Model-based, Composable Simulation for the Development of Autonomous Miniature Vehicles

Christian Berger, Michel Chaudron, Rogardt Heldal, Olaf Landsiedel, Elad M. Schiller*
Chalmers University of Technology | University of Gothenburg, Sweden
Department of Computer Science and Engineering
christian.berger|chaudron@gu.se, olafl|elad|heldal@chalmers.se

ABSTRACT

Modern vehicles contain nearly 100 embedded control units to realize various comfort and safety functions. These vehicle functions consist of a sensor, a data processing, and an actor layer to react intelligently to stimuli from their context. Recently, these sensors do not only perceive data from the own vehicle but more often also data from the vehicle's surroundings to understand the current traffic situation. Thus, traditional development and testing processes need to be rethought to ensure the required quality especially for safety-critical systems like a collision prevention system. On the example of 1:10 scale model cars, we outline our model-based and composable simulation approach that enabled the virtualized development of autonomous driving capabilities for model cars to compete in an international competition.

Keywords

Model, Simulation, Testing, Autonomous Miniature Vehicles

1. INTRODUCTION

Modern vehicle functions increase not only the driver's comfort by providing automated parking features but these functions are also responsible for the reduction of traffic accidents by, e.g., the adaptive cruise control or a pedestrian collision warning system. In common for these functions is that they depend heavily on perceived data from their system context to act reliably and safely. Inspired by the 2007 DARPA Urban Challenge [11], the yearly competition "CaroloCup"[1] for universities was initiated at Technische Universität Braunschweig to foster the research and education for this type of embedded systems on the example of 1:10 scale model cars. Within this competition, participating autonomous miniature vehicles need to demonstrate the reliable capability to follow their own lane, to overtake stationary vehicles blocking their own lane, to handle intersections safely according to the right-of-way traffic rule, and to park the vehicle at a sideways parking strip as fast as possible. Furthermore, the teams shall have an eye on the components and manufacturing costs as well as the vehicle's energy consumption.

These driving tasks are inspired by the aforementioned international competition for regularly sized vehicles but they reflect also the technical state-of-the-art for all major automotive original equipment manufacturers (OEM). In the upcoming future, even autonomous driving for less demanding traffic situations like traffic jams on highways will not only contribute positively to the driver's personal comfort level but also help to reduce rear-end collisions [2]. Moreover, these mature vehicle functions will be enhanced in the future by incorporating data, which are provided by vehicle-to-infrastructure communication or by vehicle-to-vehicle communication. While these interconnected and thus collaborating vehicle functions will help to reduce travel time on the one hand and to reduce the risk of collision with partly occluded vehicles on the other hand, their development is more demanding than ever before. The reason is that test scenarios on real proving grounds require not only several other vehicles to carry out a specific test case but it must also be ensured that a specific choreography can be reproduced with a precise accuracy to measure e.g. differences between two versions of the same algorithm over time. This hurdle was earlier addressed by the Gulliver test-bed [9] where off-board infrastructure, such as a positioning system, was assumed to be accessible. No such assumptions are considered in this work and the capabilities presented are based on on-board equipment, i.e. autonomous driving.

Moreover, the amount of possible and probable traffic situations, which might be encountered by these interconnected and perceiving algorithms, is not only unlimited but also hard to reduce to a set of some "representative" situations, which are carried out on real proving grounds. Furthermore, even the common practice of automotive OEMs to run long-term evaluation drives with pre-series vehicles will not address this challenge sufficiently because the probability to encounter a majority of all important traffic situations for these interconnected and perceiving functions is rather small.

Instead, these existing methods of validating a vehicle's proper functionality in real test-runs must necessarily be extended by appropriate simulative approaches. However, before a certain simulation tool is chosen to be used during the development and evaluation of a vehicle function, it must be clarified which questions should be addressed with simulative approaches, what are the necessary quality levels and level of details that must be achieved by such a simulative approach to answer these questions, and finally, how does the expected development and testing process look like that is based on the intended simulative approach.

In this contribution, we outline our methodical and technical approach to provide a simulation environment during the

*Partially supported by the EC, through project FP7-STREP-288195, KARYON (Kernel-based ARchitecture for safetY-critical cONtrol)

[1] www.carolocup.de

design and development of an autonomously driving 1:10 scale miniature vehicle. Therefore, the rest of the paper is structured as follows: Firstly, related work is discussed followed by a description of methodical and technical concepts for our simulation environment. Next, the applicability of this simulative approach is discussed on the example of the development of a lane following algorithm, before the paper finishes with a conclusion and an outlook of our future plans.

2. RELATED WORK

First methodical concepts for a virtual testing approach were developed and evaluated during the development of an autonomously driving vehicle for the 2007 DARPA Urban Challenge competition [11]. These concepts were further elaborated and tested together with the University of California, Berkeley in a subsequent project during the development of a navigation algorithm for an autonomous ground vehicle (AGV) [1, 3, 4]. Furthermore, some of the system components for the miniature vehicles used in this project are based on earlier knowledge and technology developed for the Gulliver Testbed [2] [12].

During the case study at the University of California, Berkeley, no lane marking detection was required due to the fact that these information were provided within the digital map. Moreover, the available mounting space on the AGV as well as the available energy supply allowed for a very powerful sensors concept and computation setup. In the "CaroloCup" competition, the limitations on the miniature vehicle like mounting space and energy supply influenced significantly the chosen hardware architecture and the sensor layout, which is described in Sec. 3.2. Therefore, the simulation environment that was used for the regularly sized AGV had to be adapted to provide data according to the chosen sensor setup for the small size version.

In contrast to the approach outlined in this paper, the authors of [7] describe Player/Stage, which is used to program and experiment with experimental robotic platforms. The framework offers a standardized interface to several robots as well as sensors like ultrasonic and laser scanner. In contrast to the solution described here, Player/Stage does not primarily support automotive function development because models for the vehicle's surroundings like roads with lanes and lane markings are missing. Furthermore, the supported motion models in the simulation are focusing on robotic platforms, which are able to turn on the z-axis which is impossible for regular vehicles.

The Robot Operating System (ROS) as described by [10] aims for providing a standardized communication and configuration platform especially for experimental robotic platforms. Therefore, it provides transparent access to a variety of sensors and includes also an image processing library. However, comparable to the aforementioned approach and in contrast to the approach described in this paper, simulative models for vehicle motion and its environment are missing.

In [5], the authors outline a concept to provide a consistent model-based approach for simulation system modelers and simulation system users. By the described transformation process, the complexity of the considered simulation environment could be managed. The outlined methodical approach of manually modeling the environment is related to this approach but the outlined transformation process focuses on the actual configuration of a discrete event system simulation while our approach is dealing with the formal description of concrete elements from the environmental system context.

[2]www.Gulliver-Testbed.net

3. MODELING VIRTUAL WORLD

In this section, conceptual and technical aspects of our simulation environment are outlined to support the development of the autonomously driving miniature vehicle. The timeline of approximately four months for the project on the one hand, the ordering and shipping of all required hardware components, as well as the manufacturing of the actual miniature vehicle on the other hand enforced us to parallelize the algorithm conception phase, the functional implementation, and the realization of the hardware architecture. Thus, a simulation environment turned out to be the enabling method to divide the development efforts.

3.1 The Virtual Test-Track

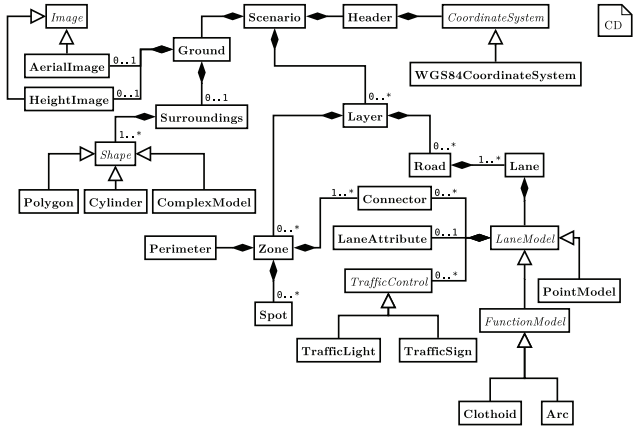


Figure 1: Class diagram describing the environmental model: The structure from this class diagram was modeled as a domain-specific language (DSL) for the simulation environment realized by Boost::Spirit for C++ [1]. Instances serve as input data to construct an OpenGL scene graph or high-level object-oriented data structures depending on the required level of details for the simulation.

First, a virtual test track needed to be modeled. Therefore, the domain-specific language depicted by Fig. 1 was used to describe the vehicle's surroundings [1]. Instances from this language describe visible elements like lane markings and stationary obstacles with a specific shape and color, structural elements like the mathematical shape of a lane (e.g. a straight line or an arc), and topological elements like the relation between two lanes and their surrounding road or the interconnection of lane segments to form a graph.

An example for the real test track is provided within the official rules & regulations document [8] that also reflects the real proving ground used in the last years' competitions. Besides the overall layout, all necessary parameters like color

and thickness of the lane markings, the distance between two center lines, the individual lane width, and the minimum curvature are provided by the official rules for the competition. These parameters were used to create a model for our simulation environment from the exemplary track, which consists of 26 lane segments that are either of the type "straight" or "arc" lane with a minimum curvature of 1,000mm according to the rules & regulations document.

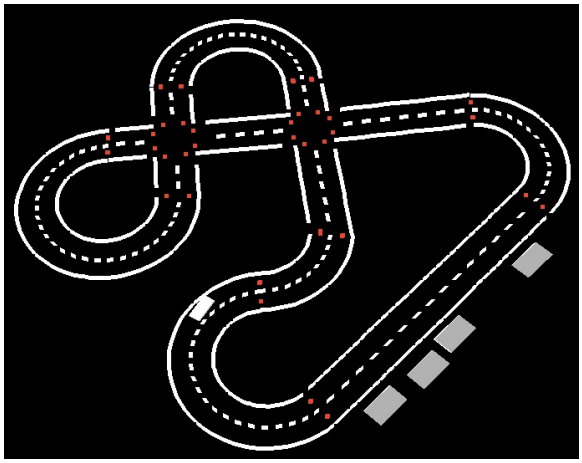


Figure 2: Modeled test track in our simulation environment with curvatures as specified in the official rules & regulations document. Furthermore, we added stationary obstacles as white boxes for the overtaking and sideways parking situation.

An excerpt from an instance of the DSL is shown in Appendix A. This instance is parsed by Boost::Spirit to generate a graph-based representation, which is traversed afterwards to construct an OpenGL scene-graph to be rendered in 3D. The result is shown in Fig. 2. Besides the actual two-lane track layout, we modeled also some obstacles for the sideways parking situation in the lower right part and the blocked lane situation in the center of the track.

3.2 The Virtual Miniature Vehicle

As mentioned before in Sec. 1, the autonomous miniature vehicle shall have the different capabilities to interact safely and reliably with its surroundings. In the following, the hardware architecture is described followed by the corresponding sensor models for the simulation environment.

The basic capability is the lane following feature, which requires an algorithm that is keeping the vehicle on its own lane. Due to the fact the the competition will be carried out indoors without providing a digital map of the track, GPS combined with an inertial measurement unit would not be helpful to realize that task. Therefore, a sensor is required to detect the lane markings to calculate the correct steering and acceleration operations for the car. As shown in Fig. 3, we decided to use a vision-based approach realized by a Logitech C525 HD WebCam.

The advanced driving tasks rely on the capability to detect stationary and dynamic obstacles. After analyzing the rules & regulations document, we decided to focus on the right

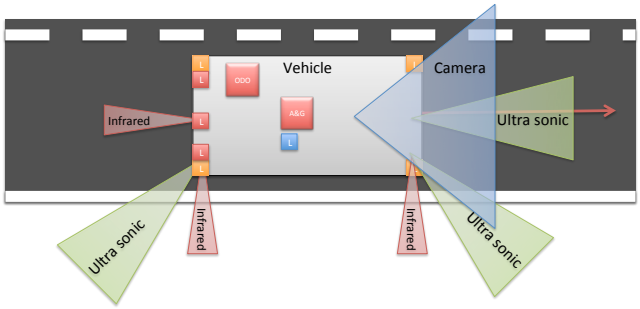


Figure 3: Concept of the sensor layout for the autonomously driving model car

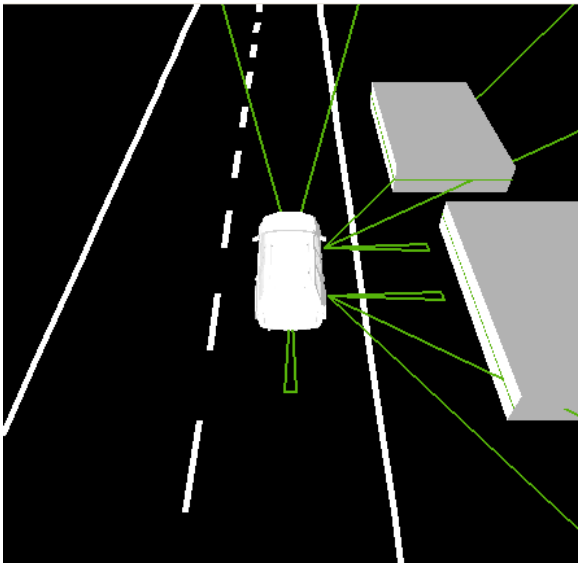
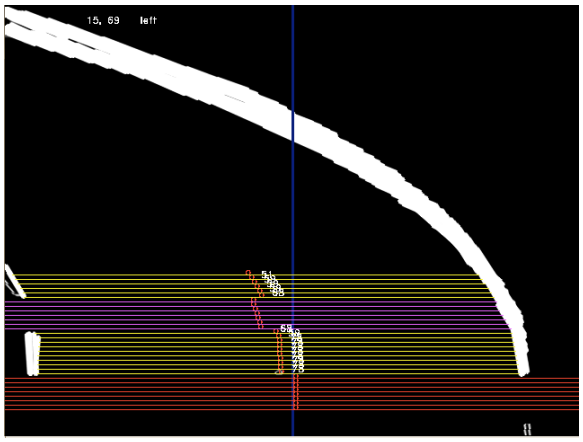


Figure 4: The derived sensor layout from the official requirements was modeled in the simulation environment: The small triangles are the infrared sensors and the larger triangles are the ultrasonic sensors, which are jointly used to detect obstacles.

hand side as well as the vehicle's front due to the sideways parking situation and the right-of-way situation at intersections. Thus, it is necessary to get distances from other obstacles to create a consistent environment representation for identifying the right spot for the sideways parking or the check whether the intersection is clear to pass. As depicted by Fig. 3, we decided to use ultrasonic distance sensors to cover the range between 0.3m and 6m with an opening angle of 30° in combination with infrared distance sensors to cover the range between 0.04m and 0.3m. Thus, the total energy consumption is in the range of approximately 4W for the environmental sensors.

To parallelize the ordering and manufacturing process of the autonomous miniature vehicle and the algorithm development, we modeled the chosen sensor setup in the simulation environment. Therefore, both distance-based sensors are modeled in such a way that the set of intersection points is

(a) In the sensor layout depicted by Fig 3, a vision-based approach is used to detect the lane markings. The input data can be grabbed directly from the 3D simulation environment.



(b) Based on the virtual camera input data, the lane detecting algorithm was developed and tested: The horizontal lines indicate whether a left and right lane marking is present (yellow), only one lane marking was detected (purple), or none lane marking was detected (red).

Figure 5: Simulated camera sensor for the development and test of the lane detection algorithm.

calculated for all stationary and dynamic obstacles within the modeled field of view, which is depicted by the green triangles in Fig. 4. Next, the closest intersection per triangle is selected and its relative distance to the virtual mounting position of the sensor is determined. According to the technical specification of the real sensors, the simulation models are executed at 20Hz, respectively. The simulation model for the camera can be realized fairly easily by defining an appropriate OpenGL camera for the 3D scenery. In Fig. 5(a), the bare input image for the lane detecting algorithm is shown. In the simulation environment, this sensor model is executed with a resolution of 640x480 pixels at 10Hz.

The actual simulation environment is based on the same methodical and technical approach that was used during the development of the navigation algorithm for the AGV at University of California, Berkeley [1]. As shown in Fig. 6, first, the environment model is created using a graphical tool for visual feedback to the modeler. In that tool, all required elements like roads, lanes with appropriate lane markings, stationary and dynamic obstacles are modeled; for dynamic obstacles, a predefined route that needs to be followed can be specified for instance. At run-time, the resulting instance of the DSL is then used by independently running components, which realize one specific aspect of the simulation. This design concept relies on separation of concerns [6] to divide and handle the overall complexity. These components are coupled by developers on demand at run-time by a UDP multi-cast session. As described in [1], the entire simulation can also be automated and executed unattendedly as part of a continuous integration service.

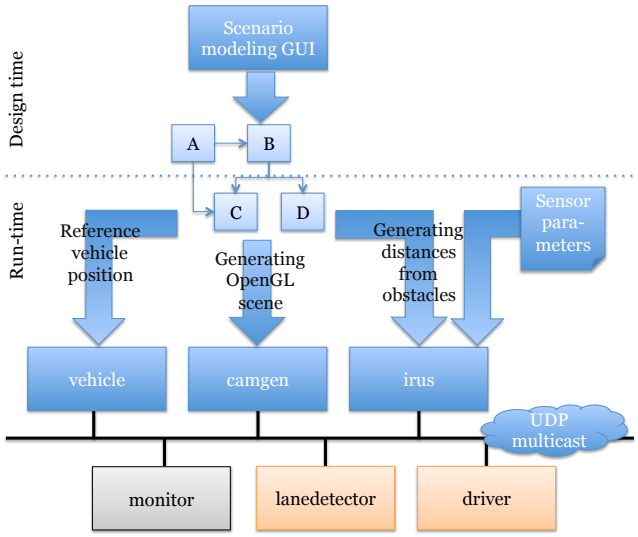


Figure 6: Our model-based, composable simulation approach at design time and run-time: Firstly, we model the environment using a graphical editor and set up sensor parameters; at run-time, the simulation environment consists of several independent components, which can be run selectively according to the required details and which share the same UDP multi-cast session. During the development of the lane detector algorithm, the distance-based sensor simulation "irus" is not executed for instance.

To realize the simulation for the autonomous miniature vehicle, the following components were used: "vehicle", "monitor", "irus", and "camgen". The first component simply provides the absolute position data of the virtual vehicle; however, the position data is not used by the actual algorithms for the real car but it is necessary to realize the other simulation components. "monitor" is a visualization environment for the modeled instance of the DSL together with all other exchanged packets from the live UDP multi-cast session. Therefore, it derives a corresponding OpenGL 3D scene-graph to allow the inspection of the current traffic situation in 3D. Furthermore, the exchanged UDP packets can be recorded for a later playback.

When the component "irus" is additionally activated by a

developer, it initially reads the modeled instance from the environment DSL to get all modeled obstacles. These obstacles are either simplified polygons with a specific height or complex 3D models in Wavefront OBJ format. Furthermore, "irus" reads its sensor setup consisting of the sensor's heading, mounting position wrt. to the vehicle's center, and the angle and distance for the field of view from a configuration file. Combined with the continuously received absolute position data for the virtual car, these simplified sensors' viewing areas are positioned accordingly to calculate the relative distance between its virtual mounting position and an intersecting surface from one of the environmental obstacles. The resulting distance data is broadcasted to autonomous miniature vehicle's algorithms.

The implemented concept of "camgen" to provide an image stream from a virtualized camera is comparable to the previously described visualization application: Firstly, it reads the modeled instance of the environment DSL to construct its OpenGL 3D world; next, it is continuously receiving the updated absolute position from "vehicle" to move its OpenGL camera to the corresponding position in the 3D environment to provide a continuous image sequence to the lane detecting algorithm.

3.3 Performance of the Simulation

In the following, preliminary results of the lane following algorithm consisting of "lanedetector" and "driver" are presented. In Fig. 5(b), the feature detection results from the bird's eye perspective are shown: yellow horizontal lines indicate that lane markings could be successfully detected on the left and right hand side, purple lines indicate that one side could not be detected, and red lines indicate that no lane markings were found at all. Based on this input data, the desired steering angle for the control algorithm is derived and fed back to the simulation component "vehicle".

In Fig. 7, the driven path in absolute coordinates is depicted for five test-runs of the lane following algorithm during the beginning of the test-track on the right hand side in Fig. 2 where the sideways parking spot is located. The dark and light blue curves represent the skeleton lines for each lane. In the best case, the vehicle would follow perfectly this skeleton lane. In our example, the resulting absolute vehicle positions of all five test-runs lay mostly on top of each other.

In Fig. 8, the absolute deviation from the lane's skeleton line is shown for all five test-runs. Comparable to the previous chart, all test-runs are very similar. The slight differences between each other are caused by the algorithmic design of the lane following algorithm: Its feature detection operates on the most recent perceived image and thus, depending on the provided images from the simulation environment, the derived steering angles are slightly different for several test-runs. Since the absolute deviation of all test-runs is very similar for the test-runs, this limitation is acceptable for the development and evaluation of the algorithm.

For carrying out the real test-runs for the autonomous miniature vehicle (cf. Fig. 10), a real world test-track on a 4mx4m pond liner as shown in Fig. 9 was prepared. Currently, experiments with the algorithms in reality are carried out to achieve two goals: Firstly, the right parameters for the lane

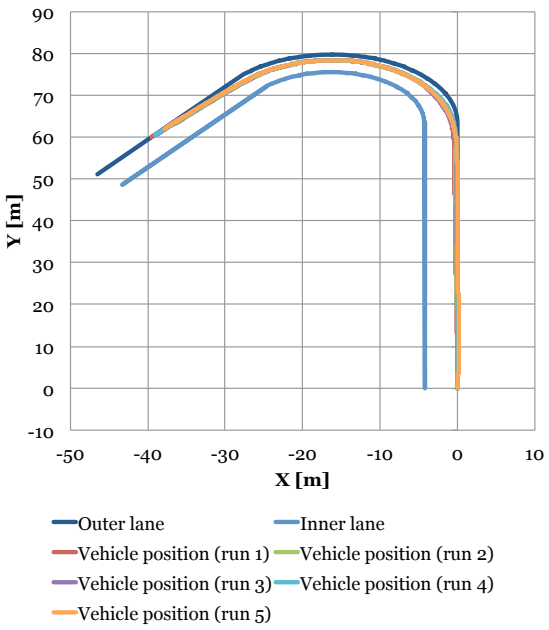


Figure 7: Driven path of the lane following algorithm at the beginning of the course for five different test-runs: The dark blue line is the skeleton line of the outer lane and the light blue line is the one from the inner lane. The orange path occludes the lines from the other test-runs due to the fact that all five runs are very similar.

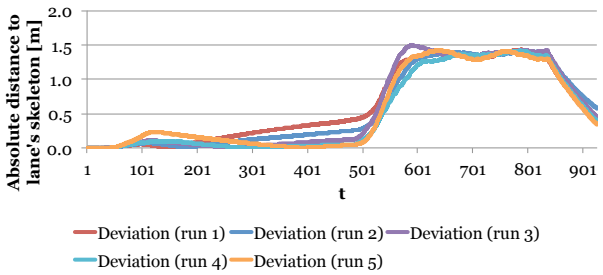


Figure 8: Absolute deviation of the vehicle's center from the lane's skeleton line for five test-runs in the simulation.

following algorithm shall be found to exhibit a reliable vehicle motion in reality; secondly, the model parameters in the simulation shall be optimized to provide a virtual development and testing environment that is even closer to the reality.

4. CONCLUSION AND OUTLOOK

In this paper, we presented a methodical and technical approach to support the development and testing of sensor-based vehicle functions. The concept was originally developed for an autonomous ground vehicle at the University of California, Berkeley but we extended it to be applicable for the "CaroloCup" competition of 1:10 scale autonomous miniature vehicles. Due to the tight schedule in the competition on the

Figure 9: Part of the real test track, which was realized by a black pond liner and white tape.

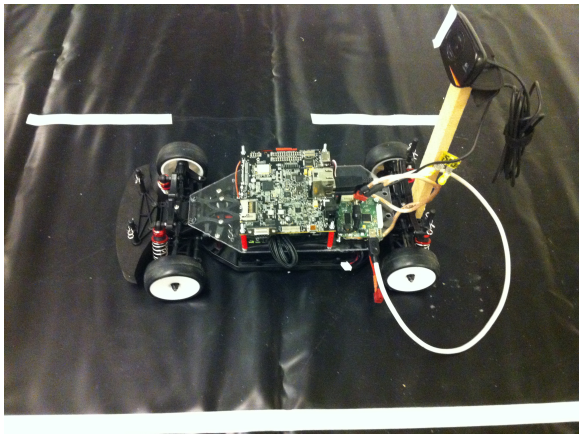


Figure 10: 1:10 scale miniature vehicle with PandaBoard main ECU and camera for lane detection.

one hand and because of the manufacturing process of the vehicle on the other hand, the software development for the algorithms to handle lane following, intersections, and parking at a sideways parking strip required the utilization of an approach that enabled a distributed and parallel development of hardware and software.

The outlined approach based on a domain-specific language to formally model the traffic situation. These instances are the input for composable simulation components that fulfill certain aspect of the entire simulation, respectively. Furthermore, simulation models for sensors were outlined and their usage during the development and test a lane following algorithm for the autonomous miniature vehicle was described.

The next steps are to standardize the software interface of the autonomous miniature vehicle to simplify the integration of prototypical algorithms. Furthermore, a formal language to describe a sensor setup is planned to simplify the configuration process on the one hand and to carry out sensor concept optimization strategies on the other hand. Moreover, our outlook includes a unified framework for simulation and development of both autonomous and cooperative capabilities [9]. As a future goal, we will look into the effect of autonomous driving in very large scale traffic systems, and by that further validate the proposed solutions before studying them on full-scale vehicular systems with many vehicles.

Acknowledgments

The authors would like to thank team Dr Meili for their help to develop an autonomous miniature vehicle. Furthermore, we would like to our sponsor HiQ AB, Göteborg, especially Anders Bengtsson, Jonas Tisell, Mohsen Nosratinia, and Anders Palo. Moreover, we would like to thank Gulliver test bed hosting this project, and Benjamin Vedder, who designed the vehicle's computer system setup and developed some of the key components.

5. REFERENCES

[1] C. Berger. *Automating Acceptance Tests for Sensor- and Actuator-based Systems on the Example of Autonomous Vehicles*. Shaker Verlag, Aachener Informatik-Berichte, Software Engineering Band 6, Aachen, Germany, 2010.

[2] C. Berger and B. Rumpe. Autonomous Driving - 5 Years after the Urban Challenge: The Anticipatory Vehicle as a Cyber-Physical System. In U. Goltz, M. Magnor, H.-J. Appelrath, H. K. Matthies, W.-T. Balke, and L. Wolf, editors, *Proceedings of the INFORMATIK 2012*, pages 789–798, Braunschweig, Germany, Sept. 2012.

[3] C. Berger and B. Rumpe. Engineering Autonomous Driving Software. In C. Rouff and M. Hinchey, editors, *Experience from the DARPA Urban Challenge*, pages 243–271. Springer-Verlag, London, UK, 2012.

[4] J. Biermeyer, H. Gonzales, N. Naikal, T. Templeton, S. Sastry, C. Berger, and B. Rumpe. Rapid Integration and Automatic Calibration for new Sensors using the Berkeley Aachen Robotics Toolkit BART. In *Proceedings des 11. Braunschweiger Symposiums "Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel"*, volume 11, pages 71–88, Braunschweig, Germany, Feb. 2010. Gesamtzentrum für Verkehr Braunschweig e.V.

[5] A. D'Ambrogio, D. Gianni, J. L. Risco-Martín, and A. Pieroni. A MDA-based Approach for the Development of DEVS/SOA Simulations. In *Proceedings of the 2010 Spring Simulation Multiconference on - SpringSim '10*, pages 142:1–142:8, New York, New York, USA, Apr. 2010. ACM Press.

[6] E. W. Dijkstra. On the role of scientific thought. *Selected Writings on Computing: A Personal Perspective*, pages 60–66, 1982.

[7] B. P. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, Coimbra, Portugal, 2003.

[8] R. Matthaei. "Carolo-Cup" - Regelwerk (Rules & Regulations) 2013. June 2012.

[9] M. Pahlavan, M. Papatriantafilou, and E. M. Schiller. Gulliver: A Test-bed for Developing, Demonstrating and Prototyping Vehicular Systems. In *Proceedings of*

the MobiWac' 11, pages 1–8, Miami, Florida, Oct. 2011.

[10] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS : an open-source Robot Operating System. In *Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA)*, number Figure 1, Kobe, Japan, May 2009.

[11] F. W. Rauskolb, K. Berger, C. Lipski, M. Magnor, K. Cornelsen, J. Effertz, T. Form, F. Graefe, S. Ohl, W. Schumacher, J.-M. Wille, P. Hecker, T. Nothdurft, M. Doering, K. Homeier, J. Morgenroth, L. Wolf, C. Basarke, C. Berger, T. Gülke, F. Klose, and B. Rumpe. Caroline: An Autonomously Driving Vehicle for Urban Environments. *Journal of Field Robotics*, 25(9):674–724, Sept. 2008.

[12] B. Vedder. Gulliver: Design and Implementation of a Miniature Vehicular System. Master's thesis, Chalmers | University of Gothenburg, 2012.

APPENDIX

A. EXEMPLARY EXCERPTS FROM THE ENVIRONMENT DSL

In the following, a short excerpt of the environment DSL is depicted. In this example, the formal description of a lane segment is shown, which is based on a radius, the arc's length in radians, and its rotations around the z-axis to fit seamlessly to a preceding lane segment.

```
...
LANE
LANEID 2
LANEWIDTH 4.0
LEFTLANEMARKING broken_white
RIGHTLANEMARKING solid_white
(1.1.2.1) -> (1.1.1.2)
(1.1.2.2) -> (1.1.3.1)
FUNCTIONMODEL
ARC
RADIUS 16.2
[ 0 2.443460953 ]
START
ID 1
VERTEX2
X 0
Y 63.5
END
ID 2
VERTEX2
X -27.700000762939453
Y 75
ROTZ 1.570796327
ENDFUNCTIONMODEL
ENDLANE
...
```

Listing 1: Excerpt from environment DSL that describes an arc-based lane model.

```
...
SURROUNDING
SHAPENAME Box_0
POLYGON
HEIGHT 1
COLOR
VERTEX3
X 1
Y 1
Z 1
VERTEX2
X 3.5999999046325684
Y 25.100000381469727
VERTEX2
X 7.599999904632568
Y 25.100000381469727
VERTEX2
X 7.699999809265137
Y 32.20000076293945
VERTEX2
X 3.5999999046325684
Y 32.20000076293945
...
```

Listing 2: Excerpt from environment DSL that describes a polygon-based obstacle.