2019 IEEE Intelligent Vehicles Symposium (IV)
Paris, France. June 9-12, 2019

CUbE: A Research Platform for Shared Mobility and Autonomous Driving in Urban Environments

Andreas Hartmannsgruber[1], Julien Seitz[2], Matthias Schreier[2] Matthias Strauss[2], Norbert Balbierer[1], and Andree Hohm[2]

***Abstract*— The number of people living in urban environments is continuing to grow, which leads to rising demand for intelligent transportation solutions. By flexibly supplementing public transport, driverless shuttles are widely regarded as a big building block for seamless mobility in the future, e.g. as first and last mile delivery systems. Therefore, we would like to introduce the Continental Urban Mobility Experience (CUbE), our research platform for driverless shuttles in urban environments. In contrast to the evolutionary automation of traditional cars, the CUbE helps to investigate new challenges and opportunities for fully autonomous driving in cities with a different kind of vehicle. Herein, we describe its system setup, early algorithmic approaches and share some of our real-world experiences, which we gained by extensive driving in an urban-like scenario.**

I. Introduction

Since 2009, more than half of the world's population is living in urban areas, and the trend to urbanization is further continuing [1]. This brings up new challenges, like the need for better mobility and transportation solutions. As observed in [2], self-driving vehicles have a huge potential on improving people's lives in cities, by mitigating traffic congestions, improving safety and lowering mobility costs. With all these benefits in mind, research on autonomous driving is on an all-time high, and universities, the automotive industry and IT companies are ever more automating our traditional cars. But without a driver, the shape and form of a vehicle are not bound to our traditional car anymore, and other concepts seem worth exploring. Small scale busses or guided transit systems, commonly dubbed "people movers", are often applied on airports, theme parks and similar private grounds, and break now free from their strict guidance, proofing to be an alternative approach for the research on fully autonomous driving vehicles. Recently, the US Department of Transportation has launched its own investigation into this topic [3], since in several cities early prototypes are already tested. At the moment, their public application is usually in a controlled environment with limited complexity, e.g., the whole driving area is previously known and slowly changing, communication to the infrastructure is possible, and most of the time an onboard attendant is watching out. Nevertheless, this way important real-world experience is gathered early on while introducing the public to driverless vehicles at the same time. [3] also points out current drawbacks, like the prototypical nature of many vehicles, or the momentary need for onboard attendants, which negate the economic benefits.

[1]The Authors are with Continental Automotive GmbH, Corporate Systems & Technology, 93055 Regensburg, Germany

[2]The Authors are with Continental Teves AG & Co. oHG, Chassis & Safety, Systems & Technology, 60488 Frankfurt, Germany

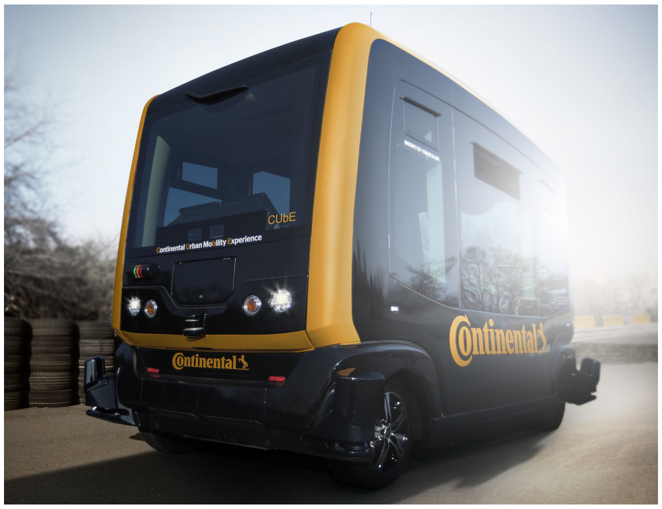

Fig. 1: Our research platform CUbE (abbrev. Continental Urban Mobility Experience)

For deeper investigations, a research platform was built up at Continental. The CUbE (abbrev. Continental Urban Mobility Experience, Fig. 1) is a real-world example for shared mobility and fully autonomous driving in urban environments. The project's main focus is to bring automotive technology into this kind of vehicles and increase its capabilities, moving around in urban environments and interacting with traffic participants. Especially series production sensors are still challenging in this respect, and most autonomous driving prototypes cope with high cost, non-automotive laser scanners. In the following, we would like to share our approach for the CUbE, as well as our real-world experiences.

II. System Architecture

The prototype vehicle CUbE in Fig. 1, is considered a test bed for autonomous driving in urban environments. Even though functional safety in systems engineering for this kind of vehicles is a very interesting topic and currently under research, gained insights along the way are not yet reflected in this prototype and are therefore left out of scope in this publication.

A. *Vehicle Platform*

As a basic vehicle platform, an Easymile EZ10 people-mover is used. It is driven by one asynchronous 6 kW electrical machine powered by two 9.6 kWh Lithium Iron Phosphate (LiFePo) batteries. This enables the vehicle to drive with a theoretical maximal velocity of 40 km/h. Since passive safety measures, like seatbelts and airbags, were widely neglected so far, the maximal operating velocity is limited to 20 km/h and in certain areas well below that. With a full battery, the vehicle can operate for around 8 hours, depending on the scenario. The Easymile EZ10 is in its base setup already able to autonomously follow a predefined route, while stopping for nearby static and dynamic obstacles. Via CAN and Ethernet, it is possible to read out the sensor data from its original sensor setup, which is shown in Fig. 2. The vehicle motion is controlled by a three-dimensional interface (steering angle, target velocity, acceleration) over CAN. A low-level controller adapts the torque on the electrical motor until the target velocity is reached. With the target acceleration, the ramp up and ramp down to the target velocity can be adjusted as needed. Considering the low-velocity range that the vehicle operates in, the target steering angle is adjusted relatively fast, so the low-level control loop has negligible influence. Also, the vehicle is steered on the front and rear axis separately, which enables a much smaller turning radius.

B. *Hardware Architecture*

A goal for CUbE is autonomous driving in urban environments with automotive grade components. Therefore, the base Easymile EZ10 platform was extended with additional sensors and related computation units. Trying out new components and variations in the hardware setup are essential to gather as much experience as possible, so there is a constant flux of components under test. Nevertheless, the basic setup with radar sensors and camera for perception, shown in Fig. 2, stayed the same and its components are explained in Tab. I. Over time, different sensor variants, i.e. short- versus long-range radar, or stereo versus mono camera, were evaluated and computation units became more embedded. Here, the focus is on the most recent hardware architecture. At the lowest level, a proprietary Rapid Prototyping Electronic Control Unit (RPECU) reads in data from the wheel tick sensors and information from the Easymile EZ10 Electronic Control Unit (ECU) from CAN and transmits a consolidated vehicle state over CAN to the Nvidia Drive PX 2. It also runs the motion control software outputting the three-dimensional control signal on the CAN. The Nvidia Drive PX 2 reads in the signals from environment sensors over Ethernet or LVDS and acts as the main computational unit within the functional chain. An additional DSM Nano Server is spent as Car PC for logging and the HMI. Most of the added sensors are automotive grade, series production sensors, except the differential global positioning system (DGPS) that serves as a reference positioning system.

From the original Easymile EZ10 setup, it is possible to input the Velodyne Puck 16-line laser scanners and use

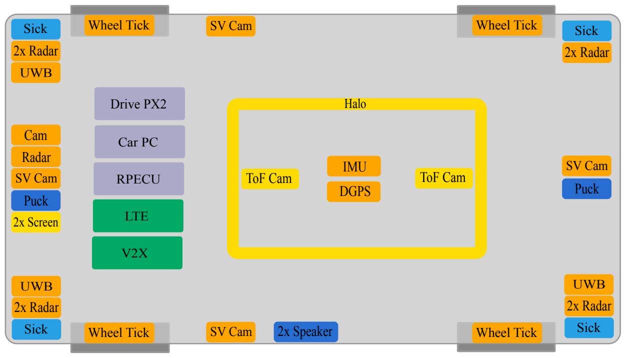

Fig. 2: A hardware overview of CUbE. Sensors (orange), computation units (grey), communication units (green) and HMI (yellow) were all added to the platform and a description, including network connections, is in Tab. I. The Velodyne Puck lidar (blue) is already built into the platform and can be read in for safety features and as reference. The built-in speakers (blue) are used for the HMI. The Sick 1-line laser scanners (light blue) are only used in the Easymile EZ10 safety layer.

them for an additional safety layer, or as reference for the camera and radar sensors. Via switch, an Easymile E10 safety function can be turned on or off, which brings the vehicle to a sudden full stop, using the Sick 1-line laser scanners in the corners. Besides autonomous driving, some effort was put into a holistic HMI concept. For that Time-Of-Flight (TOF) cameras observe the interior of the CUbE and people standing up are asked to sit-down by a voice over the speaker system. An LED band, integrated into a handrail attached to the roof, indicates the current state of the vehicle, e.g., opening/closing doors, emergency, or normal operation to the passengers. On the outside, two screens at the front windshield can also communicate the vehicle state to traffic participants nearby. It is possible to drive the CUbE manually, too. An Easymile provided controller enables an onboard safety driver to drive in walking speed. This was extended by an Xbox controller that is able to control the vehicle at higher velocities (up to 20 km/h) and adds other useful features for the safety driver, like a smooth coasting to stop or control over the doors. In this setup for the CUbE, we assume a preferential driving direction, i.e. a designated front and back of the vehicle. This is not evident in the shape of the chassis and also opposite to many similar vehicles, but it has two advantages. First of all, a full sensor setup has not to be mirrored to both ends of the vehicle, since likely more sensors for a larger field of view and for redundancies are needed in driving direction. Second, a preferential driving direction helps other traffic participants anticipating the CUbE's actions and therefore with the integration of the CUbE into an environment with human traffic participants.

TABLE I: Added components of the CUbE's hardware setup.

Label	Type	Connections	Description
Radar	Continental ARS430	Ethernet	Independent scan for long and short range (+/-9°, 250m; +/-45°, 70m), 76 - 77 GHz operation frequency.
Cam	Sekonix Camera SF3324-101	Ehternet	2.3M Pixel resolution, 120° field of view
SV Cam	Continental SVC210	LVDS (to Drive PX2)	Wide dynamic range CMOS 1M Pixel resolution, 195° field of view
ToF Cam	PMD CamBoard pico flexx	USB (to CAR PC)	Time-Of-Flight camera (depth sensor), 0.1 - 4 m measurement range, 45 fps
DGPS	OXTS RT4000	CAN	Differential global positioning system with inertial measurement unit, 250 Hz update frequency
IMU	Continental SC13S	CAN	6 Degree of freedom inertial measurement unit
UWB	Decawave Trek 1000 Evaluation Kit	USB (to CAR PC)	Ultra-Wideband anchor
Wheel Tick	IFM RB3100	CAN	Incremental rotary encoder with 0.8mm per impuls (tick)
V2X	Car Communication Unit (CCU)	Ethernet	Prototype of a gateway for outside communication with the infrastructure and other vehicles.
LTE Router	WELOTEC TK525L	Ethernet	LTE Router for backend connection
RPECU	Rapid Prototyping Electrical Control Unit	CAN	Proprietary embedded control unit for prototyping in development vehicles
CAR PC	DSM Nano Server	Ethernet, CAN	Embedded PC with Intel Core i5-520M 2.4GHz
Drive PX2	Nvidia Drive PX2 (AutoChauffeur)	Ethernet, LVDS, CAN	2x Tegra X2 (Parker) CPU and 2x Pascal GPU

C. *Functional Architecture*

The software architecture is similar to our work in [4], or the work of [5], with the difference being that in our case, the HMI is intended for passengers only and no fallback driver is available. So the system qualifies for level 4 autonomous driving according to [6]. Only for development purposes, a safety driver is considered, but is overall excluded from the concept. A big part of the HMI for CUbE is an Android smartphone application that sends a destination to the backend, which in turn transforms this information to a concrete mission for a specific CUbE in the fleet. Other than that, the HMI is mostly concerned with informing passengers, or even other traffic participants with the current vehicle state. Besides the HMI and modules for communication with the backend or infrastructure, the CUbE follows a Sense-Plan-Act approach, well known from robotics [7]. In the sense stage, sensor signals are gathered and fused to a comprehensive model of the system state and its environment, which is the input to the planning modules. The planning modules are also arranged in hierarchical layers, like in [8], but part of the planning that is concerned with the overall mission, is done in the backend, whereas the maneuver and trajectory planning happens on each CUbE in an iterative fashion. A feasible, comfortable and safe trajectory is then passed on to motion control, whose goal is to follow said trajectory closely. Most algorithms of the functional chain are running on the Nvidia Drive PX 2. Motion control is divided into two parts, where the first one is calculating the control error on the Nvidia Drive PX 2 and the second one is minimizing said control error by setting the actuators over the RPECU. Fig. 3 shows the logical software architecture in a block diagram and following Section III describes the used algorithms in more detail. The diagram encapsulates functions into logical units for more clarity, but often these logical units are composed of more than one submodule. Functional decomposition, as stated in [9], is a necessary tool to handle the emerging complexity of autonomous driving systems. [10] also defined modularity and testability as the most important requirements for ADAS software architectures and helped with the overall software architecture design process.

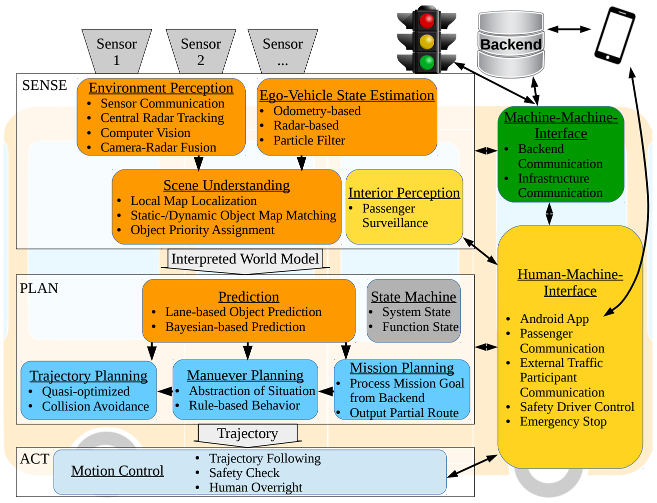

Fig. 3: The high level software architecture follows the well known Sense-Plan-Act approach. Modules for situational awareness are orange and modules for motion planning are blue.

D. *Software Framework & Development*

While the previous subsection reviewed the software architecture of CUbE, very little was mentioned about the framework and the development environment. Nevertheless, these are important for rapid and stable progress. Whereas former Car PCs were running a standard Windows 7 operating system, they were switched to Ubuntu 16.04 LTS, as it is on the Nvidia Drive PX 2. The proprietary middleware, called "enhanced communication abstraction layer" (eCAL) runs on both operating systems, which helped with porting of the algorithms. It enables fast communication of distributed, asynchronous tasks by UDP multicast. Tools for recording and playback of measurements or observing network communication are also included. Data serialization and de-serialization in form of messages is done using Google Protobuf. On top is an algorithm framework that decouples the algorithms from the communication layer, offers visualization capabilities using OpenGL and Open Scene Graph (OSG) and provides help with the setup of the build environment. The build process itself is based on CMake and a continuous integration (CI) process that ensures a stable code base in a globally distributed development.

III. Functional Description

In this section, we explain the most important algorithms running on the Drive PX2 in the CUbE or in the backend. This is done according to sense-plan-act, i.e. the flow of information, from sensor input to control of the actuators. The algorithms are all developed in-house. In the future, more open-source and proprietary software, like Nvidia Driveworks will be included for bench-marking-purposes.

A. Environment Perception

Since the CUbE is equipped with a multitude of environment sensors, see Tab. 1 and Fig. 2 for an overview of the current setup, it constitutes an ideal platform for testing and comparing various environment perception, sensor fusion and environment modeling approaches [11]. The main goal of the CUbE platform is to gain experience how low-cost, series-production sensors such as radars, standard mono cameras and fisheye surround view cameras enable the creation of a robust and reliable environment model. The modular system, hardware and software architecture allows to easily switch between different sensors and data fusion algorithms available at Continental.

The CUbE's environment model consists of three main parts: i) A so-called traffic participant list, which contains (potentially) dynamic objects such as other vehicles, pedestrians and bicyclists in an object-based manner with their estimated state vectors, ii) a grid-based and parametric free space map-based representation of arbitrarily structured static environments and free spaces [12][13][14] and iii) a so-called road model, which represents the drivable lane geometry, topology and lane-specific traffic rules such as traffic light states obtained from the camera or V2X systems. Note that the latter road model representation is not always available since the CUbE's operating area is not restricted to road-like scenarios. If, however, it operates on dedicated lanes, an additional high-level fusion of sensor-based road models (obtained via a camera-based lane and park marker detection) and map-based road models (obtained via localization on a digital HD map) as shown in [15] for additional robustness.

The generated i) traffic participant list, ii) grid and parametric free space map and iii) road model provide a consistent 360° view of the driving environment in a robust way and form the Comprehensive Environment Model (CEM) that is used as the basis for prediction, criticality assessment and planning as explained further in the following.

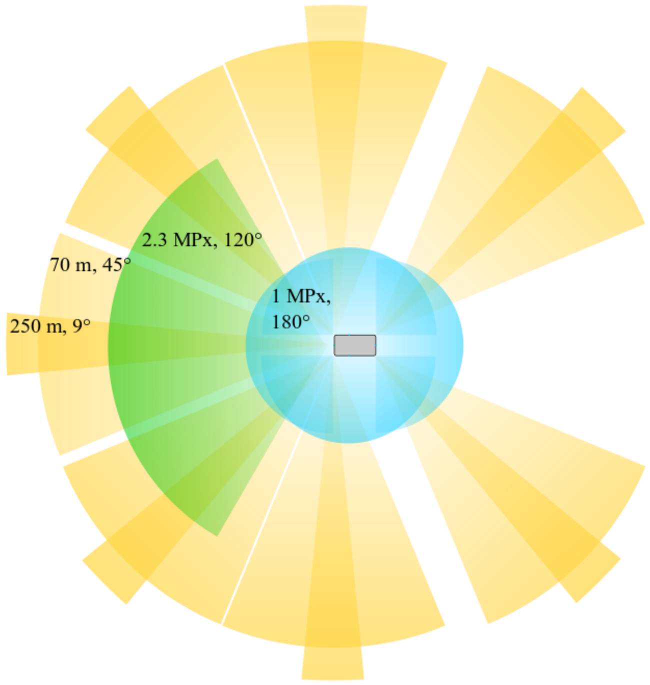

Fig. 4: The CUbE's perception relies on radars (yellow), a mono camera (green), and surround view cameras (blue).

B. Prediction

The focus of the prediction and criticality assessment lies in a reasonable anticipation of the future behavior of other traffic participants over longer time-spans. As highlighted in [16][17], a prediction purely based on physical motion models as done in target trackers is not suited for this purpose since these only give reasonable predictions for short time intervals of less than a second. In the current state, on-road users are therefore predicted along the lanes similarly to [8], whereas lane change predictions are done according to a simple driver model. In the future, an integration of a full Bayesian prediction and criticality assessment approach as available from [16][18][17] for vehicles is planned, which is based on a Bayesian network-based maneuver estimation, an uncertain trajectory prediction modeled by stochastic processes and a criticality assessment via the TTCCP (Time-To-Critical-Collision-Probability) metric. For the uncertain prediction of pedestrians that might cross the street, an approach has already been presented in more detail in [19].

C. Localization & Mapping

Driving functions require an accurate and reliable absolute position estimation as a basis for autonomous driving. Especially in the field of operation of driverless shuttles in urban environments, several specific characteristics have to be considered when designing such a localization solution. The vehicle has to operate in unstructured environments without having typical lane markings and without regulatory traffic elements such as traffic signs (e.g., when operating on non-public grounds). There will probably be no HD map available in advance for such an environment and additionally, in some cases, the vehicle needs to be able to localize even when driving off the road (e.g., on small pedestrian walks or large places). Furthermore, the vehicle has to deal with limited precision and reliability of GPS, caused by buildings, and should also be able to operate independently of weather conditions. A beneficial circumstance is that the operating area is well known and that the fleet operator has the ability to modify the infrastructure to support the localization, e.g., can place visible landmarks or C2X modules.

A modular particle filter-based [20] localization framework has been developed that is able to fuse multiple localization input sources based on a plugin concept. The overall procedure works as follows: The framework provides an initial set of randomly distributed pose-hypotheses (particles) around the GPS position or the last known vehicle position. Afterward, the plugins evaluate how good the current observation fits an environment map, given the position assumption of each particle. The particle likelihoods of the individual plugins are combined, and the most likely hypothesis can be identified. For the next cycle, the particle poses are predicted based on vehicle dynamics input from an Ackermann model based on wheel-ticks and steering-angle fused with an IMU sensor. These postponed particles are the base for the plugin-based particle likelihood estimation in the next cycle. Here, the focus is set on two specific plugins. On the one hand, a radar plugin has been created that requires a radar occupancy grid [21] of the whole operating area. It requires the current sensor observations in the form of radar clusters and performs a scan-to-map matching algorithm with the overall radar grid. This approach works independent of weather and is designed for unstructured environments even if there is no standardized infrastructure. Changes in the environment (like parked cars) are a major challenge. However, even if parked cars change their position, parts inside of the field of view will stay the same. Additionally, the radar is capable to receive reflections from objects behind parked cars that can also be used for localization purposes. Additionally, an Ultra-Wideband (UWB) plugin was created. This plugin makes use of the fact that the fleet operator can place additional infrastructure. This plugin requires so-called UWB-Anchors to be installed in the vehicle and UWB-Tags to be placed at the infrastructure (like on traffic signs, traffic lights or street lamps). A simple time of flight measurement is performed to calculate the distance between pairs of tags and anchors. Because the tag position is well known, the plugin is able to calculate the likelihood of a given position based on that measurement. For evaluation purposes as well as for redundancy, an Oxford Technologies RT4000 RTK/DPGS system is installed in the vehicle. Map creation is performed using a basic Graph SLAM algorithm. This also requires RTK to fit the generated map into a global coordinate frame.

Similar to [22], we want to use our maps not only for localization, but also to aid our motion planning. On top of possible actions, we calculate a reference velocity profile as part of the mapping process. This can be done for specific routes, or partial routes by minimizing the cost function over time t

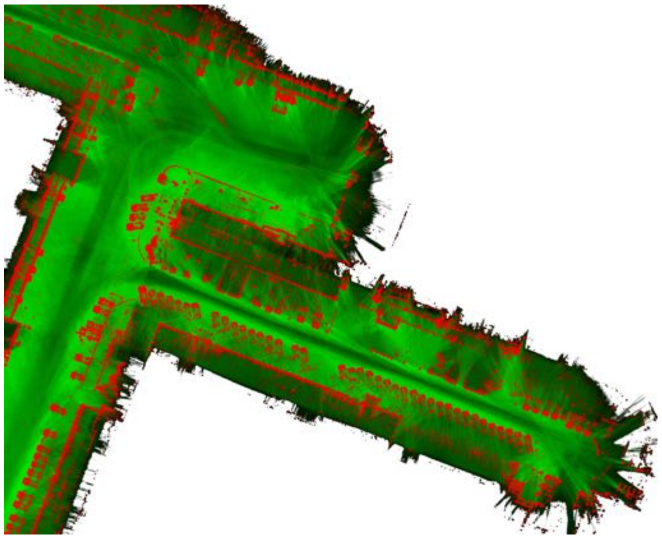
Fig. 5: A radar map of the Continental Frankfurt site. The map snippet fits the scenario shown in Fig.7.

$$J = \int_{t_0}^{t_0+T} \omega_{\mathrm{v}} \cdot \triangle v(s(t))^2 + \omega_{\mathrm{a}} \cdot a(t)^2 + \omega_{\mathrm{j}} \cdot j(t)^2 \mathrm{d}t \quad (1)$$

with $\triangle v(s(t)) = v(t) - v(s(t))_{\mathrm{des}}$ being the difference between current velocity $v(t)$ and desired velocity $v(s(t))_{\mathrm{des}}$ at the arc length $s(t)$ along the route, acceleration $a(t)$, jerk $j(t)$ and the respective cost factors $\omega_v, \omega_a, \omega_j$. The time horizon T is set appropriately to cover partial, or even the whole route. Calculation time is not critical here since the minimization is done offline. Also, the constraints

$$v(t) \leq v(s(t))_{\mathrm{des}}, a(t) \leq a_{\mathrm{max}}, j(t) \leq j_{\mathrm{max}} \quad (2)$$

must hold. The values for maximal acceleration a_{max} and maximal jerk j_{max} are chosen well beyond the dynamic restrictions of the vehicle to accomplish a comfortable motion for the passengers. The target velocity v_{des} is calculated by

$$v(s(t))_{\mathrm{des}} = \min(v(s(t))_{\mathrm{law}}, v(s(t))_{\mathrm{curv}}, v(s(t))_{\mathrm{haz}}) \quad (3)$$

with allowed velocity by traffic laws $v(s(t))_{\mathrm{law}}$, maximal velocity based on curvature $v(s(t))_{\mathrm{curv}}$, and a possible safety velocity in hazardous regions $v(s(t))_{\mathrm{haz}}$. The maximal velocity based on curvature depends on the maximal allowed lateral acceleration $a_{\mathrm{lat}_{\mathrm{max}}}$, which is also chosen for passengers comfort and way below the lateral acceleration stemming from the friction limit (i.e. from Kamm's Circle). It is calculated by

$$v(s(t))_{\mathrm{curv}} = \sqrt{\left|\frac{a_{\mathrm{lat}_{\mathrm{max}}}}{\kappa(s(t))}\right|} \quad (4)$$

with the curvature κ. In regions where the vehicle is very likely to stop, for example, due to pedestrians at a crosswalk, but not limited to that, it is advantageous to define hazardous zones, to slow down the CUbE, which in turn enables more sudden, but still comfortable stopping maneuvers. These hazardous regions are added manually to the map for now.

D. Fleet Management & Mission Planning

Usually, the CUbE is part of a fleet with a varying number of autonomous vehicles, providing the user shared mobility on demand. To do that, an Android application was developed, which allows ordering a CUbE remotely. It reports the current position and the target destination to a backend server, which is in charge of the fleet management. This means, it is choosing the appropriate CUbE and planning a route including the pick-up location and target destination in accordance with the routes of other guests sharing the ride. Therefore, each order results in a mission for the CUbE and at each time it may fulfill several missions at once. This Transportation on Demand (TOD) is also known as Dial-a-Ride Problem (DARP) and is a variant of the general Vehicle Routing Problem with Pickup and Delivery (VRTPPD) [23], [24]. The problem is dynamic by nature, since at any time a new request could come in and change the previous outcome of the optimization problem, but as often in practice, it can be treated as sequential static VRTPPD [23]. We follow this approach and treat the problem as static for now. The core algorithm of this fleet management uses Dijkstra to find the best route according to travel time and a form of time slicing to fit the user request into a mission buffer. The average time for a specific request is calculated and compared with other requests. If the algorithm finds a free slot, or overlapping mission in the mission queue, the request is accepted, and the mission dispatched to the appropriate CUbE from the fleet. If no CUbE is available for the request, alternative proposals will be sent to the user, who may accept a proposal or abort the ordering process. A basic schematic of the fleet management is shown in Fig. 6. Since the problem was rather small and most of the time only one CUbE was operated, the basic function of the algorithm was verified in simulation with the microscopic traffic simulator SUMO [25]. For the current use case the fleet management algorithm is sufficient and can easily be extended to [26]. Due to the modular nature of the backend software, in the future other fleet management algorithms, like [27] can also be tested. Within the CUbE, a fleet management agent is communicating with the fleet management in the backend server. It reports the current position and operational state back to the fleet management and provides the current mission to motion planning and the internal state machine. If a current mission is in the mission handling queue, it starts simply by closing the door. Otherwise, the CUbE is in a waiting mode. It can also be useful, to always have a return-home-mission at the end of a queue, so that the CUbE doesn't wait anywhere, but at a specific location.

E. Motion Planning & Control

When a mission starts, the internal state machine switches to driving mode, where the maneuver and trajectory planner

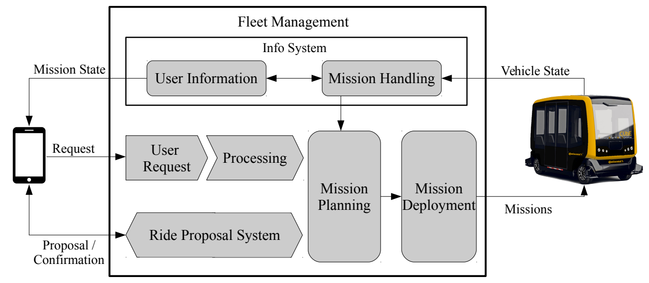

Fig. 6: The basic concept of the fleet management, running in the backend, and its communication with users and a CUbE.

take over the actual motion planning. The maneuver planner abstracts the dynamic objects, road model, traffic light and ego-vehicle state and puts it in context. Considering the planned mission and traffic rules, it determines the driving strategy, i.e. the target lane and possible stop/wait positions. The target lane and possible stop positions are obtained from the planned route and the map. For now, the target lane depends only on the planned route, but additional processing is in order, to determine whether a stop has to be initiated or not. Possible stop positions obtained from the map are:

- **Bus stop**: Stop only, if the bus stop is operated, i.e. target of a current mission.
- **Stop line**: Stop depending on traffic light state, or if yielding to other traffic participants is necessary. Each intersection has a stop line. In case no stop line is marked, where a lane is crossed, the stop line is virtual. This holds even in case of a turning maneuver that is crossing an oncoming lane.
- **Crosswalk**: Stop, if the predicted crossing probability of a pedestrian that is close to the crosswalk is above a certain threshold.

The path and velocity profile from the map, together with the simple, rule-based driving strategy, are combined to a reference trajectory, which the trajectory planner module aims to follow. Its responsibility is to efficiently plan a feasible, comfortable and most of all safe trajectory [28] that reacts to the dynamic environment and follows the reference trajectory closely. The main trajectory planning algorithm is similar to [29], [30] and samples in a street relative state space, although, because of the slow velocity range, the lateral goal states are not sampled independent from the longitudinal motion, but instead are sampled over the arc length. This is favorable to reduce the number of infeasible trajectories in the trajectory set, due to the non-holonomic vehicle kinematic of the vehicle. Trajectories are generated by quartic and quintic polynomials, depending on the sampling mode, which is either velocity or position sampling. The remaining goal state values are obtained from the target manifold of the reference trajectory. A switch from velocity to position sampling is only done in proximity to a dedicated stop location, for a precise stopping motion. In the lateral direction, only deviations to the reference trajectory are sampled. Originally, the cost functional consists of costs for the overall jerk, time to reach the target state and its distance to the reference trajectory. Here, an additional cost term combines longitudinal and lateral motion in velocity sampling mode, without lane change:

$$J_{vd} = \int d(s) \mathrm{d}s \cdot v_{\text{sample}} \tag{5}$$

J_{vd} is a cost for velocity, with d as the lateral deviation to the reference path and v_{sample} the longitudinal velocity sample. Since lateral deviations are only accepted in proximity to obstacles, this cost term leads to a slightly slower velocity when evading an obstacle, which feels more natural for passengers in the CUbE. The resulting trajectory set is sorted according to its costs and checked for kinematic and dynamic feasibility, as well as impending collisions. Each obstacle is checked for collision in a hierarchical order, as described in [31]. First, an inexpensive axis-aligned bounding box is checked for collision and only if one is detected here, the more precise oriented bounding box of the obstacle is checked for collision. After this, the first feasible and collision-free trajectory in the ordered trajectory set is the quasi-optimal solution. Start state of the trajectory planning process is under normal circumstances the point in time, where the CUbE should be according to the planned trajectory from the previous cycle.

The arising control errors are handled by a trajectory tracking controller that consists of a control deviation calculation module running on the Drive PX 2 and the longitudinal and lateral PID controller on the RPECU. The controller has also built-in safety features, like a control value saturation and the possibility to overwrite the planned trajectory by inputs from a safety driver via the Xbox controller.

IV. Evaluation

Extensive tests were carried out on the Continental site in Frankfurt, with regular traffic and many pedestrians, as well as on test tracks. Also, in September 2017, the project was presented to a large audience at the International Motor Show (IAA 2017) in Frankfurt and in September 2018 on the ADAC test facility in Hannover for many invited guests from all over the automotive industry. During these and other special events the vehicle performed over days and weeks on end, which also adds valuable experience. This section focuses on our experiences, the accomplishments, perceived short-comings and the insights gathered along the way.

The scenario for the Frankfurt production site, where also the IAA presentation was held, is shown in fig. 4. It can be considered as a typical urban environment with lite traffic and many pedestrians walking around, obeying the traffic rules for the most part. But depending on time and day, pedestrian traffic is very high and not limited to walkways and crosswalks anymore, e.g., around noon.

Using our Android application ”Call A CUbE”, a ride can be ordered within the facility. Usually, the starting position is set by GPS to the closest address or can be chosen on a map, as can the destination. But, since the whole facility has the same address, certain bus stops were implemented that can be chosen as destination and also serve as pickup locations. Basically, there are two distinctive bus stops at location B and C in fig. 4. A third bus stop A was added to better showcase the pickup capability and was used as waiting station, until a CUbE is called to the pickup location B.

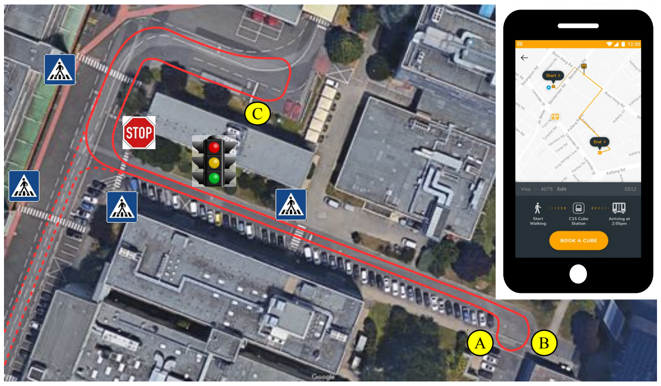

Fig. 7: The scenario on the Continental site in Frankfurt. The CUbE connects the front entrance with the canteen entrance and the park house (red,solid line) and the test course (red, dashed line). A ride with CUbE can be ordered with an Android application.

Overall, the CUbE showed good performance during the IAA 2017, regular tests carried out on site and other special events. During the showcases, the safety driver wasn’t called into action at all. Most of the time, this was also the case for our regular test drives, although on rare occasions a safety driver was still necessary. Situations, where a safety driver was necessary were mainly with oncoming traffic in the small road section from and to stations A and B. Depending on the dimensions of the oncoming vehicle and the parking situation on the side, the road became too narrow for the motion planning, to find a collision-free trajectory that passed the oncoming vehicle. In an event like this, if no collision-free trajectory is found, the CUbE stops on its track, although a solution might just be to swerve to the side and stop near the road boundary to make room for the oncoming vehicle. Other viable solutions would be to wait on either end of the narrow passage, before entering it, if an oncoming vehicle was detected early enough, or to evade onto the free area in the middle of the narrow passage that doesn’t belong to the road. But these solutions are not yet possible in the CUbE, because no trajectory is planned for long distances like this and also no high-level maneuver planning covers such situations, at the moment. Besides the planning algorithms, it is still challenging to reliably estimate the dimensions of oncoming traffic from afar. Even though the radar sensors would theoretically cover this range, in addition to the distance, the narrow passage also climbs a small hill. Furthermore, a general perception of the drivable area that is not a road is not trivial, too, although this could be added to the map in our case.

In the end, the CUbE was in a safe state at all times, but performed over-cautiously in some situations, or in worst case, was driving into dead-locks. On the other hand, the

CUbE was able to drive amid a plenitude of pedestrians, even following them in a safe distance and passing them, if enough space available. Other traffic, like bicycles, cars and trucks, was also handled according to traffic rules.

V. Conclusion & Future Work

All in all, the CUbE not only brought attention to driverless shuttles, but already got lots of great feedback from passengers from the automotive community and beyond. This is a precious side effect to the development work since it generates important trust amongst the public for autonomous driving. In 2018, Continental locations in Auburn Hills, USA and Yokohama, Japan also joined the CUbE project and more will come, developing and testing technology for urban autonomous driving and aiding with their location-specific experience. Since the CUbE is intended to drive in previously known environments, for now, maps that provide already a lot of location-specific information can be leveraged, thus slightly lowering the complexity of autonomous driving. On the other hand, this implies that careful thought must be put into the problem of scaling the CUbE's deployment.

Future research topics for the CUbE are plentiful. For one, new driverless shuttles will join the family, bringing emphasis to systems engineering and with it functional and passive safety. Also, as pointed out in Section IV, the CUbE manages safe operation, but improvements in motion planning and perception might help it to interact better with other traffic in a meaningful way.

Finally, by ever increasing the complexity of driving scenarios under test and overcoming new challenges, driverless shuttles, like the CUbE, will be another important piece in a seamless mobility experience for everyone in the future.

References

[1] *Urban and Rural Areas*. United Nations, Department of Economics and Social Affairs, Population Division, 2009.

[2] N. Lang, M. Ruessmann, A. Mei-Pochtler, T. Dauner, S. Komiya, X. Mosquet, and X. Doubara, *Self-Driving Vehicles, Robo-Taxis, and the Urban Mobility Revolution*. Boston Consulting Group, 2016.

[3] *Low-Speed Automated Shuttles: State of the Practice*. John A. Volpe National Transportation Systems Center, US Department of Transportation, 2018.

[4] A. Hohm, F. Lotz, O. Fochler, S. Lueke, and H. Winner, "Automated driving in real traffic: from current technical approaches towards architectural perspectives," in *SAE World Congress & Exhibition*, 2014.

[5] E. Bauer, F. Lotz, M. Pfromm, M. Schreier, B. Abendroth, S. Cieler, A. Eckert, A. Hohm, S. Lüke, P. Rieth, V. Willert, and J. Adamy, "PRORETA 3: An integrated approach to collision avoidance and vehicle automation," *at - Automatisierungstechnik*, vol. 60, no. 12, pp. 755–765, 2012.

[6] *SAE J3016 Levels of Driving Automation*. SAE International, 2014.

[7] D. Kortenkamp, R. Simmons, and D. Brugalli, *Handbook of Robotics*, ch. 12. Robotic Systems Architecutres and Programming, pp. 283–305. Springer, 2016.

[8] C. Urmson, J. Anhalt, D. Bagnell, and C. Baker, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics - Special Issue on the 2007 DARPA Urban Challenge, Part I*, vol. 25, no. 8, pp. 425–466, 2008.

[9] C. Amersbach and H. Winner, "Functional decomposition: An approach to reduce the approval effort for highly automated driving," in *8. Tagung Fahrerassistenz*, 2017.

[10] L. F., *Automotive Systems Engineering*, ch. System Architectures for Automated Vehicle Guidance Concepts. Springer, 2013.

[11] M. Schreier, "Environment representations for automated on-road vehicles," *at - Automatisierungstechnik*, vol. 66, no. 2, pp. 107–118, 2018.

[12] M. Schreier, V. Willer, and J. Adamy, "From grid maps to parametric free space maps - a highly compact, generic environment representation for adas," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2013.

[13] M. Schreier and V. Willert, "Robust free space detection in occupancy grid maps by methods of image analysis and dynamic b-spline contour tracking," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2012.

[14] M. Schreier, V. Willert, and J. Adamy, "Compact representation of dynamic driving environments for adas by parametric free space and dynamic object maps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 367 – 384, 2016.

[15] M. Schreier and R. Grewe, "A high-level road model information fusion framework and its application to multi-lane speed limit inference," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[16] M. Schreier, *Bayesian Environment Representation, Prediction, and Criticality Assessment for Driver Assistance Systems*. phdthesis, Technische Universität Darmstadt, 2015.

[17] M. Schreier, V. Willert, and J. Adamy, "An integrated approach to maneuver-based trajectory prediction and criticality assessment in arbitrary road environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, pp. 2751–2766, 2016.

[18] M. Schreier, V. Willert, and A. J. Systems, "Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems," *Intelligent Transportation Systems* , 2014.

[19] S. Kerscher, N. Balbierer, S. Kraust, A. Hartmannsgruber, N. Müller, and B. Ludwig, "Intention-based prediction for pedestrians and vehicles in unstructured environments," in *4th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS)*, 2018.

[20] O. Capp, S. Godsill, and E. Moulines, "An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo," *Proceedings of the IEEE*, vol. 95, pp. 899–924, 2007.

[21] R. Grewe, A. Hohm, S. Hegemann, S. Lueke, and H. Winner, "Towards a generic and efficient environment model for ADAS," in *2012 IEEE Intellignet Vehicles Symposium*, pp. 316–321, 2012.

[22] M. Wang, T. Ganjineh, and R. Rojas, "Action annotated trajectory generation for autonomous maneuvers on structured road networks," *The 5th International Conference on Automation, Robotics and Applications*, 2011.

[23] J.-F. Cordeau, G. Laporte, J.-Y. Potvin, and M. Savelsbergh, *Handbooks in Operations Research and Management Science*, vol. 14, ch. Chapter 7 Transportation on Demand, pp. 429–466. sciencedirect, 2007.

[24] S. Parragh, K. Doerner, and R. Hartl, "A survey on pickup and delivery problems - part II: transportation between pickup and delivery locations," *Journal für Betriebswirtschaft*, 2007.

[25] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using SUMO," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

[26] O. Madsen, H. Ravn, and J. Rygaard, "A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives," *Annals of Operations Research*, vol. 60, no. 1, pp. 193–208, 1995.

[27] S. Muelas, L. A, and J.-M. Peña, "A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city," *Expert Systems with Applications*, vol. 40, no. 14, pp. 5516–5531, 2013.

[28] T. Fraichard and T. M. Howard, *Handbook of Intelligent Vehicles*, ch. Iterative Motion Planning and Safety Issue, pp. 1435 – 1458. Springer, 2012.

[29] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, 2011.

[30] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*, 2010.

[31] S. M. LaValle, *Planning Algorithms*, ch. 5.3 Collision Detection, pp. 209 – 217. Cambridge University Press, 2006.