An Automatic Scenario Generator for Validation of Automated Valet Parking Systems

Andrea Tagliavini[1], Donato Ferraro[1], Tomasz Kloda[2] and Paolo Burgio[1]
[1]*Università degli studi di Modena e Reggio Emilia, Italy*
[2]*Technical University of Munich, Germany*

Keywords: Simulators, Autonomous Driving, Scenario Generation, Valet Parking, Virtual Test Drive, OpenSCENARIO.

Abstract: A primary goal of self-driving car manufacturers is to create an autonomous car system that is clearly and demonstrably safer than an average human-controlled car. The real-world tests are expensive, time-consuming and potentially dangerous. The virtual simulation is therefore required. The autonomous driving valet parking is expected to be the first commercially available automated driving function without a human driver at the wheel (SAE Level 4). Although many simulation solutions for the automotive market already exist, none of them features the parking environments. In this paper, we propose a new software virtual scenario generator for the parking sites. The tool populates the synthetics parking maps with objects and actions related to these environments: the cars driving from the drop-off point towards the vacant slots and the randomly placed parked cars, each with a given probability of exiting its slot. The generated scenarios are in the OpenSCENARIO format and are fully simulated in the Virtual Test Drive simulator.

1 INTRODUCTION

The mandatory requirement for the approval of the highly automated and autonomous driving is to demonstrate that the self-driving cars do not pose more of a risk to pedestrians and other cars than the cars piloted by humans. The validation of such systems needs to be as extensive and as complete as possible to cover every corner-case testing scenario. This is usually done by millions of miles of road tests resulting in very high costs in terms of both money and time required by the field data collection. The real-world tests are potentially dangerous, especially if full- or semi-autonomous functionalities must be validated. The simulation cannot fully replace the real-world tests and driveable millage but can significantly help to reduce potential safety risks, time-to-market and the number of prototypes required in the development and verification process.

1.1 Driving Simulation

The obvious advantage of the simulation methods for the self-driving compared to the real-world testing is that the simulation does not technically require a real vehicle and has no running costs (e.g. fuel consumption, maintenance). Simulation approach also widens the spectrum of the test-cases allowing to recreate many situations which would be too dangerous and too expensive in the real-world involving pedestrians, cyclists or high-speed maneuvers under different weather, visibility and traffic conditions. Last but not least, the testing process is significantly accelerated by running the simulations at a faster pace than real-time.

Full-scale simulation comes with the test catalog covering the scene diversity and the variety of edge cases. A relevant scenario is based on the gathered data and the safety requirement specification. Creating each scenario statically can be long and allow to test only a narrow set of the most common behaviors. The machine learning algorithms driving autonomous cars can be overfitted and fail in unseen scenarios due to the little or no data variance. Random sub-scenario generation for simulators is quite promising in this respect.

The automatic augmentation of the gathered real-world data can help to overcome this limitation by generating the new mixed-reality scenarios avoiding time-consuming and expensive human interaction. In essence, the set of recorded data (weather conditions, visibility, traffic) is mixed up to create a wider set of test scenarios. The randomized elements in these scenarios are, principally, the actors (vehicles of various

Tagliavini, A., Ferraro, D., Kloda, T. and Burgio, P.
An Automatic Scenario Generator for Validation of Automated Valet Parking Systems.
DOI: 10.5220/0009417004890496
In *Proceedings of the 6th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2020)*, pages 489-496
ISBN: 978-989-758-419-0

types and pedestrians), their behaviors and, to a lesser extent, the surrounding scenery with the road network. However, a non-controlled randomization of generated scenarios may lead to the unrealistic scenarios or to the scenarios that may be out of testing scope, potentially increasing the necessary time for training and testing of the model. The parametrizable scenario generators let the user define the boundaries of the generation accordingly to the safety specification and the use case under test.

1.2 Automated Valet Parking System

The automated valet parking system is about to become the first commercially available facility at SAE 4 level (Society of Automotive Engineers, 2018) (a human is not required to take over in any situation) (Automotive World, 2018). The system drives autonomously a vehicle from the drop-off point to the assigned parking slot and returns it to the dedicated pick-up point when required. The parking site infrastructure retrofitted with sensors and connectivity communicates constantly with the vehicles and guides them to the vacant parking slots. The prior knowledge of the environment reduces the number of test scenarios (compared to urban driving) but the physical testing is still expensive and may still pose a serious risk for the humans and the environment. This operational domain requires also a set of its specific models of objects and behaviors.

1.3 Contribution

In this paper, we introduce the *UNIMORE Map Populator*, a virtual scenario generator for the parking sites. The tool fills the scenario maps with objects and actions related to the parking environments: parked cars, pedestrians, moving cars looking for an empty slot or the cars that are exiting their slots and reenter the traffic again. It permits a user-defined ratio of traffic participants randomizing their placement and their traffic behaviors within a parameterized set of values (e.g. the distances between the exiting cars and the cars under test). The tool is fully integrated with the *OFFIS StreetArt* synthetic parking maps generator. The scenarios are stored in the *OpenSCENARIO* format and can be played in the *Virtual Test Drive* (*VTD*) from *Vires* (VTD, 2019), a well-established toolchain for driving simulation. To our knowledge, the *UNIMORE Map Populator* and the *OFFIS StreetArt* are the first scenario generators specifically targeting driving in the parking areas. We provide the necessary abstraction to build the models of traffic in the parking areas and enable the safety analysis required by the international standards (ISO, 2011).

1.4 Paper Organization

This document is structured as follows. In the next section, we present the related work. In Section 3, we describe the scenario generator for the static and dynamic content of the parking sites. Section 4 outlines the simulation of the basic parking maneuvers. We conclude the paper in Section 5.

2 RELATED WORKS

In this section, we describe the works that focus on scenario-based simulation automated or autonomous driving software with respect to the proposed virtual scenario generator for the parking sites.

2.1 Driving Simulators

In the fields of automated and autonomous driving, simulators are widely adopted for both training and testing purposes. The simulators like *Dynacar* (Tecnalia, 2019) or *Gazebo* (Syed Ahamed et al., 2018) target dynamic aspects of the vehicle (e.g. electrified powertrain systems) while others, like *VTD* (VTD, 2019), *PreScan* (TASS International, 2019) or *Carla* (Dosovitskiy et al., 2017) allow traffic behavior testing. One of the most important features of a driving simulator is realism, both in terms of visual resemblance with the real world, physics and vehicle behavior. In recent years, researchers started also to exploit the capabilities of the video games graphical engines. The simulators like *Carla* (urban driving simulation) or *Torcs* (Wymann et al., 2015) (racing simulation) are built on game engines such as *Unity* and *Unreal Engine*. In this work, we decided to use *VTD* to play the generated scenarios but any simulator supporting *OpenSCENARIO* format can be used instead.

2.2 Parameterized Simulation

The *ASM Traffic* adds traffic to *dSPACE's Automotive Simulation Models (ASM)* (dSPACE, 2017). It simulates the behavior of the vehicle under test in a traffic situation involving other traffic participants. The simulation can be performed by varying the model parameters. During the simulation, the scenario segments and road features can also be altered by means of scripting language. In our approach, the scenario generation process is performed directly on the

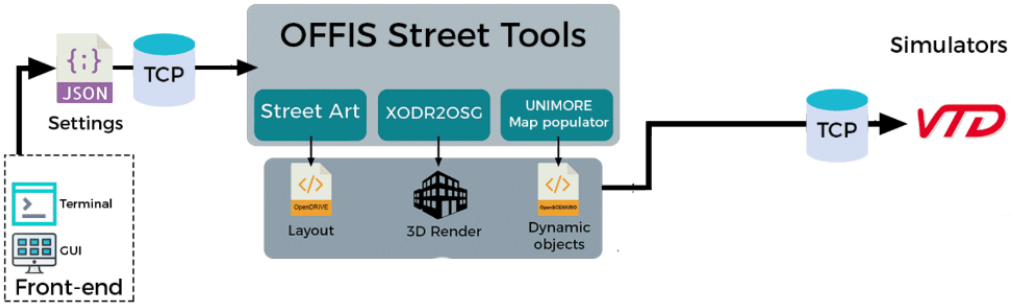

Figure 1: The scenario generation toolchain.

OpenSCENARIO structures and parking environment model is proposed.

The example in (Matlab, 2019) shows how *Matlab Simulink Automated Driving Toolbox* can be used to build and test an automated parking valet system. The toolbox provides algorithms for designing, simulating, and testing multiple aspects of autonomous driving systems like perception, sensor fusion, path planning and traffic simulation. It features also the *Driving Scenario Designer* to design synthetic driving scenarios. In our tool, the scenarios are generated automatically given the input parameters and can be further loaded in *Matlab* through the *OpenDRIVE* format.

In (Zhou and del Re, 2018) the authors identify the critical scenarios based on the data collected from the accidents. These data can be used in our approach to set the scenario generation parameters.

2.3 Parking Sites Simulation

Several works propose the simulation techniques for the parking maneuver (Heinen et al., 2015; de Oliveira Andrade et al., 2011; Schwesinger et al., 2016). For instance, in (Schwesinger et al., 2016) the required parking spot width for the parking maneuver is evaluated in simulation. None of these works consider other traffic participants. The focus of our paper is on the simulation of the entire traffic in the parking environments. In (Schönemann et al., 2019) the authors study the parking maneuvers and derive analytically the minimum safety distance between traffic participants. Our framework can be used to derive empirically certain parameters important from the safety perspective but its main goal is in identifying malfunction and corner cases of the autonomous driving algorithms in complex traffic situations.

Stanford's robot Junior (Montemerlo et al., 2009) was equipped with separate planners for common road navigation and parking slot navigation. In (Löper et al., 2013; Tcheumadjeu et al., 2018; Min and Choi, 2013; Banzhaf et al., 2017; Chirca et al., 2015) the valet parking prototypes with a fully automated navigation and monitoring are described. The focus of these works is on system architecture and implementation. The aspects related to the safety and simulation are not covered.

3 SCENARIO GENERATOR

In this section, we present the scenario generation process, breaking it into two separate parts: *static* and *dynamic* content generation.

The *static content* is constituted by the road network (e.g. roads) with the road objects (e.g. traffic signs, road surface marking, cars) and their detailed topological and topographical order. Since most of the driving simulators require a similar static content, several standards were developed. The one that has gained traction in recent years is *OpenDRIVE*. The format stores the static content data in an XML file that describes various features and geometry of the road and its surrounding.

The *dynamic content* specifies the time-variant behavior of all the active entities that take part in the simulation (e.g. vehicles and pedestrians). Actions of traffic participants (e.g. lane change) and road infrastructure (e.g. change of a traffic control signal) are triggered by conditional events (e.g. vehicle's position is ten meters to the pedestrian). *OpenSCENARIO* is a commonly used description of the dynamic content for driving and traffic simulators. It describes the traffic actions in a hierarchical structure of XML file as a storyboard subdivided into stories, acts, and sequences.

In our framework, the scenario generation process is orchestrated by the *Street Tools*, a non-free academic tool suite from *OFFIS* [1]. It includes three plugins: static content generation, dynamic content generation, and 3D rendering. Figure 1 depicts the structure of the scenario generator. The first plugin, map-

[1]OFFIS – Institut für Informatik, www.offis.de

generator, randomly creates a road network, while the second plugin, map-populator, populates it with dynamic objects. The 3D rendering plugin is beyond the scope of this work. Because of its open and modular design, we decided to use *VTD* as environment simulation software. Hereafter, we detail the plugins for map-generation and map-population.

3.1 Static Content: Map Generation

We use the *StreetArt* from OFFIS to generate the synthetic maps of the parking sites. The maps are automatically assembled from the tileset composed of straight roads, 90 degrees bends, roadside parking slots, three-way and four-way junctions. Each tile of the set has its own definition in an OpenDRIVE file, with absolute position, fixed-length width, and parking slots. The generated layout file is simply a collection of these tiles which have been properly arranged and offsetted by the algorithm. The generation parameters (e.g. parking area size, number of parking slots) are specified by the user and the final layout is saved in the *OpenDRIVE* format.

An OpenDRIVE file begins with a *header* element that has time and version as the attributes. The road network is specified with street elements (*road*) and intersections (*junction*). Each element has a unique ID attribute and can be linked with the other elements (*link*). Street elements can have another street or intersection element as a predecessor and successor.

Figure 2 shows an example of a synthetically generated map.

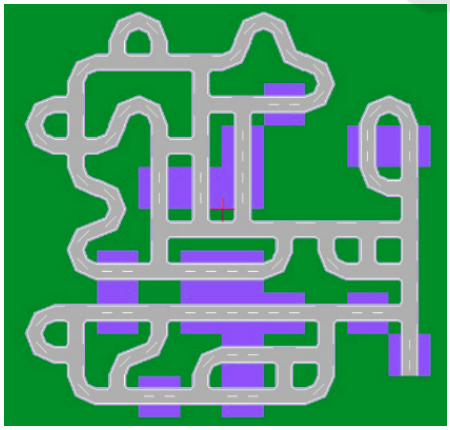

Figure 2: A synthetic map generated by StreetArt.

3.2 Dynamic Content: Map Population

The *map populator* is the plugin that we implemented to automatize the vehicle placement on the parking site maps and the behavior assignment for the various traffic participants. The safety engineer is no longer burdened with a tedious task of manually placing the vehicles and specifying their behaviors. The scenario is now automatically generated according to the user-defined parameters.

The *map populator* permits to specify the number of parked cars, the number of driving cars, and the probabilities of different actions that these cars can undertake.

The coordinates of the parking slot, its orientation with regards to the road and the car parked on it are calculated and added to the structure describing the road map of a given scenario. Figure 3 shows an XML snippet with a sample parking slot.

```
<objects>
    <object type="ParkingSpace" name="RdParkingBox_90Deg_White.flt"
            id="10" s="1.25" t="-6.5" zOffset="-1.9"
            length="5" width="2.5" height="4"
            hdg="1.57" pitch="0" roll="0">
    </object>
</objects>
```

Figure 3: XML snippet with parking slot definition.

The driving cars are randomly placed on the road network and one of them is selected as the *ego* car (i.e. the car under test). The user defines the ratio of the cars that are parking and those that are cruising in the autonomous mode.

Each parking car is assigned a path towards a distinct unoccupied parking slot (reachability from the initial point is verified). In the XML file, a path is defined as a sequence of roads and a routing strategy (e.g. shortest, fastest, least intersections or random). The path is generated by selecting randomly the start and the destination from the set of road elements (the destination is chosen from the road subset with lateral parking slots).

At the destination, the car performs a parking maneuver. The type of parking maneuver depends upon the car relative position to the parking slot (see detailed description in the next section). For parking maneuver and for exiting maneuver specific trajectories are designed. A trajectory is defined as a sequence of vertexes with a specific shape (e.g. polyline, spline or clothoid). We model the parking trajectory by three way-points and a counter that is incremented at each way-point and triggers a predefined action (e.g. reduce the speed, stop).

Once the parking maneuver is completed and the car is stopped, the driver can exit after a random amount of time and walk along a simple pathshape towards the parking exit. Figure 4 shows a driver that has exited a parked car. The ratio of exiting drivers is tunable and can be disabled for the fully autonomous parking systems.

The cars cruising in the autonomous mode can make sudden stops when the ego car is within a given

Figure 4: Driver exiting a parked car.

range to simulate the behavior of an undecided driver or a car malfunction.

Figure 5: Car exiting its slot at the approach of the ego car.

A specified number of parked cars is randomly placed on the free parking slots. Some of them can suddenly exit their parking slots at the approach of the ego car and re-enter the traffic (see Figure 5). The distance to the ego car that activates the action, as well as the number of cars implementing this behavior, are specified by the user.

During our test/experiments, we were able to generate batches of 10 parking scenarios with 3+1 (ego) vehicles within approximately 70-80 seconds.

4 PARKING MANEUVERS

The extensive simulation allowed us to identify a set of different parking maneuvers to park the vehicle aligned and accurately within the spot. In this section, we detail three basic parking maneuvers implemented in the *map populator*.

The driveable path of the car towards its parking slot is given in the scenario by the identifier of the initial and the final road tile. The simulator finds and assigns to the car a path between these two tiles. When the car reaches its final destination, it must enter the parking slot. However, the parking maneuver is not defined in the *OpenSCENARIO* standard. We therefore specify it as a sequence of absolute coordinates and associated actions. The parking driveable path depends on the vehicle's position relative to the target parking slot, the road geometry, and the vehicle's turning radius.

4.1 Forward Bay Parking

The forward bay parking maneuver is used when the target parking slot is situated perpendicularly to the road. For the standard bay parking maneuver, we select three control points (P_0, P_1 and P_2) on the parking trajectory (see Figure 6) at which the vehicle undertakes the following actions:

P_0 - disable autonomous driving mode and reduce the vehicle speed,

P_1 - follow the predefined arc trajectory,

P_2 - stop.

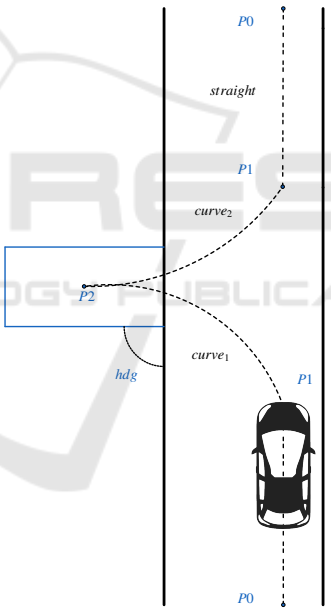

Figure 6: Bay parking maneuver.

The coordinates of the control points are given as:

$$P_0 = \begin{cases} x_0 = EndRoad.x \\ y_0 = EndRoad.y \end{cases}$$

$$P_1 = \begin{cases} x_1 = x_0 + \Delta \cdot \cos(EndRoad.hdg) \\ y_1 = y_0 + \Delta \cdot \sin(EndRoad.hdg) \end{cases}$$

$$P_2 = \begin{cases} x_2 = ParkingSlot.x \\ y_2 = ParkingSlot.y \end{cases}$$

with:

- **EndRoad** - the destination road element,
- **ParkingSlot** - the selected parking slot,
- **x** - x coordinate of road element,
- **y** - y coordinate of road element,
- **hdg** - the angle of the ParkingSlot relative to the EndRoad direction,
- **Δ** - parameter depending on the distance between the beginning of the EndRoad and the ParkingSlot, the default width of a single parking slot, the vehicle's turning radius.

The shape of the curvature (*spline*) depends on the definition of the steering angle of the car's model. As the figure shows, for a single parking maneuver we generate two symmetric trajectories. This is because the car driving in the simulator autopilot mode can arrive from one of two opposing directions and the actual path cannot be determined before running the simulation. The trigonometric functions, sine and cosine, are used to determine P_1 in more complex scenarios when the road is not perpendicular to the X or Y axes. By using sine and cosine, we can apply the same method to determine the parking trajectory for any orientation of the road.

4.2 Junction Parking Slot

When the car approaches the parking slot from the front (Figure 7), it performs the parking maneuver with the coordinates of the points P_0 and P_1 modified as follows:

$$P_0 = \begin{cases} x_0 = FrontRoad.x \\ y_0 = FrontRoad.y \end{cases}$$

$$P_1 = \begin{cases} x_1 = ParkingSlot.x + \Delta \cdot \cos(hdg) \\ y_1 = ParkingSlot.y + \Delta \cdot \sin(hdg) \end{cases}$$

4.3 Margin Parking Slot

As shown in Figure 8, in this case, the space is too narrow to park the car with a single turn. Therefore, the entire maneuver is decomposed into three segments: two forward and one reverse. The first segment trajectory transposes the vehicle parallel to the parking spot. Then, the reverse segment trajectory moves the car away from the parking spot but, at the same time, turns the car's front towards it. The extra distance and the new car orientation resulting from moving away from the parking slot makes the next phase easier. In the end, the forward-path drives the car into the target slot.

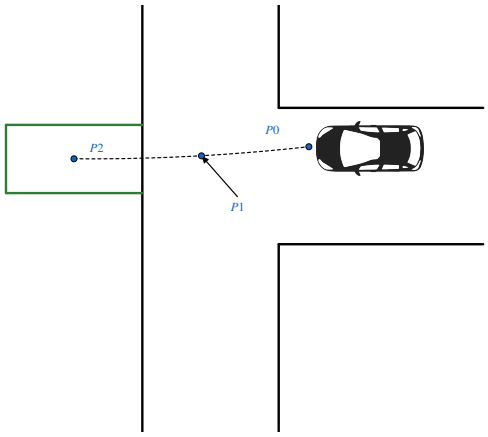

Figure 7: Parking maneuver on junction.

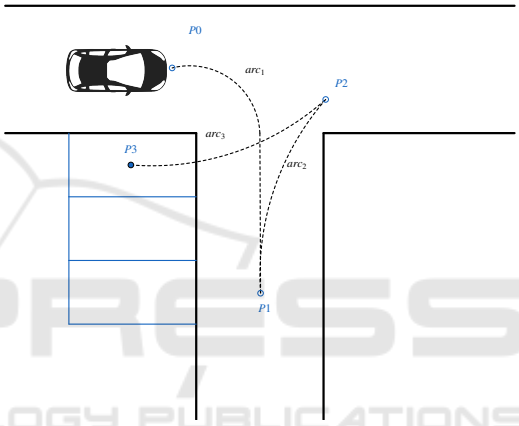

Figure 8: Maneuvers for margin parking slots.

The first arc starts from the end of the penultimate road element in the path leading to the parking slot (PrevEndRoad) and its coordinates are given as:

$$P_0 = \begin{cases} x_0 = PrevEndRoad.x \\ y_0 = PrevEndRoad.y \end{cases}$$

$$P_1 = \begin{cases} x_1 = EndRoad.x + \Delta \cdot \cos(hdg) \\ y_1 = EndRoad.y + \Delta \cdot \sin(hdg) \end{cases}$$

The second arc forms a "V" with the first arc and is done in reverse. Its end coordinate is given by[2]:

$$P_2 = \begin{cases} x_2 = PrevEndRoad.x + \Delta \cdot \cos(hdg) \\ y_2 = ParkingSlot.y \quad - \Delta \cdot \sin(hdg) \end{cases}$$

Finally, the path to drive into the parking space:

$$P_3 = \begin{cases} x_3 = ParkingSlot.x \\ y_3 = ParkingSlot.y \end{cases}$$

[2]The parameter Δ can be replaced by the separate parameters Δ_x and Δ_y for x and y coordinates.

5 CONCLUSIONS

In this work, we introduced a toolchain for automatic generation of random scenarios for autonomous driving in parking sites. The scenarios are described in *OpenSCENARIO* format and can be played in various driving simulators. The toolchain was integrated into a broader test system comprising of autonomous vehicle control logic, vehicle dynamic, sensors, traffic observer, real data database (Schönemann et al., 2019; Esen et al., 2020).

In our ongoing work, we are extending the catalog of parking maneuvers (e.g. reverse and parallel bay) and exporting the generated scenarios to the simulators based on the commercial game engines.

ACKNOWLEDGEMENTS

This work has been conducted within the ENABLE-S3 project that has received funding from the ECSEL Joint Undertaking under Grant Agreement no. 692455. This Joint Undertaking receives support from the European Union's HORIZON 2020 research and innovation programme and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.

This work was also supported by the Prystine Project, funded by Electronic Components and Systems for European Leadership Joint Undertaking (ECSEL JU) in collaboration with the European Union's H2020 Framework Programme and National Authorities, under grant agreement no. 783190.

Tomasz Kloda was supported by the Chair for Cyber-Physical Systems in Production Engineering at TUM and the Alexander von Humboldt Foundation.

REFERENCES

Automotive World (2018). Daimler and Bosch jointly premiere automated valet parking in China. https://www.automotiveworld.com/news-releases/daimler-and-bosch-jointly-premiere-automated-valet-parking-in-china/.

Banzhaf, H., Nienhüser, D., Knoop, S., and Zöllner, J. M. (2017). The future of parking: A survey on automated valet parking with an outlook on high density parking. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1827–1834.

Chirca, M., Chapuis, R., and Lenain, R. (2015). Autonomous valet parking system architecture. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2619–2624.

de Oliveira Andrade, K., Hernandes, A. C., and Becker, M. (2011). A rule-based controller simulation for an autonomous parallel parking of a car-like robot using laser sensors. In *American Scientific Research Journal for Engineering, Technology, and Sciences*.

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

dSPACE (2017). Automotive Simulation Models Traffic. https://www.dspace.com/shared/data/pdf/2017/dSPACE_ASM_Traffic_Product_information_2017_English.pdf.

Esen, H., Kneissl, M., Molin, A., vom Dorff, S., Böddeker, B., Möhlmann, E., Brockmeyer, U., Teige, T., Padilla, G. G., and Kalisvaart, S. (2020). *Validation of Automated Valet Parking*, pages 207–220. Springer International Publishing, Cham.

Heinen, M. R., Osório, F. S., Heinen, F. J., and Kelber, C. (2015). Seva3d: Autonomous vehicles parking simulator in a three-dimensional environment.

ISO (2011). ISO 26262: Road vehicles - functional safety. Geneva, Switzerland. International Organization for Standardization.

Löper, C., Brunken, C., Thomaidis, G., Lapoehn, S., Fouopi, P. P., Mosebach, H., and Köster, F. (2013). Automated valet parking as part of an integrated travel assistance. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 2341–2348.

Matlab (2019). Automated Parking Valet in Simulink. https://www.mathworks.com/help/driving/examples/automated-parking-valet-in-simulink.html.

Society of Automotive Engineers (2018). Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.

Min, K. and Choi, J. (2013). Design and implementation of autonomous vehicle valet parking system. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 2082–2087.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A., and Thrun, S. (2009). *Junior: The Stanford Entry in the Urban Challenge*, pages 91–123. Springer Berlin Heidelberg, Berlin, Heidelberg.

Schönemann, V., Duschek, M., and Winner, H. (2019). Maneuver-based adaptive safety zone for infrastructure-supported automated valet parking. In *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems, VEHITS 2019, Heraklion, Crete, Greece, May 3-5, 2019*, pages 343–351.

Schönemann, V., Winner, H., Glock, T., Otten, S., Sax, E., Boeddeker, B., Verhaeg, G., Tronci, F., and Padilla, G. G. (2019). Scenario-based functional safety for automated driving on the example of valet parking. In

Arai, K., Kapoor, S., and Bhatia, R., editors, *Advances in Information and Communication Networks*, pages 53–64, Cham. Springer International Publishing.

Schwesinger, U., Bürki, M., Timpner, J., Rottmann, S., Wolf, L., Paz, L. M., Grimmett, H., Posner, I., Newman, P., Häne, C., Heng, L., Lee, G. H., Sattler, T., Pollefeys, M., Allodi, M., Valenti, F., Mimura, K., Goebelsmann, B., Derendarz, W., Mühlfellner, P., Wonneberger, S., Waldmann, R., Grysczyk, S., Last, C., Brüning, S., Horstmann, S., Bartholomäus, M., Brummer, C., Stellmacher, M., Pucks, F., Nicklas, M., and Siegwart, R. (2016). Automated valet parking and charging for e-mobility. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 157–164.

Syed Ahamed, M. F., Tewolde, G., and Kwon, J. (2018). Software-in-the-loop modeling and simulation framework for autonomous vehicles. In *2018 IEEE International Conference on Electro/Information Technology (EIT)*, pages 0305–0310.

TASS International (2019). PreScan: simulation of ADAS & active safety. https://www.tassinternational.com/prescan.

Tcheumadjeu, L. T., Andert, F., Tang, Q., Sohr, A., Kaul, R., Belz, J., Lutz, P., Maier, M., Müller, M. G., and Stürzl, W. (2018). Integration of an automated valet parking service into an internet of things platform. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 662–668.

Tecnalia (2019). Dynacar. http://www.dynacar.es.

VTD (2019). VTD Vires Virtual Test Drive. https://vires.com/vtd-vires-virtual-test-drive/.

Wymann, B., Dimitrakakisy, C., Sumnery, A., and Guionneauz, C. (2015). Torcs: The open racing car simulator.

Zhou, J. and del Re, L. (2018). Safety verification of adas by collision-free boundary searching of a parameterized catalog. In *2018 Annual American Control Conference (ACC)*, pages 4790–4795.