering the benefits of autonomous convoys. Especially, in the field of logistics, convoying can reduce fuel consumption and has the potential to cut transportation costs and carbon dioxide emissions significantly [4]. In regular road traffic, convoy systems can improve the efficiency of traffic flow and safety [5].

FIGURE 1. Our first test vehicle (left) autonomously following our second test vehicle (right) through an off-road area.

Another common application is the military convoy, where the goal is to navigate a group of autonomous vehicles through dangerous territory (e.g., war zones or remote regions) to deliver supplies and aid [6]. This challenging scenario poses several difficulties to an autonomous convoying system, including driving in unstructured areas (see Fig. 1), handling of varying path conditions (e.g., driving on dirt-covered roads, asphalt, or through tall grass), and maintaining accurate path following to navigate through narrow passages. In addition, there may be no map data nor a reliable Global Navigation Satellite System (GNSS) signal available due to jamming or occlusions. Therefore, the convoying system can only use data from the vehicle's onboard sensors, like radar, camera, and light detection and ranging (LiDAR).

The European Land Robot Trial (ELROB)[1] is an event that aims to test the latest research and development in the area of autonomous off-road ground systems in challenging real-world military tasks. The event is organized as a trial and not a competition, where participants can test the performance of their systems in up to five different scenarios, including the military convoy. In order to recreate the challenges of those scenarios as realistically as possible, the organizers develop them in close consultation with members of the German Federal Armed Forces. The participants receive a brief description of some of the expected situations beforehand. To make it harder and more realistic, the actual challenges arise during the trial run. For the 12th ELROB 2024 event, the following description was given.

1) A group of at least two vehicles has to deliver goods to an approximately 6-km distant goal.
2) The expected environment is a mixture of semi-urban, nonurban, and hilly terrain with roads and paths covered by asphalt, dirt, bushes, trees, grass, sand, water, ditches, and pot holes.
3) There will be dynamic and static obstacles. Dead ends, sharp turns, road blockages, and narrow passages might occur.
4) The environment will contain GNSS-denied areas.
5) At least one vehicle should be remotely controllable from a control station, or the other vehicle.
6) The convoy should be able to disconnect and reconnect during operation, and it may be necessary to swap the roles of lead and follower vehicle.
7) The gap between the vehicles needs to be adjustable and might go up to a distance of approximately 100 m (nonline-of-sight).

The goal of our work was to develop a full-stack system for the above-mentioned application of autonomous military convoying. We wanted to create a software architecture that is capable of overcoming the challenges associated with this task. To test our system under realistic conditions, we participated in the 2024 ELROB challenge. But first, we started our research by analyzing related work.

When examining other publications in the field of autonomous convoying, it is noticeable that most studies only deal with individual aspects of this task, such as the reduction of fuel consumption [7], [8], the formation of platoons of multiple connected and automated vehicles [9], [10], [11], vehicle control [12], or the tracking for vehicle following [13]. Studies like this provide useful insights but are often based on simulations or simulate missing components of the full-stack system and thus assume ideal conditions. When integrated into a complete system in a real vehicle, neglecting factors such as latency times, data loss, limited computing power, or measurement noise will lead to errors and insufficiencies. It is therefore important to carry out practical experiments where the complete system is tested in real life. In the following, only works that do so are examined.

A significant milestone in the area of convoying was the 2011 Grand Cooperative Driving Challenge (GCDC), where the objective was to develop a system for cooperative adaptive cruise control (CACC) for automated driving within a heterogeneous platoon [14]. All GCDC participants used GNSS positioning data for localization and exchanged it with each other via V2V communication. The automated platoon drove along a closed-off 6-km section of a highway with speeds ranging from 0 to 23 m/s. Team AnnieWAY, the winners of the GCDC, used a model predictive controller (MPC) to calculate a desired acceleration and two feedforward controllers for the low-level control of brake and throttle [15]. During the challenge, occlusions from a bridge and low-quality GNSS data from other vehicles led to major issues. To overcome these issues, Team AnnieWAY used a radar sensor and merged the other vehicles' communicated positions with the measured radar data.

A different promising team participating at the GCDC was Team Chalmers [16]. This team also used GNSS for localization and an MPC for the control of the ego vehicle's longitudinal motion. In addition, a sensor data fusion of LiDAR, radar, camera, and real-time kinematic (RTK) GNSS data was used to estimate the states (position, velocity, and acceleration) of the lead, ego, and proceeding vehicle and the relative distance between them. We point out that all systems presented at the GCDC are intended for CACC and therefore

[1] https://elrob.org

neither take into account other road users apart from the lead or preceding vehicle nor lateral movement.

Zhao et al. [17] focused on military convoying. The idea is that all vehicles in the platoon follow the path of the first vehicle in the convoy and not the preceding vehicle's path. To achieve this, all vehicles exchange their position through wireless radio communication. The ego position is estimated by fusing data from a GNSS sensor, an Inertial Navigation System (INS), and odometry. This is done to compensate for poor or lost position data in GNSS-denied areas. The exchanged position is also used as an initial guess for LiDAR and camera-based tracking. Tracking data is fused together with exchanged position data using an extended Kalman filter (EKF) to estimate the pose and speed of the preceding vehicle. A proportional–integral–derivative (PID) controller is managing the ego vehicle's speed based on the distance to the preceding vehicle. A second PID controller is managing the lateral movements. Obstacles are avoided by calculating a set of trajectories and selecting the optimal trajectory that avoids obstacles at the lowest cost.

Another approach on how to achieve autonomous convoying is presented in [18]. There, a Monte Carlo particle filter is used to localize the ego vehicle in an offline map [19]. To do so, the filter receives data from an RTK GNSS sensor, a LiDAR, and an odometer. The authors point out that the RTK GNSS data is only used once during the initialization of the system. A mapping module updates a priori recorded occupancy grid map (OGM) with new obstacle information. Obstacles are avoided by slowing down and stopping until the blocked path is free again. Tracking of the lead vehicle is done by fusing LiDAR data with the output from a pretrained deep regression network [20] applied to the camera image. Longitudinal movement is controlled by using a PID controller and lateral movement by using an MPC.

So far, all of the mentioned works require GNSS to function, or at least to initialize. This means the system will not be able to work properly when started or restarted in a large GNSS-denied area. Furthermore, none of the systems will be able to cope with the nonline-of-sight criterion of the ELROB. To achieve localization without relying on GNSS, [21] employs a visual simultaneous localization and mapping (SLAM) algorithm that uses monocular vision. On the lead vehicle, the SLAM algorithm is coupled with a trajectory creation procedure, and both map and trajectory updates are exchanged online through Wireless Fidelity (WiFi). Doing so, the system does not require a tracking algorithm and the convoy can drive with large distances between the vehicles where no line of sight is given. Control of lateral and longitudinal movement is decoupled and nonlinear control techniques are applied. Although the idea of using SLAM for convoying is great, the use of vision-based SLAM will lead to problems under poor visibility conditions, such as driving in the rain or darkness. In addition, SLAM relies on landmarks, which are not always sufficiently available when driving offroad (e.g., flat steppe). The proposed system also only deals with obstacles in the longitudinal plane by slowing down or stopping, just like in [18].

A second popular strategy for convoying without relying on GNSS is the use of perception-based tracking for trajectory generation paired with dead-reckoning for localization. Here, the drift of the ego motion estimation can be ignored if the convoying system uses a moving reference frame for the lead vehicle's trajectory [22]. Typically, this type of system also does not apply V2V communication. Such a system is used in [6] for autonomous military convoying. There, a camera system with a color tracker [23] measures the bearing and range to the lead vehicle. This information is fed together with the ego vehicle's speed and heading into a nonlinear observer, which then estimates the lead vehicle's pose and speed, the ego vehicle's position, and the heading of a look-ahead point. Those values are used by a state space controller with gain scheduling to calculate desired speed and steering signals.

A different perception-based tracking approach is presented in [24]. Fries et al. [13] employed a combination of template- and model-based vehicle tracking using a monocular camera to create points from the lead vehicle's estimated movement. Sequential quadratic programming is applied to generate a continuous-curvature trajectory from those points. Lateral and longitudinal vehicle movements are again decoupled and separately controlled by a state and a PID controller.

The most recent example of this type of strategy is [25]. This system measures the lead vehicle's position with radar, camera, and LiDAR sensors and stores the measurements over time. A novel strategy is proposed on how to rate the relevance of those measurements and a spline-approximation is applied to achieve continuity in the estimated path. Once more, a PID controller is used for the longitudinal movement and a state space controller with gain scheduling for the lateral one. This and the two previously mentioned systems do not deal with dynamic obstacles. The main disadvantage of using perception-based tracking for convoying is that the system will fail if visual contact is lost, whether due to occlusions (bushes or other vehicles) or large desired distances between the vehicles.

In summary, all systems have one or more of the following drawbacks. They either rely on prerecorded maps or GNSS, ignore dynamic obstacles, or are incapable of dealing with large intervehicle distances. Also, none of the works that refrain from using GNSS evaluate their systems performance during bad light and weather conditions. Therefore, the novelty of our work comes from the proposal of an improved combination of modules into a system for autonomous driving in military convoys, which:

1) combines perception-based tracking and SLAM…;
2) …and therefore does not rely on GNSS or prerecorded map data…;
3) …and also enables driving with large intervehicle distances;
4) tracks and deals with dynamic obstacles; and
5) achieves a high path-following accuracy.

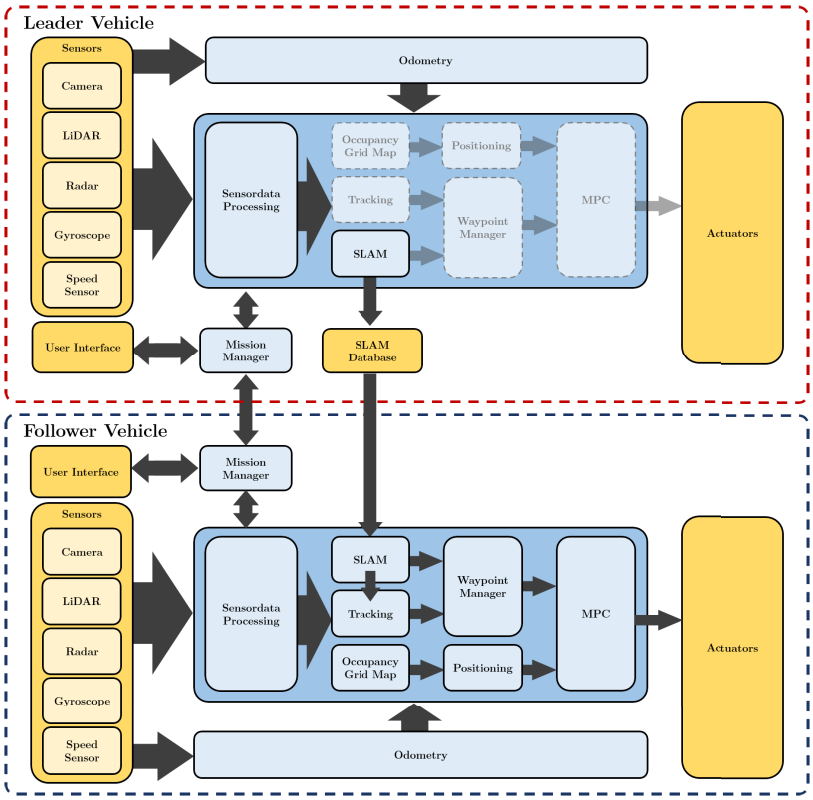


FIGURE 2. Overview of the full-stack system running on the leader and follower vehicle in the case of two vehicles. All blue boxes are part of our software architecture. The first vehicle in the convoy is operated manually (leader vehicle). Accordingly, some modules of the system are in idle mode (dashed grayed-out boxes). They are activated immediately in the event of a role swap.

Even more novelty comes from conducting various practical experiments on a real vehicle, including testing the system's behavior under bad weather and light conditions and participating at the ELROB to test our system under challenging conditions. The rest of this work has the following structure. Our proposed system and its individual components are explained in Section II. In Section III, we show our experimental results and compare them to those of other works. Our performance at the ELROB and the lessons learned during testing and the trial are covered in Section IV and V. Finally, Section VI concludes this article.

II. SOFTWARE ARCHITECTURE

In this section, we discuss our proposed full-stack system and its components. An overview of the system running on a leader and a follower vehicle in the case of two vehicles is depicted in Fig. 2. Since our institute only has two fully functional autonomous vehicles, we were only able to verify our system on a convoy consisting of two vehicles. Nevertheless, the system was designed with scalability to multiple vehicles in mind, which will be taken up and discussed later in Sections IV and V-D. We start with a brief description of how our system works, followed by a more detailed explanation of each individual module.

All vehicles in the convoy are connected to each other and use V2V communication. The communication itself is handled by the Mission Manager module. Besides that, the Mission Manager also acts as an interface between the convoy system and the graphical user interface (GUI). This allows an operator to interact with each vehicle of the entire platoon from any vehicle using the GUI. It is also possible to run the system on a separate computer to use it as a control station. For ego motion estimation, the odometry module uses a simple dead reckoning algorithm. The sensor data processing module consists of a panoptic segmentation and a clustering algorithm for preprocessing of the data from the camera, LiDAR, and radar sensors.

A multiobject-tracker (MOT) is used to estimate the pose, speed, and acceleration of other dynamic objects, including the lead vehicle. At system start, the MOT receives the lead vehicle's pose from the SLAM module and uses it to initialize the tracking. If SLAM is, for whatever reason, not available, a region of interest is set up in front of the ego vehicle, in which the MOT searches for a moving object. Technically, this way any vehicle could be tracked and, if so desired, assume the role of the lead vehicle without having to rely on V2V. This is supported by the Mission Manager. The downside is that driving with large intervehicle distances will not be possible due to SLAM not being available. Panoptic

segmentation is used by the MOT to ensure that only moving objects of the class ''vehicle'' can be labeled as the lead vehicle and not a random pedestrian that was moving inside the region of interest. Once the tracking is set up, points are created based on the estimated pose of the lead vehicle. Those points and the estimated speed of the lead vehicle are forwarded to the Waypoint Manager module for reference trajectory generation.

To enable driving with large intervehicle distances, the system also employs an SLAM algorithm. On the leader vehicle, the SLAM module identifies landmarks and uses them for mapping and to estimate the vehicle's ego pose in the map. Estimated pose, measured speed, and generated map are locally stored and frequently updated in a database on the lead vehicle's own computer. On the follower vehicle, the SLAM module accesses the database through V2V communication and identifies landmarks to estimate its own ego pose on the map. The lead vehicle's speed and estimated pose, and the estimated ego pose, are forwarded to the Waypoint Manager. In case of multiple vehicles, the SLAM module would create a copy of the lead vehicle's database and locally store it with its own estimated ego pose and measured speed in its own database so that the information could then be accessed by the proceeding vehicle. This way, all vehicles in the convoy would receive the driven route of the leader vehicle as a reference path and the pose and speed of the preceding vehicle for distance keeping.

Another advantage of using SLAM and tracking simultaneously is the additional redundancy. The Waypoint Manager can use data from both modules to create a reference trajectory. If the MOT fails and tracking is interrupted (e.g., because of occlusions), then the Waypoint Manager will use the information from the SLAM module. The MOT will also try to use data from the SLAM module to reinitialize the tracking if possible. If the SLAM fails because there is an insufficient number of landmarks or because V2V communication has been interrupted or jammed, then the Waypoint Manager will use the information from the tracking module. Regardless of which module provides the data, a curvature-corrected moving average filter (CCMAF) [26] is used by the Waypoint Manager to filter the noise and smooth the reference path before generating a trajectory.

One of the requirements specified for ELROB 2024 was that at least one vehicle must be able to be controlled remotely from a control station or the other vehicle. With our proposed system, any vehicle can be remotely controlled by any vehicle of the platoon if necessary. The operator can simply use a GUI to select a vehicle in the convoy and then click anywhere on a map to specify a desired pose. The Mission Manager handles everything else. A Hybrid A* algorithm is executed and calculates a collision-free path from the vehicle's starting point to the desired pose. To do so, a different module creates an OGM and provides the Hybrid A* with the necessary information about drivable space.

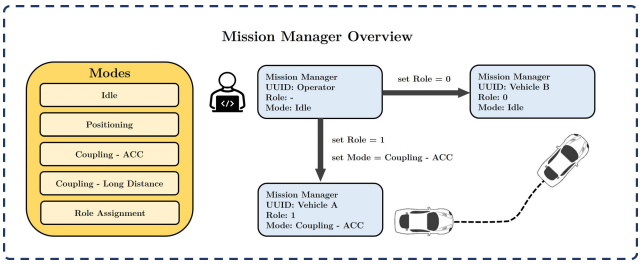


FIGURE 3. Overview of the Mission Manager. The box titled "modes" displays the different modes that can be assigned to the vehicles of the convoy. The blue boxes show an external operator assigning roles and modes to the vehicles through the corresponding Mission Manager modules.

Depending on the mode commanded by the Mission Manager an MPC with dead-time compensation uses data from either the Hybrid A* or the Waypoint Manager. The MPC uses a mathematical model of the vehicle to make predictions about its future states. Through optimization, the best trajectory that respects the vehicle's dynamic, kinematic, and spatial constraints is found, and the respective control inputs to navigate the vehicle along this trajectory are computed. The control commands calculated by the MPC are then converted into signals for the vehicle's actuators. This is done on an embedded computer and this process is not part of our proposed system.

A. MISSION MANAGER

The Mission Manager handles communication between all convoy members, sharing of essential information for monitoring, maneuver commanding, and convoy role assignments. Each convoy member runs its own Mission Manager instance, which communicates with other Mission Managers. This distributed approach enhances robustness, ensuring continued operation if a single convoy member fails. An overview is given in Fig. 3.

Each Mission Manager shares its unique identifier and convoy role at a specified frequency while also listening to updates from other members to build a local fleet data record. In this way, each actor continuously updates its understanding of the fleet. Once the fleet data is available, an operator can assign roles to each member using their unique identifier. Each role specifies a position in the convoy formation, such as the lead vehicle (role $= 0$), the first follower (role $= 1$), the second follower (role $= 2$), and so on. The operator must also ensure that the vehicles are properly aligned with respect to their assigned roles. This can be achieved either by initializing them in the desired formation or by positioning the vehicles manually through the interface (as described in Section II-H).

Each Mission Manager also shares its global position with the others. Once two entities are localized in a global reference frame (or frames with known transformations), positioning commands can be effectively communicated. In our case, the system uses the SLAM frame as the global reference frame (details in Section II-F). Besides that, the

system also supports GNSS as the global reference frame if wanted. We do not use that, as in our scenario GNSS is not available. Although global positioning is not required for nominal convoy operation (when the convoy is in coupling mode), it is advantageous during the initial setup phase to position the convoy members correctly. Further details on the importance of initialization alignment for convoy operation are provided in Section II-G.

The convoy system supports multiple modes that an operator can assign remotely. Initially, when the convoy starts, each vehicle is *idle* and waits for further instructions. When the vehicles are aligned (the preceding vehicle is properly oriented in front of the follower), the operator can start the convoy by assigning roles and initiating *coupling*. Two coupling modes are available: automatic cruise control (ACC), which provides a tightly coupled convoy where vehicles remain within line of sight, and long-distance mode, which is used when large intervehicle distances are required. In long-distance mode, tracking relies on SLAM-only, as a loss of line of sight is expected. The positioning mode can be used to maneuver the vehicles of the fleet to a desired goal point. If desired, vehicles can be decoupled from the fleet by putting them in *idle*.

B. ODOMETRY

The odometry module uses yaw rate $\dot{\psi}$ and speed v to estimate the vehicle's ego motion. The currently measured yaw rate comes from the gyroscope of the vehicle's electronic stability control. To reduce the effects of sensor drift, an offset adjustment is applied. The offset value $\dot{\psi}_{\text{off}}$ is determined with a Kalman filter. The speed used for the estimation is the average of the four measured wheel speeds of the vehicle. Yaw rate and speed are integrated to estimate the ego vehicle's pose

$$x_{k+1} = x_k + v_k \cos(\psi_k)\, \Delta t_s \tag{1}$$

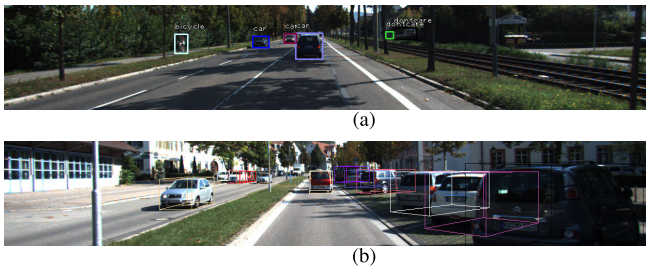
$$y_{k+1} = y_k + v_k \sin(\psi_k)\, \Delta t_s \tag{2}$$

$$\psi_{k+1} = \psi_k + \left(\dot{\psi}_k - \dot{\psi}_{\text{off}}\right) \Delta t_s. \tag{3}$$

Here, x_k, y_k, and ψ_k denote the vehicle's position in a fixed coordinate system and the yaw angle at time step k. Δt_s is the sampling time of the faster measurement. As stated in [6], the drift of the ego motion estimation can be neglected since the target points of the reference path are created from the lead vehicle's relative position to the ego vehicle. We use this strategy for ego motion estimation instead of our SLAM algorithm because this method provides a jump-free pose as an input for the MPC, which then results in a more precise and smoother vehicle control.

C. OBJECT DETECTION

Object detection plays a pivotal role in enabling accurate perception and tracking of multiple objects within the environment of an ego vehicle. In our system, we utilize both camera and LiDAR sensors to perform object detection, taking advantage of their complementary characteristics.



(a)

(b)

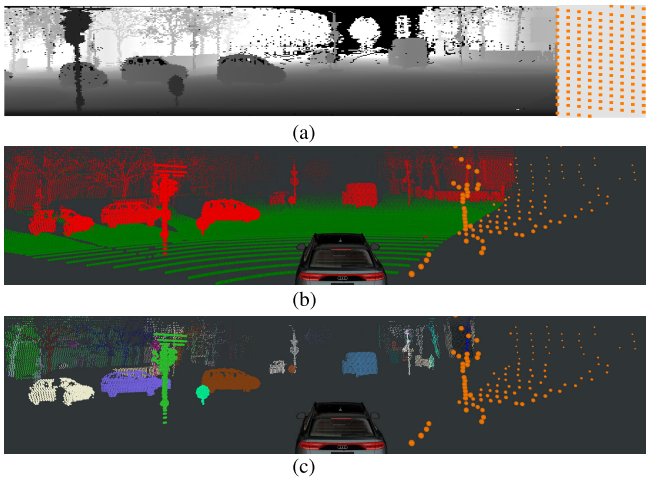
FIGURE 4. Examples of object detections using monocular camera images, (a) showcasing both 2-D bounding boxes and (b) their corresponding 3-D representations. These detections were generated by CenterTrack [27], a monocular 3-D object detector.

1) CAMERA

In our work, we employ off-the-shelf neural networks for 2-D object detection. Such 2-D bounding boxes are depicted in Fig. 4. Our system is designed to be compatible with a wide range of 2-D object detectors. Even though our object tracking architecture operates in the 3-D domain, we are able to leverage 2-D detections in the measurement update step of the EKF. Optionally, our system can also utilize monocular 3-D object detections, such as those generated by CenterTrack [27], to initialize track hypotheses with camera sensors. These detectors operate on single RGB images and extend the detected 2-D bounding boxes by estimating additional information, including object distance, 3-D dimensions, and aspect angles. This polar representation of a 3-D bounding box can be converted into a Cartesian representation, as shown in Fig. 4(b). Such an approach mimics aspects of human perception at larger distances, where depth perception relies more heavily on prior knowledge about the size and shape of an object as the stereo vision is limited to a few meters.

2) LiDAR

For object detection in LiDAR data, we propose our own custom approach specifically created for this work to overcome the high latency challenges typically encountered in deep learning-based methods. Deep learning approaches using common backbones, such as PointPillars [28] or sparse convolutions [29], can exhibit substantial computational delays due to their complex processing pipelines. These delays can hinder real-time performance, making them less ideal for time-critical applications. Our approach, detailed in [30], is called continuous clustering and is designed to address these latency challenges while maintaining robust detection capabilities. The continuous clustering builds on our previous work [31] by clustering LiDAR points based on their distances to extract coherent object instances efficiently. The clustering step is followed by fitting bounding boxes around these instances. The bounding box fitting is guided by identifying the longest straight line found within the corresponding LiDAR points. In contrast to other methods, where LiDAR



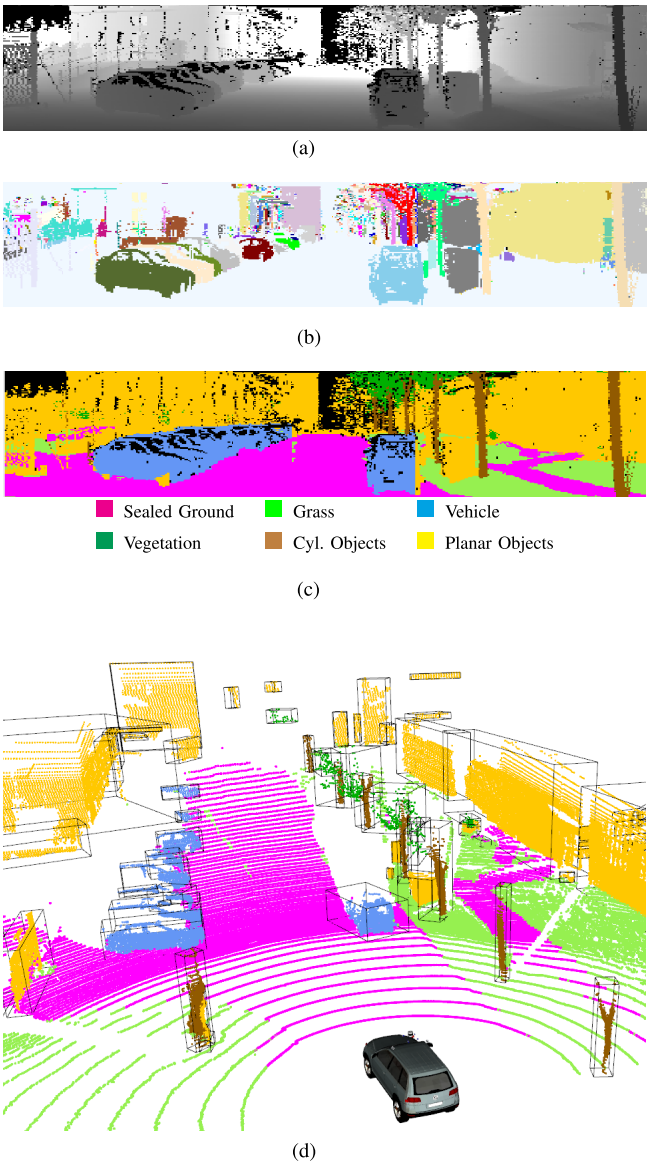
(a) (b) (c)

FIGURE 5. Visualization of our continuous clustering approach for object detection in LiDAR data, demonstrating (a) processing of laser firings, (b) ground classification, and (c) final clustering results.

points are accumulated over a full sensor revolution which leads to an artificial latency of 100 ms, our approach is able to process LiDAR points continuously. The continuous clustering begins processing as soon as a new column of the LiDAR range image becomes available. The processing steps are visualized in Fig. 5. It outputs completed clusters for which no new points are expected based on a specified distance threshold. This incremental processing reduces the latency of individual object instances to as low as 5 ms relative to the timestamp of the most recent LiDAR point within the cluster. This enhancement ensures rapid detection and improved responsiveness, making our system suitable for real-time object detection tasks in dynamic environments.

D. PANOPTIC SEGMENTATION

The panoptic segmentation is a combination of instance and semantic segmentation. It results in a set of individual object instances, each labeled with a class label. Whereas many other approaches focus on a combined model for both tasks, we use our own specially developed two-stage approach [32], [33]. Instead of fully relying on one model, we combine classic algorithms from the instance segmentation (described in Section II-C) and a semantic segmentation. Due to the continuous clustering approach from the instance segmentation, we also use continuous data processing for the semantic segmentation. We use the CENet [34] as a semantic segmentation model, mainly due to its high inference speed. Instead of using the full rotation (e.g., 1700 columns) as input, we use 81 columns from the continuous range image. Subsequently, we combine both the instance and semantic segmentation results and extract bounding boxes for each object instance. The merging of the instance and semantic segmentation is done by combining similar classes and assigning the class with the highest confidence, if they are within one cluster. Distinct classes are separated into different



(a) (b) (c) (d)

FIGURE 6. Visualization of the panoptic segmentation. The depth image is shown in (a), the instance segmentation in (b), and the semantic segmentation in (c). The result of the merging step can be seen in (d). The instance segmentation does not differentiate between vegetation and tree trunks, whereas the panoptic segmentation does. Furthermore, wrongly classified points from the semantic segmentation (e.g., the vehicles) are improved.

panoptic instances. As a result, we also split, for example, the tree trunk and the tree crown into two separate objects. Both objects can now be handled individually by the SLAM. The resulting semantic objects o with the semantic class c are denoted as o^c. A more detailed description of our panoptic segmentation can be found in [33]. A result of the panoptic segmentation is shown in Fig. 6(d). However, we do not make use of the low-latency possibilities of the panoptic segmentation. Instead, we accumulate the instances for one rotation in order to simplify the processing and to reduce the complexity

FIGURE 7. **Tracking of the leader vehicle using data from both the camera (left) and LiDAR (right) for dynamic object tracking with an EKF. The blue LiDAR cluster in the right image is the tracked leader vehicle.**

of the system. Nevertheless, this still decreases the latency compared to the full rotation processing. A visualization of our approach is shown in Fig. 6.

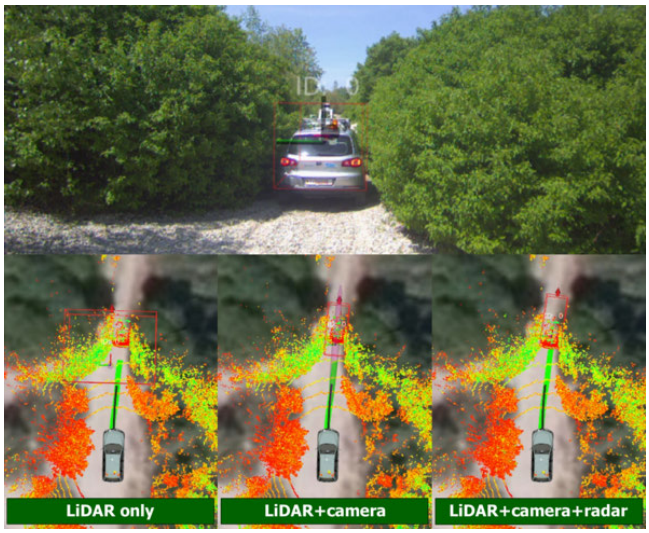
E. TRACKING

The tracking system employs a tracking-by-detection approach to monitor and estimate the positions of multiple objects around the ego vehicle. Operating in a 3-D coordinate space, it utilizes detections from both camera and LiDAR sensors (see Fig. 7) while using data from the camera, LiDAR, and radar sensors for state updates. The fusion of these multiple sensors helps maintain stable tracking even under challenging conditions, such as poor visibility, while also reducing false positives. Fig. 8 shows a scenario that nicely confirms this claim. Here, the lead vehicle drives through a narrow passage covered by bushes. If only the LiDAR sensor is used, the LiDAR points of the bushes are clustered with the points of the lead vehicle, which then leads to a loss of the track (see Fig. 8(a), bottom left). Using LiDAR and a camera together helps to avoid this problem. However, this time the camera detection has problems to properly estimate the depth and the track is lost again (see Fig. 8(b), bottom middle). Only when using LiDAR, camera, and radar together, tracking of the lead vehicle is possible in this challenging situation (see Fig. 8, both images, bottom right).

Various object classes, including pedestrians, bicycles, and vehicles, have unique state vectors and motion models. Here, we focus specifically on vehicle tracking, as it is the primary scenario for our convoy. The state vector for a vehicle is represented as

$$\mathbf{x}^{\text{track}} = \begin{pmatrix} x & y & z & l & w & h & \psi & v & \delta & a \end{pmatrix}^{\mathrm{T}} \quad (4)$$

where (x, y, z) denotes the position within a fixed odometry coordinate system, (l, w, h) specify the dimensions of the bounding box, ψ is the yaw angle, v is the longitudinal velocity, δ is the steering angle, and a represents the acceleration. Upon receiving sensor detections, the state of all track hypotheses, $\mathbf{x}_k$, is predicted to the state $\mathbf{x}^*_{k+1}$ for the time of the next measurement. The motion model employed is based on the bicycle kinematic model as found in [35, p. 21]. Next, expected measurements are computed for each track, and data association is performed using the Mahalanobis distance metric, accounting for the covariance of the



(a)

(b)

FIGURE 8. **Showcasing the advantage of using three different sensors for tracking. In narrow passages, a LiDAR-only approach results in the lead vehicle point cloud being clustered together with point clouds from other nearby objects. Adding a camera sensor prevents the clustering problem but still fails due to uncertain depth estimation. Only the use of all three sensors prevents the track from being lost. The green line is the reference trajectory generated from the lead vehicle's movement. The line is interrupted once the track is lost. The black line is the trajectory planned by the MPC. (a) Scene 1. (b) Scene 2.**

EKF innovation. To resolve associations between tracks and measurements, the Hungarian Algorithm [36] is used. The associated track's state vector is then updated using the corresponding measurement, and unassociated measurements lead to the creation of new track hypotheses. Tracks with a high Mahalanobis distance relative to all available measurements are excluded from updates. The MOT framework is shown in Fig. 9.

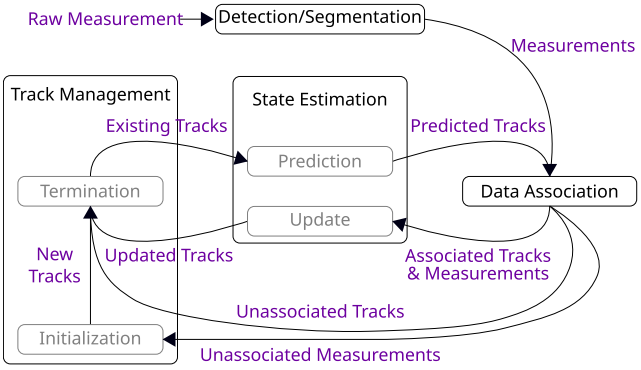


FIGURE 9. Architecture of the MOT. The illustration shows new raw measurements going through segmentation and detection, data association, resulting in a state estimation. The track management handles termination of unassociated tracks and measurements.

1) LiDAR

Measurements from the LiDAR are represented as

$$\mathbf{y}_{\text{LiDAR}} = \begin{pmatrix} x & y & z & l & w & h & \psi \end{pmatrix}^{\mathrm{T}}. \tag{5}$$

These measurements provide the geometric basis for initializing new track hypotheses. During the EKF update step, the measurement $\mathbf{y}_{\text{LiDAR}}$ is predicted from the state $\mathbf{x}^*_{k+1}$. For each track hypothesis, four measurements are predicted, corresponding to different sides of the bounding box. Only the most visible side is retained for association, whereas others are discarded after data association.

2) CAMERA

Track initialization from monocular 3-D object detections, such as those generated by CenterTrack [27], results in 3-D bounding boxes characterized by their center, dimensions, and orientation. Uncertainty is accounted for by assigning larger positional uncertainty along the line of sight, reflecting the challenges of monocular distance estimation. For measurement updates, we use 2-D bounding box detections

$$\mathbf{y}_{\text{cam2d}} = \begin{pmatrix} u_{\min} & v_{\min} & u_{\max} & v_{\max} \end{pmatrix}^{\mathrm{T}}. \tag{6}$$

These measurements are predicted from the state $\mathbf{x}^*_{k+1}$ through projection into the image frame, which helps reduce systematic biases and maintain state correlations.

3) RADAR

At the moment, radar measurements are used only for state updates and not for detection. All Doppler velocities incorporated within the track's bounding box are used as a measurement

$$\mathbf{y}_{\text{radar}} = \begin{pmatrix} v_1 & v_2 & , \ldots , & v_N \end{pmatrix}^{\mathrm{T}}. \tag{7}$$

Outlier velocities are filtered, and expected Doppler velocities are computed from the predicted state $\mathbf{x}^*_{k+1}$, enhancing motion estimation accuracy.

F. SLAM

Introducing SLAM into a convoy architecture has two main applications. One of them is enabling backward driving over long distances. In a stand-alone manner using a single vehicle, SLAM eliminates drift while driving backwards. To generate a path for driving backwards, most other approaches simply store the driven path of the ego vehicle in the odometry coordinate system while driving forward. When driving backwards, the vehicle is localized in the odometry coordinate system and tries to follow the stored path. Nevertheless, the drift of the odometry results in an offset between the actual driven path and the backward path. This can be avoided by using SLAM. Even though SLAM also has a drift, this drift is reduced and only occurs when driving into unmapped terrain. This is not the case when driving backwards, as by definition, we only drive along an already visited path.

The second use case of SLAM is cooperative localization for convoying. Having multiple vehicles that are equipped with similar sensor setups allows the use of SLAM in a cooperative manner, as it has already been introduced in [37]. The leader–follower principle of convoy simplifies the cooperative SLAM drastically. The same SLAM algorithm runs on all vehicles, but only the lead vehicle is responsible for the global map. All the other following vehicles only have to localize themselves in the global map. If new landmarks occur for one of the follower vehicles, they are only stored locally. If all vehicles use the same coordinate system, information of the relative position between the vehicles can be extracted. For the remainder of this article, the resulting leader for each follower is called SLAM-object.

Our system's SLAM algorithm is based on our own approach [38], which is a LiDAR-only SLAM that combines the GraphSLAM algorithm [39] and panoptic segmentation for the extraction of geometric primitives. We made some adjustments to our algorithm to make it work for the convoy scenario. In the following, we will explain the SLAM algorithm in more detail. As is common in the literature, we will differentiate between the front-end and the back-end of the SLAM algorithm. While the front-end is responsible for the data association and the feature extraction, the back-end is responsible for the optimization of the map and the trajectory.

1) FRONT-END

The front-end of the SLAM algorithm is responsible for the data association and the feature extraction. The main idea of the SLAM algorithm is to extract geometric primitives and directly use them in the optimization. Because this is a computationally expensive task, a panoptic segmentation is used to extract semantic and instance information from the LiDAR data (see Section II-D). From the panoptic segmentation, a semantic object o^c is received, which is a set of 3-D points. Based on that, a geometric representation is extracted. Walls and fences are represented by four corner points of a plane, poles, and tree trunks are represented by cylinders, and

vegetation is represented by a single point. Doing so, dynamic objects are automatically neglected.

For the data association between the observations and the map, a simple nearest-neighbor search is used. This is sufficient enough, as we only have a small number of landmarks and we differentiate between the different kinds of them. Hence, only planes are matched with planes, cylinders with cylinders, and vegetation with vegetation.

2) BACK-END

Our localization is an SE(2) localization and therefore the vehicle's pose (x, y, ψ), described by the state $\mathbf{x}$, is estimated. The main idea is described in [38]. For the localization, a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is constructed, which consists of a set of vertices $\mathbf{V} = \{\mathbf{v}_0, \ldots, \mathbf{v}_n\}$ and edges $\mathbf{E} = \{e_0, \ldots, e_m\}$. The vehicle state $\mathbf{x}$ and the landmarks $\mathbf{L}(\mathbf{o}^c)$ are represented as vertices $\mathbf{v}_i$. The edges describe the relationship between different vertices. In this case, this is either the odometry for the relation between two vehicle states or the observations for the relation between vehicle state and landmark. The edges are weighted by the information matrix Ω_{ij}. Using our observations ${}^{\mathcal{G}}\mathbf{y}_{ij}$ and the expected observation ${}^{\mathcal{G}}\hat{\mathbf{y}}_{ij}$, which we receive from the map result in the following error function:

$$e_{ij}\left({}^{\mathcal{G}}\mathbf{y}_{ij}, \mathbf{v}_i, \mathbf{v}_j\right) = {}^{\mathcal{G}}\mathbf{y}_{ij} - {}^{\mathcal{G}}\hat{\mathbf{y}}_{ij} \tag{8}$$

with $\mathbf{v}_i = \mathbf{x}_i$ and $\mathbf{v}_j = L_j$. The error function is then weighted by the information matrix Ω_{ij} and results in the constraint J

$$J_{ij} = e\left({}^{\mathcal{G}}\mathbf{y}_{ij}, \mathbf{v}_i, \mathbf{v}_j\right)^{\mathrm{T}} \Omega_{ij} e\left({}^{\mathcal{G}}\mathbf{y}_{ij}, \mathbf{v}_i, \mathbf{v}_j\right). \tag{9}$$

For each kind of measurement, a specific error function is defined. J^{o} for the odometry, J^{cy} for cylinders, J^{b} for vegetation, and J^{p} for planes. During optimization, we use the center of the vegetation and the poles as a single point while we optimize the plane-corner point distance of the planes. We define the overall function to minimize as

$$J^{\mathrm{slam}}(\mathbf{V}) = \sum J^{\mathrm{o}} + \sum J^{\mathrm{cy}} + \sum J^{\mathrm{b}} + \sum J^{\mathrm{P}}. \tag{10}$$

The Gauss–Newton algorithm is used for optimization with the optimization framework g2o [40]. While this setup works for a pure SLAM system, several adaptations for the convoy architecture needed to be made, such as differentiating between the leader and all followers.

a) Leader

The leader is responsible for the global map. It is the only vehicle that writes the map to the map server and starts its position with the zero-pose. Our system does not use static obstacle avoidance when driving in coupling mode. The path driven by the lead vehicle is expected to be free from static obstacles, just like the path driven by the follower vehicles is expected to be free from static obstacles and therefore suitable for driving backwards. Therefore, it is important that the position relative to the environment is as accurate as possible. To avoid a drift of the map while driving backwards, landmarks are fixed during optimization after they have been observed long enough by the leader.

b) Follower

The followers are only responsible for their own localization and the relative position to the leader. For a simplification of the cooperative SLAM problem, the follower does not optimize the map any further. Instead, the landmarks are expected to be fixed. This has multiple advantages. For one, the follower does not have to communicate with the map server, which reduces the communication overhead. Furthermore, changing the map would also result in a forced relocalization of all participants of the convoy, which may lead to a jump in the position.

3) LOCALIZATION STATES

For the convoy scenario, we had to adjust the SLAM algorithm to differentiate between the following states: initialization, localization, and role-switching.

a) Initialization

The initialization is the first step of the localization. All followers wait until the leader has created a map. Whereas the leader starts with the zero-pose, each follower needs an initial guess for its own position. Here, the pure nearest-neighbor search is not sufficient. Therefore, a probabilistic matching approach is used. The leader shares information about its position $\mathbf{x}^{\mathrm{leader}}$ and the map. The panoptic segmentation of the follower returns semantic objects of the surrounding n vehicles. For each vehicle, we extract the bounding box and therefore the center and the orientation, which results in the hypotheses $\hat{\mathbf{x}}_{0:n}^{\mathrm{leader}}$. Nevertheless, the panoptic segmentation only returns observations and no tracked vehicles, which means that the orientation of the vehicle is unknown. Therefore, further hypotheses are introduced, which are rotated by 90°, 180°, and 270°. Hence, a full set of $4n$ hypotheses $\hat{\mathbf{x}}_{0:4n}^{\mathrm{leader}}$ is obtained. As the leader's hypotheses are relative observations from the follower vehicle, a hypotheses of the follower's position in the map can be calculated by

$$\hat{\mathbf{x}}_{0:4n}^{\mathrm{follower}} = \left(\hat{\mathbf{x}}_{0:4n}^{\mathrm{leader}}\right)^{-1} \cdot \mathbf{x}^{\mathrm{leader}}. \tag{11}$$

Each hypothesis is weighted by matching the current observations with the map. The weight is calculated by the distance between the current observation and the expected observation. The most likely hypothesis is used as the initial guess for the follower's position. This approach allows us to initialize the convoy without any manual interaction or the use of GNSS.

b) Localization

The localization is the regular operation mode of the convoy. In this mode, the leader continuously optimizes the map and transmits the updates and its own current position to the followers. The followers continuously receive map updates and localize themselves in the map. Moreover, a health check

is performed. Both the follower and the leader need to see the same landmarks at similar positions. This is done by comparing the landmark IDs from the leader and the follower at similar or the same positions. We calculate the graph similarity SG by

$$\mathrm{SG} = \frac{\text{number of same landmarks}}{\text{total number of observations}}. \tag{12}$$

If SG is below a certain threshold but a sufficient amount of landmarks is observed, the position of the follower is set to be invalid and needs to be reset.

c) Role-Switching

The role-switching is the last state of the localization. If a role-switching is initiated, a chosen follower becomes the leader, and the leader becomes a follower. In this case, the new leader updates the map to the map server and the follower initializes itself. This time, a hypothesis of the new follower's position is not needed. As the new follower used to be a leader before, its initial position is assumed to be the last position when the vehicle was still a leader.

4) COMMUNICATION

The communication of the SLAM is decentralized. Each vehicle has its own database, just as the map server is also a database, which always runs on the leader. If the leader changes, the map server is also changed. The communication is done by a simple table structure. The leader writes the map to the map server, and the followers read the map from the map server. Furthermore, the followers only read the information since the last query. This is ensured by an increasing ID in the map server.

Nevertheless, the communication itself has its drawbacks. While it is easy to use, it is not the most efficient way of communication due to the overhead of the database and the writing and reading on the hard drive. Therefore, the database itself introduces a latency to the system. However, in our convoy architecture, this latency can be neglected, as the vehicles are driving either at a moderate speed or with a high distance to each other. Fast and stable communication is not a topic that is covered in this article.

5) SLAM-OBJECT-COORDINATOR (SLOC)

In our proposed system, the Waypoint Manager module decides whether data from the cooperative SLAM or from the MOT is used for the generation of the reference trajectory. In order to keep everything in one coordinate system, we came up with our own approach that we call the SLOC. The SLOC is the main interface between the SLAM and the Waypoint Manager and fulfills two tasks. First, it receives the position of the lead vehicle and stores it in a buffer. Second, it publishes a so-called *Delayed-Object*, which is based on the buffered positions of the lead vehicle. It is desirable that all vehicles are controlled in one common coordinate system, which in our case is the odometry coordinate system. The SLAM coordinate system is not suitable for this, as it is prone to potential jumps. Therefore, a transformation of the SLAM-object to the odometry coordinate system needs to be done. However, there is a drift between the SLAM and the odometry coordinate system. If the SLAM-object were used as it is, a drift would be introduced to the path. Especially for far-distant objects, the resulting error would be significant. Therefore, the Delayed-Object, which is a virtual object in the odometry coordinate system, is introduced. The Delayed-Object is published to the Waypoint Manager as soon as the distance on the lead-vehicle trajectory between the follower and the buffered object is below a certain threshold. We are aware that this way errors from the follower position and the leader position are combined. However, the Waypoint Manager deals with the accumulated errors and smoothes the path.

G. WAYPOINT MANAGER

The Waypoint Manager is a module that we built from scratch for the convoying scenario. It is responsible for managing the follower task of its lead vehicle. More specifically, it interprets the driving mode of the lead vehicle (e.g., *holding*, *driving forward*, or *driving backward*) and adjusts its own behavior accordingly. The driving mode is estimated using a simple hidden Markov model (HMM), where the three driving modes are the hidden states. Observations include positive velocity ($v_{\text{lead}} > v_{\text{threshold}}$), negative velocity ($v_{\text{lead}} < v_{\text{threshold}}$), and no velocity otherwise. At first glance, a direct correlation could be assumed, e.g., if positive velocity is observed, the mode is *driving forward*. However, due to measurement noise and tracking issues, this approach could lead to undesired behaviors. For example, if its lead vehicle is stationary but the follower briefly observes a negative velocity below the threshold $v_{\text{threshold}}$, the follower could incorrectly start driving backwards. Increasing the threshold to prevent this would reduce the system's responsiveness. Instead, driving backwards should only be executed if the HMM detects that its lead vehicle intends to drive backwards and the follower is stationary. Transition and emission probabilities for the HMM were manually chosen, resulting in robust and reliable performance during testing and the ELROB trial.

The Waypoint Manager is also responsible for path generation. Previous methods typically gathered a history of the lead vehicle's tracked positions and fit noisy trajectories using models like clothoids or splines. Instead, we pursued a different approach, assuming the lead vehicle operates under the same or more restrictive kinematic constraints as the follower. Rather than performing complex path generation, we focused on reconstructing the original path. First, we tried the moving average (MA), as this is the most used model-free reconstruction algorithm for regular curves. However, MA suffers from an inward-bending effect on curves. Therefore, we came up with our own approach, which is the curvature corrected MA (CCMA) [26]. The CCMA is a symmetric smoothing algorithm that overcomes the issue of inward-bending via curvature correction. CCMA offers similar smoothing

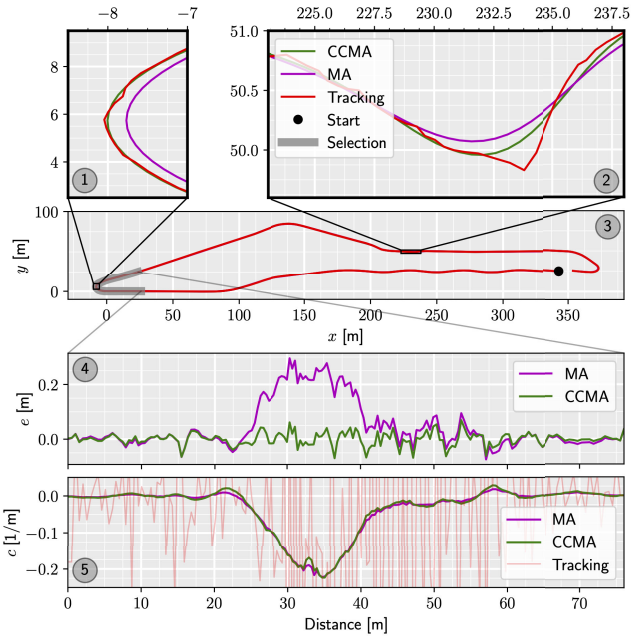


FIGURE 10. Qualitative results of the CCMA during a convoy scenario on our test site. In (3), the traversed path with its starting point is shown. Two noteworthy areas are highlighted in (1) and (2): in (1), the CCMA's superior accuracy over the MA is demonstrated, while (2) shows the largest reconstruction error, caused by strong outliers. The errors and curvatures for the gray-highlighted area in (3) are plotted in (4) and (5), respectively. In (4), it becomes clear that CCMA produces errors symmetrically distributed around the unsmoothed path, while the MA bends inward. Both CCMA and the MA exhibit similar smoothing properties, as shown in (5).

properties to MA but with greater accuracy, which is essential for accurate path following (see Fig. 10).

While tracking combined with CCMA provides a target path, we did not discuss initialization of the convoy, where no positional measurements between two consecutive vehicles are available yet. In this case, a simple clothoid optimization with three consecutive clothoids is used to fit a kinematically feasible path between follower and leader poses. This method requires proper initial alignment of the vehicles, which is ensured either by the vehicles already standing at a proper position or by using the planning functionality (see Section II-H) to position them at a desired starting point. The same path creation technique is applied when switching from backward to forward driving. This is necessary because when the lead vehicle resumes forward movement, the follower truncates the history of the path up to its current position.

Finally, the Waypoint Manager is also responsible for target velocity generation. The target velocity consists of ACC and curvature-based velocities. The formula to calculate the ACC velocity, inspired by [41], is as follows:

$$\text{forward}: v_{\text{acc}} = \max\big(0,\ v_{\text{leader}} + K_v\left(v_{\text{leader}} - v_{\text{ego}}\right) + K_d\left(d_{\text{actual}} - d_{\text{desired}}\right)\big) \quad (13)$$

$$\text{backward}: v_{\text{acc}} = \min\big(0,\ v_{\text{leader}} + K_v\left(v_{\text{leader}} - v_{\text{ego}}\right) + K_d\left(d_{\text{actual}} - d_{\text{desired}}\right)\big) \quad (14)$$

$$d_{\text{desired}} = d_{\text{safe}} + T_{\text{min}}\, v_{\text{leader}} \quad (15)$$

where K_v and K_d are the gains, d is the distance between the follower and the lead vehicle along the path, d_{safe} is the minimum safety distance, and T_{min} is the minimum time gap between two consecutive vehicles. For mission distance, we modify this as follows: set $K_v = 0$ to reduce strict following, and adjust the desired distance:

$$d_{\text{desired}} = \max\left(d_{\text{safe}} + T_{\text{min}} v_{\text{leader}},\ d_{\text{mission}}\right). \quad (16)$$

Another constraint on target velocity is curvature-based speed limits, ensuring safe lateral accelerations. Calculating this requires robust curvature estimations along the target path. The raw path, gathered from noisy lead vehicle positions, would cause unexpected braking if used directly. However, after CCMA filtering, curvature calculations yield reliable values for safe driving (see Fig. 10).

Additional techniques used to improve the generated path and target velocity include the following aspects.

1) Collecting a new lead vehicle position for path reconstruction only if it is sufficiently distant from the previous point ($|p_{\text{new}} - p_{\text{prev}}| > d_{\text{threshold}}$) to prevent excessive noise accumulation when the convoy is stationary.
2) Reducing K_d for mission distance to allow a more flexible driving behavior while maintaining approximate desired spacing between vehicles.

H. OGM AND POSITIONING

During the convoy scenario, it may become necessary to rearrange vehicles or to control one of the follower vehicles remotely. This could be due to a sudden road blockage between two vehicles, a dead end with limited space for maneuvering, or a track loss. When controlling a follower vehicle independently of the lead vehicle, we can no longer benefit from the perception and path planning capabilities of the human driver and need to autonomously plan an obstacle-free path. Thus, we utilize an occupancy grid that maps static obstacles in the environment and implement a path planning algorithm based on this OGM.

Our OGM is based on the terrain maps implemented by Jaspers et al. [42], but was reimplemented using the Universal Grid Map Library from [43] and its ROS interfaces. The library manages 2-D grid maps with multiple layers and takes care of moving and wrapping the grid for efficient memory usage. Our grid is centered around the ego vehicle and has a resolution of 20 × 20 cm. Its layers include an occupancy probability as well as the minimum and maximum height of the obstacle in the grid cell, making the grid a 2.5-D grid. The 2.5-D grid is more efficient than a voxel grid and can still be visualized in 3-D by displaying the obstacles as boxes between their minimum and maximum height.

The input for the OGM is a LiDAR point cloud segmented into ground points, obstacle points, and other points (see

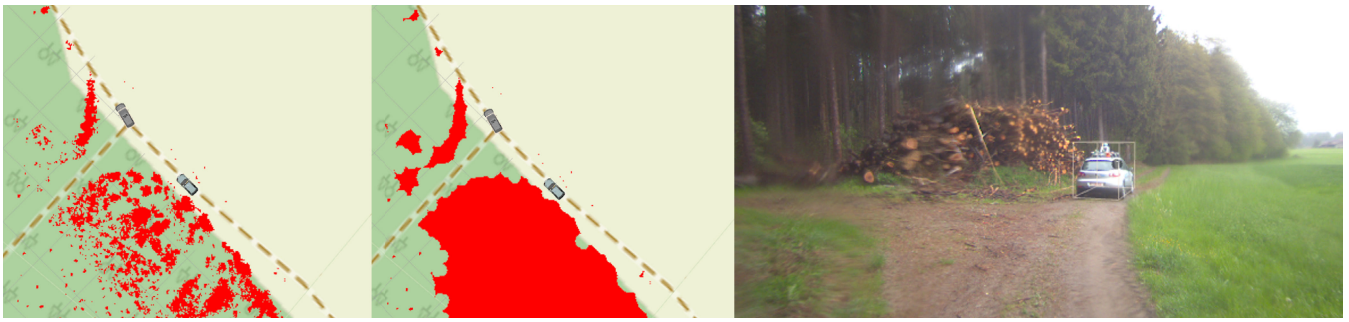


FIGURE 11. **Convoy driving along a forest. The left image shows the obstacle map with the occupied grid cells colored in red. The middle image shows the same map after postprocessing, where cells of unreachable areas are marked as occupied. The right image is the corresponding ego camera image of the follower vehicle.**

Section II-C). Based on the classification of all points that fall into one grid cell, we calculate the inverse sensor model $p(\text{occ}^{(c)}|\ \mathbf{y}_k)$, which gives the probability that a grid cell c is occupied based on extracted measurements $\mathbf{y}$ at time step k. If a grid cell contains two or more obstacle points with a height difference of 30 cm, the grid cell is measured as occupied with $p(\text{occ}^{(c)}|\ \mathbf{y}_k) = 0.95$. If it contains only ground points, it is considered to be possibly free with $p(\text{occ}^{(c)}|\mathbf{y}_k) = 0.4$. If there are no point measurements falling into the grid cell, its measured state is considered unknown with $p(\text{occ}^{(c)}|\ \mathbf{y}_k) = 0.5$. Since we do not want to include dynamic objects in the occupancy grid, we ignore LiDAR measurements within the predicted bounding boxes of all tracked dynamic objects.

Based on this inverse sensor model, we recursively estimate the occupancy probability $p(\text{occ}^{(c)}|\ \mathbf{y}_{1:k})$ using a Bayes filter. Applying the log odds

$$l_k\left(\text{occ}^{(c)}\right) = \log\left(\frac{p\left(\text{occ}^{(c)}|\ \mathbf{y}_{1:k}\right)}{1-p\left(\text{occ}^{(c)}|\ \mathbf{y}_{1:k}\right)}\right) \tag{17}$$

the update step of the Bayes filter is given by

$$l_k\left(\text{occ}^{(c)}\right) = \gamma \log\left(\frac{p\left(\text{occ}^{(c)}|\mathbf{y}_k\right)}{1-p\left(\text{occ}^{(c)}|\mathbf{y}_k\right)}\right) + (1-\gamma)\ l_{k-1}\left(\text{occ}^{(c)}\right). \tag{18}$$

The exponential decay factor $\gamma = 0.1$ causes the mapped obstacles to become more uncertain over time and allows them to be forgotten eventually. This is necessary in case of an incorrect measurement or if the environment changes.

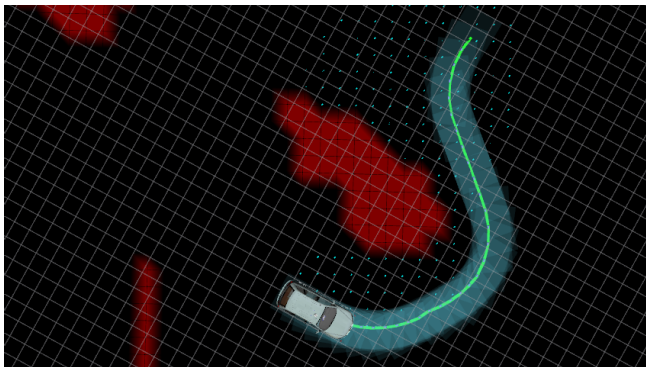
The resulting occupancy probability indicates whether a grid cell is likely to contain a static object. However, it does not specify whether this object is actually an obstacle for the autonomous vehicle. For example, high grass, which the vehicle could easily drive over, can also cause LiDAR measurements and thus may invoke an occupancy probability. Because of this, semantic segmentation is fused into the obstacle grid in order to obtain a real obstacle probability [44]. Furthermore, the OGM does not provide any information as to whether an obstacle-free grid cell is passable at all or whether there is water, for example. This can be part of a dedicated terrain model like we usually use [45]. However, to reduce the overall computational load, we stick to the occupancy probability for the convoy scenario and accept false positive obstacles due to high grass and neglect terrain that is not drivable.

To prepare the occupancy grid for a subsequent trajectory calculation, we must also close off all the areas of the grid map that are traversable but unreachable. To do this, we follow some postprocessing steps proposed in [46] as follows.

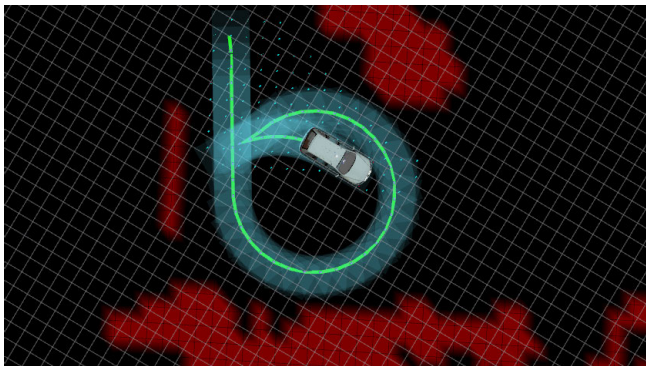
1) Morphological dilation of occupied grid cells using a dilation kernel at least as wide as the vehicle. This closes gaps that the vehicle is too wide to fit through.
2) Flood filling all connected free cells from the vehicle.
3) Morphological erosion of the nonflooded area with the same kernel from step 1. This ensures that the reachable area is not too conservative.

The Universal Grid Map Library enables conversion between grid map objects and OpenCV images, allowing image operations on grid maps. Since the reachable area can drastically differ between two vehicle positions, it is not tracked over multiple time steps via the Bayes filter, as opposed to the obstacles. The resulting grid map is displayed in Fig. 11.

To fulfill the task of obstacle-free path planning in the created grid map, a Hybrid A* algorithm is deployed. The Hybrid A* is an algorithm based on the classical A*, which is an efficient path planning algorithm to find the shortest path from a start to a goal point in a grid map or graph [47]. To summarize briefly, the A* plans expansions from the starting grid cell to neighboring cells. For each expansion, a total cost function is calculated, which is made up of the estimated distance to the goal point and the actual total movement cost from the starting point to the current grid cell. The algorithm works with a priority queue and selects an expansion to the grid cell with the lowest costs in each step. It examines the neighbors of the currently cheapest node, updates their costs, and adds them to the queue if they have not yet been visited or a shorter path has been found. Grid cells that contain an obstacle are marked as occupied and an expansion to this point is forbidden. The algorithm ends when the destination node has been reached or no reachable path exists.



(a)



(b)

FIGURE 12. Sample results of path planning done by the Hybrid A*. The red fields are the occupied grid cells with static obstacles. The light blue dots are the expansions of the A* algorithm. The green line is the planned path of Hybrid A*. The light blue boxes are the visualization of the planned vehicle movement. (a) Example 1. (b) Example 2.

While the A* is ideal for path planning in discrete, grid-based environments, it does not take into account the physical and kinematic constraints (e.g., turning radius) that a real vehicle has. In [48], the A* was extended to better incorporate these constraints into the path planning, ultimately creating the Hybrid A*. The Hybrid A* uses a simple vehicle model to calculate very short forward and backward movements. These movements form branches that expand independently of the grid cells in the OGM. The branches do try to follow the path planned by the A*, as this is one of the heuristics in the total cost function. The total cost function of the A* is also adjusted to incorporate cost for orientation, change of direction (left and right), and gear change (forward and backward). To avoid collisions, the endpoint of each branch is checked to see whether it is located in an occupied grid cell or not. These adjustments result in paths that not only represent the shortest route but are also physically feasible for vehicles.

Our system uses a Hybrid A* version based on the work of [49], which we have adapted to work with our grid map. Fig. 12 depicts two examples of paths planned by the Hybrid A* on our vehicle. For the sake of clarity, we have refrained from visualizing the LiDAR point cloud, the satellite image of the area, and the expansions done by the Hybrid A*.

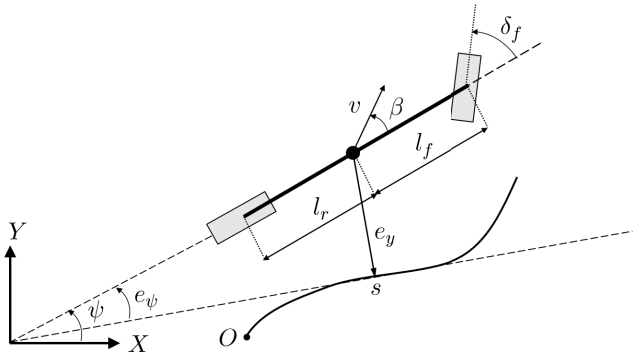


FIGURE 13. Kinematic bicycle model in the Frenet frame.

I. MODEL PREDICTIVE CONTROLLER

As shown in our previous works [50], [51], the MPC is a suitable algorithm for trajectory planning for autonomous convoying, especially when formulated in the Frenet frame. The reason for that is that in this formulation, lateral displacement is a controllable state, thereby making it possible to penalize cutting corners in the cost function of the optimization problem. This is an important benefit, since in tracking-based approaches, corner cutting is a common problem that can, if not resolved, lead to large displacement errors due to error propagation of multiple vehicles in the convoy cutting corners [6]. Furthermore, obstacles and lane boundaries can be better formulated as hard constraints when using the Frenet frame formulation [52]. We note that ego pose and reference paths are in Cartesian coordinates and therefore need to be transformed into the Frenet frame before being used by the MPC.

The mathematical model we use to describe the ego vehicle's dynamics is the kinematic bicycle model [35, p. 26]. This model was chosen because of its good balance between complexity and accuracy. In [53], [54], and [55], different authors proved that this model can achieve an accuracy comparable to higher fidelity models, which consider tire and road interaction. This is especially the case when the lateral acceleration a_{lat} is limited to values smaller than 0.5 gμ, where μ is the coefficient of friction of the surface being driven on [55]. In addition, in our intended use case, the system has to drive on paths with changing road conditions. As we have neither a reliable ground terrain estimation nor map data available, a modeling of the tire-road interaction would be prone to errors. Our used nomenclature refers to the model depicted in Fig. 13.

We define s as the curvilinear abscissa and e_y as the lateral displacement between the ego vehicle's center of mass and the closest point on the reference path. The term e_ψ represents the difference between the orientation of the tangent at the closest point and the yaw angle ψ. The distances from the vehicle's center of mass to the rear and front axles are

denoted as l_r and l_f. The angle between the velocity v and the vehicle's longitudinal axis is the slip angle β. To obtain a finite-dimensional optimal control problem, the continuous-time equations of the model are discretized using the explicit Euler method, resulting in the following nonlinear discrete-time equations (where $\kappa(s)$ is the lane curvature and Δt_d the model's discretization time)

$$s_{k+1} = s_k + \dot{s}_k \Delta t_d \tag{19}$$
$$e_{y,k+1} = e_{y,k} + v_k \sin\left(e_{\psi,k} + \beta_k\right) \Delta t_d \tag{20}$$
$$e_{\psi,k+1} = e_{\psi,k} + \left(\frac{v_k}{l_r} \sin\left(\beta_k\right) - \kappa_k \dot{s}_k\right) \Delta t_d \tag{21}$$
$$v_{k+1} = v_k + a_k \Delta t_d \tag{22}$$
$$\beta_k = \tan^{-1}\left(\frac{l_r}{l_r + l_f} \tan\left(\delta_k\right)\right) \tag{23}$$
$$\dot{s}_k = \frac{v_k}{1 - \kappa_k e_{y,k}} \cos\left(e_{\psi,k} + \beta_k\right). \tag{24}$$

We use (19)–(24) to describe the nonlinear vehicle dynamics for the remainder of this article as

$$\mathbf{x}^{\text{mpc}}_{k+1} = f\left(\mathbf{x}^{\text{mpc}}_k, \mathbf{u}_k\right) \tag{25}$$

where $\mathbf{x}^{\text{mpc}} = [s, e_y, e_\psi, v]^{\text{T}}$ is the state vector and $\mathbf{u} = [a, \delta]^{\text{T}}$ the input vector.

Dynamic obstacles are accounted for by the same strategy as in [56], wherein the obstacles and the ego vehicle are modeled as multiple circles. The number of used circles and their radii r depend on the size of the bounding box of the obstacle. The future movement of those obstacles must also be considered. Assuming constant velocities for the short period of time between measurement updates, an obstacle's future position in the Frenet frame can be estimated using

$$s^{\text{obs}}_{k+1} = s^{\text{obs}}_k + v^{\text{obs}}_s \Delta t_d \tag{26}$$
$$e^{\text{obs}}_{y,k+1} = e^{\text{obs}}_{y,k} + v^{\text{obs}}_{ey} \Delta t_d \tag{27}$$

where v^{obs}_{ey} and v^{obs}_s are the obstacle's currently measured velocities in the direction of the path and the lateral displacement. A collision can be avoided by ensuring that the circles of the ego vehicle do not overlap with the circles describing the dynamic objects at any time. This is achieved by formulating a hard (28) and a soft constraint (29), where $o \in \{1, \ldots, N_{\text{obs}}\}$ with N_{obs} being the number of obstacle circles

$$\left(r^{\text{ego}} + r^{\text{obs}}_o\right)^2 - \left(s^{\text{ego}}_k - s^{\text{obs}}_{k,o}\right)^2 - \left(e^{\text{ego}}_{y,k} - e^{\text{obs}}_{y,k,o}\right)^2 \leq 0 \tag{28}$$

$$J^{\text{obs}}_k = e^{K\left[\left(r^{\text{ego}}+r^{\text{obs}}_o\right)^2 - \left(s^{\text{ego}}_k - s^{\text{obs}}_{k,o}\right)^2 - \left(e^{\text{ego}}_{y,k} - e^{\text{obs}}_{y,k,o}\right)^2\right]}. \tag{29}$$

During practical testing, we have found that using the soft constraint (29) does a good job keeping the ego vehicle away from obstacles while at the same time providing feasibility. Only using the hard constraint (28) will result in suboptimal solutions when the circles of the ego vehicle and object are very close to each other and suddenly overlap because of a noisy position measurement. This is especially a disadvantage when maneuvering in tight spaces, which is a very common occurrence in unstructured environments.

Each control cycle, the MPC solves the following constrained optimization problem over a finite prediction horizon N_p, generating a sequence of optimal control inputs, of which only the first one is applied:

$$\min_{\boldsymbol{u}_t} J^{\text{mpc}} = \sum_{k=t}^{t+N_p} \|\mathbf{x}^{\text{mpc}}_{k|t} - \mathbf{x}^{\text{ref}}_{k|t}\|^2_Q + J^{\text{obs}}_k \tag{30}$$
$$+ \sum_{k=t}^{t+N_p-1} \|\mathbf{u}_{k|t}\|^2_R + \|\Delta\mathbf{u}_{k|t}\|^2_S \tag{31}$$
$$\text{subject to } \mathbf{x}^{\text{mpc}}_{t|t} = \mathbf{x}^{\text{mpc}}(t) \tag{32}$$
$$\mathbf{x}^{\text{mpc}}_{k+1|t} = f\left(\mathbf{x}^{\text{mpc}}_{k|t}, \mathbf{u}_{k|t}\right) \tag{33}$$
$$\forall k = t, \ldots, t + N_p - 1 \tag{34}$$
$$\mathbf{x}^{\text{mpc}}_{\min} \leq \mathbf{x}^{\text{mpc}}_{k+1|t} \leq \mathbf{x}^{\text{mpc}}_{\max} \quad \forall k \tag{35}$$
$$\mathbf{u}_{\min} \leq \mathbf{u}_{k|t} \leq \mathbf{u}_{\max} \quad \forall k \tag{36}$$
$$\dot{\mathbf{u}}_{\min} \leq \Delta\mathbf{u}_{k|t} \leq \dot{\mathbf{u}}_{\max} \quad \forall k \tag{37}$$
$$h^{\text{obs}}\left(\mathbf{x}^{\text{mpc}}_{k+1|t}\right) \leq 0 \quad \forall k \tag{38}$$

where Q, R, S, and K are diagonal matrices of suitable dimensions for weighting and tuning. The state and the input vector at time step k predicted at time t are denoted as $\mathbf{x}^{\text{mpc}}_{k|t}$ and $\mathbf{u}_{k|t}$. The reference state is $\mathbf{x}^{\text{ref}}$ and $\|\mathbf{x}\|^2_Q$ the quadratic function $\mathbf{x}^T Q\mathbf{x}$. The cost function J contains penalties for the deviation from the reference states, and short distances between the ego vehicle and obstacles (30), as well as excessive use of acceleration, steering, and high jerk and steering rates (31). Equation (32) defines the initial state. The cost function minimization is subject to (34), defined by (25). Equations (35)–(37) restrict the ego vehicle's states, inputs, and input rates by upper and lower bounds to enforce dynamic, kinematic, and spatial constraints. Equation (38) is the hard constraint in (28) written compactly.

To take into account the distance between ego and lead vehicle, s^{ref} is calculated for all time steps using

$$s^{\text{ref}}_{k|t} = s^{\text{lead}}_{k|t} - d^{\text{des}}_t \tag{39}$$

where future s^{lead} are estimated using (26). The desired distance at time t is assumed to be constant throughout the control cycle and is determined by (15) from Section II-G.

To compensate for dead-time caused by actuator and communication delays, we employ a prediction-based delay compensation strategy as proposed in [57]. The MPC's generated control commands are stored in a buffer. During each control cycle, the buffered control commands are applied to (25) to predict the ego vehicle's future states after dead-time. Instead of using the vehicle's current states in (32), the estimated future states are used. This way, the vehicle's states during control command execution are anticipated, enabling a proactive adjustment to the dead-time.

The optimization problem [(30) to (38)] is solved by the MPC in real-time (20 ms) using the open source solver IPOPT [58]. The prediction horizon of the MPC is $N_p = 10$ with a discretization time of $\Delta t_d = 300$ ms. We have intentionally not listed all used MPC parameters, as some of them are vehicle-specific (e.g., minimum and maximum steering angles and actuator and communication delays) or subjectively chosen based on comfort level (e.g., minimum and maximum acceleration and weighting factors to punish jerky movements). Besides that, some of the values are subject to change based on the current speed and driving mode. For instance, the slower the vehicle drives, the more we penalize the lateral displacement e_y to achieve higher accuracy when driving through narrow curves or passages. In the presence of dynamic obstacles and during high speeds, we allow a larger lateral displacement from the reference path in order to maximize driving comfort and safety. All weighting factors were determined by careful experimental parameter tuning.

III. EXPERIMENTAL RESULTS

In this section, we present the results of our practical experiments carried out on our two autonomous test vehicles. In one experiment, we showcase our system's ability to keep desired intervehicle distances ranging from 5 to 100 m while driving with speeds of up to 20 m/s. This experiment is carried out multiple times, including during bad weather and light conditions. We also evaluate the system's path-following accuracy and compare it with the performance of other approaches. In addition, we demonstrate dynamic obstacle avoidance during an overtaking scenario and drift-free backwards driving over a long distance. Readers are encouraged to watch the recorded video footage of the experiments.[2]

A. EXPERIMENTAL SETUP

The aforementioned practical experiments to examine the performance of our proposed architecture were carried out on our two robotic vehicles (see Fig. 1). The sensor setup used by our architecture is depicted in Fig. 14, using one of those vehicles as an example. Our second test vehicle uses a similar sensor setup. To avoid confusion, we will refer to the follower vehicle as CAR1 and the lead vehicle as CAR2 for the rest of the article.

Our software system runs on a single-CPU 32-core computer and uses the popular middleware ROS [59]. On a dSPACE MicroAutobox embedded computer, the control commands generated by the MPC are converted to signals for the vehicle's actuators. For V2V communication, our vehicles use a simple WiFi connection. We are aware that WiFi communication is unsafe, particularly for military applications. However, secure V2V communication is not the subject of this article and should be addressed in other works. The SLAM module in our system uses PostgreSQL for its database [60]. The main benefits of PostgreSQL are the easy setup and the automatic handling of the connection, connection loss, and reconnection. To examine the performance and accuracy of our architecture, we equipped both vehicles with an RTK GNSS sensor to collect ground truth data. Our architecture does not use any of the data from those sensors.

[2] https://www.mucar3.de/journal-fr2025-convoy

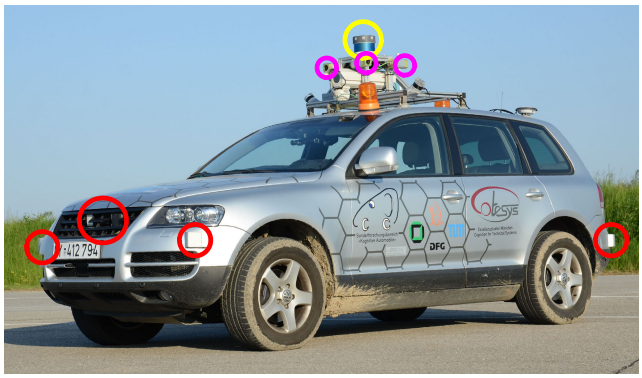


FIGURE 14. **Sensor setup of our test vehicle: LiDAR Velodyne VLS-128 (yellow), color cameras (magenta), and short- and long-range radars (red). Stock sensors (e.g., wheel speed sensors and gyroscope) are not marked.**

B. CHANGING INTERVEHICLE DISTANCES AND SPEEDS

In the first experiment, the system's ability to maintain different intervehicle distances at different speeds was tested. To do so, CAR1 had to follow CAR2 autonomously over a 2.8-km-long route (see Fig. 15).

Both vehicles started at position A. While driving to position B, CAR1 used tracking and maintained short distances to the lead vehicle, which varied depending on the driven speed. At B, the convoy entered a winding track and an intervehicle distance of 100 m was set, making the use of tracking not possible anymore. Somewhere on the winding track, a pedestrian ran between the vehicles to block the road. This shows that the MOT still operates, even when the lead vehicle is out of sight. Once the lead vehicle reached position C, a command was given to the follower vehicle to catch up and get back within tracking distance and go into tracking mode. At D, both vehicles accelerated until they reached a speed of 20 m/s. As a result, the desired distance between the vehicles changed according to (15). Arriving at E, the convoy entered an off-road section that was partially covered by dirt or grass. This section was exited at F, where CAR2 started to drive backwards before stopping and finishing the experiment. Plots analyzing the desired and measured distance between the vehicles, as well as the measured speeds of both vehicles, can be seen in Fig. 16.

As already mentioned, the experiment started with CAR1 following CAR2 in tracking mode. In the beginning, the speeds of both vehicles match, while the desired and actual distance overlap well. At about $t = 25$ s, a command to increase the distance between the vehicles to 100 m was given and the system switched to using SLAM-only. At the same point in the speed plot, it can be seen how CAR1 slows down

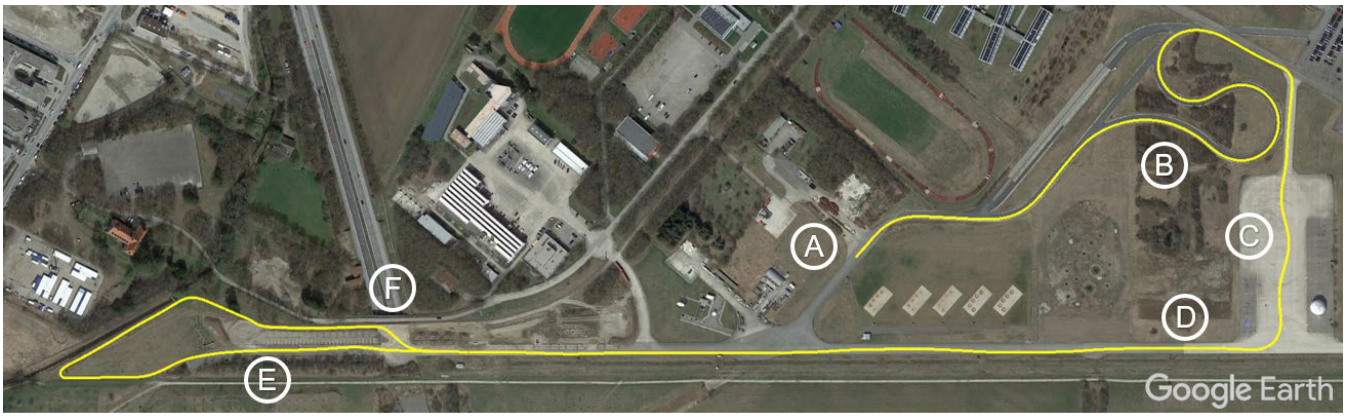


FIGURE 15. **2.8-km-long path (yellow) autonomously driven by CAR1 as it was following CAR2.**

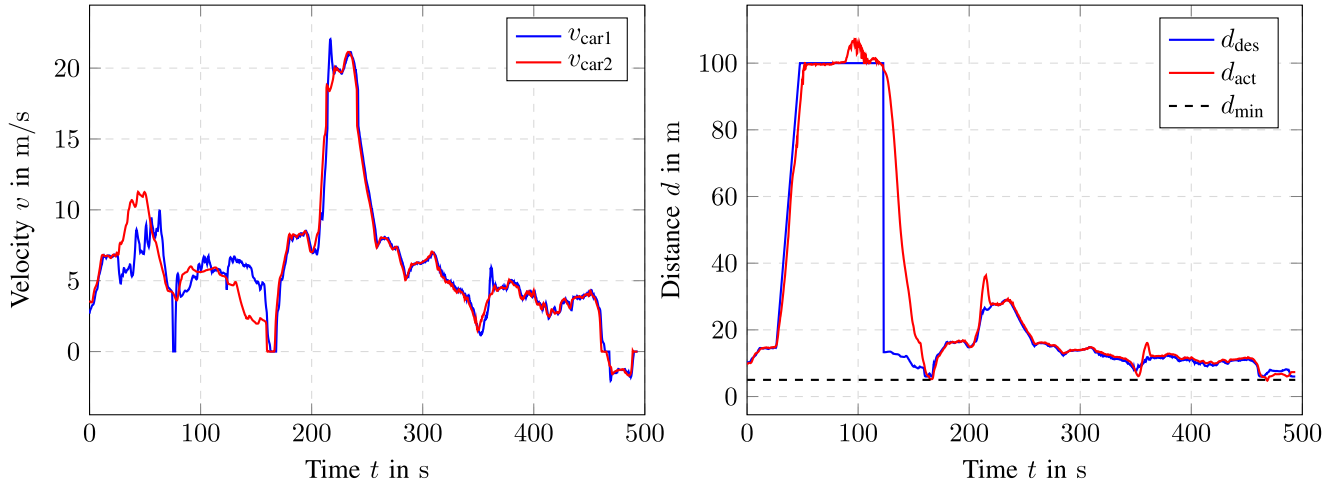


FIGURE 16. **Data recorded during the practical demonstration of driving with different intervehicle distances and speeds. The left plot shows the measured speeds of both vehicles, while the right plot shows the desired and actual distance between both vehicles.**

to widen the gap between both vehicles. The distance plot shows a linear increase of desired and actual distance until both reach 100 m. At about $t = 80$ s, a pedestrian jumped between the convoy and blocked the road. The speed plot shows how CAR1 instantly brakes until it stops, while the distance between the vehicles goes higher than the desired value. As soon as the pedestrian left, the follower vehicle made sure that the desired and actual distance match again. At about $t = 120$ s, the operator commanded CAR1 to catch up to CAR2 immediately. This time, the speed plot shows how CAR1 drove with a higher speed than CAR2 until it caught up at $t = 165$ s. Once CAR1 caught up, the system switched from using SLAM-only back to using the MOT with SLAM as a backup. At about $t = 200$ s CAR2 drove around a corner and began to accelerate rapidly. Because CAR1 was still taking the same corner, it was prohibited from accelerating fast. It can be seen in the distance plot how the actual distance goes higher than the desired one. This showcases how the system limits speed during cornering to increase comfort and safety. Once CAR1 left the corner, the vehicle increased its speed to catch up to CAR2. In the speed plot this is noticeable by the peak in v_{car1} at $t = 220$ s. The same situation happens again at about $t = 350$ s. At about $t = 460$ s both vehicles stop and CAR2 starts to drive backwards. CAR1 reacts almost immediately and also starts to drive backwards. In the plots, this can be seen by both speeds being negative, while the desired and actual distance go up. At about $t = 500$ s both vehicles stop again and the experiment is finished. The same experiment was repeated two more times to investigate the influence of poor visibility conditions on our system (see Fig. 17). It was observed that even driving at night or in heavy rain had no major impact on the performance of our convoy stack. The desired distance was kept as expected and neither tracking nor SLAM had problems.

C. PATH-FOLLOWING ACCURACY

For the evaluation of the accuracy of our system, we did an experiment in which both vehicles again drove the same course as in Fig. 15 in convoy. This time, no pedestrian interfered with the convoy and the intervehicle distance was always in line-of-sight. We did the experiment once

FIGURE 17. **CAR1 autonomously following CAR2 during rain (left) and at night (right). To increase the difficulty while driving in the dark, both vehicles turned off their headlights. The only source of light was the flashing signal lamp on top of CAR1, which signals that the vehicle is in autonomy mode.**

using tracking-only and once using SLAM-only. After that, we compared the recorded GNSS ground truth data of the leader and follower vehicles and calculated the lateral displacement between both driven paths, as is common in the literature. It is noted that at point C of the course (see Fig. 15), there were already not many landmarks available. This affected the accuracy of the SLAM localization and led us to limit the driven velocity at 12 m/s for safety reasons. As we drove SLAM-only, big jumps in the localization of CAR2 could have led to sudden steering movements. Testing the accuracy of SLAM-only had to be stopped halfway between points D and E because not enough landmarks were available for proper localization. The lateral displacements of both test drives are plotted over the driven path in Fig. 18. We have chosen this type of presentation because, in addition to the speed, the lateral displacement also depends on the path curvature and the surface being driven on.

It was determined that over the entire length of the track, the root mean square of the lateral displacement (RMSE) was 10.8 cm and the mean absolute of the lateral displacement (MAE) was 7.3 cm, for driving with tracking-only. When driving SLAM-only, the RMSE was 14.4 cm and the MAE 10.9 cm. It is no surprise that using tracking-only for path generation leads to better path-following accuracy than using SLAM-only, as the MOT is much more precise in estimating the lead vehicle's position. Nevertheless, the path-following accuracy when using SLAM-only is also very satisfactory. We also redid the accuracy test during rain and at night, and found that in both cases, no significant influence on the accuracy could be determined (see Table 1). Changes in the measured errors can be attributed to the accuracy and measurement noise of our used GNSS sensor. Besides that, when doing our tests, the lead vehicle was operated by a human driver. Therefore, it was not possible to recreate the exact same situation with the exact same speeds and curvatures. We do acknowledge that thick fog or very heavy rain would change the accuracy of SLAM-only for the worse, as our SLAM algorithm is only using the LiDAR sensor.

TABLE 1. **Results of the accuracy experiment.**

	sunny	rainy	night
RMSE MOT	10.8 cm	11.2 cm	10.6 cm
MAE MOT	7.3 cm	8.1 cm	7.5 cm
RMSE SLAM	14.4 cm	15.5 cm	14.7 cm
MAE SLAM	10.9 cm	12.3 cm	11.1 cm

To better assess the obtained results, we compare the accuracy of our system with the accuracy archived by other systems that were also tested on real vehicles. Here, we will only look at the accuracy of our system when using the MOT, as this is the default case most of the time, and because the systems we are comparing ourselves against almost all are using tracking exclusively. The results are summarized in Table 2. Besides the RMSE and MAE, the table also provides some additional information and comments. Not all works provide consistent metrics. The absence of information is marked as not provided (n.p.).

D. BACKWARDS DRIVING

When driving off-road and in an unfamiliar area, it can happen that the convoy gets stuck in a dead end where the space is too narrow for maneuvering on the spot. In this case, the vehicles of the convoy need to drive backwards until they reach a point (which may be far away) where maneuvering is possible again or a different path can be taken. When relying on dead-reckoning for localization, the drift of the ego motion estimation can only be ignored when driving forward. Instead, the Waypoint Manager module buffers the latest few points of the driven path that were localized by the SLAM module. This way, they can be reused for backwards driving over longer distances. An experiment was conducted where CAR1 autonomously followed CAR2 for 800 m before the convoy reversed and drove backwards for about 500 m (see Fig. 19). The GNSS ground truth data of both vehicles was used to compare the follower vehicle's paths that were

TABLE 2. Accuracy comparison of different convoy systems. The abbreviation n.p. stands for not provided.

Method	RMSE	MAE	Plot provided	Path length	$v_{\max}$	comment
Method [21]	n.p.	8.7 cm	yes	130 m	1 m/s	
Method [17]	20 cm	n.p.	yes	n.p.	4.5 m/s	convoy drove off-road, our RMSE during off-road driving is 12.1 cm
Method [24]	n.p.	11.4 cm	no	2800 m	15 m/s	
Method [18]	21 cm	n.p.	yes	2200 m	6 m/s	
Method [25]	n.p	n.p	yes	100 m	8 m/s	authors state that lateral displacement stays well within 40 cm
Our Method	10.8 cm	7.3 cm	yes	2800 m	20 m/s	

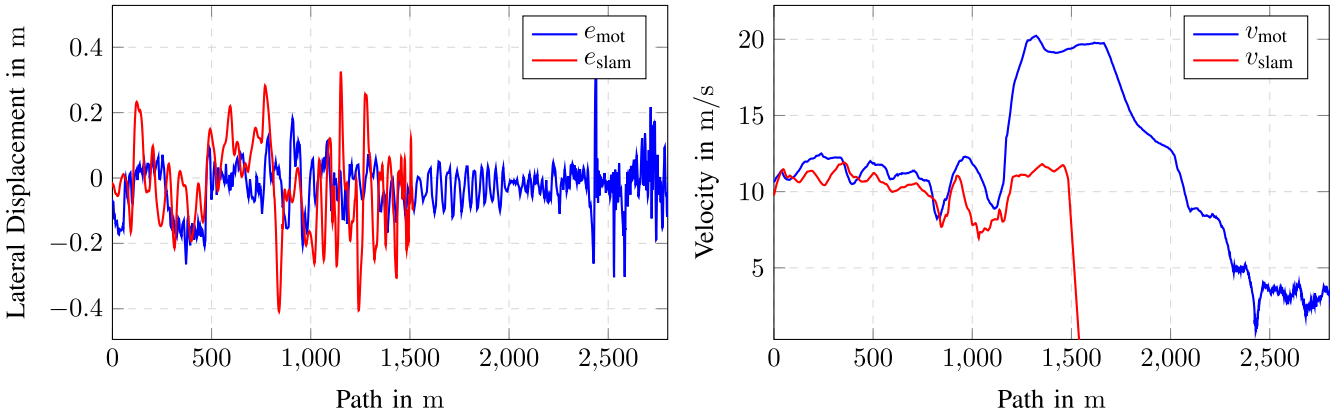


FIGURE 18. Data recorded during the accuracy determination. The left plot shows the lateral displacements when driving in convoy using tracking-only (e_{mot}) and SLAM-only (e_{slam}). The right plot shows the measured velocities of CAR1 for both experimental drives. Measurements for SLAM-only stop halfway through because not enough landmarks were available for the SLAM to work properly. The distance between the vehicles was not plotted, as it does not influence the lateral displacement.

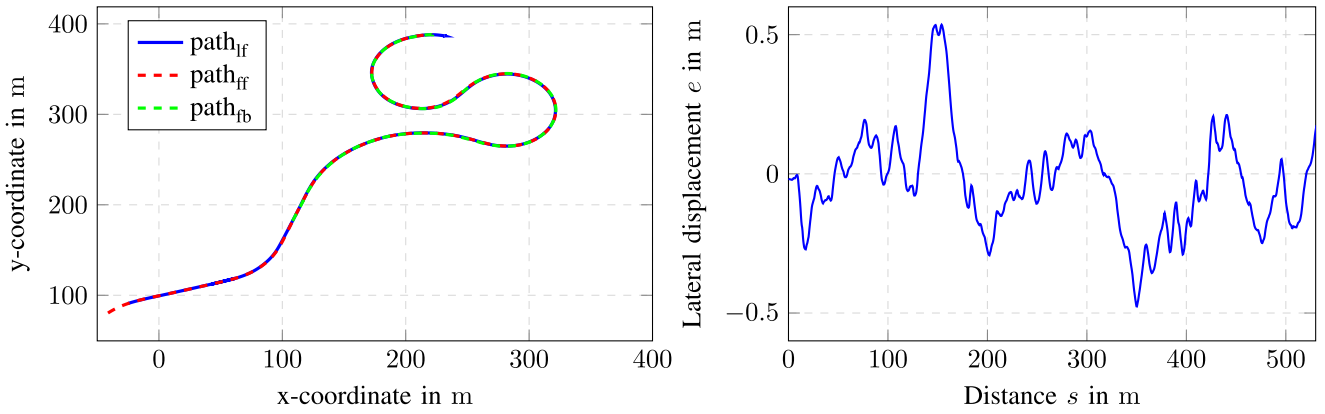
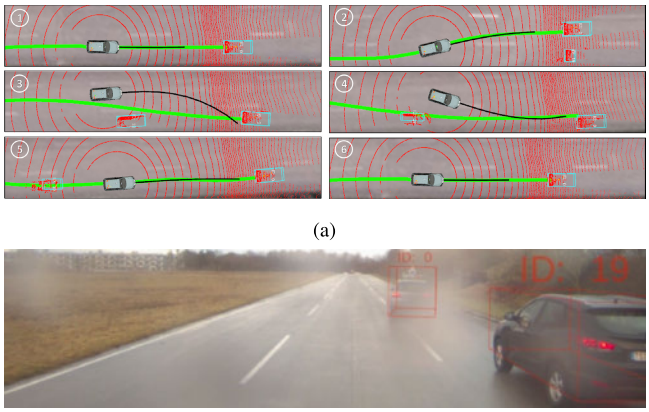


FIGURE 19. Data recorded during the practical demonstration of driving backwards. The left plot shows the path driven by the lead vehicle forward (blue straight line), the follower vehicle forward (red dashed line), and the follower vehicle backwards (dashed green line). The right plot shows the lateral displacement between the forward path of the lead vehicle and the backward path of the follower vehicle, showcasing how the follower vehicle is able to drive the same path backwards without drift, even for long distances.

driven forward and backward to the lead vehicle's forward path. The lateral displacement between the backwards driven path of the follower vehicle and the forward driven path of the lead vehicle was calculated and plotted over the driven distance. The results of the experiment are very satisfactory. The follower vehicle was able to keep itself on the path without drifting away and without large displacements.

E. DYNAMIC OBSTACLE AVOIDANCE

Considering dynamic obstacles during convoying is crucial. The importance of this feature can be particularly well demonstrated in the event of an overtaking scenario. The scenario starts with the leading vehicle overtaking a slower vehicle and merging in front of it. The follower vehicle begins the overtaking maneuver at the same point as the leading



(a)

(b)

FIGURE 20. **CAR1 autonomously follows CAR2, as both vehicles drive with a speed of about 10 m/s. Once the convoy encounters a slower vehicle (8 m/s), the lead vehicle starts overtaking. Our system tracks the slower vehicle and executes a collision-free overtaking. The black line is the MPC's trajectory, while the green line is the path driven by CAR2. The camera image (bottom picture) was taken shortly before (3). (a) Overtaking scenario as seen in rviz. (b) Camera Image during the overtaking scenario.**

vehicle did, since it blindly follows the reference path for now. If the follower vehicle were to also blindly follow the same path when merging in, it would cause a collision with the slower vehicle, as the slower vehicle would have moved forward in the meantime. Considering dynamic obstacles in the MPC formulation allows the follower vehicle to deviate from the reference path long enough to finish the overtaking maneuver properly. This specific scenario was recreated during one of our practical experiments to showcase our system's ability to solve this situation as intended (see Fig. 20).

IV. ELROB TRIAL

To test our system under conditions that resemble the intended use case as realistically as possible, we took part in the 12th ELROB 2024 trial (see Section I), where we participated in the military convoying scenario. In this section, we discuss the system's performance during the trial, going into detail on where the system succeeded and where it could have done better. During the ELROB run, the convoy had to drive a total of about 6 km.

We started our run with CAR2 as the lead and CAR1 as the follower vehicle. In addition, we also had a notebook serving as a base station, which was running the Mission Manager module to get a visualization of the camera data and the driven route of both vehicles. The convoy operator received a list of points with UTM coordinates, which the convoy had to drive to in 10 min.

The trial started off simply. Both vehicles drove along a paved road until they reached the first obstacle. On the way to the first obstacle, it was already noticeable that the GNSS signal was very poor due to being somewhere in a forest. At one point, GNSS was even actively blocked by a jammer.

FIGURE 21. **Convoy encounters its first obstacle. One straight (top picture) and one zigzag (bottom picture) narrow passage.**

FIGURE 22. **Off-road section of the trial. The top picture shows a road with big potholes. The bottom picture shows a dust cloud that made tracking harder.**

This was no problem for our convoy, as our system does not rely on GNSS at all. The first obstacle that the convoy encountered was a narrow passage (see Fig. 21). This was also not a problem, as our system has a high path-following accuracy (see Section III-C).

After passing the narrow passage, the convoy drove until it reached a roadblock. The instruction here was to drive around the blockage across an off-road section with rough terrain (see Fig. 22). The bumpy path was covered in potholes that caused the vehicle to shake heavily, leading to high pitch and roll angles. The sunny weather led to dry dust being stirred up from the ground, which also interfered with the sensors (better seen in the demonstration video). The MOT mastered

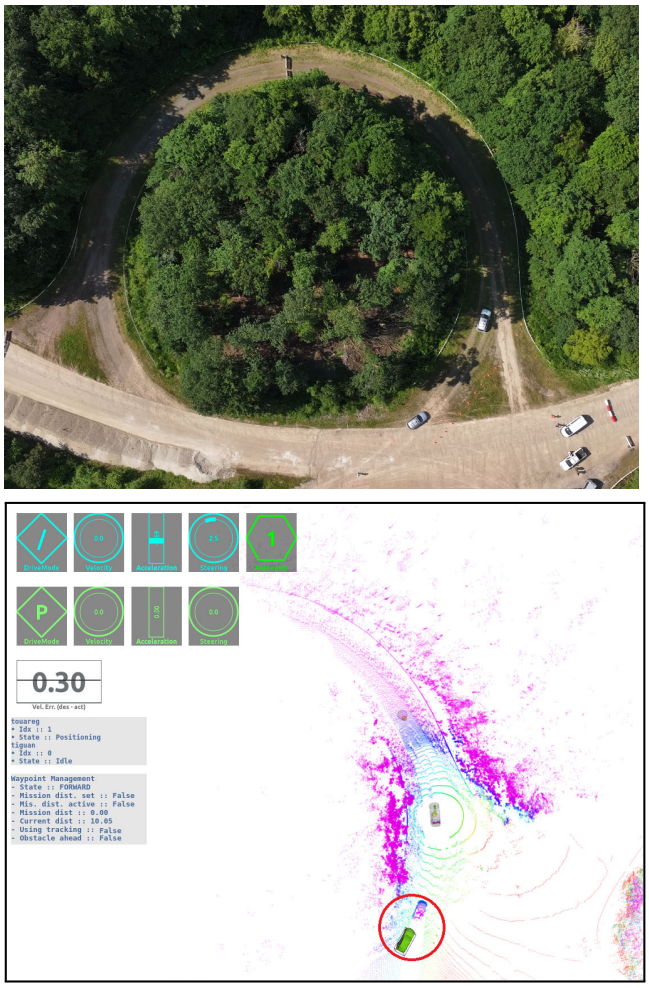


FIGURE 23. **Top picture shows a drone shot of the large circle that CAR2 had to drive while CAR1 was waiting in idle. The bottom picture shows the surrounding environment as perceived by CAR1 in rviz. A drift between the real position of the vehicle (LiDAR points) and the estimated SLAM position (green vehicle) can be observed in the red circled area.**

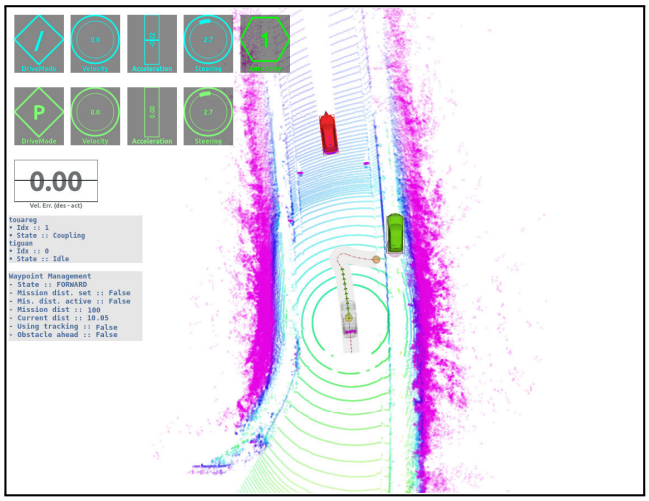


FIGURE 24. **Visualization of the environment around CAR1 in rviz. The red vehicle is the position of the lead vehicle estimated by the MOT. The green vehicle is the position of the lead vehicle estimated by SLAM. While the MOT predicts the position accurately, the SLAM suffers from a drift. The created reference path tries to follow the estimated SLAM position of the lead vehicle, as the convoy is in long-distance mode, where it drives using SLAM-only.**

this challenging situation flawlessly and kept on tracking the lead vehicle without problems.

At one point in the off-road section, the convoy had to turn left at a junction in order to reach one of the specified UTM coordinate points. This route turned out to be a dead end with no possibility of making a U-turn. Accordingly, the system's ability to drive backwards came into play. Both vehicles reversed and drove back to the junction, where this time they had to take the right path. From there, the vehicles were required to drive with an intervehicle distance of 100 m. Now, the system had to fully rely on the SLAM module, as the lead vehicle was out of sight for the MOT.

After a while, CAR1 stopped and was put into idle mode to wait for further commands. In the meantime, the lead vehicle, CAR2, drove a large circle (see Fig. 23, top picture) until it emerged behind CAR1. The operator in the lead vehicle used the positioning mode to position CAR1 behind CAR2 and activated the coupling mode (ACC) again. After a few hundred meters of driving with short intervehicle distances and using the MOT for trajectory generation, the command to widen the gap between the vehicles to 100 m was given again. Here, we encountered our first error. While the lead vehicle was driving the circle, it kept on updating the map. As the follower vehicle was standing and waiting, it could not localize itself due to missing information about landmarks along the path of the circle. This led to a drift of the lead vehicle's position in the map created by the SLAM module (see Fig. 23, bottom picture). Since the system relies on SLAM-only when driving with large intervehicle distances, the drift almost caused CAR1 to drive off the road into the forest (see Fig. 24), leading to the first intervention of the safety driver. This situation led us to later update the system. Now the SLAM module periodically checks if the panoptic segmentation can find an object of the class "vehicle" at the position, where SLAM thinks that the lead vehicle should be.

The operator reinitialized the system and the trial continued. The next obstacle was a parked vehicle that positioned itself between the convoy. CAR2 stopped and waited until the operator positioned CAR1 around the static obstacle. We are aware that the more elegant solution would have been to use the MPC's capability to drive around dynamic obstacles. We decided against that, as our system does not use a road tracking algorithm, from which it could have derived the lane boundaries that need to be taken into account when driving around the standing obstacle. We do have such an algorithm, but due to limited computation power, it was not integrated into the system. On the way to the next obstacle, the convoy passed a jammer that disrupted radio communications. This

FIGURE 25. **Blocked off road between CAR2 and CAR1.**

FIGURE 26. **Off-road path on which CAR1 had to drive autonomously running next to a regular road where CAR2 was driving.**

did not affect our system, as our WiFi emits signals at different frequencies.

The next obstacle was a blocked road again, except this time the blocking was between the lead and follower vehicle (see Fig. 25). The new command given was that the follower vehicle needs to drive on an off-road path running parallel to the regular road (see Fig. 26). Technically, the MPC is capable of driving in parallel formations. This can be achieved by setting an appropriate desired lateral displacement from the reference path. The problem was that both roads were not perfectly parallel. We could have used the positioning command to navigate CAR1 across this section, but that seemed tedious. Instead, we decided to intervene a second time to save time.

After a while, the off-road path merged into the regular road. There both vehicles met again. Now, a role switch needed to be performed between the vehicles. This procedure can be done quickly thanks to the UI of the Mission Manager module. Both vehicles drove in convoy until they reached the last UTM coordinate located at the starting point of the trial, thus ending our run.

V. LESSONS LEARNED AND SCALING

In this section of the article, we cover the lessons learned during our own tests and during the trial. Based on them, we give remarks on how we feel that our system would scale to a convoy of multiple autonomous vehicles.

A. BEFORE THE TRIAL

When we started to design our convoy system, we intended to have the poses of each vehicle in a unified coordinate system. Since in our use case we could not rely on GNSS, we chose the SLAM frame as the global reference. Early in testing, we observed that using a unified SLAM coordinate system introduced jumps in the localization and a significant loss of tracking accuracy for the MOT. Therefore, we decided to use the odometry frame as the primary coordinate system for both self-localization and reference trajectory generation. The MOT inherently performs tracking in the odometry frame, while the pose of the lead vehicle, as localized by the SLAM, must be transformed into the odometry frame for consistent processing. The drift in the odometry frame is negligible for small relative distances [6]. Later on, we found that it is not sufficient to simply transform the path traveled by the lead vehicle from SLAM coordinates to odometry coordinates when there are large intervehicle distances. Transforming the reference path from SLAM to odometry at the time of reception means that any subsequent drift in the follower's odometry leads to increasing errors in the transformed path points, especially those further ahead. While the original SLAM coordinates remain consistent, their corresponding odometry representations degrade over time as the follower moves. To address this, we began buffering the full SLAM-based path of the lead vehicle and incrementally transforming and publishing short segments of it in the odometry frame (see Section II-F5). The same applies when driving backwards. The SLAM frame is still used as a global reference by the individual Mission Manager modules of each vehicle. This is done to unify everything for easier visualization and to make sure that every Mission Manager can command a desired pose to any vehicle of the convoy using the positioning mode.

Speaking of SLAM. For our system, we chose a sparse SLAM algorithm because it requires less bandwidth for V2V communication. On the other hand, this approach is not without its disadvantages. The sparse SLAM algorithm has large localization errors or, even worse, cannot properly localize the lead vehicle if there are not enough landmarks available (e.g., when driving in a steppe or desert). We do argue that using SLAM in a full-stack autonomous convoy system is crucial. Therefore, the sparse SLAM algorithm might need to be replaced by a dense SLAM algorithm. This is only recommended if stable V2V communication that supports fast exchange of large data packages is available.

Another finding was that MPC is a particularly well-suited algorithm for convoying tasks. In contrast to our approach, systems in related work often decouple lateral and longitudinal motion planning into separate control problems. While this may simplify implementation, it fails to capture the inherent coupling in vehicle dynamics, which can lead to worse path-following accuracy and less maneuvering when handling dynamic obstacles. Optimizing both steering and velocity jointly enables more precise and stable convoy behavior and more maneuvers. Since the reference trajectory is largely determined by the motion of the lead vehicle, the complexity of the optimization problem is reduced. During the preparations for the ELROB trial, we covered a total of over 200 km in convoy, including 20 km in one stretch. Throughout these tests, the MPC exhibited no stability issues.

B. DURING THE TRIAL

An important lesson learned during the trial was that if vehicles from the convoy do not take the same route, it affects the accuracy of the localization of the SLAM algorithm. At one point during the trial, the follower vehicle remained stationary while the lead vehicle drove a large circle until it reappeared behind the follower vehicle (see Fig. 23). Although the SLAM module of the lead vehicle continued to update the map, the SLAM module of the follower vehicle lacked sufficient observations of the relevant landmarks along the circular section. As a result, it could not localize the pose of the lead vehicle without drift. This incident taught us the risk of map desynchronization in cooperative SLAM when vehicles are idle or separated for extended periods. After the trial, we updated our system to perform periodical health checks. Now the SLAM module checks if the panoptic segmentation can detect an object of the class ''vehicle'' at the position where the SLAM algorithm is localizing the lead vehicle. This approach is not without flaws. In an environment with many other vehicles, which are not part of the convoy fleet, it could happen that the incorrectly localized position falls on another vehicle, resulting in an incorrect assumption. In addition, if the convoy consists of multiple vehicles, it can be checked whether the pose of the lead vehicle is localized at an identical position by all other vehicles of the convoy. If this is not the case, a drift is detected. Unfortunately, we were unable to implement this in practice as we only have two test vehicles. Alternatively, relying on GNSS as an additional source for verification can be done in other applications, but not in ours.

The second lesson that we learned is the importance of short reaction times. Other participants of the trial praised how fast our system detects a change from forward to backwards driving and reacts accordingly. This responsiveness becomes particularly critical in multivehicle convoys, where delays in reaction times can propagate and amplify along the chain. For example, when the lead vehicle initiates backward driving, the immediate follower must first detect and respond to this change before it can begin moving in reverse. If there is a bigger delay in this response, the third vehicle must then wait even longer for the second vehicle to move, and so on. As a result, each vehicle in the convoy introduces additional latency, causing the overall movement to degrade into a staggered, stop-and-go pattern. This delay is particularly significant for scenarios in which time losses are critical, such as search and rescue or when the military convoy is under attack.

Another observation that we had concerns the separation of the static and dynamic environment into two different modules. In our system, bounding boxes from the MOT are used to ignore certain LiDAR measurements for the creation of the OGM, which effectively prevents moving objects from leaving trail artifacts. However, this approach is not without flaws. Vegetation such as tall grass or bushes can sway in the wind and, therefore, be misclassified as dynamic. As a consequence, relevant static grid cells are suppressed frequently, and their status is set to unoccupied. While this has minimal impact on structured environments, where the position of the vegetation rarely falls into the drivable area, it becomes problematic in off-road environments, where vegetation may occupy significant portions of the traversable terrain. This can be addressed by semantic filtering, where bounding boxes are only used to mask static grid cells if the class label of the bounding box corresponds to a genuinely dynamic object (e.g., vehicle and pedestrian). But one issue still remains. Cells directly beneath dynamic objects are marked as unoccupied, which leads the Hybrid A* to consider paths that pass directly through them. The task of handling the dynamic object is passed to the MPC. The result is increased computational complexity, tighter coupling between modules, and a system that becomes more prone to integration errors.

C. AFTER THE TRIAL

An insight from our development process was that integrating multiple individually functional modules into a cohesive full-stack system presents significant challenges. During the design of the individual modules, it is often assumed that required data will be reliably provided by other components or that computationally intensive tasks can be offloaded to other modules without affecting overall performance. However, such assumptions frequently do not hold once modules are integrated. In practice, compromises are inevitable, and in our case, we had to omit certain features from individual modules due to hardware limitations. Subsequent changes to fully functional modules can introduce errors and slow down the implementation process, as those changes also affect other modules in a modular architecture. This experience highlighted two key lessons. First, the goal should be to keep the number of modules low. This helps reduce dependencies, system complexity, and integration effort. Perhaps the use of machine learning to unify some modules into a single one, like in [61], would help address this problem. On the other hand, machine learning requires a large amount of data for model training, and the amount of datasets for outdoor driving is very limited [62], [63]. Second, early integration into a full system is essential to uncover architectural limitations and functional mismatches at an early stage. Retrofitting or debugging mature, standalone modules proved to be substantially more difficult and time-consuming than working with components still in active development.

In addition, we also had to deal with the limitations of the middleware. While ROS1 provides a flexible publish-subscribe communication model [59], its architecture is limited when it comes to handling multiple robots in a distributed manner. ROS1 is designed around a centralized master (roscore), which maintains the global topic registry and facilitates node discovery. Using one centralized master for every vehicle of the convoy introduces a single point of failure and bloats the communication traffic. Instead, each

vehicle runs its own local instance of the roscore. In this case, ROS1 does not provide a native mechanism for cross-master synchronization, topic remapping, or namespace conflict resolution. To resolve this problem, we used the ROS1 multimaster extensions [64]. Unfortunately, this did not solve all of the problems. The networking model in ROS1 relies on TCPROS, which is a custom ROS-specific protocol layered on top of standard TCP/IP. In an environment with unreliable or asymmetric communication links, the networking model does a poor job at handling packet loss, changing IP addresses, and variable latencies. We observed situations in which the SLAM of the follower vehicle did not receive any updates from the lead vehicle for a certain time, just to receive multiple points of the lead vehicle's driven path in a sudden burst of communication. ROS2 seems to be a more promising solution for a multirobot system [65]. In the future, we will switch our middleware to ROS2 and verify this assumption.

D. SYSTEM SCALABILITY

When designing our system for autonomous convoying, we always kept its scaling to multiple vehicles in mind. In the following, the term ''lead vehicle'' refers to the very first vehicle in the convoy. All other subsequent vehicles are referred to as followers. In our system, each vehicle hosts its own instance of the Mission Manager module, which communicates with its counterparts on the other vehicles. This way, each Mission Manager can assign any role to any vehicle of the convoy fleet, making sure there is no single point of failure. The decision as to which role and which position in the order of the convoy is taken by which vehicle is the responsibility of the convoy operator. As of now, the Mission Manager does not support the separation of one big convoy into a smaller number of convoys with their own lead vehicles. This functionality can be added quickly. We just refrained from doing so because we only have two vehicles.

In the convoy, only the SLAM module of the lead vehicle performs landmark-based mapping. The generated map is stored and frequently updated on the lead vehicle's local SLAM database, as are its estimated ego pose and measured speed. The SLAM module of every other vehicle (followers) accesses the SLAM database of the preceding vehicle (the one in front of it), copies its data, independently identifies landmarks to estimate their own pose within the shared map, and stores a copy of the map on its own local SLAM database, together with the ego vehicle's estimated pose, measured speed, and the estimated pose of the lead vehicle. We decided against the lead vehicle sharing its map with every vehicle of the convoy directly because of the limited communication range. In the case where the vehicles of the convoy need to maintain an intervehicle distance of 100 m, the lead vehicle will not be able to distribute its map to every vehicle, since they are too far away.

The Waypoint Manager on each follower vehicle uses the lead vehicle's pose to define the reference path and combines it with the preceding vehicle's pose, as it was estimated by SLAM and tracked by the MOT, to derive a velocity profile. This helps to reduce the influence of error propagation. The MPC also has no scaling problem. Dunbar and Caveney [66] proved mathematically that a nonlinear and heterogeneous approach to MPC for convoying is string stable. A disturbance in longitudinal motion is not amplified as it travels down the convoy. Stability of the MPC is ensured by appropriately selecting the weighting parameters in the cost function of each local MPC formulation. Notably, the authors also proved that the MPC does not require acceleration data from the lead vehicle or continuous information from the lead vehicle to the followers. This stands in clear contrast to other string-stable controllers that inherently depend on information about the lead vehicle's acceleration, velocity, and position error, to maintain stability [67]. The authors indicate that convoys of tens and hundreds of vehicles are possible.

The first restriction of our system is in the CCMA filter. Although the size of the convoy does not influence the filter, the type of vehicle does. During the reconstruction of the preceding vehicle's path, it is assumed that the preceding vehicle operates under the same or more restrictive kinematic constraints as the follower. Also, when using our system for a convoy consisting of different vehicle types (e.g., cars and trucks), the vehicle model and the weights of the MPC need to be adjusted accordingly. In a nonhomogeneous convoy, the Mission Manager of each vehicle could store a ''vehicle type'' variable, which then influences the used vehicle model and other parameters inside the MPC. The biggest restriction is the middleware. As already explained in detail in the previous chapter, ROS1 is simply not capable of handling big, distributed multirobot applications.

VI. CONCLUSION

In this article, we presented an improved architecture for the task of autonomous military convoying. Our architecture uses a tracking and a SLAM algorithm to estimate the position of the lead vehicle, which adds redundancy and enables both very precise driving and driving without a clear line of sight. As specified by the use case, neither GNSS nor a prerecorded map of the environment is used. We evaluated the performance of our architecture in several tests and experiments on our real autonomous cars. All of the conducted experiments showed satisfactory results. Furthermore, we took part in the 2024 ELROB military convoying scenario to test our system even more and to put it up against a realistic challenge. Here, too, we are generally satisfied with the system's performance during the trial.

Of course, there were also hurdles and setbacks, as no system is perfect from the start. The trial and the preparation for it helped us to gather valuable insights into where our architecture could still be improved and what challenges we need to deal with. At the moment, the MPC does not consider lane boundaries or static obstacles. Our logic was that if the lead vehicle can drive around a static obstacle, then the other vehicles in the convoy will do the same. However, this is not

always the case, especially not for the particular application that we designed the architecture for. A suddenly fallen tree or debris after a landslide can block a path that was previously passable. With our current system, the autonomous follower vehicle would just stop because the path is blocked, and the operator would need to navigate it around the blocked path by activating positioning mode and clicking on the map. It would be more preferable if the follower vehicle simply drove around the static obstacle, like it does with dynamic obstacles.

Regardless of whether a static or dynamic obstacle is avoided, the lane boundaries must be taken into account. This means that we need to find a way to overcome the encountered hardware limitations to make use of our improved OGM module for free-space detection. We chose free-space detection instead of a road detection algorithm because, when driving off-road, situations might arise where there is no clearly visible road or path. On the other side, lane boundaries can be constructed from a drivable free-space.

Besides that, with the increasing use of machine learning for autonomous driving [68], the question arises as to whether the tasks of some of our modules could not be better solved by neural networks. It would also be interesting to use a reinforcement learning-based approach to replace the Hybrid A* for the positioning task or to use a diffusion model [69] to create a reference trajectory for the MPC directly from sensor input. These are the topics on which we will be focusing our research in the near future to improve our system even further.

REFERENCES

[1] S. E. Shladover, "PATH at 20—History and major milestones," *IEEE Trans. Intell. Transp. Syst. (ITSC)*, vol. 8, no. 4, pp. 584–592, Jan. 2006.

[2] K. S. Chang et al., "Experimentation with a vehicle platoon control system," in *Proc. Vehicle Navigat. Inf. Syst. Conf.*, Troy, MI, USA, Oct. 1991, pp. 1117–1124.

[3] S. Nahavandi et al., "Autonomous convoying: A survey on current research and development," *IEEE Access*, vol. 10, pp. 13663–13683, 2022.

[4] S. Tsugawa, S. Jeschke, and S. E. Shladover, "A review of truck platooning projects for energy savings," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 68–77, Mar. 2016.

[5] S. W. Smith, Y. Kim, J. Guanetti, A. A. Kurzhanskiy, M. Arcak, and F. Borrelli, "Balancing safety and traffic throughput in cooperative vehicle platooning," in *Proc. 18th Eur. Control Conf. (ECC)*, Naples, Italy, Jun. 2019, pp. 2197–2202.

[6] H. K. Goi, J. L. Giesbrecht, T. D. Barfoot, and B. A. Francis, "Vision-based autonomous convoying with constant time delay," *J. Field Robot.*, vol. 27, no. 4, pp. 430–449, May 2010.

[7] M. Lammert, B. McAuliffe, P. Smith, A. Raeesi, M. Hoffman, and D. M. Bevly, "Impact of lateral alignment on the energy savings of a truck platoon," Detroit, MI, USA, SAE Technical Paper 2020-01-0594, Apr. 2020.

[8] V. Turri, Y. Kim, J. Guanetti, K. H. Johansson, and F. Borrelli, "A model predictive controller for non-cooperative eco-platooning," in *Proc. Amer. Control Conf. (ACC)*, Seattle, WA, USA, May 2017, pp. 2309–2314.

[9] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen, "A survey on platoon-based vehicular cyber-physical systems," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 263–284, 1st Quart., 2016.

[10] O. Pauca, A. Maxim, and C.-F. Caruntu, "Control architecture for cooperative autonomous vehicles driving in platoons at highway speeds," *IEEE Access*, vol. 9, pp. 153472–153490, 2021.

[11] E. Chan, P. Gilhead, P. Jelínek, P. Krejci, and T. Robinson, "Cooperative control of SARTRE automated platoon vehicles," in *Proc. 19th ITS World Congr.*, Jan. 2012, pp. 22–26.

[12] M. Goli and A. Eskandarian, "MPC-based lateral controller with look-ahead design for autonomous multi-vehicle merging into platoon," in *Proc. Amer. Control Conf. (ACC)*, Philadelphia, PA, USA, Jul. 2019, pp. 5284–5291.

[13] C. Fries, T. Luettel, and H.-J. Wuensche, "Combining model- and template-based vehicle tracking for autonomous convoy driving," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Gold Coast, QLD, Australia, Jun. 2013, pp. 1022–1027.

[14] E. van Nunen, M. R. J. A. E. Kwakkernaat, J. Ploeg, and B. D. Netten, "Cooperative competition for future mobility," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1018–1025, Sep. 2012.

[15] A. Geiger et al., "Team AnnieWAY's entry to the 2011 grand cooperative driving challenge," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1008–1017, Sep. 2012.

[16] 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, Sep. 2012.

[17] X. Zhao, W. Yao, N. Li, and Y. Wang, "Design of leader's path following system for multi-vehicle autonomous convoy," in *Proc. IEEE Int. Conf. Unmanned Syst. (ICUS)*, Beijing, China, Oct. 2017, pp. 132–138.

[18] F. Mutz et al., "Following the leader using a tracking system based on pre-trained deep neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Anchorage, AK, USA, May 2017, pp. 4332–4339.

[19] L. D. P. Veronese et al., "A light-weight yet accurate localization system for autonomous cars in large-scale and complex environments," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 520–525.

[20] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Proc. 14th Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands, 2016, pp. 749–765.

[21] P. Avanzini, E. Royer, B. Thuilot, and J.-P. Dérutin, "Using monocular visual SLAM to manually convoy a fleet of automatic urban vehicles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2013, pp. 3219–3224.

[22] N. C. Griswold, N. D. Kehtarnavaz, and K. M. Miller, "A transportable neural network controller for autonomous vehicle following," in *Proc. Intell. Vehicles Symp.*, Paris, France, Sep. 1994, pp. 195–200.

[23] J. L. Giesbrecht, H. K. Goi, T. D. Barfoot, and B. A. Francis, "A vision-based robotic follower vehicle," *Proc. SPIE*, vol. 7332, pp. 451–462, May 2009.

[24] D. Fassbender, B. C. Heinrich, T. Luettel, and H.-J. Wuensche, "An optimization approach to trajectory generation for autonomous vehicle following," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 3675–3680.

[25] G. Nestlinger, J. Rumetshofer, and S. Solmaz, "Leader-based trajectory following in unstructured environments—From concept to real-world implementation," *Electronics*, vol. 11, no. 12, p. 1866, Jun. 2022.

[26] T. Steinecker and H.-J. Wuensche, "A simple and model-free path filtering algorithm for smoothing and accuracy," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Anchorage, AK, USA, Jun. 2023, pp. 1–7.

[27] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 474–490.

[28] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 12689–12697.

[29] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, Oct. 2018.

[30] A. Reich and M. Maehlisch, "Low latency instance segmentation by continuous clustering for LiDAR sensors," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jeju Island, South Korea, Jun. 2024, pp. 1871–1877.

[31] A. Reich and H.-J. Wuensche, "Fast detection of moving traffic participants in LiDAR point clouds by using particles augmented with free space information," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Kyoto, Japan, Oct. 2022, pp. 538–543.

[32] L. Beer and H.-J. Wuensche, "General panoptics: Combining semantic segmentation and classical methods for a fast LiDAR panoptic segmentation," in *Proc. Tagungsband Workshop Fahrerassistenzsysteme (FAS)*, Berkheim, Germany, May 2022, pp. 119–128.

[33] L. Beer, T. Luettel, and M. Maehlisch, "Toward feature-based low-latency localization with rotating LiDARs," *IEEE J. Indoor Seamless Positioning Navigat.*, vol. 3, pp. 105–116, 2025.

[34] H. Cheng, X. Han, and G. Xiao, "Cenet: Toward concise and efficient LiDAR semantic segmentation for autonomous driving," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Taipei, Taiwan, Jul. 2022, pp. 01–06.

[35] R. Rajamani, *Vehicle Dynamics and Control*. New York, NY, USA: Springer, 2011.
[36] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logistics (NRL)*, vol. 52, no. 1, pp. 7–21, Feb. 2005.
[37] P. Burger, "Kooperative Lokalisierung und Kartierung in unstrukturierter Umgebung," Ph.D. thesis, Inst. Auton. Syst. Technol. (TAS), University of the Bundeswehr Munich, Neubiberg, Germany, 2022.
[38] L. Beer, T. Luettel, and H.-J. Wuensche, "GenPa-SLAM: Using a general panoptic segmentation for a real-time semantic landmark SLAM," in *Proc. IEEE 25th Int. Conf. Intell. Transp. Syst. (ITSC)*, Macau, Oct. 2022, pp. 873–879.
[39] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures," *Int. J. Robot. Res.*, vol. 25, nos. 5–6, pp. 403–429, May 2006.
[40] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 3607–3613.
[41] K.-H. Siedersberger, "Komponenten zur automatischen Fahrzeugführung in Sehenden (semi-)autonomen Fahrzeugen," Ph.D. thesis, Inst. Syst. Dyn. Flight Mech., University of the Bundeswehr Munich, Neubiberg, Germany, 2003.
[42] H. Jaspers, M. Himmelsbach, and H.-J. Wuensche, "Multi-modal local terrain maps from vision and LiDAR," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Los Angeles, CA, USA, Jun. 2017, pp. 1119–1125.
[43] P. Fankhauser and M. Hutter, "A universal grid map library: Implementation and use case for rough terrain navigation," in *Proc. Robot Operating Syst. (ROS)-Complete Reference*, vol. 1, 2016, pp. 99–120.
[44] B. Forkel, "Wahrnehmung der statischen Umgebung eines autonomen Fahrzeugs auf Feldwegen durch die Fusion von Kamera und LiDAR," Ph.D. thesis, Inst. Auton. Syst. Technol. (TAS), University of the Bundeswehr Munich, Neubiberg, Germany, 2024.
[45] B. Forkel and H.-J. Wuensche, "Dynamic resolution terrain estimation for autonomous (Dirt) road driving fusing LiDAR and vision," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Aachen, Germany, Jun. 2022, pp. 1181–1187.
[46] M. Schreier, V. Willert, and J. Adamy, "Compact representation of dynamic driving environments for ADAS by parametric free space and dynamic object maps," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 367–384, Feb. 2016.
[47] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Feb. 1968.
[48] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," in *Proc. 1st Int. Symp. Search Techn. Artif. Intell. Robotics*, Jun. 2008, pp. 18–80.
[49] K. Kurzer, "Path planning in unstructured environments: A real-time hybrid A implementation for fast and deterministic path generation for the KTH research concept vehicle," M.S. thesis, Integr. Transp. Res. Lab, ITRL, KTH Royal Institute of Technology, Stockholm, Sweden, 2016.
[50] A. Bienemann and H.-J. Wuensche, "Model predictive control for autonomous vehicle following," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Anchorage, AK, USA, Jun. 2023, pp. 1–6.
[51] A. Bienemann, A. Reich, T. Luettel, and M. Maehlisch, "Perception-based accurate autonomous vehicle following in GNSS-denied environments," in *Proc. IEEE 27th Int. Conf. Intell. Transp. Syst. (ITSC)*, Edmonton, AB, Canada, Sep. 2024, pp. 3908–3914.
[52] F. Micheli, M. Bersani, S. Arrigoni, F. Braghin, and F. Cheli, "NMPC trajectory planner for urban autonomous driving," *Vehicle Syst. Dyn.*, vol. 61, no. 5, pp. 1387–1409, May 2023.
[53] C. M. Kang, S.-H. Lee, and C. C. Chung, "Comparative evaluation of dynamic and kinematic vehicle models," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 648–653.
[54] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2015, pp. 1094–1099.
[55] P. Polack, F. Altché, B. d'Andréa-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *Proc. IEEE Intell. Vehicles Symp.*, Los Angeles, CA, USA, Jun. 2017, pp. 812–818.
[56] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying MPC," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1586–1595, Jun. 2017.
[57] R. Findeisen, L. Grüne, J. Pannek, and P. Varutti, "Robustness of prediction based delay compensation for nonlinear systems," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 203–208, Jan. 2011.
[58] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Apr. 2005.
[59] M. Quigley, "ROS: An open-source robot operating system," in *Proc. ICRA workshop open source Softw.*, Kobe, Japan, May 2009, pp. 1–5.
[60] PostgreSQL Global Development Group. (2024). *PostgreSQL*. [Online]. Available: https://www.postgresql.org/
[61] N. Mu et al., "MoST: Multi-modality scene tokenization for motion prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2024, pp. 14988–14999.
[62] P. Mortimer and M. Maehlisch, "Survey on datasets for perception in unstructured outdoor environments," in *Proc. IEEE ICRA Workshop Field Robot.*, Yokohama, Japan, May 2024, pp. 1–8.
[63] P. Mortimer, R. Hagmanns, M. Granero, T. Luettel, J. Petereit, and H.-J. Wuensche, "The GOOSE dataset for perception in unstructured environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Yokohama, Japan, May 2024, pp. 14838–14844.
[64] A. Tiderko, F. Hoeller, and T. Röhling, "The ROS multimaster extension for simplified deployment of multi-robot systems," in *Robot Operating System (ROS)—The Complete Reference*, vol. 1. Cham, Switzerland: Springer, 2016, ch. 24, pp. 629–650.
[65] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Sci. Robot.*, vol. 7, no. 66, May 2022, Art. no. eabm6074.
[66] 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, Mar. 2012.
[67] D. Swaroop and J. K. Hedrick, "String stability of interconnected systems," *IEEE Trans. Autom. Control*, vol. 41, no. 3, pp. 349–357, Mar. 1996.
[68] J. Zhao et al., "Autonomous driving system: A comprehensive survey," *Expert Syst. Appl.*, vol. 242, May 2024, Art. no. 122836.
[69] B. Liao et al., "DiffusionDrive: Truncated diffusion model for end-to-end autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Nashville, TN, USA, Jun. 2025, pp. 12037–12047.