2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)
October 8-12, 2022, Macau, China

The OPA^{3}L System and Testconcept for Urban Autonomous Driving

Andreas Folkers[1], Constantin Wellhausen[2], Matthias Rick[1], Xibo Li[1], Lennart Evers[3], Verena Schwarting[2], Joachim Clemens[2], Philipp Dittmann[4], Mahmood Shubbak[1], Tom Bustert[1], Gabriel Zachmann[4], Kerstin Schill[2], Christof Büskens[1]

2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC) | 978-1-6654-6880-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITSC55140.2022.9922416

***Abstract*— The development of autonomous vehicles for urban driving is widely considered as a challenging task as it requires intensive interdisciplinary expertise. The present article presents an overview of the research project OPA^{3}L (Optimally Assisted, Highly Automated, Autonomous and Cooperative Vehicle Navigation and Localization). It highlights the hardware and software architecture as well as the developed methods. This comprises algorithms for localization, perception, high- and low-level decision making and path planning, as well as model predictive control. The research project contributes a cross-platform holistic approach applicable for a wide range of real-world scenarios. The developed framework is implemented and tested on a real research vehicle, miniature vehicles, and a simulation system.**

I. Introduction

Global road safety statistics show that about 1.35 million deaths and 50 million injuries yearly result from road traffic crashes. Traffic accidents are considered the leading cause of death for children and young adults aged 5-29 years [1]. Moreover, the economic cost of traffic accidents is estimated at 3% of the gross domestic product of most countries [2]. The main risk factors behind road traffic crashes are human error, speeding, driving under the influence of alcohol and other psychoactive substances, distracted driving, and non-compliance with traffic rules [1].

Against these facts, the development of safer driving technologies, in which human behavioral risk factors are eliminated, is inevitable. Autonomous vehicles are widely seen as a revolutionary technology that not only holds the promise of safer roads and fewer traffic accidents, but also less traffic congestion, more effective usage of vehicles and space, better fuel efficiency and environment friendliness, as well as an easier life for elderly and disabled people [3].

This work was partially supported by the German Aerospace Center (DLR) Space Administration with financial means of the German Federal Ministry for Economic Affairs and Energy (BMWi), project OPA^{3}L (grant No. 50 NA 1909).

[1]The authors are with the Group for Optimization and Optimal Control, University of Bremen, Bremen, Germany, contact afolkers@uni-bremen.de

[2]The authors are with the Cognitive Neuroinformatics Group, University of Bremen, Bremen, Germany, contact wellhausen@uni-bremen.de

[3]The author is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Projectnumber 281474342/GRK2224/1, contact levers@uni-bremen.de

[4]The authors are with the Computer Graphics and Virtual Reality Research Lab, University of Bremen, Bremen, Germany, contact dittmann@uni-bremen.de

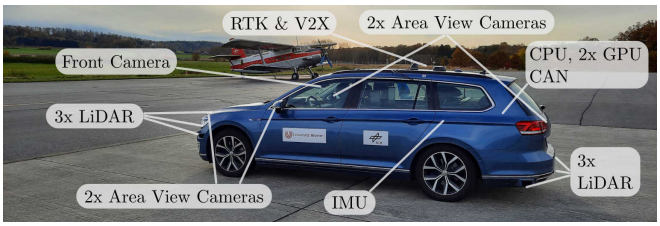

Fig. 1. Sensor setup of the research car

The development of autonomous vehicles for urban driving is widely considered as a challenging mission due to the highly dynamic nature of such environments. Over the past years, a considerable progress in developing autonomous driving systems has been made [4]. This has been realized by intensive collaborations of expertise from various fields, such as electronics: sensing and actuating systems [5], computer science and artificial intelligence [6] [7], mathematical optimization and optimal control systems [4] [8] [9], as well as law and social sciences [3].

This article presents an overview of the research project OPA^{3}L (Optimally Assisted, Highly Automated, Autonomous and Cooperative Vehicle Navigation and Localization). It explains the system architecture of the research vehicle developed within the scope of the project for urban driving applications, in terms of its hardware and software architecture. Accordingly, the present article describes the used methods and algorithms for localization, perception (based on LiDAR and camera), object tracking, strategic and tactical decision making, motion planning, and model predictive control. It is shown how simulated and miniaturized environments are utilized for the rapid development of the system. Finally, the article presents a real-world example for fully autonomous urban driving.

The project contributes a cross-platform holistic approach that is applicable for a wide range of real-world scenarios. This is due to its ability to abstract both static and dynamic information in the vehicles surroundings in a very generalized fashion. The planning and control algorithms, accordingly, are suited to provide an anticipatory guidance of the vehicle even in unknown situations.

II. System Overview

A. Research Vehicle

The research car is a customized hybrid VW Passat GTE (Fig. 1). Vehicle specific sensor information (e. g. current

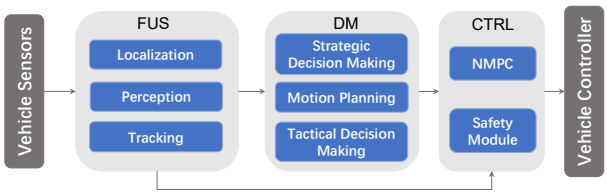

Fig. 2. Overview of the software architecture.

steering angle or wheel speeds) as well as control commands (e. g. desired steering angle or acceleration) are transferred via CAN. A camera in the windshield records the area in front in high resolution. This is supplemented by four ValeoVis area view cameras with medium resolution at low driving speeds (up to $10\,\mathrm{km/h}$). Six Ibeo ScaLa LiDAR sensors, measuring at $25\,\mathrm{Hz}$, cover the entire surroundings of the vehicle, with an increased density in direct front and rear. Two u-blox F9P receivers are used for obtaining Real-time kinematic (RTK) at $10\,\mathrm{Hz}$. An ADIS 16488 IMU provides accelerometer and gyroscope measurements at $800\,\mathrm{Hz}$. For V2X (Vehicle to everything) a Nordsys waveBEE, is used to communicate with other traffic participants. All computations are performed on an on-board computer with an Intel Core i9-9900K CPU and two NVIDIA GeForce RTX 2080 Ti GPU's.

B. Sketch of Software Architecture

The proposed software architecture is a successor of the autonomous driving system presented in [9], where the framework is divided into three main modules: Multi-Sensor Fusion (FUS), Decision Making (DM) and Control (CTRL), as shown in Fig. 2. However, within the scope of OPA^{3}L all modules were extended by additional functionalities according to the urban driving scenarios.

Firstly, an image processing part is added, which enables the autonomous vehicle to perform road segmentation, object detection and motion prediction. Secondly, DM is rearranged into three parts, where the strategic decision making provides the global routing by planning a route from the current location of the vehicle to the desired destination. The motion planning and tactical decision making perform the local trajectory planning as before. Finally, the CTRL part generates feasible control signals computed by a nonlinear model predictive controller (NMPC). The software architecture benefits from a well-structured modular design with a scalable deployment capability, making it suitable for a variety of driving platforms.

III. Localization

A precise estimate of the vehicle's state as well as its surroundings is required for safely maneuvering an autonomous vehicle. While state estimation is described in this section, perceiving the surroundings and estimation of the state of other traffic participants is explained in Sect. IV and Sect. V respectively.

The localization module determines the state of the vehicle in its environment which is then used directly or indirectly by most subsequent algorithms. Much research is currently focused around combining an inertial navigation system (INS) with information from a global position sattelite system (GNSS) [10], [11]. While this traditional approach yields very good results in areas where GNSS signals are strong, the estimate may jump in areas where the satellites are occluded or multi-pathing occurs [12]. Such jumps may produce dangerous behavior in subsequent algorithms that control the vehicle based on the state estimate.

In order to address this issue, two $\boxplus$-Kalman Filters are used in parallel. One filter solves the classic INS/GNSS problem, and produces a global estimate which may contain jumps where the GNSS signal is erroneous. This filter achieves lane-accurate localization and is used to make known information about the environment usable, such as lane information and street signs, which are stored in global coordinates. Since the output of this filter is not directly used for vehicle control, the occuring jumps do not lead to dangerous behaviour.

The second filter only estimates odometry, without including any global information and thus does not inherently suffer from jumps. The approach is realized by moving the reference state (towards which the uncertainty is estimated) forward in time periodically, as described in detail in [12]. This reference is always only a few meters behind the current state, which bounds the uncertainty in the vehicle pose that would otherwise grow unbounded without any correction data. However, due to the lack of any global correction data this filter will drift over time. Nevertheless, the state estimate is very precise over short time frames, making the result of this filter well-suited for vehicle control.

IV. Perception

An understanding of the surroundings of the vehicle are of utmost importance for safe and reliable driving. Traffic participants, obstructions and the areas in which to possibly drive need to be detected. The vehicles perception is based on LiDAR and front-area camera data.

A. Visual Perception

To get a semantic understanding of the cars surroundings two camera based deep learning approaches are used.

The first one facilitates the detection of a local driveable area based on perceived roads without borders or lane markings in images from the front area camera. To obtain an estimate of this area, each camera frame is passed through a deep convolutional neural network (CNN) that performs semantic segmentation. The segmentation model architecturally is a scaled down variant of the Deeplab network [13] with a resnet34 backbone [14], specifically tuned to achieve real time capability on the research vehicle on an input resolution of 1280x384. With only 23.08M parameters,

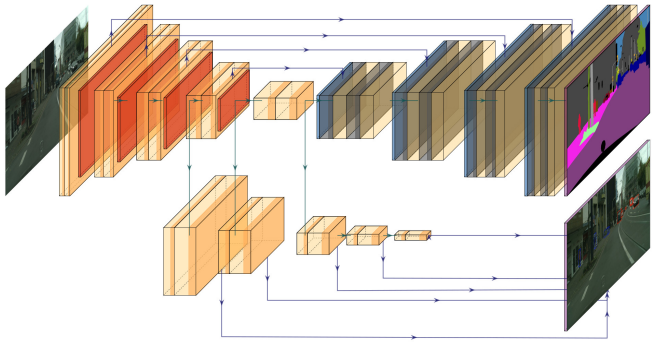

Fig. 3. Deep CNN architecture used for image processing. The upper part consists of the Resnet34 encoder network as well as the deeplab-style decoder used for (road-)segmentation. At different depths of the network, features are passed to the object detection decoder, i. e. the lower path, at which objects are predicted at five different output layer depths.

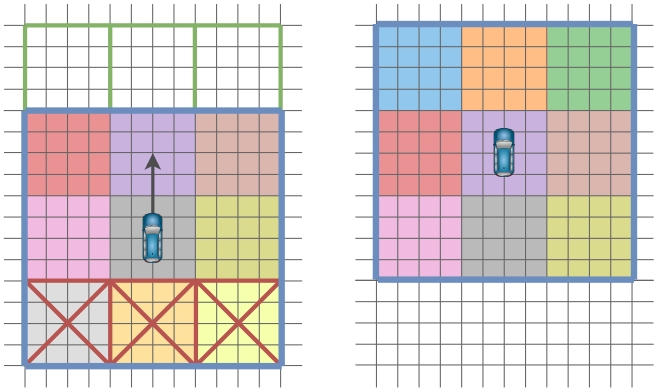

Fig. 4. Moving the current submap window. The currently active window is shown as a blue outline in two timesteps. As the vehicle moves, the submaps behind it are discarded (red crosses), and new ones are allocated in the direction of travel (green outlines), from [21].

this model achieves an mIOU[1] of 76.4% on the Cityscapes [15] validation set with inference speed in excess of 20fps [16]. On the car, the camera images get passed through the model, thereby get segmented into the 19 classes of the Cityscapes training dataset and subsequently transformed into a top-down view using a simple linear transformation. Assuming a reasonably flat road around the car, edges of the pixels segmented into the road class are passed to the DM as edges of the driveable area via a simple grid map. A more advanced process of transforming to a top-down view might be implemented in future work by taking local elevation data obtained from the LiDAR sensors into account.

A second decoder network performs single shot 2D object detection similar to FCOS [17] to get an understanding of other traffic participants. It is trained to detect instances of other cars, buses, trucks, persons as well as bicycles. The encoder parameters and features are shared with the segmentation model, giving an advance in inference speed by making only one expensive forward pass through the backbone network per frame necessary, giving us an overall single-encoder multi-decoder network structure which is visualized in Fig. 3. Objects are predicted at five different levels of scale and depth, with prediction feature map sizes being between 1/8th to 1/128th of the input image resolution.

The joint training procedure consists of 100 epochs on the finely annotated training images of the Cityscapes dataset with considerable application of data augmentation (e.g. scale jittering, random noise, random flips etc.) to improve generalization to the vehicle camera. Object bounding box ground truth is generated from the finely annotated instance segmentation labels of the dataset. To naturally train only on objects with a reasonable size in the image, any objects which are too small to have a corresponding feature map cell contained within them, i.e. by being smaller than 16px in any dimension, are discarded during training.

[1]mIOU (*mean intersection over union*): area of intersection of ground truth and prediction divided by the area of union of ground truth and prediction, averaged by detected class.

B. Mapping

In addition to camera-based perception, a mapping approach utilizing the six LiDAR sensors is used. These offer more robust distance information and are especially useful for detecting obstacles in the immediate surroundings of the vehicle. The common mapping representation is the grid map, which divides the environment into cells. In this work, evidential grid maps are used [18], which can be seen as a generalization of the classical probabilistic grid maps [19, Chap. 9]. Instead of representing only the probability for a cell to be free or occupied, belief mass can be assigned to any of the subsets of the set $\Theta_Y := \{o, f\}$. This includes mass on the empty set $\emptyset$, representing conflicting information, as well as mass on the superset Θ_Y, which represents no information in that area. Mapping is performed using *mapping with known pose* [19, Chap. 9], where the LiDAR scan is transformed into the map using the current state estimate, which is obtained as described in Sect. III. The uncertainty of this state estimate is ignored, as the pose is assumed to be known. The map is then updated according to the equations given in [20].

A unique challenge in mapping for autonomous driving lies in the large areas that can be covered in a single session. Often these areas can not be fully kept in memory due to computational hardware constraints. Instead, only the direct environment of the vehicle is put into focus by using a periodically updated moving window that is kept roughly centered around the vehicle. This is implemented by using a hierarchical map organization, which can be seen in Fig. 4. The map is split into submaps, which are allocated or discarded according to the movement of the vehicle. The equations as well as a performance evaluation for different underlying datastructures can be found in [21].

V. Object Tracking

While the map already contains information about the static parts of the environment, dynamic parts such as other traffic participants require seperate handling. This is done by late fusion using multiple ⊞-Kalman Filters, which is similar

to the one used for state estimate in Sect. III with slightly modified models, since no intrinsic information is available. In this approach each traffic participant is tracked by their own filter.

A. LiDAR-based Object Tracking

There are two different possible approaches for detecting traffic participants. The first is to obtain the objects directly from the sensors in the form of precomputed objects containing a size, pose and velocity vector. These object can then be used directly as input for the late fusion.

The second is still under development, where objects are detected in a specialized grid map that represents the dynamics of the environment. For this, evidential dynamic maps are used as presented in [22]. These maps are an extension to the evidential maps presented in Sect. IV, in that they extend the set of possible states to $\Theta_Y = \{f, s, d\}$, with d representing belief that the corresponding area contains a dynamic object, and s representing static objects. The state of each cell is determined using a particle filter, where each particle is assigned a velocity and occupancy probability, and only the particles which match the observations over time remain. This results in a belief for each cell to be free, dynamically occupied or statically occupied, as well as their estimated velocity. These cell attributes can then be used to find clusters of dynamic cells with similar velocities and directions of travel. Velocity, orientation, position and size of the resulting cluster can be easily calculated from its contained cells. Therefore, all required information for a filter update can be derived directly from the map.

B. Camera-aided Object Tracking

Using the detected objects from Sect. IV, we currently investigate improving the birth-detection of the tracking filter. In the LiDAR-based approach a simple clustering is performed in order to find areas with similar dynamic properties. However, this method relies on the evidential dynamic map having detected dynamic properties of the cells with a reasonable certainty, which may take several observations of the same object to reach that threshold. Using the detected objects from the camera, it seems reasonable to choose a significantly lower threshold for the detection in areas that are known to belong to an object. This may speed up the initialization process, which becomes more important with increasing vehicle velocities.

In addition to the improved birth-detection we are currently investigating using the detected bounding boxes directly for correcting the filter. Using the extrinsic and intrinsic parameters of the camera, the 3D bounding-box of the tracked traffic participant in world coordinates is transformed to a 2D bounding-box in image coordinates. This bounding-box may be used to correct the width and length of the tracked object by using the 3D-to-2D transformation as the measurement function of the Kalman Filter.

VI. Planning and Decision Making

Conceptually three sub-modules are responsible for high- and low-level planning and decision making. The strategic decision making computes long-term routes for one or multiple vehicles, mainly considering static information from annotated maps and traffic flow data. The tactical decision making, on the other hand, is responsible for computing specific short-term maneuvers that follow the specifications provided by the strategic decision making and at the same time consider the immediate surroundings of the vehicle. To compute such maneuvers in arbitrary and dynamic environments, the tactic module is closely coupled to a motion planner that constantly computes a series of feasible vehicle maneuvers based on specific requests from the tactic.

A. Strategic Decision Making

The main task of the strategic decision making is to plan a route from the current location of the vehicle to one or several destinations specified by the user. A graphical user interface for the operation of the onboard system by the passenger (input of destination) is therefore part of the module.

The route planning and decision making is based on mostly static data. This includes a reference map of the area as well as a map of the street network. The maps are annotated to include driving lanes and their driving direction, lane markings like stop lines, as well as traffic lights and traffic signs. It is planned to take into account information about the traffic or changes to the map from cooperating cars or other sources (e.g. about traffic jams or blocked roads).

Planning a route includes selecting the most advantageous lanes and corresponding desirable speeds, taking into account traffic and priority rules. Depending on predefined objectives or user input, the routing algorithm has to optimize for a given criteria like shortest distance or shortest travel time. In scenarios with multiple users or multiple cooperating vehicles this extends to multiple vehicle routing. This module implements various different existing routing algorithms to compare their performance when solving these problems.

The strategic decision making module does not issue any control commands directly. All decisions made are relayed to the tactical decision making module, which in return provides constant feedback while the planned route is executed. Based on the received information the planned route can change at any time, e. g. in case a road is blocked, when a lane change is impossible or something else causes the vehicle to diverge from the planned route.

Depending on the algorithm used, re-routing in real-time can be accomplished by calculating a number of different routes at all times instead of solving the optimization problem for a single route. After possible routes are planned, characteristics like total distance driven and expected driving time are calculated and used to rank the different options according to the specified optimization criteria. All possible routes are kept as backups and handed to the tactical decision making module to use in case a chosen route has to be abandoned suddenly. When an algorithm is used for which this multi-planning approach is not feasible, the strategic

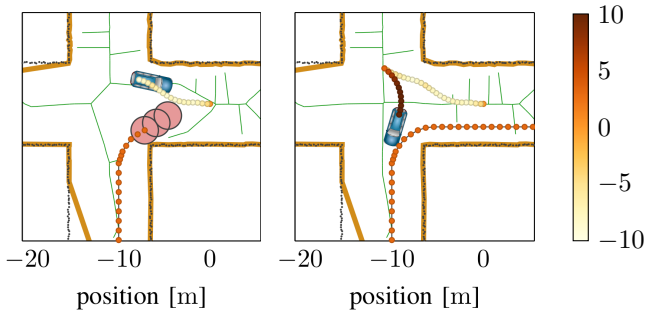

Fig. 5. Simulated maneuver generated by the motion planner involving an inversion of the driving direction and considering the predicted motion of a moving object (approximated by red circles). The colored points describe the past trajectories of both vehicles with the colors representing the speed values at the corresponding time. Data about the vehicle environment is given in black, the automatically generated free-space polygon is shown in orange and the (dynamic) Voronoi edges in green accordingly.

decision making module will have to re-plan as soon as a deviation from the planned route is reported by the tactical decision making module.

B. Motion Planning

In its core, the motion planning module is based on the well-known hybrid state A* algorithm, which was originally designed for the computation of feasible paths for an autonomous car [23]. That method itself, however, cannot compensate for dynamic environments (like other cars), needs a post-optimization step to ensure safety to obstacles and is highly specialized to consider only specific spatial restrictions. To overcome these drawbacks, this work utilizes a generic version of the hybrid state A* algorithm which is built on three main features [4].

First, all static spatial restrictions, like lane information or obstacles generated by the sensor fusion module, are combined into a unique representation through a free-space polygon. This not only allows to consider arbitrary types of constraints without extending the algorithm itself, but most importantly reduces the complexity of collision checks inside the planner, especially for large point clouds as generated by mapping of LiDAR data.

Second, a Voronoi field similar to [23] can be efficiently computed from the edges of the free-space polygon. For the approach presented in this work the space occupied by the anticipated trajectory of moving objects at discrete time points is also considered. This leads to the definition of a dynamic Voronoi potential which is then used to compute costs of explored nodes as well as to perform collision checks of the corresponding states directly within the hybrid state A* algorithm.

Finally the exploration strategy to generate new nodes takes information about the vehicle's speed v into account by using the kinematic vehicle model

$$\left(\dot{x}, \dot{y}, \dot{\varphi}, \dot{v}\right)^\top = \left(v\cos(\varphi), v\sin(\varphi), {}^{v}/_{L}\tan(\delta), a\right)^\top,$$

considering the wheel base L and the controls being the acceleration a and the steering angle δ. The performance of the resulting motion planner is demonstrated by an example solution for an intentionally complex maneuver (involving forward and backward driving as well as dynamic obstacle avoidance) in Fig. 5.

C. Tactical Decision Making

Utilizing the motion planner from Sect. VI-B, the tactic module computes a set of trajectories and decides upon the best solution by considering aspects like safety, strategic specifications or consistency between consecutive decisions. Eventually, the resulting trajectory is forwarded to the model predictive controller for execution, compare Sect. VII.

In order to properly setup the motion planning framework, the knowledge about the vehicles surroundings is thoughtfully preprocessed. As a key feature, spatial restrictions are categorized into soft (e. g. lane boundaries) and hard (i. e. actual obstacles) constraints, both having a separate accessible area. The dynamic Voronoi potential is computed regarding the intersection of both. This prevents collisions and at the same time allows the vehicle to cross lane boundaries if necessary while otherwise being centered within the lane which is enforced through the node costs introduced by the potential. Furthermore, by analyzing the distance of static and moving obstacles to the Voronoi edges, an adaptive definition of a current speed target is possible. This does not only allow to drive slowly in narrow areas but also to detect and consider the velocity of leading vehicles, effectively implementing the functionality of an adaptive cruise control (ACC).

VII. Model Predictive Control

In order to handle the complex situations that arise during autonomous driving, a most generic controller is needed. For this, a single, adaptive algorithm that is used to compute control commands for every occurring maneuver is implemented. This represents a significant difference to other approaches, where specialized controllers are used depending on the given situation. Predestined for such a generic approach is the formulation of every driving task as an optimal control problem. In (OCP) the standard problem formulation of an optimal control problem is shown.

$$\begin{aligned} \min_{z,u,T} \quad & J(z,u,T) \\ \text{s.t. } & \dot{z}(t) = f(z(t),u(t)), \\ & z_{\min} \le z(t) \le z_{\max}, \\ & u_{\min} \le u(t) \le u_{\max}, \\ & \Psi(z(0),z(T)) = 0, \\ & C(z(t),u(t),t) \le 0 \text{ for all } t \in [0,T], \end{aligned} \tag{OCP}$$

An objective function J which is completely customizable is minimized giving states z, controls u and the process time T. The states are subject to the potentially nonlinear dynamics f. Conditions for start and goal states are formulated in Ψ. Additionally, general path constraints can be described by means of C and have to be fulfilled for every time point in $t \in [0,T]$.

To describe the dynamic behavior of the car a kinematic single track model is used. For this the formulation in [4] is utilized, which includes time delays for the acceleration and the steering angle. This model is accurate enough since the driving maneuvers considered within OPA^{3}L are in a low speed range.

Utilizing the objective J all controls and selected states are combined and penalized respectively. A proper weighing is chosen to ensure safe and robust driving and forces the optimizer to find a optimal solution in a given time range.

The path constraints C are formulated such that the car maintains a safety distance to both static and dynamic obstacles. For that, the same free-space polygon as introduced in Sect. VI-B is reused to describe the currently drivable area.

Within a model predictive control (MPC) setting, (OCP) is solved periodically based on the respective current vehicle state. For this the software framework TransWORHP [24] is utilized which applies direct methods to transform the infinite dimensional problem into a finite one. Afterwards the software WORHP [25] is used to solve the resulting nonlinear optimization problem.

Within the setting of (OCP) j and ω_δ are used as controls. Depending on the platform, i. e. research vehicle, model car or simulator, the actual control signals differ and might need further processing to be computed (e.g. steering wheel angle for the research vehicle, power of electric motor for the model car). The resulting controls are ensured to be generated in a frequency of $50\,\mathrm{Hz}$.

VIII. Rapid Development & Remote Driving

This section introduces two platforms, a simulation and miniature vehicles, that are used for rapid development, evaluation and demonstration of the OPA^{3}L system for autonomous driving. Additionally, a remote driving concept is presented, which will in the future be utilized as a fallback to the autonomous system.

A. CARLA Simulator

In order to test and demonstrate complex and dynamic driving scenarios, the open-source software CARLA [26] is utilized, which is a simulator developed for autonomous driving research. The simulated vehicle is equipped with six LiDAR sensors, similar to the research vehicle described in Sect. II-A. Based on the artificial LiDAR data and vehicle state information provided by CARLA, the aforementioned algorithms perform mapping, planning and control to steer the simulated car through its environment. In addition, street networks and lane information from actual testing areas are considered to increase the realism of the scenarios. This is further enhanced by a sophisticated modeling of the area as illustrated in Fig. 6.

B. Miniature Vehicles

A miniaturized 1:8-scale car enables the evaluation of new algorithms or cooperative maneuvers on a real-world system. The vehicle is equipped with a simplified yet similar setup of sensors and computational units. A missing global reference system such as GNSS is substituted by visual markers corresponding to fixed positions in the environment. Fig. 7 shows how this is used within the software framework of OPA^{3}L to navigate the vehicle within its test field.

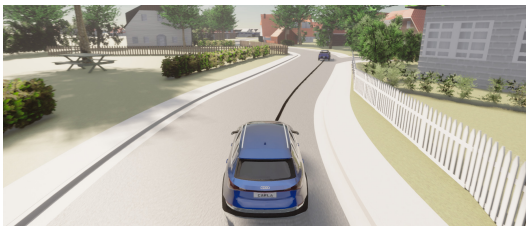

Fig. 6. A vehicle steered by the OPA^{3}L software in Borgfeld, a suburb of Bremen, within the CARLA simulator.

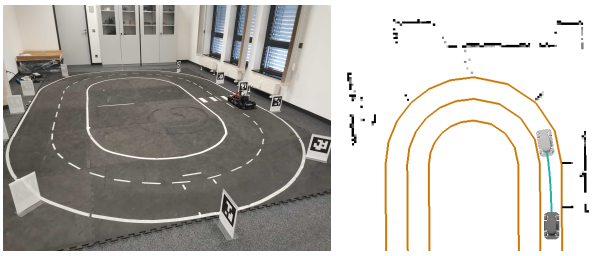

Fig. 7. Test field (left) and autonomous navigation (right) of the miniature vehicle. The output of the mapping module (black), the road network (orange) and the trajectory computed by the MPC (cyan) are displayed.

C. Remote Driving Examples

A remote driving functionality is developed as a backup solution for unexpected scenarios and for monitoring. For this, sensor and preprocessed data are transferred to a control center to be visualized on a standard display or in virtual reality, as shown in Fig. 8. With this an operator will be able to interact with the vehicle directly (in the form of control signals) or indirectly (in the form of waypoints for the car to follow by itself).

IX. Driving Examples

The previously introduced systems form an autonomous driving stack which is capable of autonomous navigation in suburban environments. We showcase this in Borgfeld, a suburb of Bremen, Germany, by driving a route of $1.6\,\mathrm{km}$, which takes around 5 minutes at maximum speeds of $30\,\mathrm{km/h}$.

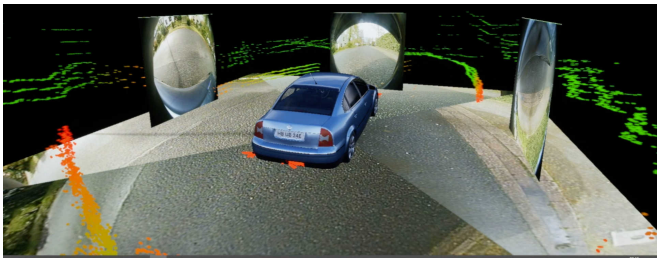

Fig. 8. Sensor data displayed in the remote control center. LiDAR scans are rendered as point clouds, cameras are mapped to planes.

Fig. 9. © GeoBasis-DE / Landesamt GeoInformation Bremen 2019. Driven route (yellow) overlaid with the lanes (red) provided by the strategic decision making. The route starts in the east and leads to a turning circle to the west. After passing through it, the route returns to the starting point.

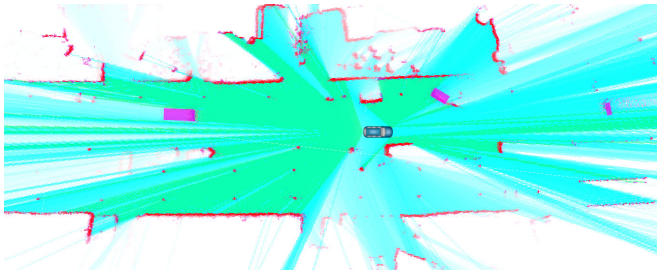

Fig. 10. Example of the evidential dynamic map. Green areas are observed as free, static cells are marked red, while dynamic cells are blue. Cyan represents areas with no information in the current scan, where previously information was available. There are three dynamic objects in the scene, which are detected and marked with a purple bounding box. Additionally there are a number of parked vehicles, which are correctly excluded from the detection.

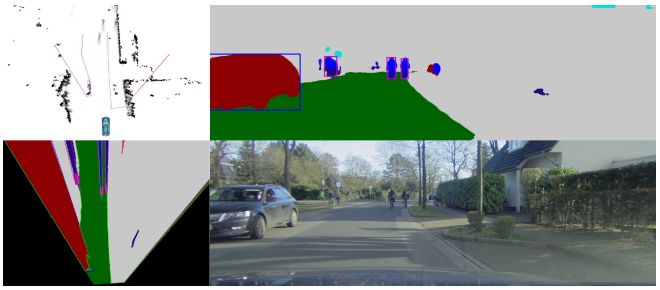

Fig. 11. Exemplary output of image processing module. The camera image (lower right) gets transformed in a semantic representation (top right) via a deep CNN and subsequently mapped into a top down view (bottom left) from which the road edges (top left, in purple) are inferred and passed on into the tactical decision module.

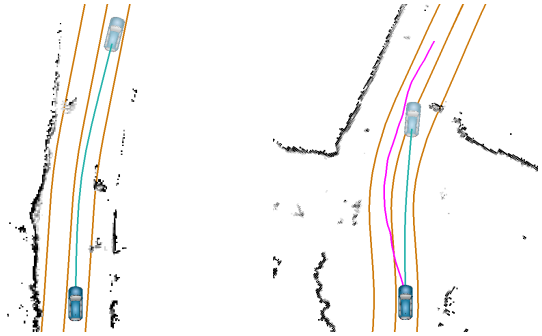

Fig. 12. Examples for automatic obstacle handling of the tactical decision making. The lane information and the result of the mapping module are displayed in orange and black respectively. The green line represents the optimized trajectory of the model predictive controller. For lanes that are only partially blocked by obstacles, the planned movement is automatically adapted to maintain a safety distance (left). If a lane is actually blocked on the other hand, the motion planner computes a dedicated lane change maneuver (right, in pink).

The major challenges of this scenario include tight roads with no lane markings, cars parking directly on the street and sharing the road with pedestrians and cyclists.

Fig. 9 shows the driven route. It was created using the result of the localization module, or more specifically of the INS/GNSS fusion. Since the estimate is corrected using RTK, lane-accurate positioning is achieved, which can be seen in the image.

Using the estimated odometry state as well as the LiDAR data, an evidential dynamic map of the surroundings is computed, which can be seen in Fig. 10. In addition to this grid map, the figure shows other traffic participants, which are detected and tracked based on the estimated dynamics in the map.

Data from the front area camera is fed into the image processing module and forwarded through the multi-task neural network to obtain a semantic segmentation of the image and detected objects. The segmentation is then used to generate a local description of the road, which is passed to the motion planner. All stages of the image processing can be seen in Fig. 11.

Based on the joint information about localization, mapping and lanes, short-term maneuvers that comply with the strategic plan are computed by the tactical decision making. One of the main requirements of this is to implement an adaptive behavior with respect to unforeseen obstacles along the route. As shown in Fig. 12, this is either ensured by the motion planner implicitly (for smaller obstacles) or explicitly by computing a lane change maneuver in case of a blocked lane.

In case of slower traffic participants driving in front, the analysis of the tactical planner leads to an adaption of the currently driven speed (i. e. an ACC functionality). Fig. 13 shows an exemplary situation where this is realized for a leading car providing its state information via V2X communication. The corresponding speed values of both vehicles over time are displayed in Fig. 14. It shows that the cars approach each other, both in position and speed, for the first 10 seconds. After a phase of driving behind each other for 14 seconds, the leading car turns, hence allowing the ego vehicle to accelerate again. Note, that the same principle can be applied to traffic participants detected by the tracking module.

X. Summary and Outlook

We present the holistic approach for an autonomously driving vehicle for suburban areas developed as part of the OPA^{3}L project. Therein, a multi-sensor fusion module generates a representation of the static and dynamic environment

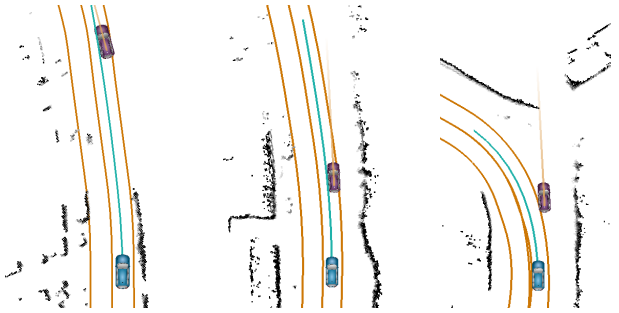

Fig. 13. Example for the application of an ACC functionality for a slow driving leading car (displayed in purple).

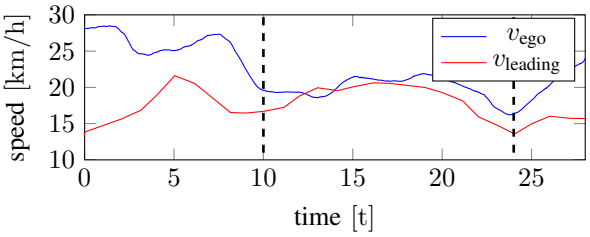

Fig. 14. Speeds of the ego vehicle (v_{ego}) and leading vehicle (v_{leading}) for the scenario displayed in Fig. 13.

of the vehicle, utilizing camera, LiDAR, V2X, RTK, IMU and odometric information. Based on this, algorithms for long- and short-term decision making build the foundation to guide the vehicle by strategic as well as tactical decisions. Finally, all planned maneuvers are executed by a nonlinear model predictive control approach, making the autonomous system very robust to even unknown situations. The developed methods are continuously evaluated in a simulated and a miniaturized environment, and will in the future be assisted by a remote driving concept. We show the validity of the overall approach and the performance of the current state of the software by a fully autonomous ride of $1.6\,\mathrm{km}$ in a real suburban scenario. Upcoming work will, for instance, lead to a stronger coupling of semantic information obtained from cameras and V2X-communication with the LiDAR-based environment representation. This enhanced basis of knowledge will further be used to improve the capabilities of the decision making modules in dynamic situations.

References

[1] WHO, *Global status report on road safety 2018*. Geneva: World Health Organization, 2018.

[2] W. Wijnen and H. Stipdonk, "Social costs of road crashes: An international analysis," *Accident Analysis and Prevention*, vol. 94, pp. 97–106, 2016.

[3] M. Shubbak, "Self-driving cars: Legal, social, and ethical aspects," *Artificial Intelligence - Law, Policy, & Ethics eJournal*, vol. March-2017, pp. 1–9, 2017. [Online]. Available: https://doi.org/10.2139/ssrn.2931847

[4] A. Folkers, M. Rick, and C. Büskens, "Time-dependent hybrid-state A* and optimal control for autonomous vehicles in arbitrary and dynamic environments," in *21st IFAC World Congress*, vol. 53, no. 2, 2020, pp. 15 077–15 083.

[5] B. Fleming, "New automotive electronics technologies," *IEEE Vehicular Technology Magazine*, vol. 7, no. 4, pp. 4–12, 2012. [Online]. Available: https://doi.org/10.1109/MVT.2012.2218144

[6] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *2018 ACM/IEEE 40th International Conference on Software Engineering*, 2018, pp. 303–314.

[7] A. Folkers, M. Rick, and C. Büskens, "Controlling an autonomous vehicle with deep reinforcement learning," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Jun. 2019.

[8] P. Falcone, F. Borrelli, H. Tseng, J. Asgari, and D. Hrovat, "Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 862–875, 2007.

[9] M. Rick, J. Clemens, L. Sommer, A. Folkers, K. Schill, and C. Büskens, "Autonomous driving based on nonlinear model predictive control and multi-sensor fusion," in *10th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2019.

[10] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2nd ed. Artech House, 2013.

[11] J. Breler, P. Reisdorf, M. Obst, and G. Wanielik, "GNSS positioning in non-line-of-sight context—A survey," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1147–1154.

[12] J. Clemens, C. Wellhausen, T. L. Koller, U. Frese, and K. Schill, "Kalman filter with moving reference for jump-free, multi-sensor odometry with application in autonomous driving," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, 2020.

[13] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 801–818.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[15] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

[16] L. Evers, "Benchmarking pretrained encoders for realtime semantic road scene segmentation," *PAMM*, vol. 20, 01 2021.

[17] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9627–9636.

[18] D. Pagac, E. M. Nebot, and H. Durrant-Whyte, "An evidential approach to map-building for autonomous vehicles," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 623–629, 1998.

[19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[20] J. Clemens, T. Reineking, and T. Kluth, "An evidential approach to SLAM, path planning, and active exploration," *International Journal of Approximate Reasoning*, vol. 73, pp. 1–26, 2016.

[21] C. Wellhausen, J. Clemens, and K. Schill, "Efficient grid map data structures for autonomous driving in large-scale environments," in *2021 24th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2021, pp. 1–8, in press.

[22] S. Steyer, G. Tanzmeister, and D. Wollherr, "Object tracking based on evidential dynamic occupancy grids in urban environments," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1064–1070.

[23] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, Jan. 2010.

[24] M. Knauer and C. Büskens, "From WORHP to TransWORHP," in *5th International Conference on Astrodynamics Tools and Techniques*, 2012.

[25] C. Büskens and D. Wassel, "The ESA NLP Solver Worhp," in *Modeling and Optimization in Space Engineering*. Springer, 2012, pp. 85–110.

[26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *1st Annual Conference on Robot Learning*, 2017, pp. 1–16.