3DCoAutoSim: Simulator for Cooperative ADAS and Automated Vehicles

Ahmed Hussein[1], Alberto Díaz-Álvarez[2], José María Armingol[1]
and Cristina Olaverri-Monreal[3], Senior Member IEEE

***Abstract*— The field of intelligent technologies that range from intelligent control systems to wireless communications and sensing technologies is rapidly increasing, advancing thus the research on the Intelligent Transportation Systems (ITS) field and the related innovative intermodal transport services, traffic management and road safety. Driving simulators make it possible to study driver reactions in a controlled environment. Accordingly, this paper presents the design and development of a flexible, modular tailored simulation tool to the specific requirements for investigating the effect of automation and V2X communication on drivers. Moreover, the simulator is linked to Simulation of Urban MObility (SUMO) for micro traffic emulations and connected to Robot Operating System (ROS) for architecture management and nodes handling. The proposed simulator was validated through various driving experiments, which were carried out over a selected scenario trajectory. The obtained results were compared with the results from field tests and different path tracking algorithms for automated driving showing the outcome a good and efficient performance.**

I. Introduction

Road safety is one of the top priorities for the European Commission. Even if the number of fatalities in the European Union (EU) has decreased a lot in recent years (by 52% between 2001 and 2015), and the EU has the lowest fatality rate of any region in the world [1], in 2016, traffic accidents are still the fifth cause of death [2].

Accordingly, researchers in the Intelligent Transportation Systems (ITS) field are investigating the implementation of novel approaches, Advanced Driving Assistance Systems (ADAS) and algorithms to increase road safety. In this context, the introduction of autonomous vehicles on our roads represents an opportunity to alleviate the number of accidents as the automation will make driver intervention in the control of the vehicle unnecessary [3].

This paper introduces 3DCoAutoSim, which is an abbreviation for "3D Simulator for Cooperative ADAS and Automated Vehicles Simulator". It is a vehicle simulator with high quality 3D visualization based on Unity [4], which makes it possible to emulate a variety of controlled driving environments. The version presented in this paper is an extension of the capabilities presented in [5], where a driver-centric driving platform visualized the mobility behavior of other vehicles based on traffic models and a TraCI protocol allowed communication between Unity3D and the microscopic traffic simulator SUMO; in [6], in which the optimal speed while approaching an intersection was calculated by retrieving the traffic light timing program from the road infrastructure and in [7], in which Vehicular Ad Hoc Network (VANET) communication capabilities were used to assess different information paradigms.

Driving simulators have been used in many applications, including traffic safety, ADAS implementation, driver distraction, human-machine-interaction, among-others. For example, to evaluate the usability of head-up displays (HUDs) in forward collision warning systems [8] or to assess a user interface for a novel traffic regulation system [9]. Several platforms that simulate V2X have been also developed in recent works. Some example simulation models are described in [10]. The authors in [11] combined the network simulator ns-2 [12] with the open source traffic simulator SUMO [13] to evaluate Vehicle Ad-Hoc Networks (VANET) and developed a TraCI in which SUMO and ns-2 communicated over a Transmission Control Protocol (TCP) connection to simulate vehicle-to-vehicle connections. In a further work SUMO was also integrated with the network simulator OMNeT++ to evaluate inter-vehicle communication (IVC) protocol [14]. In the field of Robotic simulation the combination of the 3D simulator Gazebo and ROS as interface make it possible to create realistic simulations [15]. Furthermore, the Udacity self-driving car nanodegree program offers a simulator to teach students how to train cars how to navigate road courses using deep learning [16], the authors utilized Unity game engine for the simulation environment. On the other hand, authors in [17], utilized Unreal engine to develop CARLA simulator, which has been developed to support training, and validation of self-driving urban systems.

All these simulations platforms have been created to address specific needs of the systems to be tested. In the same way, through the approach presented in this paper a tailored simulation platform has been developed to contribute to research in the field. We contribute to the research field by introducing 3DCoAutoSim, a modular simulation platform that integrates all the capabilities that other simulators offers. 3DCoAutoSim includes cooperative ADAS capabilities, which use vehicular data connection among multiple simulators to be able to test a variety of applications that are based on Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) or Vehicle-to-Pedestrian (V2P) communication, it also makes it possible to evaluate operational interconnectivity

[1]Intelligent Systems Lab (LSI) Research Group, Universidad Carlos III de Madrid (UC3M), Spain {ahussein,armingol}@ing.uc3m.es
[2]Universidad Politécnica de Madrid, Instituto Universitario de Investigación del Automóvil (INSIA), Spain alberto.diaz@upm.es
[3]Cristina Olaverri-Monreal was with the UAS Technikum Wien. She is now with the Johannes Kepler University, Linz, Austria. Chair for sustainable transport logistics 4.0
cristina.olaverri-monreal@ieee.org

 DOI: 10.1109/ITSC.2018.8569512

that results from the collected data coming from various vehicle, pedestrians and infrastructure. The 3DCoAutoSim simulator is also linked to Simulation of Urban MObility (SUMO) [18] for microscopic road traffic simulation and traffic congestion identification. Furthermore, the simulator is connected to Robot Operating System (ROS) to get access to various software libraries and tools related to intelligent vehicle applications.

The reminder of this paper is organized as follows: Section II describes the proposed simulator in terms of implementation, features and capabilities. The experimental setup, selected scenario and evaluation metrics follow in Section III. Section IV presents the obtained results for the validation process. Finally, Section V concludes the paper.

II. 3DCoAutoSim

The simulator is implemented using Unity, since it is a powerful 3D visualization tool, which is platform independent and has a strong physics engine. Figure 1 depicts the proposed simulator architecture hierarchy. It is divided into four main categories; features, devices, outputs and mode. The required configuration can be selected from the main menu. After clicking on the Start button the selection is compiled and the pertinent experiment loaded. The following sub-sections describe all the implemented technologies in the proposed simulator.

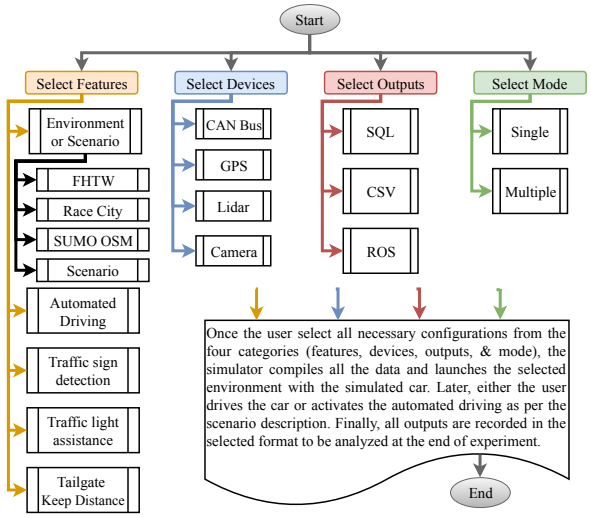

Fig. 1. Proposed simulator architecture hierarchy

A. Environments

There are several options of environments available for selection, which are summarized as follows:

- FHTW Environment, which is a 3D constructed map of the campus of University of Applied Sciences, Technikum Wien (FHTW) in the city of Vienna. The map construction was carried-out using the CityEngine software [19], achieving thus a one-to-one scale with high-quality visualization to emulate the real environment.
- Race City, which is provided as a testing environment by the Realistic Car Controller asset in Unity [20]. It includes several buildings, a parking lot and a large circular racing track for high-speed experiments.
- SUMO Open Street Maps (OSM) based environments, are dynamically configured by the user through adjusting a configuration file from any where in OSM and link it SUMO, further details are shown in the next sub-section.
- Scenario based selection are environments that are designed for a specific use-case, for instance, all experiments in this work were carried out using the validation scenario, which is explained in the experimental work section.

B. SUMO

The communication between Unity3D and the microscopic traffic simulator SUMO makes it possible to use real-world road networks together with realistic traffic models to simulate a variety of driving conditions for evaluating interaction with other road users (e.g. the uses cases mentioned in the next subsections).

Integrating traffic environment into the driver centric simulator has a series of restrictions, SUMO overcomes most of them as illustrated below.

- Macroscopic simulators lose the ability of referring to a single element in the environment, however SUMO is of a microscopic granularity nature, which allows identifying each different element.
- Time discrete and space continuous simulator. Whereas the latter can be simulated, the former is a requirement because the status of the whole simulator will be asked in a regular pace of 60Hz.
- A way to establish a bidirectional communication. Through the TraCI API, the simulator is able to provide and embody information related to each of the elements in the environment.

The SUMO OSM option requires a previously created scenario. When this option is selected and the simulation initiated, a new SUMO simulation runs in server mode. The loading operation is explained in [5]. The simulation process is as follows:

1) SUMO is started in server mode (i.e. stopped and waiting for a connection to a newly create socket in an free port), specifying the scenario the user has entered in the configuration box.
2) A new connection is made to the port from the 3DCoAutoSIM simulator.
3) All the roads are loaded to create all the network in the 3d environment.
4) All the traffic lights are loaded : the states and configuration of the traffic lights are cached in order to decrease the calls to the SUMO server.
5) All the initial cars are loaded.

The simulation is then started. For each game loop a new simulation step is performed in the micro-simulator through

the API, asking for the position of the System-Controlled Vehicles (SCV) and communicating the current position of the User-Controlled Vehicle (UCV).

C. Automated Driving and ROS

Automated driving is a feature that controls the vehicle navigation to follow a set of waypoints, generated by an offline path planning. In the proposed simulator, there are two methods of path tracking.

1) The first method is path following behavior available from Unity [21]. The logic consists of moving the object from one point to another, and upon reaching that point, moving to the next one. All the calculations are on the direction vectors among the points.
2) The second method is based on the ROS link to the simulator, which is explained in detail as follows:

In order to link ROS with the implemented simulator, a communication bridge is implemented as depicted in Figure 2. Based on the first bridge package back in 2011, `rosbridge`, the server node uses WebSocket as transport layer and enables a two-way communication between a client and a server [22]. Unlike HTTP, WebSocket keeps the connection open, and do not require HTTP handshakes for every message. In [23], [24], the authors built on the aforementioned package and implemented `rosbridgelib`, which is a library for working with JSON (JavaScript Object Notation). In this work, the library was utilized and extended to share back-and-forth ROS standard messages between ROS nodes and the Unity simulator.

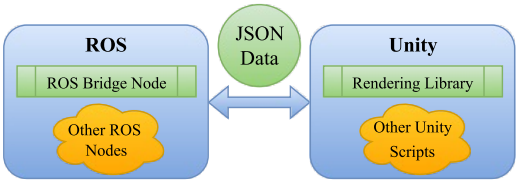

Fig. 2. Link from ROS environment to Unity simulator

By sharing messages the simulator outputs can be retrieved to be recorded through the ROS environment, thus allowing an easy comparation with data from real platforms. Moreover, messages sharing allows the simulator to read high-level control commands and transform the vehicle into an autonomous entity.

Accordingly, if the automated driving feature is selected while the ROS link is active, the path tracking algorithm based on the time-elastic-bans is executed. This approach has been implemented and tested over electric golf carts platforms [25]. The algorithm is adaptive to the environment changes, since it uses the vehicle on-board sensors to create a local map and avoids the obstacles in the environment.

D. Traffic Sign Detection

3DCoAutoSim is also able to detect and recognize traffic signs or obstacles such as construction cones in the simulated environment, using the mounted frontal camera and the works completed in [26], [27]. The objective is to increase drivers' awareness by informing them about the detected traffic signs or obstacles in the environment.

E. Traffic Light Assistance

The Traffic Light Assistance (TLA) option is an ADAS, which communicates with the traffic lights of the environment and provides the driver with information regarding the optimal speed to arrive at the intersection in the green phase [6].

F. Tailgate (Keep-Distance)

The tailgate feature is based on the works presented in [28] and [29]. The feature utilizes vehicular communication or mounted sensors, to measure the distance between the driven vehicle and the car ahead, relying on the work in [7].

G. Devices

3DCoAutoSim provides an extension API to create devices to be attached to the cars. Each new device has to be comprised at least of a prefab element (to be attached to the vehicles) and of a subclass of the Device abstract class, as described in Figure 3.

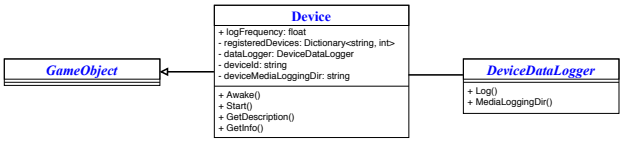

Fig. 3. `Device` abstract class as the parent for all the device classes

The figure depicts the relationship between the devices and the associated loggers to dump their data. The reason for it is that API is internally connected with the logging framework (as described in II-H). As a consequence any data related to a device can be easily created and logged.

The 3DCoAutoSim vehicles include four devices that can be enabled or disabled and allow to record their associated data for further analysis:

- **CAN Bus**. This is the only device that cannot be disabled. It gathers internal information about the vehicle, such as current speed, odometry, wheels torque and consumption. Some of these values are extracted directly from the simulator itself, whereas some others (such as the fuel consumption) are extracted from SUMO. The logging rate defaults to 10Hz.
- **GPS**. Emulates a GPS located in the middle of the vehicle. It provides the 3D position of the vehicle inside the scenario at a regular rate (defaults to 10Hz).
- **LiDAR**. Emulates a LiDAR located on top of the vehicle with the frontal position heading to the forward and with the 0-plane horizontal to the ground. It has some parameters that are configurable such as the number of planes (defaults to 8 planes), the horizontal resolution (defaults to 1deg) or the vertical separation of planes (defaults to 2.5deg). The default logging rate is 1Hz.
- **Camera**. A monocular camera which takes photos at a regular rate (defaults to 2Hz).

H. Output

As introduced in section II-E, 3DCoAutoSIM logs data for each of the devices that are used in the experiments.

To this end, methods and attributes of the `Device` subclass need to be implemented (i.e. `Row` (attribute for getting the last row of data) of `GetDescription()` (the textual information of the device object)). An overview of the logging framework that defines the methods required for the subclasses to implement is presented in Figure 4. In the framework `DataLogger` and `DeviceDataLogger` are the abstractions of the design for which `CsvDataLogger` and `CsvDeviceDataLogger` are concrete implementations.

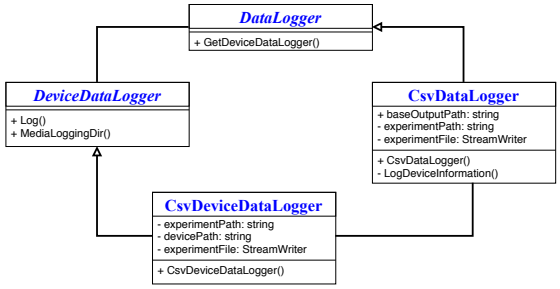

Fig. 4. Logging framework design

CSV logging is a default implementation. For each experiment, it creates a new folder with the summary of the enabled devices and their configurations as well as the time stamp for starting and ending. Each of the devices create a new csv file with all the data logged at the specified step.

The SQL option is a further implementation. It requires a query string and the drivers' device class to connect to the database. It adds a new row to the experiment tables with the configuration and devices. In case of missing device tables or data columns, the implementation will create them.

I. Multiple Simulators

Networked driving simulation represents an effective virtual prototyping tool, which supports the development of intelligent vehicles and accelerates system deployment [30]. Accordingly, 3DCoAutoSim supports the options of single or multiple nodes. The two systems have all aforementioned features and capabilities, in addition to Unity Multiplayer Service, which creates real-time networked instances of the simulator, each on a separate computer.

III. Experimental Work

In this section, the experiment setup is explained, followed by the scenario description and the evaluation metrics.

A. Setup

The utilized software are: Unity 2017.3 [4], SUMO v0.32.0 [18] and ROS Kinetic Kame [31].

In order to control the vehicle in the simulator, a Thrustmaster T500 RS controller is used as the manual steering wheel, in addition to its throttle, brake and clutch pedals, and the TH8 RS gear shifter, as shown in Figure 5. It has unprecedented 1080-degree rotation and powerful force feedback effects. The controller is connected to a car play seat to simulate a real vehicle and the simulator visuals are displayed using overhead HD beamer with resolution of 1400 x 1050, in addition to a five point one surround sound system.

Fig. 5. Thrustmaster T500 RS wheel, pedals and TH8 RS gear shifter

B. Scenario

The selected scenario was the surroundings of the FHTW in the city of Vienna. The trajectory was a total distance of 2.6 km, which included intersection, traffic lights, a roundabout and pedestrians crossing. The route was defined over OSM, as shown in Figure 6, which is considered as the theoretical path in all experiments. The theoretical path is represented as the waypoints that the center of the vehicle should reach, in the center line of the road lane. This path is obtained through defining multiple destination points over the map, and run trajectory planning algorithm [32].

Fig. 6. Selected path over the OSM of Vienna, where the green pin is the starting point and the red pin is the ending point

Multiple experiments were carried-out for the selected scenario, which are summarized as follows:

- **Driver Real Car**, users drive through the selected route using a car equipped with ADAS system. Recorded data are the vehicle GPS coordinates, orientation, velocity, acceleration, and CAN bus data.
- **Driver Simulator**, users drive through the selected route using a car in the simulator, same data parameters are recorded for later comparison.

- **Automated Simulator**, the simulator uses the route waypoints and the path tracking methods of Unity to navigate the same car on its own. Same data parameters are recorded for later comparison.
- **Automated ROS**, the simulator connects to ROS, where optimal path tracking nodes use the route waypoints to navigate the same car on its own. Same data parameters are recorded for later comparison.

C. Metrics

In order to evaluate the functionality and efficiency of the simulator, for the Ackermann modeled vehicles [33] evaluation metrics are calculated for the vehicle position. Accordingly, the mean and maximum relative position error percentages are calculated as shown in Equations (1) and (2) respectively.

$$PE_{mean}[\%] = \frac{\frac{1}{N}\sum_{k=1}^{N}\left\| \begin{bmatrix}\hat{x}_k \\ \hat{y}_k\end{bmatrix} \quad \begin{bmatrix}x_k \\ y_k\end{bmatrix} \right\|_2}{TotalDistance} \tag{1}$$

$$PE_{max}[\%] = \frac{\frac{1}{N}max(\left\| \begin{bmatrix}\hat{x}_k \\ \hat{y}_k\end{bmatrix} \quad \begin{bmatrix}x_k \\ y_k\end{bmatrix} \right\|_2)}{TotalDistance} \tag{2}$$

where $\hat{x}_k, \hat{y}_k$ are the coordinates of the vehicle and x_k, y_k are the theoretical coordinates of the vehicle at time step k.

The metrics represent the vehicle lateral deviation from the center of the lane in which the vehicle drives. They characterize driving performance and performance degradation. The lane position was measured as the distance between the vehicle center and the lane center and depends on the lane geometry. Coordinates re-sampling and interpolation were implemented to calculate the lateral path deviation for all the vehicles that were driving at different velocities. To this end each point in the experiment path was orthogonally projected to the lane center. The standard deviation of lateral position (SDLP) is calculated as the root mean square of the error.

IV. Results and Discussion

In this section, the obtained results from all experiments are presented for validation purposes of the simulation functionality and efficiency.

A. Qualitative Analysis

Figure 7 depicts the followed trajectory by the vehicle for each carried-out experiment, where the green point represents the starting position, and the red point represents the final position. The compared trajectories correspond to the four experiments (Driver Real Car, Driver Simulator, Automated Simulator and Automated ROS) mentioned in Section III-B, in addition to the theoretical path, which is the waypoints of the path shown in Figure 6.

The Automated Simulator experiment using ROS path tracking delivered the best results in terms of deviation from the lane center. This was due to the fact that it relied on the Ackermann modeling for the simulation of kinematic and dynamic parameters. The Driver Real Car and

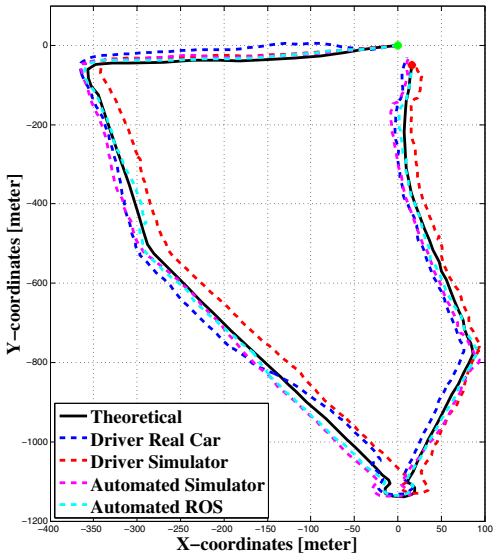

Fig. 7. Paths comparison against the theoretical one

Automated Simulator experiments delivered similar results in terms of smoothness and overall error. The Driver Simulator experiment delivered however a constant lateral-error during a period of 13 minutes on average per driver.

B. Quantitative Analysis

Table I summarizes quantitative results for the trajectories shown in the aforementioned section. The obtained results emphasis the qualitative analysis, where the automated driving using ROS obtained PE_{mean} of 0.38% (SD: 0.21%), followed by the automated driving using Unity with PE_{mean} of 0.64% (SD: 0.28%). The manual driving of the cars in both real and simulator obtained less accurate results, due to the fact that human error is involved. Driving the simulator car obtained the maximum PE_{mean} of 1.01% (SD: 0.48%).

TABLE I
EVALUATION METRICS QUANTITATIVE RESULTS

Metrics	PE_{mean} [%]	PE_{max} [m]	v_{mean} [m/s]
Driver Real Car	0.88%	37.692	4.79
Driver Simulator	1.01%	25.523	5.69
Automated Simulator	0.64%	16.879	5.11
Automated ROS	0.38%	13.421	4.89

The obtained results shows the viability of the proposed simulator to emulate ideal conditions (such as trajectories) that can be defined for autonomous vehicles and being then compared with manual driving in a real test field or in a simulator environment. The simulator functionality and efficiency was validated by the performed experiments. The performance decrease shown in the Driver Simulator experiment was due to the settings difference in terms of velocity in the simulator.

V. Conclusion and Future Work

In this work, the development of a the "3D Simulator for Cooperative ADAS and Automated Vehicles Simulator" (3DCoAutoSim) has been presented. It is based on the 3D graphic engine Unity, connection to the microscopic traffic simulator SUMO, and connection to the ROS architecture for intelligent vehicles.

3DCoAutoSim was validated by carrying out four driving experiments with the same trajectory using four different experiments; two for which manual driving was required (with a real car and with a simulated ca) and two that involved automated driving (Unity and ROS). The comparative analysis of the obtained results showed the good performance and efficient functionality of the simulator under different conditions.

The findings from the decrease in performance that resulted from the Driver Simulator experiment will help to adjust the calibration of the simulator in future research experiments in different controlled environments, with different scenarios and more users. Additionally, driving behavior will be evaluated by using different vehicles, since the proposed simulator have a variety selection of vehicles from small cars to big buses and trucks. Last but not least, the connection of multiple simulators will provide cues related to the interaction of several road users.

ACKNOWLEDGMENT

This research work was partially supported by the Madrid Community project SEGVAUTO-TRIES (S2013-MIT-2713) and by the Spanish Government CICYT projects (TRA2015-63708-R and TRA2016-78886-C3-1-R).

References

[1] E. Commission, "Road safety in the european union," *Road Safety in the European Union: Trends, statistics and main challenges*, 2016.

[2] E. U. CARE, "Road accidents in europe: Statistics and facts," *European Commission General for Mobility and Transport*, pp. 1–5, 2017.

[3] C. Olaverri-Monreal, "Autonomous vehicles and smart mobility related technologies," *Infocommunications Journal*, vol. 8, no. 2, pp. 17–24, 2016.

[4] U. Technologies, "Unity user manual (2017.3)," *Unity3D Docs*, pp. [Last Accessed: 2018–04–30], 2018. [Online]. Available: http://docs.unity3d.com/Manual/index.html

[5] C. Biurrun, L. Serrano-Arriezu, and C. Olaverri-Monreal, "Microscopic driver-centric simulator: Linking unity3d and sumo," in *World Conference on Information Systems and Technologies*. Springer, 2017, pp. 851–860.

[6] C. Olaverri-Monreal, J. Errea-Moreno, and A. Diaz-Alvarez, "Implementation and evaluation of a traffic light assistance system in a simulation framework based on v2i communication," *Journal of Advanced Transportation*, 2018.

[7] F. Michaeler and C. Olaverri-Monreal, "3d driving simulator with vanet capabilities to assess cooperative systems: 3dsimvanet," in *Intelligent Vehicles Symposium (IV'2017)*. IEEE, 2017, pp. 999–1004.

[8] P. R. Alves, J. Gonçalves, R. J. Rossetti, E. C. Oliveira, and C. Olaverri-Monreal, "Forward collision warning systems using heads-up displays: Testing usability of two new metaphors," in *Intelligent Vehicles Symposium (IV) Workshop*, 2013, pp. 1–6.

[9] C. Olaverri-Monreal, P. Gomes, M. K. Silvéria, and M. Ferreira, "In-vehicle virtual traffic lights: a graphical user interface," in *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on*. IEEE, 2012, pp. 1–6.

[10] R. Maia, M. Silva, R. Araújo, and U. Nunes, "Electric vehicle simulator for energy consumption studies in electric mobility systems," in *Integrated and Sustainable Transportation System (FISTS), Forum on*. IEEE, 2011, pp. 227–232.

[11] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "Traci: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th communications and networking simulation symposium*. ACM, 2008, pp. 155–163.

[12] NSNAM, "ns-2: The network simulator," *NSNAM*, pp. [Last Accessed: 2018–04–30], 2018. [Online]. Available: http://nsnam.sourceforge.net/wiki/index.php/Main_Page

[13] B. Pattberg, "Institute of transportation systems - SUMO Simulation of Urban MObility," http://www.dlr.de/ts/en/desktopdefault .aspx/tabid-9883/16931_read-41000/, accessed on: 10-02-2017.

[14] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved ivc analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, 2011.

[15] BGR, "Robotic simulation scenarios with gazebo and ros," *Generation Robots*, pp. [Last Accessed: 2018–04–30], 2018. [Online]. Available: https://www.generationrobots.com/blog/en/category/robotics-tutorials/

[16] A. Brown *et al.*, "Udacity self-driving car simulator," *GitHub Repository*, pp. [Last Accessed: 2018–04–30], 2018. [Online]. Available: https://github.com/udacity/self-driving-car-sim

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

[18] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo (simulation of urban mobility): an overview," *The Third International Conference on Advances in System Simulation (SIMUL)*, 2011.

[19] Esri, "Esri cityengine — 3d modeling software for urban environments," http://www.esri.com/software/cityengine, (Accessed: 01- Jan-2016).

[20] B. Ozdodanlar, "Realistic car controller v3.1," *Bone Cracker Games*, vol. 3, pp. 1–36, 2017.

[21] A. S. Kyaw, *Unity 4.x Game AI Programming*. Packt Publishing Ltd, 2013.

[22] P. Saint-Andre, "Rfc standard 6455: The websocket protocol," RFC 6455 (Proposed Standard), Tech. Rep., 2011.

[23] M. C. Thorstensen, "Visualization of robotic sensor data with augmented reality," Master's thesis, Universitetet i Oslo, 2017.

[24] M. C. Thorstensen, "Ros bridge lib," *GitHub Repository*, pp. [Last Accessed: 2018–04–30], 2018. [Online]. Available: https://github.com/MathiasCiarlo/ROSBridgeLib

[25] P. Marin-Plaza, A. Hussein, D. Martin, and A. de la Escalera, "Complete ros-based architecture for intelligent vehicles," in *Iberian Robotics conference*. Springer, 2017, pp. 499–510.

[26] M. S. Aminian, A. Allamehzadeh, M. Mostaed, and C. Olaverri-Monreal, "Cost-efficient traffic sign detection relying on smart mobile devices," in *International Conference on Computer Aided Systems Theory*. Springer, 2017, pp. 419–426.

[27] A. Allamehzadeh, J. U. de la Parra, A. Hussein, F. Garcia, and C. Olaverri-Monreal, "Cost-efficient driver state and road conditions monitoring system for conditional automation," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 1497–1502.

[28] C. Olaverri-Monreal, R. Lorenz, F. Michaeler, G. C. Krizek, and M. Pichler, "Tailigator: Cooperative system for safety distance observance," in *International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, 2016, pp. 392–397.

[29] C. Olaverri-Monreal, G. C. Krizek, F. Michaeler, R. Lorenz, and M. Pichler, "Collaborative approach for a safe driving distance using stereoscopic image processing," *Future Generation Computer Systems*, 2018.

[30] K. Abdelgawad, J. Gausemeier, R. Dumitrescu, M. Grafe, J. Stoecklein, and J. Berssenbruegge, "Networked driving simulation: Applications, state of the art, and design considerations," *Designs*, pp. 1–17, 2017.

[31] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," *ICRA workshop on open source software*, pp. 1–5, 2009.

[32] P. Marin-Plaza, A. Hussein, D. Martin, and A. d. l. Escalera, "Global and local path planning study in a ros-based research platform for autonomous vehicles," *Journal of Advanced Transportation*, vol. 2018, 2018.

[33] P. F. Muir and C. P. Neuman, "Kinematic modeling of wheeled mobile robots," *Journal of Field Robotics*, vol. 4, no. 2, pp. 281–340, 1987.