A Software Architecture for an Autonomous Racecar

Johannes Betz, Alexander Wischnewski, Alexander Heilmeier, Felix Nobis, Tim Stahl, Leonhard Hermansdorfer, Markus Lienkamp

***Abstract*—This paper presents a detailed description of the software architecture that is used in the autonomous Roborace vehicles by the TUM-Team. The development of the software architecture was driven by both hardware components and usage of open source languages for making the software architecture reusable and easy to understand. The architecture combines the autonomous software functions perception, planning and control which are modularized for the usage on different hardware and for the purpose of using the car on high speed racetracks. The goal of the paper is to show which software functions are necessary for letting the car drive autonomously and fast around a racetrack.**

I. Motivation

Autonomous driving dominates the headlines of daily news. Almost all major car manufacturers are working on the step-by-step integration of ADAS (Advanced Driver Assistance Systems) functions that will enable autonomous driving in the future. According to the definition of SAE [1], the levels of autonomous driving are defined from level 0 (the driver drives himself) to level 5 (no driver in the vehicle anymore). The missing driver in level 5 creates a multitude of challenges that have to be considered during the development of the vehicle hardware and software. The first step is to completely convert the driving task into software: The vehicle must be able to plan its own path (navigation), follow a path (path tracking) and stabilize itself (control) based on environmental information (perception). In the second step, the successful introduction of these algorithms in series production vehicle requires great robustness and self-diagnosis capabilities to handle rare situations safely. Furthermore, the software collection has to show descent computational performance as hardware will always be limited and the autonomous driving task poses several large scale problems. In the last step, the software has to be tested so that a safe use of the vehicle in a real environment can be guaranteed. For step two and three, simulators can be used or real test drives can be carried out. The problem is that there is no complete simulation environment covering all relevant tests for autonomous driving so far. Furthermore, a lot of know-how and information can be obtained from real test drives with the racecars. The problem is, that driving with level 5 vehicles is not permitted everywhere on public roads, the test vehicles are expensive and require a lot of maintenance and the recording of measurement data takes a long time. In addition to all these challenges, as a university research institute it is particularly difficult to get the time, money and hardware to test autonomous driving functions in real world scenarios.

For these reasons, two chairs of the Technical University of Munich (TUM) decided to participate in a new racing series called *Roborace* with their own team. Roborace is a support series of the FIA Formula E and takes place on the same tracks [2]. The goal of Roborace is to provide the first racing series for autonomous vehicles. The teams that take part in this competition develop the software for the provided autonomous racecars called *Robocars* [3].

The concept of offering a vehicle that can be used as a development platform allows developing and evaluating algorithms for autonomous driving without own hardware. The software therefore needs to integrate the holistic autonomous driving functions for perception, planning and control. This offers various advantages. Firstly, a deeper understanding and further development of state of the art algorithms in these areas can be carried out. Secondly, because of the racing environment, the software has to consist of high-performance, robust and secure algorithms working at high clock rates to keep real time capability and fast response times at high speeds (max. 300 kph). In addition, the absence of a driver eliminates a limiting factor in the vehicle and thus enables the technical potential of the algorithms regarding utilizing longitudinal and lateral accelerations. Thirdly, the racetrack provides a safe and reproducible terrain (racetrack) in which various scenarios can be created.In summary taking part in the Roborace racing series provides a lot of advantages especially for research institutes and allows them to learn fast from the motorsports environment [4]. However, there is no detailed description of a software architecture that provides the basic structure for autonomous driving with a detailed focus on driving on a racetrack. Furthermore, this software architecture needs to be modular such that software modules can be exchanged easily.

Johannes Betz, Alexander Heilmeier, Felix Nobis, Tim Stahl, Leonhard Hermansdorfer and Markus Lienkamp are working at the Chair of Automotive Technology, Technical University of Munich, Garching, BY 85748 Germany, (Corresponding author: Johannes Betz- email: betz@ftm.mw.tum.de).

Alexander Wischnewski is working at the Chair of Automatic Control, Technical University of Munich, Garching, BY 85748 Germany

978-1-7281-1217-6/19/$31.00 ©2019 IEEE

II. Related Work

Getting information about a holistic software architecture for autonomous vehicles is quite difficult. Initially, one would assume these tasks with the car manufacturers but today's passenger cars show only level 1-3 functions. For example, the *Tesla Model S* has an autopilot that enables automatic parking as well as a longitudinal guidance and lane keeping function on the motorway [5]. The *Audi A8* is the world's first production car that enables autonomous longitudinal and transverse control in slow-flowing traffic at up to 60 kph (congestion pilot) [6, 7]. Both cars are available but the knowledge about the hardware and software structure is non existent as well as the extension of the software to further functionality .

If one switches to the universities and research institutions, one realizes that the focus is more on the development of isolated driving functions for autonomous driving than on the holistic framework. The Center for Automotive Research (CARS) at Stanford University [8] focuses on functions for image processing (automatic camera calibration, vehicle and object detection [9, 10] and methods for localization [11, 12]. Using a driving simulator, the University of Leeds [13] investigates both human factors and functions to increase safety. UC Berkeley [14] researches in the development of algorithms based on artificial intelligence for the detection of vehicles from videos, for a vehicle control system based on real driver input as well as for security and privacy in the deep learning area.

Nevertheless, there are publications and projects that show a holistic software framework for autonomous driving. In the Darpa Grand Challenge the Stanford vehicle Stanley [15] was one of the first level 5 vehicles with a research purpose. The authors give an overview of the hardware and software structure of the vehicle but missing a detailed description of the ECU hardware and individual software functions. In the Darpa Urban Challenge Carnegie Mellon University's vehicle Boss drove 97 km through an urban environment, interacting with other moving vehicles and obeying the California Driver Handbook [16]. The authors give a detailed overview of the individual autonomous driving functions in the perception, motion planning and control part. However, detailed information concerning the software architecture are missing. Equipped with a drive-by-wire system based on an architecture and hardware developed for the DARPA Urban Challenge [15] and previously implemented in the Junior3 vehicle [17] Stanford University published a racing vehicle based on an Audi TTS [18]. The authors present an overview of their system architecture but focus more on the detailed path planning and control of the vehicle [19].

Besides these research vehicles there are some open-source software frameworks that can be used for the development of autonomous driving functions. The Robot Operating System (ROS) [20] provides both a structured communication layer and a library for autonomous driving functions. Maybe because of its good documentation it is getting more and more attention during the last few years. Autoware [21, 22] is based on ROS and provides a rich set of self-driving modules composed of sensing, computing, and actuation capabilities. Although Autoware is used in different research projects and autonomous vehicles, it is unknown how the different software modules work together. The last to be mentioned is Apollo [23] which has a highly performant, flexible architecture that should help to accelerate the development, testing, and deployment of autonomous vehicles. The Apollo 3.0 software is supplied with modules for perception, localization and control of the vehicle and offers a reference hardware for getting started with the software framework.

In summary, it can be said that there is a lot of literature dealing with the software and the software architecture for autonomous driving. Unfortunately, a lot of literature is not state of the art anymore regarding the hardware and misses a detailed description of the hardware. The focus of today's literature is on developing algorithms for autonomous driving. In addition, there is no literature that shows the software structure for autonomous driving on a racetrack.

III. Hardware

The Robocar is a battery electric vehicle with various sensors for perceiving the environment and localizing itself. It has two front and four surround cameras, five LiDAR sensors (four around the front wheels and one in the back of the car), two RADAR sensors (one in the front and one in the back) as well as 17 ultrasonic sensors all around the car. All the information from the sensors is united in the main ECU, the *Nvidia Drive PX2* [24]. The Drive PX2 is an Ubuntu 16.04-based ECU with two separate GPUs for high performance computing applications. With this ECU, a platform for deep learning, sensor fusion and surround vision is available for the software developers of the Robocar. The software running on the Drive PX2 provides the information for the real time motion controller *Speedgoat mobile target machine* [25]. This Simulink Real Time-based developer ECU offers real-time computation performance and allows to easily integrate MATLAB/Simulink models. A Short summary of the specifications of the two ECUs is provided in Table 1.

Table 1: Robocar ECU specs

	Drive PX2	**Speedgoat Mobile Target Machine**
Operating System	Ubuntu 16.04	Simulink Real-Time
CPU	4x ARM Denver 8x ARM Cortex A57	1x Intel Core i7 2.5 GHz, 2 cores
GPU	2x Tegra X2 (Parker) 2x Pascal GPU	-
Memory	16 GB LPDDR4	12 GB DDR3 RAM
Storage	128 GB eMMC	64 GB SSD

Interfaces	CAN, Ethernet	CAN, Ethernet
Performance	20 TFLOPS FP16	-
TDP	250W	-

IV. SOFTWARE

Pendelton et. al [26] define three major tasks an autonomous vehicle has to perform and that have to be implemented in the corresponding software: Perception, planning and control. Each of these tasks includes different subtasks which we call autonomous driving functions (Figure 1).

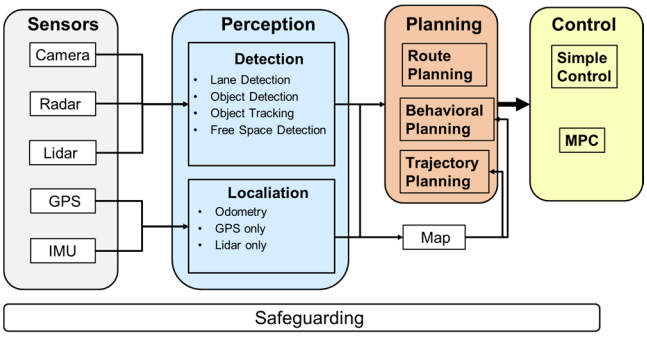

Figure 1: Software functions overview [27]

A. *Overall structure: Combining hardware and software*

First, we want to present how the three software parts work together and how they are allocated to the hardware in the Robocar (Figure 2). In general, it can be said that the software components are splitted into two groups. The first, running on the NVIDIA Drive PX2, consists of data and/or computational intense algorithms. This includes the complete perception and planning parts of the software. In contrast, highly time critical but less complex algorithms are running on the Speedgoat Real-Time ECU. It allows time consistent execution with high frequencies which is necessary for robust control performance during fast driving.

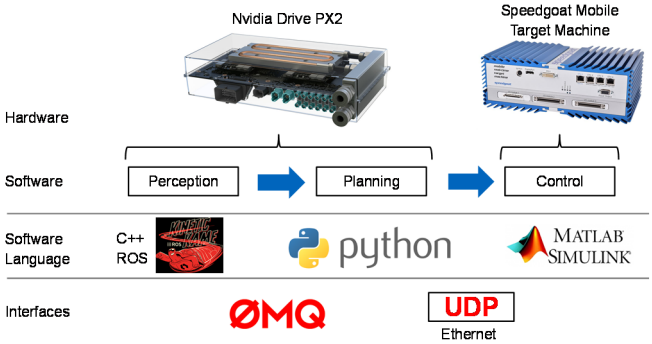

Figure 2: Overall hardware and software structure

The goal of the perception part is to gain a holistic representation of the environment around the car (e.g. lane detection algorithms [28–30]). Therefore, the data acquired from the various sensors have to be unified in the perception functions. The sensors produce many data in every time step, so the Nvidia Drive PX2 for this purpose. Besides the computing capability, the Drive PX2 runs a Linux operating system that allows us to use open source ROS functions. In this context, we are using the *gmapping* SLAM (Simultanous Localization and Mapping)-algorithm [31] for mapping an occupancy grid map [32] of the environment and an AMCL (Adaptive Monte Carlo Localization)-algorithm [33], for localizing the vehicle on LiDAR only.

Every information from the perception module that is needed in the planning module is sent via *ZeroMQ [34]*. This is a high-performance asynchronous messaging library providing a message queue, but unlike message-oriented middleware it can run without a dedicated message broker. The information coming through the ZeroMQ socket is processed in the planning module running also on the Drive PX2. The focus of this software part is calculating a fast and safe trajectory. Right now, we are using a globally minimum curvature path with separate speed profile planning for a static environment that will be accompanied by local trajectories later in the project for planning in dynamic environments. In an additional behavioral planner, which is similar to a state machine at the moment, time critical behavioral decisions have to be made, e.g. overtaking, lane changing or safety braking. Both algorithms are written in Python 3 using the NumPy package, which makes C-functions usable in Python and therefore very efficient, especially for large arrays.

Both localization and trajectory information are sent via *UDP (Ethernet)* to the control software running on the Speedgoat. UDP is advantageous because it does not retransmit lost packets and has a smaller delay than TCP. In addition, UDP offers multicast applications, which are hard to implement in TCP. The control part includes both longitudinal and lateral control of the car based on acceleration and curvature signals and is implemented in MATLAB/Simulink. This software is compiled for the usage on the Speedgoat.

B. *Specific structure: Autonomous racecar driving functions*

On the one hand, the displayed overall software structure is a generic architecture that works for every autonomous driving purpose in different vehicles. On the other hand, the specific adaptation to the Robocar hardware and the choice of specific software languages are making this software structure beneficial for the usage on a racetrack. The first test of the software structure was at the Formula E Event in Berlin 2018 [35]. The purpose of the Berlin competition was to build a robust software architecture that can set a proper lap time in an obstacle free environment. For this purpose, we now propose a specific functional software architecture for an autonomous racecar (Figure 3).

First, the software architecture is split into offline and online parts. This is possible because there were different testing sessions where the car could drive alone on the racetrack. A human driver therefore drives three laps around the track where we fuse GPS-, LiDAR and odometry data into one ROS data file (ROS Bag) on the Drive PX2.

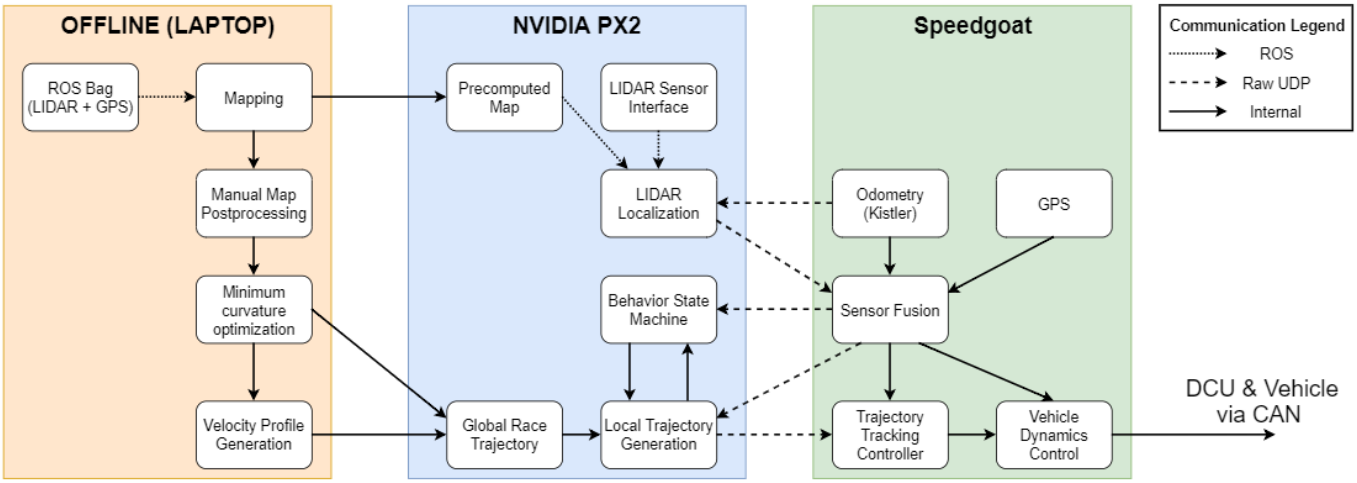

Figure 3: Functional software architecture of the TUM team for the Robocar

The ROS Bag is then used for creating a 2D occupancy grid map of the racetrack with three different states: “occupied”, “unknown” and “free of obstacles”. After that, the map is post-processed by smoothing and filtering the raw map data, applying Euclidean distance transform and extricating the centerline. This centerline is then used to determine the globally fastest trajectory around the racetrack. This trajectory generation is performed based on a minimum curvature optimization. It minimizes the squared sum of the curvature values at every intermediate point by relocating it along the centerline normals. A velocity profile is then calculated for the resulting raceline using a forward-backward-solver that considers the car's longitudinal and lateral acceleration limits dependent on the vehicle velocity. This allows to include aerodynamic effects and motor limitations. After the offline calculations, both the precomputed occupancy grid map and the global race trajectory are stored into a database on the Nvidia Drive PX2. The car is now ready to drive autonomously on the racetrack. In the autonomous driving sessions, there are mainly three active software parts.

The first software part is a behavior state machine that monitors the mission of the system (e.g. it checks if the car is ready to launch, it introduces tire warmup laps, it counts the completed laps and it adjusts the velocity profile). The local trajectory generation module extracts local parts from the global raceline for the vehicle controller. This is done with a frequency of 25Hz. To counteract possible network transmission or process failures much longer foresight emergency trajectories are generated simultaneously and are fed to the controller.

The second software part is a localization software based on a Kalman Filter (KF), which is a model based filter algorithm. The KF runs on the Speedgoat and fuses the localization information from the GPS sensors, the velocity sensors, the IMU and the LiDAR localization information from the AMCL running on the Drive PX2 (Figure 4). The former is executed at 250Hz to provide fast information to the control algorithm, while the latter is executed asynchronously depending on the incoming LiDAR measurements. This is approximately 25Hz in the current setup.

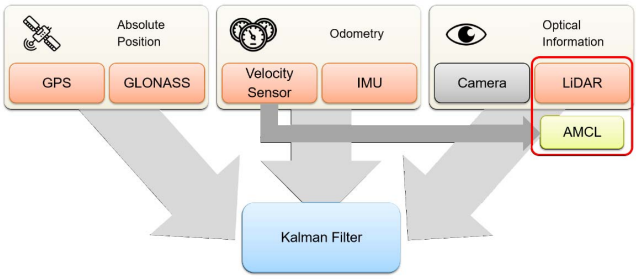

Figure 4: Sensor fusion in the Kalman Filter for localization

The third software part is the real time trajectory controller that actuates the electrical engines, the brakes and the steering of the Robocar. It is executed at 250Hz and therefore guarantees fast response times and a sufficient close loop performance. The control software is separated into a lateral path and a velocity tracking controller. Both are designed using a two-degree of freedom structure and rely heavily on a feed-forward control capturing the main vehicle characteristics. The feed-back control is based on the localization obtained from sensor fusion in the Kalman Filter. The clear separation of trajectory control and trajectory planning allows to leverage the advantages of a fast controller execution in combination with more complex planning algorithms. The latter would be significantly limited if it would be required to run within a couple of milliseconds.

V. Conclusion and Discussion

In this paper, we presented a holistic software structure for autonomous driving based on the hardware used in the Roborace

racecar. This software was implemented on the Robocar and successfully tested in the Roborace event in May 2018. We proved the robustness of the software by beating the lap time of a human driver [35]. In addition, the software structure is modular and extendable. For future work, we are planning to integrate a real time vehicle detection algorithm and to complement the global planner by a local trajectory planner to allow overtaking and evasion maneuvers taking static and dynamic objects on the racetrack into account. For this we have to calculate all possible, i.e. thousands, safe and fast trajectories without losing the real-time capability of the algorithm. Accordingly, the research activities will be focusing on solving these tasks on both of the Tegra GPUs on the Drive PX2. Also, we want to integrate new functions like a friction detection to increase utilization of the lateral an longitudinal acceleration potentials.

This paper is presenting the overall structure for and is giving an idea about the different software parts. Each software part, e.g. localization, trajectory generation and control of the vehicle, is too complex for explaining it in one paper. The authors are therefore aiming for publishing each software part in an individual paper. For further interest, we will publish our whole software architecture on Github.

CONTRIBUTION

Johannes Betz initiated the idea of this paper and contributed essentially to the overall structure of the software and this paper Alexander Wischnewski, Alexander Heilmeier, Felix Nobis, Tim Stahl, Leonhard Hermansdorfer contributed equally to different parts of the software architecture and of this paper. Markus Lienkamp made an essential contribution to the conception of the research project. He revised the paper critically for important intellectual content. He gave final approval of the version to be published and agrees to all aspects of the work. As a guarantor, he accepts the responsibility for the overall integrity of the paper.

ACKNOWLEDGEMENT

First, we want to thank the complete Roborace team for giving us the opportunity to work with them and using their Devbot vehicles for our research. We would like to thank the *Bayerische Forschungsstifung* for funding us in the research project *rAIcing*. We would like to thank *Continental Engineering Services* and the *TÜV Süd* for funding us with research projects. The work described in this paper was also supported by the basic research fund of the Chair of Automotive Technology from the Technical University of Munich. In addition, we want to thank *in-tech* for helping us with the Speedgoat software development.

REFERENCES

[1] SAE International, "Automated Driving - Levels of Driving Automation," [Online] Available: https://www.sae.org/misc/pdfs/automated_driving.pdf.

[2] e-formel.de, *Roborace - die Rahmenserie der Formel E.* [Online] Available: https://www.e-formel.de/roborace.html. Accessed on: May 07 2018.

[3] Roborace, *Robocar.* [Online] Available: http://roborace.com/. Accessed on: May 07 2018.

[4] J. Betz *et al.,* "What can we learn from autonomous level-5 motorsport?," in *Proceedings, 9th International Munich Chassis Symposium 2018*, P. Pfeffer, Ed., Wiesbaden: Springer Fachmedien Wiesbaden, 2019, pp. 123–146.

[5] Tesla, "Autopilot," 2017. [Online] Available: https://www.tesla.com/de_DE/autopilot. Accessed on: May 07 2018.

[6] t3n - Digital Pioniers, *Audi A8 soll „erstes Serienautomobil der Welt" mit Level-3-Automation sein.* [Online] Available: http://t3n.de/news/audi-a8-level-3-autonomes-fahren-837581/. Accessed on: May 07 2018.

[7] Audi, *So funktioniert der Audi AI Staupilot.* [Online] Available: http://blog.audi.de/2017/07/17/audi-ai-staupilot/. Accessed on: May 07 2018.

[8] Stanford University, *Center for Automotive Research at Stanford.* [Online] Available: https://cars.stanford.edu/. Accessed on: May 07 2018.

[9] D. Held, J. Levinson, and S. Thrun, "Precision tracking with sparse 3D and dense color 2D data," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, pp. 1138–1145.

[10] D. Held, J. Levinson, S. Thrun, and S. Savarese, "Combining 3D Shape, Color, and Motion for Robust Anytime Tracking," 2014.

[11] J. Levinson and S. Thrun, "Robust Vehicle Localization in Urban Evnironments Using Probabilistc Maps," in *2010 IEEE International Conference on Robotics and Automation*, 2005.

[12] D. Yamamoto and N. Suganuma, "Localization for autonomous driving on urban road," in *2015 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, Okinawa, Japan, pp. 452–453.

[13] University of Leeds, *Driverless Cars.* [Online] Available: http://www.leeds.ac.uk/site/custom_scripts/spotlight/driverless-cars.php. Accessed on: May 07 2018.

[14] University of California Berkeley, *Berkeley Deep Drive.* [Online] Available: https://deepdrive.berkeley.edu/. Accessed on: May 07 2018.

[15] S. Thrun *et al.,* "Stanley: The robot that won the DARPA Grand Challenge," *J. Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[16] C. Urmson *et al.,* "Autonomous driving in urban environments: Boss and the Urban Challenge," *J. Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[17] J. Levinson *et al.,* "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, Baden-Baden, Germany, pp. 163–168.

[18] J. Funke *et al.,* "Up to the limits: Autonomous Audi TTS," in *2012 IEEE Intelligent Vehicles Symposium (IV)*, Alcal de Henares Madrid, Spain, pp. 541–547.

[19] V. A. Laurense, J. Y. Goh, and J. C. Gerdes, "Path-tracking for autonomous vehicles at the limit of friction," in *2017 American Control Conference (ACC)*, Seattle, WA, USA, pp. 5586–5591.

[20] M. Quigley *et al.,* "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[21] S. Kato *et al.,* "An Open Approach to Autonomous Vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.

[22] S. Kato *et al.,* "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, Porto, Portugal, pp. 287–296.

[23] Apollo, *Apollo Open Platform.* [Online] Available: http://apollo.auto/index.html.

[24] NVIDIA, *Nvidia Drive PX 2.* [Online] Available: http://www.nvidia.de/object/drive-px-de.html. Accessed on: May 07 2018.

[25] Speedgoat, *Mobile real-time target machine.* [Online] Available: https://www.speedgoat.com/products-services/real-time-target-machines/mobile. Accessed on: May 07 2018.

[26] S. Pendleton *et al.,* "Perception, Planning, Control, and Coordination for Autonomous Vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[27] J. Betz *et al.,* "What can we learn from autonomous level-5 motorsport?,"
[28] Z. Kim, "Robust Lane Detection and Tracking in Challenging Scenarios," *IEEE Trans. Intell. Transport. Syst.*, vol. 9, no. 1, pp. 16–26, 2008.
[29] D.-K. Lee *et al.,* "Real-time lane detection and tracking system using simple filter and Kalman filter," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, Milan, pp. 275–277.
[30] J. C. McCall and M. M. Trivedi, "Video-Based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation," *IEEE Trans. Intell. Transport. Syst.*, vol. 7, no. 1, pp. 20–37, 2006.
[31] S. Thrun, "Simultaneous Localization and Mapping," in *Springer Tracts in Advanced Robotics, Robotics and Cognitive Approaches to Spatial Mapping*, M. E. Jefferies and W.-K. Yeap, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 13–41.
[32] A. Milstein, "Occupancy Grid Maps for Localization and Mapping," in *Planning for Unraveling Deformable Linear Objects Based on Their Silhouette*, H. Wakamatsu, E. Arai, and S. Hirai, Eds.: INTECH Open Access Publisher, 2008, pp. 389–409.
[33] F. Delleart, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo Localization for Mobile Robots," [Online] Available: https://www.cc.gatech.edu/~dellaert/ftp/Dellaert99icra.pdf.
[34] iMatix, *ZeroMQ: Distributing Messaging*. [Online] Available: http://zeromq.org/.
[35] K. Burke, *The Fast and the Driverless: Munich Team Takes Home Roborace Victory: Researchers at Technical University of Munich turn to NVIDIA DRIVE to fill their need for speed at Berlin event*. [Online] Available: https://blogs.nvidia.com/blog/2018/06/29/fast-and-driverless-munich-roborace-victory/.