ADORe: Unified Modular Framework for Vehicle and Infrastructure-Based System Level Automation

Mikkel Skov Maarssoe[a], Sanath Konthala[b], Marko Mizdrak[c], Giovanni Lucente[d], Matthias Nichting[e], Thomas Lobig[f] and Andrew Koerner[g]

Institute of Transportation Systems, German Aerospace Center (DLR), Lilienthalplatz 7, 38108 Braunschweig, Germany

Keywords: Automated Driving, Cooperative Automated Driving, Research Framework, Modular Software, Open-Source, System-Level Automation.

Abstract: Recent advancements in automated driving have primarily focused on achieving autonomy within individual vehicles. However, a broader paradigm shift is emerging that leverages both vehicle-level autonomy and collaboration with other road users and infrastructure to optimize traffic flow and enhance safety. This paper introduces ADORe®(Automated Driving Open Research), an open-source Automated Driving System developed by the German Aerospace Center (DLR). ADORe adopts a modular, system-level approach, enabling seamless integration between Single-Agent Automated Driving for local autonomy and Multi-Agent Autonomous Driving for infrastructure-assisted decision-making. By utilizing vehicle-to-infrastructure (V2X) communication, ADORe facilitates coordinated multi-agent planning, dynamic route optimization, and improved situational awareness through shared data. The framework supports flexible testing via simulation tools like CARLA and SUMO, alongside deployment on research vehicles equipped with advanced sensors and teleoperation capabilities. Successful demonstrations in research projects like the German national Gaia-X4ROMS and MAD Urban validate ADORe's capability to bridge the gap between isolated autonomous driving and cooperative traffic systems. This collaborative approach highlights the potential of automated driving systems as a cornerstone of intelligent transportation systems, advancing safety, efficiency, and interoperability.

1 INTRODUCTION

Recent progress in automated driving has demonstrated remarkable advancements in perception, decision-making, and control, bringing self-driving vehicles closer to widespread deployment. However, many current approaches still frame the challenge as merely replacing human drivers inside the vehicles by an artificial intelligence. This perspective alone neglects the broader potential of automated driving as a system-level innovation. Communication technologies, such as vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) connectivity, enable more comprehensive approaches to automated driving. By facilitating real-time information exchange between vehicles and infrastructure, these technologies allow for multi-agent decision-making that transcends the limitations of local ego-vehicle-based optimization. Such systems can promote collaboration, where vehicles collectively negotiate traffic flow, adapt to dynamic conditions, and coordinate their movement with infrastructure like traffic lights and edge computers. A taxonomy for the different possible classes of cooperation is provided in SAE J3216 (SAE, 2024b), distinguishing specifically 'status-sharing', 'intent-sharing', 'agreement-seeking' and 'prescriptive cooperation'. The paradigm shift redefines automated driving not merely as individual autonomy but as a cornerstone of intelligent, cooperative mobility systems, paving the way for safer, more efficient, and seamlessly integrated transportation networks.

Automated Driving Open Research (ADORe®) is the German Aerospace Center's (DLR) open-source Automated Driving System (ADS). Unlike

[a] https://orcid.org/0009-0003-9999-8711
[b] https://orcid.org/0009-0008-4096-7685
[c] https://orcid.org/0009-0004-3931-2842
[d] https://orcid.org/0000-0002-7844-853X
[e] https://orcid.org/0000-0002-2484-4203
[f] https://orcid.org/0009-0009-4158-5184
[g] https://orcid.org/0009-0000-3837-3433

Maarssoe, M. S., Konthala, S., Mizdrak, M., Lucente, G., Nichting, M., Lobig, T. and Koerner, A.
ADORe: Unified Modular Framework for Vehicle and Infrastructure-Based System Level Automation.
DOI: 10.5220/0013405200003941
In *Proceedings of the 11th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2025)*, pages 571-581
ISBN: 978-989-758-745-0; ISSN: 2184-495X

Figure 1: DLRs research CAVs, NGC-FASCar, ViewCar2, & FASCarE.

many ADS that prioritize ego-vehicle performance, ADORe adopts a system-level approach, facilitating coordinated decision-making among Cooperative Automated Vehicles (CAV) and infrastructure systems. Its modular and containerized framework supports seamless integration and easy scalability, enabling vehicles to operate both independently and collaboratively. ADORe's dynamic design allows CAVs to continuously communicate with traffic lights, other vehicles, and infrastructure components facilitating the optimization of traffic flow and the enhancement of safety and efficiency. It enables adapting to varying operating domains by seamlessly switching between local autonomy, infrastructure-based support and decision-making, and human-assisted remote operation. By bridging the gap between isolated autonomy and cooperative traffic management, ADORe allows researchers to work with a system-wide perspective of automated driving. As such tool, its goal is to be used for prototypic research testing, and not for full SAE Level 4 and above driving (SAE, 2024a).

2 STATE OF THE ART

The field of Automated Driving Systems (ADS) has seen a surge in open-source tools in recent years. Due to the complexity of the driving task, there is no perfect tool that fits all needs. Many of these tools are focused on a particular level of abstraction like traffic design, sensor processing, driving policy, and others.

Having multiple integrated tools that leverage the advantages of more than one approach is an obvious path for ADS technology development, making the journey from idea to proof-of-concept prototype faster and easier.

There are already several existing open-source automation stacks that have been accessible for years and have matured into well-developed software. Autoware (Aut, 2023) is an open-source highly featured Automated Driving Stack based on ROS2, developed at Nagoya University. It is well-documented and maintained. Another automation stack is Apollo (Apo, 2024), based on the open-source middleware Apollo Cyber RT, developed by Baidu. RoboCar (Testouri et al., 2024) is another recent modular and cost-effective Automated Driving Stack. Alternatively, there are Automated Driving Stacks that are designed around research into Cooperative Driving like OpenCDA that focus on providing an environment for testing cooperative driving algorithms (Xu et al., 2021). For a comprehensive overview, we refer the reader to the survey papers (Tang et al., 2023), (Li et al., 2023).

Alongside these automation stacks, domain-specific simulation and modeling tools have been developed to address particular ADS application requirements. SUMO (Behrisch et al., 2011) is used for traffic flow simulation (supporting V2X communication and linking to network simulation tools like ns-3 (Lücken et al., 2021)), CARLA (Dosovitskiy et al., 2017) for vehicle dynamics simulation and different scenarios testing, and CommonRoad (Althoff et al., 2017), which focuses on planning policies. Comprehensive surveys on available tools (Li et al., 2023) (Tong et al., 2020) emphasize that the need for more integration of ADS tools, such as with SUMO for traffic simulation and with CARLA for more realistic vehicle dynamics, is critical.

3 METHODS

3.1 Architecture

ADORe is a modular system designed to be easily adaptable to other frameworks and various hardware

components. To achieve this, ADORe is conceptualized as two different categories of modules each containing its own set of nodes, the interface modules, and the ADORe Core. The ADORe interface modules are all hardware or companion software specific, meaning specifically designed for the DLR-provided hardware and software, or other companion software like the CARLA Simulator. The ADORe Core contains all modules for route planning, trajectory planning, and control used to achieve automated driving, and are specifically designed to avoid hardware or companion software-specific implementations, and are therefore adaptable to other frameworks as long as they fulfill the message and topic requirements of the ADORe Core ROS2 nodes. The ADORe Core can be further categorized into two distinct sets of modules, as the planning behavior differs between them: on the one hand, the Single-Agent Automated Driving (SAAD) modules, and on the other hand, the Multi-Agent Automated Driving (MAAD) modules. A simplified overview of all the modules can be seen in Fig. 2.

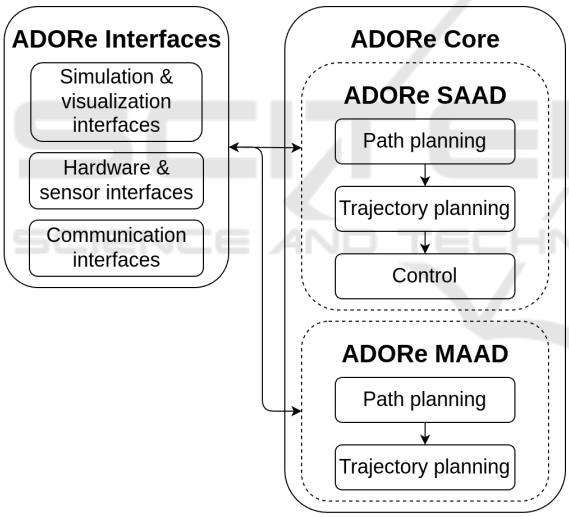

Figure 2: ADORes main categories of modules, ADORe interface and ADORe core. The ADORe Core is further separated into modules used for single-agent automated driving (SAAD) and multi-agent automated driving (MAAD).

What distinguishes ADORe from many other alternative ADS is that it is designed not only to run on local hardware in a single ego vehicle using SAAD, but also to perform planning for groups of CAVs when operating on remote hardware, such as road-based local infrastructure communicating through V2X with the individual vehicles. ADORe is designed around this collaborative relationship between SAAD and MAAD, as depicted in Fig. 3. ADORe enables a CAV running ADORe SAAD to drive autonomously on local in-vehicle hardware when out of range of cooperative infrastructure, yet supports switching to receiving its trajectories, control or maneuver advisories or commands from ADORe MAAD through V2X whenever available. The benefit of using infrastructure-provided commands and advisories is that they are based on local infrastructure sensors tailored to the related terrain and therefore enhancing safety, while they can also be influenced by traffic management systems focusing on the optimized throughput and efficiency of all traffic participants in the local area (Schindler et al., 2023).

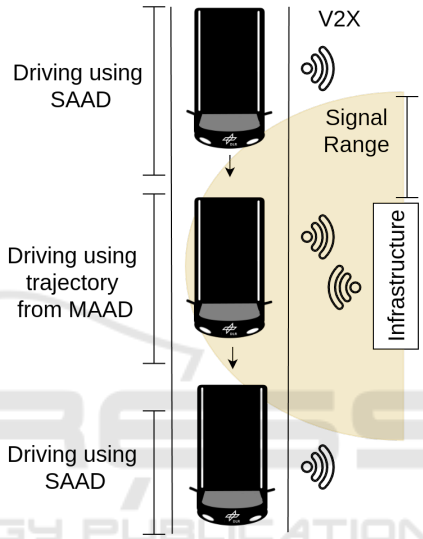

Figure 3: A scenario showing a vehicle driving along a route, running ADORe SAAD for autonomous driving, when the car is within range of the infrastructure running ADORe MAAD, the planning switches to be handled on the infrastructure and gets switched back again once the vehicle is out of range.

3.2 Integrated Build System & Development Environment

ADORe features a container-based development environment built on Docker (Doc, 2024). This system serves multiple roles, acting as a development environment with debugging and development tools preinstalled, as well as a runtime environment with all necessary dependencies. It includes a virtual display for executing commands and programs in contexts without standard input or output, enabling programs that require a display to run in a headless environment (no standard input and standard output), such as an automated build server. Known as the ADORe CLI, this build system leverages GNU Make non-conventionally as a command runner and operates within a containerized Docker context, support-

ing both x86 and ARM64 architectures. The ADORe CLI uses Docker "volumes" to link the host filesystem to the ADORe source code directory, allowing system requirements and dependencies to be dynamically added by individual components, modules, or nodes. Additionally, it integrates ccache, a tool that caches C++ compilations to reduce repeat compile times, enhancing development efficiency. The ADORe CLI comes preinstalled with ROS 2 Jazzy and Ubuntu 24.04, both of which are configurable, along with a collection of common ROS development tools, network debugging tools, and other command-line tools. This system supports not only a standardized development environment but also facilitates automation for automated building and testing.

3.3 Simulation & Visualization

3.3.1 Visualization

ADORe allows visualizing with all common ROS2 visualization tools like RViz (RVi, 2024), Foxglove (Fox, 2024), and Lichtblick (lic, 2024).

This is done via a visualization node which is responsible for subscribing to all relevant messages and converting them to visualizable messages (primarily Marker Arrays). Including:

- Ego vehicle
- Other detected traffic participants
- Road/lane boundaries
- Planned trajectories
- Goal point
- Route to goal
- Satellite image of the map area
- Driven path

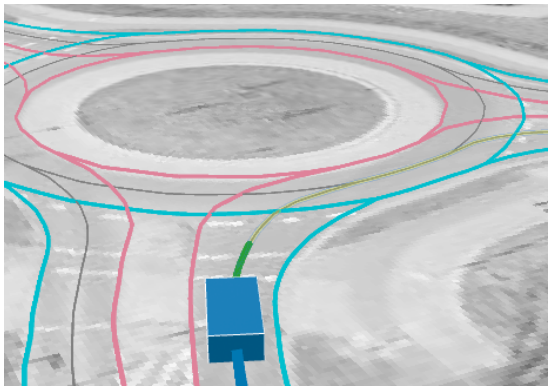

Figure 4: Example of ADORe visualization in Lichtblick.

3.3.2 Simulation

For testing in simple cases, ADORe provides a ROS2 node for simulating a vehicle with a simple vehicle model which utilizes tire forces based on a simplified dynamic bicycle model involving front and rear tire stiffness, considers the effects of yaw rate, lateral velocity, and steering angle, accounts for rotational inertia and lateral tire forces through friction and stiffness parameters, implements a simplified dynamic force balance to compute lateral acceleration and yaw rate changes (Rajamani, 2011). Many instances of the Simulated Vehicle node can be spawned to test multi-vehicle systems. The Simulated Vehicle node also has several additional features, such as manual driving through a teleoperations panel, the ability to toggle between manual and autonomous driving, and optional generated/synthetic input noise.

3.3.3 CARLA Simulator Interface

The CARLA Simulator (Dosovitskiy et al., 2017) is a popular open-source simulation tool for automated driving with a realistic graphical output. ADORe has an interface to CARLA extending its applicability in research. For example, the interface facilitates the simulation and testing of the ADORe vehicle automation using the precise vehicle models provided by CARLA and it supports the execution of complex driving scenarios. Additionally, the interface enables the evaluation of perception models by leveraging CARLA's realistic graphical environment in conjunction with the ADORe automation framework. The interface ensures bidirectional data exchange by converting information into the appropriate formats for both simulators. The CARLA Simulator is provided in a containerized way by the CARLA team allowing a straightforward deployment and execution together with the ADORe framework.

3.3.4 SUMO Interface

The SUMO (Simulation of Urban Mobility) (Alvarez Lopez et al., 2024) tool is a widely used open-source microscopic traffic simulation tool. It is capable of supporting the simulation and analysis of intermodal traffic flows within the context of large-scale networks. ADORe provides an interface to SUMO. This enables the utilization of SUMO for the lightweight simulation of vehicles and traffic flows in the vicinity of a vehicle that is simulated with ADORe. Furthermore, it permits the operation of an automated vehicle within a SUMO simulation via the ADORe automation. This allows for the easy assessment of the impact of an automated vehicle operated

by ADORe on the surrounding traffic flow. To achieve the aforementioned functionality, the ADORe SUMO bridge synchronizes both the ADORe and SUMO simulations and transfers the information of all statuses of the traffic participants in the simulations. The data exchange is based on libsumo, which is the C++ interface of SUMO. The direction of data transfer for each participant is determined based on whether a given traffic participant is simulated in SUMO or ADORe. In each time step, the vehicle status is transferred and translated.

3.4 Hardware & Sensors

3.4.1 Research Vehicles

ADORe is deployed to all three of DLR's research vehicles seen in Fig. 1. There is the NGC-FASCar, a Mercedes EQV equipped with an acceleration and steering interface solution developed by Paravan. Then there is also the FASCarE, a Volkswagen eGolf, and the ViewCar2, a Volkswagen Passat GTE. Both cars are controlled using enhanced CAN-bus interfaces. ADORe's trajectory tracker sends commands to a vendor-specific low-level interface to control the actual vehicle. While the vendor-specific interface part cannot be open-sourced, the trajectory tracker's output is simple enough to be adopted for any kind of vehicle interface. All DLR's research vehicles are outfitted with LIDARs and cameras as well as GNNS and a high-precision IMU. The research vehicles have a DLR-internal sensor perception and fusion solution which is also based on ROS, so the interface with ADORe is based on ROS2 messages to provide vehicle state information and surrounding object data. Thus the interface is generic and modular and can be adapted to a wide range of automation capable vehicles.

3.5 Single-Agent Automated Driving

The ADORe SAAD is responsible for calculating routes, trajectories, decisions, V2X, and the actual vehicle command to drive the vehicle to fulfill its goals. Fig. 5 shows the different ROS2 nodes and topics communicating with each other and gives an overview of the flow of execution. ADORe SAAD receives through its interfaces the current ego vehicle state, other traffic participants, and a variety of V2X topics. Using these topics, ADORe SAAD first sends the necessary information to the mission controller, which handles everything related to navigation. Afterwards, the decision maker is used to determine which planner is to be used. Different planners exist for different purposes, which generate appropriate trajectories, see below for details. The planned trajectories can also be published using V2X. Lastly, ADORe SAAD uses the trajectory tracker to calculate low-level vehicle commands.

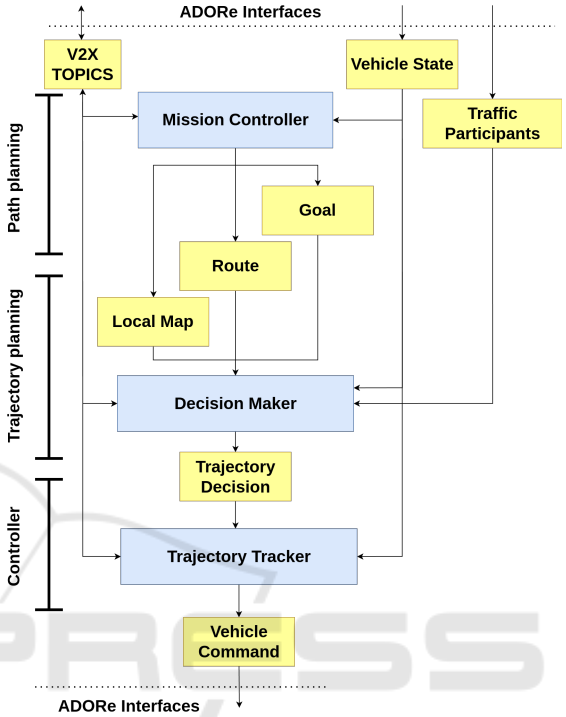

Figure 5: The ADORe SAAD nodes and topics overview, yellow boxes being ROS2 topics & messages, and blue boxes are nodes.

3.5.1 Mission Controller

The *Mission Controller* is responsible for the vehicle's high-level navigation, this includes map loading, keeping track of goals, and route planning. The *Mission Controller* keeps track of a queue of goals. This enables, for example, multiple pick up and drop-offs at different locations. When many destinations should be reached, many successive routes can be planned.

Two formats of HD maps are supported. Road2Simulation maps and OpenDRIVE maps.

When the high definition (HD) map is loaded from a file, a lane graph weighted by the lengths of the lanes is created and Dijkstra is used to find the shortest path to the goal along the drivable lanes. The *Mission Controller* keeps track of the progress along this route and adjusts accordingly, recalculating when the vehicle deviates too far from it.

A local cut-out of the larger map, the route, and the goal are sent onward to other parts of the automation.

3.5.2 Decision Maker & Trajectory Planner

The *Decision Maker* is the core component of ADORe SAAD, responsible for managing the high-level decision-making required for seamless and safe automated vehicle operation. It operates through a hierarchical state machine that dynamically determines the type of trajectory and action executed by ADORe SAAD. At its highest level of abstraction, the *Decision Maker* can transition between currently seven prioritized states:

- **Emergency Stop:** Halts the vehicle as quickly as possible.
- **Minimum Risk Maneuver:** Strategically brings the vehicle to a stationary position safely.
- **Requesting Assistance:** Signals for human intervention when the vehicle encounters a situation requiring external input (see Sec. 4.1.1).
- **Remote Operations:** Allows human operators to guide the vehicle remotely during challenging scenarios. (see Sec. 4.1.1).
- **Safety Corridor:** Engages when a safety corridor message is received, such as one issued by emergency vehicles, instructing the vehicle to clear a path (see Sec. 4.1.2).
- **Standstill:** State for stopping or remaining stationary.
- **Follow Route:** Creates a trajectory following the lane center of a route provided by the mission controller.
- **Follow Reference:** Follows a reference trajectory provided from elsewhere, typically ADORe MAAD.

The transition into and behavior within each state depends on the data received through the topics and interfaces described in Sec. 3.5. For instance, if localization is lost, the *Emergency stop* state is triggered. Alternatively, when a reference trajectory is available, the *Follow reference* state is prioritized to ensure optimal trajectory execution.

The *Decision Maker* and *Trajectory Planner* work in tandem to enable seamless transitions between local automation, infrastructure-assisted driving, and remote operations.

The *Decision Maker* determines the vehicle's current operational state and generates the appropriate reference inputs for the *Trajectory Planner*. Depending on the state, the reference can take different forms: a reference path to the goal, safety corridor lanes, or a multi-agent-based reference trajectory from MAAD. For example, when providing a reference path to the goal, the *Decision Maker* also computes a reference velocity using the Intelligent Driver Model (IDM) (Malinauskas, 2014), ensuring safe minimum distance and time headway from other vehicles. These references are then passed to the *Trajectory Planner*.

The *Trajectory Planner* uses these inputs, along with additional information such as vehicle dynamics and constraints, to generate a feasible and safe trajectory. It employs a nonlinear constraints optimizer, named OptiNLC (Opt, 2024), an OSQP-based optimization solver (Stellato et al., 2020) library developed by DLR, to solve the optimal control problem. The optimization cost function is designed to ensure the vehicle follows the center of the lane smoothly while adhering to kinematic and dynamic constraints. It prioritizes passenger comfort by minimizing lateral deviation, aligning the vehicle's heading with the reference path, and promoting smooth, efficient motion.

By optimizing control inputs at each point within the planning horizon, the algorithm iteratively solves for the trajectory that satisfies convergence criteria. While the current implementation focuses on longitudinal motion planning guided by IDM, future developments aim to include more sophisticated lateral planning and lane-changing capabilities, further enhancing the system's robustness and adaptability.

3.5.3 Trajectory Tracker

To ensure that the automated vehicle follows the planned trajectory, we have developed multiple controllers to provide flexibility. These include a PID controller, a Model Predictive Controller (MPC), and an iterative Linear Quadratic Regulator (iLQR).

The PID controller computes control commands by penalizing errors in position, velocity, and heading relative to a point on the reference trajectory, with separate PID gains for each error type. Anti-windup mechanisms prevent integral terms from growing uncontrollably, while additional terms for comfort constraints (e.g., limits on steering velocity and acceleration jerk) ensure smooth transitions.

The Model Predictive Controller improves upon this by considering not just a single reference point on the planned trajectory but a prediction horizon. This also uses OptiNLC, the same OSQP-based optimization solver library as is used in trajectory planning. This approach results in smoother and more comfortable control outputs. The MPC's cost function incorporates various factors to optimize trajectory tracking, including minimizing lateral deviation, penalizing steering angle velocity, and reducing longitudinal jerk.

The iLQR controller also works in an MPC-like way. At each time step, it calculates a control sequence over a longer horizon while only executing

the first part. It operates by iteratively refining an initial control sequence through forward and backward passes, leveraging linear approximations of system dynamics and quadratic approximations of the cost function. In the forward pass, the algorithm simulates the system's response to the current control sequence using nonlinear dynamics, generating a state trajectory. The backward pass linearizes the dynamics around this trajectory and applies dynamic programming to compute a time-varying feedback control law that minimizes a local quadratic approximation of the cost-to-go.

This implementation of iLQR incorporates a cost function that penalizes deviations in longitudinal and lateral position, velocity, and heading while adding regularization terms for jerk and steering rate to ensure smooth and feasible motion. The dynamics are represented using discretized linearized matrices, which model how states and controls influence the system over time. Cost derivatives are computed to approximate the sensitivity of the cost function, enabling precise control updates. A line search mechanism ensures stability by scaling control updates while warm-starting from previously computed trajectories accelerates convergence.

3.6 Multi-Agent Autonomous Driving

ADORe MAAD is designed for working on infrastructure or surrounding hardware that remotely communicates with ADORe SAAD on one or multiple vehicles. The idea of communication with infrastructure for planning for multiple CAVs gains multiple advantages. First, each vehicle gains access to information not achievable by a single vehicle like in the scenario seen in Fig. 6. In Fig. 6 the DLR CAV is driving along a road, and due to the viewing angle the car is unable to see the person running toward the CAV's driving direction. However, the person is within the viewing angle of the infrastructure on the traffic light, and through communication the CAV can gain an understanding of traffic that not even humans can achieve.

Secondly, since ADORe MAAD is designed for planning multiple CAVs' trajectories simultaneously, it can optimize goals that involve many agents, for example, to improve traffic and eliminate phantom queues.

Due to the nature of controlling multiple vehicles, the structure of ADORe MAAD is simplified by removing the trajectory controller, to avoid assumptions of the individual vehicles and their hardware. The ADORe MAAD nodes and topics, when controlling groups of vehicles, can be seen in Fig. 7.

The ADORe interfaces are responsible for providing ADORe MAAD with a list of traffic participants, the HD map, and a list of the traffic participants the framework can control. The respective controllability is defined through received V2X messages, which contain state, capability and goal point information (see (Schindler et al., 2024)) to be sent to the view node. A map provider node is responsible for creating a local map for the view component. This component, described in Sec. 3.6.1 generates a view message, containing information on each vehicle's driving options and where they are heading. The view message is then sent to the traffic predictor, which as

Figure 6: Example of the advantages when combining infrastructure and local autonomous driving, the CAV is driving along a road, and unable to see the person running towards it, as they are hidden behind another vehicle.

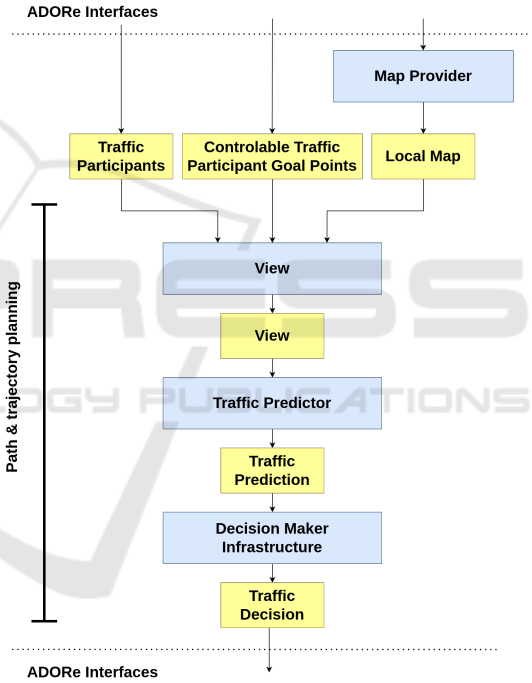

Figure 7: ADORe MAAD ROS2 nodes and topics overview.

described in Sec. 3.6.2, will generate trajectories for all traffic participants. Lastly, the MAAD decision maker, which is intended to be used for creating improvement to the trajectories in the future, but right now only has the functionality of filtering out the traffic participants not relevant for the vehicles driving using ADORe SAAD.

3.6.1 View

The purpose of the View module is to provide a localized representation of the HD map at the agent level, removing unnecessary structures and information while delivering the lane features essential for downstream planning tasks. The function of the View module can be understood by considering the type of information a multi-agent trajectory planner requires to plan the ego vehicle's trajectory and predict the trajectories of other traffic participants. This required information includes:

- The state of each traffic participant, including the ego vehicle. This comprises position, orientation, yaw rate, speed, acceleration, dimensions, and ID.
- A description of the center lane (with any potential forks) as well as the left and right lanes, if present, for each traffic participant in the scenario.

The role of the View module is to associate each traffic participant with its respective lane and incorporate this information into the published View topic.

The view message consists of a vector of vehicle items. Each vehicle item contains the state of the vehicle and vectors for its center lanes (plural, to account for forks), left lanes, and right lanes. Each lane item further includes the points defining its center line.

3.6.2 Trajectory Predictor

Planning in a multi-agent environment like traffic requires predicting the trajectories of other traffic participants. This must be done simultaneously with planning, as the trajectory of the ego vehicle affects and is affected by the trajectories of other traffic participants.

There are two ways in which this is done. Firstly, a primitive forces-based approach where each agent is pulled toward making progress on the lane centers while being repulsed by other agents. This method benefits from being fast to compute while suffering from a lack of optimality. A second, more complex approach involves modeling the traffic scenario as a dynamic game and computing its Generalized Nash Equilibrium (GNE) (Lucente et al., 2024). The GNE represents a strategy combination in which each player of the game adopts their best strategy, assuming that the strategies of the other players remain unchanged. In this context, each vehicle is modeled as a player, and its objective is to choose the optimal strategy (trajectory) that minimizes its cost while respecting collision constraints. To compute the GNE, the constrained optimization problems are reformulated as unconstrained optimization problems using the augmented Lagrangian method. These unconstrained problems are then solved using the Trust Region method (Lucente et al., 2024).
While the Trust Region method ensures local convergence, it cannot guarantee global convergence. Consequently, there is no assurance that the equilibrium found constitutes a true Nash equilibrium. For this reason, the solver's performance is sensitive to the initial solution. If the initial guess is sufficiently close to the GNE, it is reasonable to assume that the solution will converge to the global minimum and, thereby, to the GNE.

3.6.3 Middleware Service

ADORe is used in very varying projects that impose different requirements in terms of back-end connectivity. The requirements usually revolve around obtaining mission goals or propagating the vehicle status. To avoid project-specific implementation details within the ADORe core, we implemented a simple middleware service. This backend service is responsible for interfacing with the different project-specific backends while keeping the API for the ADORe side stable. Only a small portion of project-specific glue code in the form of a Python ROS2 node is needed, while the middleware service handles complex handshakes and any other kind of server communication.

4 EXPERIMENTS

ADORe has successfully been tested in multiple research projects. Also, several demonstrations have been conducted in different use cases, performed in different cities and under multiple environmental conditions.

4.1 Passenger Transport

In the German national research project Gaia-X4ROMS, which deals with the support and remote operation of automated and networked mobility services, the NGC-FASCar has demonstrated a prototypic implementation of a shuttle service (Rom, 2024). It was bookable through a mobile app to

pickup individual passengers at specific bus stops. The CAV drove around the perimeter along the green path shown in Fig. 8. During the drives, it had to interact with a mobile traffic light sending out V2X messages.

Figure 8: Gaia-X4ROMS driving demo area, showing the two different pick-up and drop-off locations and a traffic light.

A video of the vehicle driving the route observed in Fig. 8 can be seen in: https://github.com/DLR-TS/adore_videos/blob/main/links.md#hamburg-driving-demo.

4.1.1 Remote Operations

A main use case that has been worked on at DLR is remote operations, where a remote operator is taking over control of CAVs running ADORe SAAD, in situations where the CAV is unable to make decisions itself. The focus is primarily on remote assistance while remote driving plays a minor role. Detailed information regarding the related project scope is provided at (Rem, 2024). An example scenario where remote operations could be beneficial can be seen in Fig. 9, where the CAV is driving along a given route, that is blocked by temporal road construction. However, due to the solid line road marking, the CAV is not allowed to cross the line to pass the obstacle, even though there is no other way to get around. The vehicle simply may not be allowed to perform this action due to its restricted Operational Design Domain (ODD).

During the scenario, the CAV is approaching the obstacle while continuously monitoring its Operating Domain in conjunction with the defined ODD. In case there is a mismatch, the CAV is triggering a Minimum Risk Maneuver (MRM) while at the same time it is requesting remote assistance as explained in Sec. 3.5.2. The MRM consists of a smooth braking maneuver accompanied by hazard lights. Contacting of the remote operator is currently achieved by pushing related state information via MQTT wirelessly through

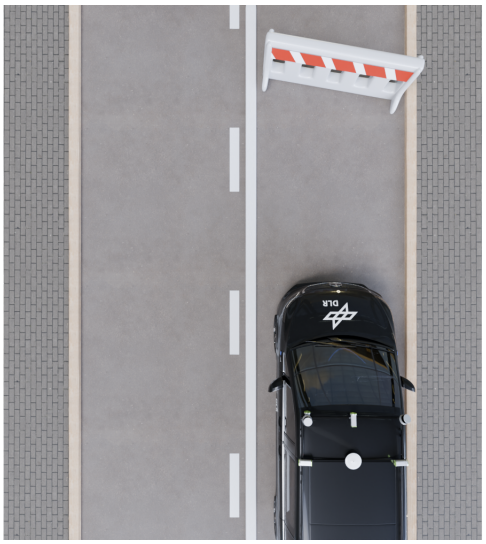

Figure 9: Example of a situation an CAV can encounter, where human intervention could be required.

the nearby infrastructure or by using mobile communication. The CAV and remote operator then begin the negotiation depicted in Fig. 10.

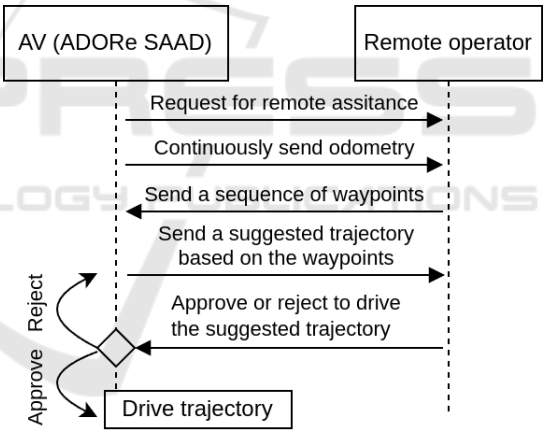

Figure 10: The communication between the AV running ADORe SAAD and the remote operator to escape a situation.

After first sending a request for remote assistance the CAV will continuously throughout the negotiation transmit odometry at 1 Hz. The remote operator, upon receiving the request and odometry will then see a visualization of the CAV in their visualization tool seen in Fig. 11. From the visualization tool, the remote operator can then enter a set of waypoints that they want the CAV to drive, as seen in Fig. 11.A, which is then sent to the CAV.

After the CAV receives the waypoints, it will calculate a trajectory locally using its own planner as described in Sec. 3.5.2, and return it to the remote

Figure 11: A. The remote operator's visualization of the AV running ADORe SAAD with placed waypoints. B. The remote operator's visualization of the received trajectory.

operator for final approval. When the remote operator receives the trajectory, it will be visualized as shown in Fig. 11.B, and the remote operator can then decide to either approve or reject. If the remote operator rejects the trajectory, the CAV will generate a new one for approval. If the trajectory is accepted, the CAV will begin driving the trajectory until it reaches the end, where it switches back to automated driving and stops contact with the remote operator. A video of the car driving the remote operations example shown in Fig. 11 can be seen here: `https://github.com/DLR-TS/adore_videos/blob/main/links.md#remote-operation-demo-braunschweig`.

4.1.2 Safety Corridor

Part of the German national project Gaia-X4AMS (Advanced Mobility Services) was about leveraging vehicle automation to create safety corridors during emergencies (Ams, 2024). In this scenario, the automated vehicle detects an emergency through V2X communication, receiving detailed information about the safety corridor. This information includes a polygonal area representing the path an emergency vehicle will take. Upon receiving this data, the automated vehicle promptly calculates and executes an evasive trajectory to vacate the polygon area. By doing so, it helps forming an emergency lane, ensuring a clear path for the emergency vehicle to pass through.

4.2 Collaborative Automated Driving

The collaborative relationship outlined in this paper between ADORe SAAD and ADORe MAAD has been developed as part of the German national research project MAD-Urban (Managed Automated Driving) (Mad, 2024). Throughout this project, protocols and the related system architecture for "Managed Automated Driving" have been developed and implemented in research vehicles and local infrastructure components of DLR and project partners (Schindler et al., 2024). First tests successfully established the link between SAAD and MAAD, so that infrastructure-based trajectories have been followed by CAVs. The work is continued to bring the technology to public roads.

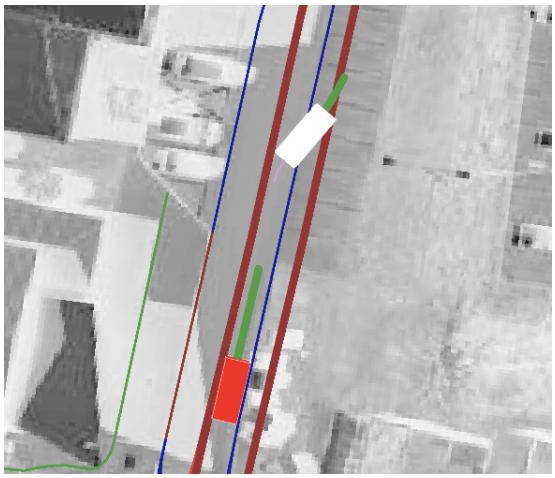

Figure 12: Ego Vehicle (white) leaving the safety corridor to form emergency lane for emergency vehicle (red) to pass by.

5 CONCLUSION

This paper introduces ADORe, German Aerospace Center's research framework for system-level automated driving of single CAVs controlled in-vehicle, and multiple CAVs controlled by surrounding infrastructure, achieving both automated driving and cooperative automated driving. ADORe is ROS2-based and comes packaged with its own integrated build system and developer environment. The software framework is divided into three stacks. First, the ADORe interfaces, tailored to specific hardware or software, are the only modules that would need modification to adapt to new vehicles or software. Secondly, the ADORe single-agent autonomous driving stack, designed for use inside of a single vehicle, which based on its conditions and available data can make decisions on how to drive, plan, and execute trajectories independently. Thirdly, the ADORe multi-agent automated driving is intended to run on infrastructure, and will generate trajectories for all CAVs in the local area, allowing for traffic optimizations not possible by a single vehicle. ADORe has been successfully used and demonstrated in several research projects. While being further developed in recently

started and upcoming research projects, it is open-source and available on Github: `https://github.com/DLR-TS/adore`.

REFERENCES

(2023). Home Page - Autoware. [Online; accessed 10. Dec. 2024].

(2024). AMS. [Online; accessed 17. Dec. 2024].

(2024). Apollo. [Online; accessed 10. Dec. 2024].

(2024). Docker: Accelerated Container Application Development. [Online; accessed 19. Dec. 2024].

(2024). Foxglove - Visualization and observability for robotics developers. [Online; accessed 20. Dec. 2024].

(2024a). J3016_202104: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles - SAE International. [Online; accessed 13. Dec. 2024].

(2024b). J3216_202107: Taxonomy and Definitions for Terms Related to Cooperative Driving Automation for On-Road Motor Vehicles - SAE International. [Online; accessed 13. Dec. 2024].

(2024). lichtblick. [Online; accessed 10. Dec. 2024].

(2024). MAD Urban. [Online; accessed 17. Dec. 2024].

(2024). OptiNLC. [Online; accessed 20. Dec. 2024].

(2024). Remote Operation: An important building block from Braunschweig for the mobility of the future. [Online; accessed 17. Dec. 2024].

(2024). ROMS Abschlussevent. [Online; accessed 17. Dec. 2024].

(2024). rviz. [Online; accessed 20. Dec. 2024].

Althoff, M., Koschi, M., and Manzinger, S. (2017). Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE.

Alvarez Lopez, P., Banse, A., Barthauer, M., Behrisch, M., Couéraud, B., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Nippold, R., and Wagner, P. (2024). Simulation of urban mobility (sumo).

Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). Sumo - simulation of urban mobility: An overview. In of Oslo Aida Omerovic, S. . U., Simoni, R. I. R. T. P. D. A., and Bobashev, R. I. R. T. P. G., editors, *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68, Barcelona, Spain. ThinkMind.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16.

Li, Y., Yuan, W., Zhang, S., Yan, W., Shen, Q., Wang, C., and Yang, M. (2023). Choose Your Simulator Wisely: A Review on Open-source Simulators for Autonomous Driving. *arXiv*.

Lucente, G., Maarssoe, M. S., Konthala, S. H., Abulehia, A., Dariani, R., and Schindler, J. (2024). Deepgametp: Integrating dynamic game theory and deep learning for trajectory planning. *IEEE Open Journal of Intelligent Transportation Systems*, pages 1–1.

Lücken, L., Schwamborn, M., Mintsis, E., Koutras, D., Karagounis, V., Correa, A., Sepulcre, M., Coll Perales, B., Thandavarayan, G., Blokpoel, R., Zhang, X., Huisken, G., Boerma, S., Maerivoet, S., Carlier, K., Pápics, P., Ons, B., Tourwé, S., Banse Bueno, O. A., and Schindler, J. (2021). TransAID Deliverable 6.2/2 - Assessment of Traffic Management Procedures in Transition Areas. [Online; accessed 11. Dec. 2024].

Malinauskas, R. (2014). The intelligent driver model: Analysis and application to adaptive cruise control.

Rajamani, R. (2011). *Vehicle dynamics and control*. Springer Science & Business Media.

Schindler, J., Dariani, R., Klein, P., Lee, A., and Fleck, T. (2024). Automated Vehicles Controlled By Smart Infrastructure – The Architecture Of Managed Automated Driving. In ERTICO, editor, *2024 ITS World Congress Paper Publications Part 2*, pages 1210–1220, Dubai, VAE. ERTICO.

Schindler, J., Klein, P., Fleck, T., and Lee, A. (2023). Managed Automated Driving (MAD) - a Concept for Empowering Road Infrastructure. In ERTICO, editor, *2023 ITS European Congress Book of Abstracts*, pages 933–944, Lisbon, Portugal. ERTICO.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2020). Osqp: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672.

Tang, S., Zhang, Z., Zhang, Y., Zhou, J., Guo, Y., Liu, S., Guo, S., Li, Y.-F., Ma, L., Xue, Y., and Liu, Y. (2023). A Survey on Automated Driving System Testing: Landscapes and Trends. *ACM Trans. Software Eng. Method.*, 32(5):1–62.

Testouri, M., Elghazaly, G., and Frank, R. (2024). RoboCar: A Rapidly Deployable Open-Source Platform for Autonomous Driving Research. *arXiv*.

Tong, K., Ajanovic, Z., and Stettinger, G. (2020). Overview of Tools Supporting Planning for Automated Driving. *arXiv*.

Xu, R., Guo, Y., Han, X., Xia, X., Xiang, H., and Ma, J. (2021). Opencda: an open cooperative driving automation framework integrated with co-simulation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1155–1162. IEEE.