2013 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)
June 23-26, 2013, Gold Coast, Australia

Extending Driving Simulator Capabilities Toward Hardware-in-the-Loop Testbeds and Remote Vehicle Interfaces

Kevin S. Swanson, Alexander A. Brown, Sean N. Brennan, Cynthia M. LaJambe

Abstract—**This paper describes the development of a Hardware-in-the-Loop (HIL) driving simulator designed with ROS and Gazebo. An analysis of current driving simulator software is conducted focusing on the requirements for HIL simulators versus traditional simulator implementations. Finally, the process for implementing an open-source HIL-centric driving simulator is documented, along with an analysis of system performance emphasizing metrics of system latency and strategies to minimize inter-hardware delays.**

Keywords—hardware-in-the-loop; driving simulator; Robot Operating System; Gazebo; tele-op

I. INTRODUCTION

Driving simulators have been an object of increasing focus in recent years because of their many applications including vehicle/traffic control testing, testing of driver performance and driver training under various circumstances, and development of realistic driving games [1]. This paper presents a novel and recent software with broad capabilities for multiple levels of simulation. These include rendering fully simulated driving environments, control of Hardware-in-the-Loop, and tele-operation as an interface to real vehicles and road situations.

Controlled driving environments such as test tracks and driving simulators share a common purpose of evaluating human performance with minimal driving risk. Test track facilities allow the driver to interact with the road. This provides a higher "sense of realism" although reproducibility of a traffic driving environment is reduced; in contrast, driving simulators allow high reproducibility of driving scenarios at the expense of realism [2]. This similarity between test tracks and driving simulators begs the question: what tools are needed to facilitate a combination of both experiences to provide a continuum of choices between a realistic experience and a replicable result?

To obtain realism, one assumption is that the simulator driver perceives risk similar to that of driving an actual vehicle. According to Risk Homeostasis Theory, drivers may not behave in a simulator as they do on the road [3] unless exposed to particularly risky driving behaviors from other drivers rather than simply ambient traffic [4]. One means of improving a driver's perceived realism is to use simulators as interfaces for humans to drive real vehicles, with clear consequences in terms of external safety and vehicle motion. Preliminary testing on rolling roadway simulators suggest that drivers are far more cautious driving actual systems knowing that physical damage will take place in the presence of poor driving choice, even if their own safety isn't involved [5][6][7], a result supported by Risk Homeostasis theory [3].

Recently, simulators have moved beyond basic driver testing and training, and now many simulators are used for specialized purposes such as controlling hardware, allowing multiple drivers-in-the-loop, or interacting with real vehicles. The tele-operated vehicle has been described as "…a fully-mobile, physical proximity (a 'real-world avatar') for the operator" [8]. Tele-operation development has accelerated in recent years due to video and computing compression improvements, and the use of high quality Light Detection and Ranging (LiDAR) optical remote sensing technology [9]. Despite these simulation advances, automotive research using live Hardware-in-the-Loop (HIL) in combination with Human-in-the-Loop testing, sometimes called H^2iL, has been underutilized [10].

Since all simulators are HIL testing systems, with the human being the "hardware" component, there is not a clear dividing line between human-HIL simulators and fully HIL simulators. In this paper, the mention of a HIL simulator refers to a system where both the driver and another vehicle hardware component are interacting with each other. According to Fathy, et al. [11], the advantages of a HIL system include higher fidelity, faster simulation speed than purely virtual systems, and greater comprehensiveness than purely physical systems.

To achieve these advantages, the system must bring together many key factors including: real-time operating systems, high-fidelity dynamic models, high bandwidth networking, and low-latency hardware/software integration. The remainder of the paper details the design of a high-fidelity, highly modular HIL simulator design with the open source software ROS and Gazebo. The use of this environment to interface real-world scenes, show data "live" to a human driver during simulator usage, and to interact with remote vehicles, is also discussed and quantified.

II. REVIEW OF EXISTING SIMULATORS

This section examines the current status of driving simulator technology focusing on the use of driving simulators as HIL systems designed for interacting with real-world data, either live or offline. The view of simulators here is more of a "virtual cockpit" to an actual and ongoing driving situation, rather than as a human testing interface. Thus, the emphasis of this review is not on the insights offered by human testing, but rather on the technical

challenges of connecting human drivers to stored or live data available from real-world measurements.

A. Professional Driving Simulators

Many vehicle testing agencies have developed high-capability simulators that are themselves wonderful demonstrations of HIL testing. The most notable federally-funded driving simulator, the National Advanced Driving Simulator in Iowa, is comprised of a 13 degree of freedom motion base that encloses a car, SUV, or truck cab inside of a 24-foot dome and projects the simulator on eight LCD projectors 360 degrees around the driver [12].

Fig. 1. Interior (left) and motion platform (right) of Toyota's CarSim DS based driving simulator [13]

Most large automotive companies have also developed driving simulators to aid in the design of their vehicles. For example, Toyota has a driving simulator seen in Fig. 1 similar to NADS at Toyota's Higashifuji Technical Center in Japan with a high degree of freedom motion platform and 360 degree view in a large hexapod enclosure [13]. These and similar large simulator systems have a significant amount of capital, as they seek to develop a driving experience as close to reality as possible with current technology. Unfortunately, these are extremely costly systems that require extensive support from dedicated engineers. Thus, these interfaces are largely beyond the range of casual academic HIL simulator studies.

B. Commercial Software

There are many commercially-developed driving simulator platforms available on the market targeting cost-effective deployment in research labs. These platforms are very popular because they are developed and tested to a high level of accuracy, and can therefore be trusted for high fidelity tests. Unfortunately, much of this software can still be very expensive both to install and to maintain.

One of the most popular software platforms for driving simulators is CarSim/TruckSim DS®. The Toyota simulator mentioned above uses CarSim DS for the driving simulator software [14]. Additionally, there are many researchers who use MATLAB® and Simulink® compiled with C++ to run their vehicle simulations [15][16][17]. Another available platform is Realtime Technologies®, a driving simulator platform very similar to CarSim DS [18]. As a low-cost solution to a research project [19], a driving simulator video game, rFactor® is used to provide accurate vehicle dynamics and a small amount of customizability.

Additionally, some commercial software platforms are specifically designed for HIL implementation. PreScan® is built for sensor simulation and is easily integrated with many existing platforms like CarSim [20]. CarMaker® is an extremely fast vehicle simulation platform with sophisticated vehicle models that is also able to integrate with programs like CarSim [21]. Table I provides an overview of these commercial platforms.

TABLE I. COMMERCIAL SOFTWARE OVERVIEW

Platform	Advantages	Disadvantages
CarSim/ TruckSim	- Very accurate vehicle dynamics - Very high fidelity - Ability to control motion base	- Expensive to buy and maintain - Difficult to customize (eg. importing tracks) - Difficult to interface with hardware (eg. LiDAR)
MATLAB/ Simulink	- Ability to simulate multiple-drivers-in-the-loop (to some extent) - Ability to control motion base	- Lower fidelity in general, or too slow for real-time - Difficult to customize (eg. importing tracks) - Difficult to interface with hardware (eg. LiDAR)
Realtime Technologies	- Accurate vehicle dynamics - High fidelity - Ability to control motion base	- Expensive to buy and maintain - Difficult to customize (eg. importing tracks) - Difficult to interface with hardware (eg. LiDAR)
rFactor	- Accurate vehicle dynamics - Small amount of customizability (eg. tracks)	- Not designed for simulator use - Difficult to interface with hardware (eg. LiDAR)
PreScan	- Full HIL simulations - Integration with many software platoforms like CarSim	- Designed for sensor integration over vehicle simulation - Low customizability from closed source - Expensive base and additional software
CarMaker	- Full HIL simulations - Integration with CarSim - Built in vehicle dynamics	- Expensive base and additional software - Low customizability from closed source

C. Open-Source Software

Open-source software has the benefit of no cost and a high level of customizability, but with a severe learning curve and little or no product support in the case of debugging.

TABLE II. OPEN-SOURCE SOFTWARE OVERVIEW

Platform	Advantages	Disadvantages
TORCS	- Accurate vehicle dynamics - Highly customizable	- Difficulty with advanced data acquisition
Delta3D	- Creates high fidelity 3D environments	- Not built for accurate vehicle dynamics
GrooveSim	- Ability to input real world maps - Ability to communicate with real vehicles	- Not built for driver-in-the-loop operation

Examples of researchers choosing this approach include Benoit et al. [22], who uses an open-source program called TORCS, Darken et al. [23] who use Delta3D Gaming Engine, and Mangharam et al. [24] who use GrooveSim which is software that is not technically a driving simulator, but has beneficial capabilities to this effect. Table II contains an overview of these platforms.

III. Requirements for HIL-Type Driving Simulators

According to Kuhl, there are three factors with which a traditional driving simulator must excel to achieve a high degree of fidelity [25]. These factors are 1) the control interfaces through which the driver operates the vehicle, 2) the dynamics of the vehicle in response to driver inputs, and 3) the environment with which the driver interacts. However, for use in an HIL environment, a driving simulator system must meet several additional requirements that both expand on the capabilities of traditional simulators and create new capabilities specific to HIL simulators. An ideal testing scenario would have the following capabilities:

1) Import real world environments: Most simulator environments allow the user to manually construct real-world environments. In general, this is a time-consuming and technical process. One can simplify this process by using CAD imports of world environments, 3rd-party reconstructions of world environments (Google Earth), vehicle-based road measurements, and/or spline-fitting of sparse world measurements.

2) Interface with advanced sensors: The best traditional and HIL driving simulators enable advanced sensors/actuators to be input to the system readily. This includes by-wire steering systems, RADAR, LiDAR, camera-detection systems, eye tracking, EEG, and videos of driver behavior. As an example, three researchers have used both commercial and open source software for eye tracking ability while testing subjects [19][20][26].

3) "Live" scene updates: "Live" recording and playback of a 3-D environment includes ability to update the presentation of the real-world geometry of the environment "on the fly", reflecting incoming measurements. Such capability allows changes in vehicle dynamics driven by real-world measurements; for example, friction measurements on a remote car could inform live updates of the simulated environment.

4) Interactions between vehicles and driver: In traditional simulators, the ability of multiple drivers to interact is sought to allow these users to be tested and trained simultaneously, or to test (and train) interactions between users. For example, law-enforcement personnel regularly train on simulators that link their environments together for tactical training [27]. Additionally, Güvenç and Kural [15] use multiple-drivers-in-the-loop in adaptive cruise control tests using MATLAB and Simulink for the simulations. For HIL simulators, the necessity for multiple drivers to interact arises because of the need to maintain fail-safes, particularly in the remote operation of vehicles, such that algorithms and/or other humans can recover operation of the vehicle in the case of communication failures.

5) Maintain real-time latency: HIL simulators must generally maintain real-time latency to external environmental inputs. While some driving simulators allow remote inputs with latencies imperceptible to human drivers, if the latencies are too large the simulator may not be suitable for sensor-to-actuator control-loop interactions to be correctly represented. To maintain such ability to co-simulate driver-input-models alongside HIL remote-control vehicle operation, generally millisecond level latencies (or less) are required. Latencies between input and user displays are usually more tolerable than latency differences between inputs or between displays.

IV. Proposed Driving Simulator Platform

In this section we focus on the design of a software system specific to HIL driving simulator usage, while maintaining capability more traditionally required by driving simulators. The platform chosen to build this driving simulator is based on the Robot Operating System (ROS) and Gazebo. ROS is a Linux-based open-source platform that provides libraries and tools to help designers create robot applications [28]. ROS primarily operates as a group of individual "nodes" that subscribe to, process, and publish information. Fig. 2 graphically displays the connections of hardware and ROS nodes during a basic vehicle simulation. This "middleware" aspect of ROS provides a highly modular design, yet can also be challenging to first learn.

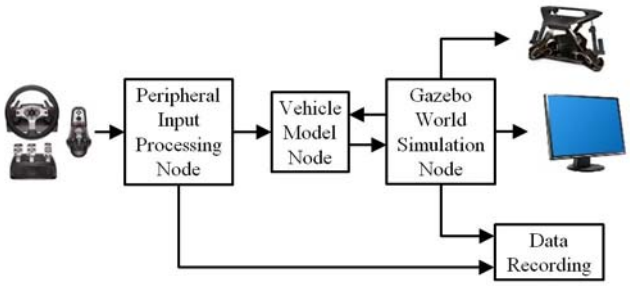

Fig. 2. Connection of nodes in ROS during basic vehicle simulation

While ROS was chosen to perform the base control and communication, Gazebo was chosen to create the visual simulation in a physics-based environment. Gazebo is a multi-robot simulator capable of representing robots, sensors, and objects. This software was designed to generate realistic sensor feedback and physical interactions between objects [29]. The choice of using ROS and Gazebo is intentional: ROS already includes a 3D visualization tool (RVIZ), but Gazebo was necessary because of its ability to model accurate physics. Additionally, ROS was designed to be integrated with Gazebo, so a significant amount of support exists to integrate these two software packages.

This project is not the first time that ROS and Gazebo have been used to create driving simulations. In designing their automated vehicle, Fernandes et al. [30] uses ROS and Gazebo to model their physical system; however, very little

information is given on the simulation itself. The remainder of this paper will be used to describe the capabilities of the ROS and Gazebo platforms in the creation of a driving simulator.

A. Basic Capabilities of ROS and Gazebo

1) Inputs: In its current state, the simulator uses a Logitech G25 Racing Wheel for steering, throttle, and brake. This peripheral is connected to the simulator PC via USB. ROS includes a package that drives any Linux-compatible joystick. This package contains joy_node which publishes a "Joy" message containing the states of all buttons and axes of the joystick [28]. Each of these states can be subscribed to, processed, and published as inputs to the vehicle in Gazebo through subsequent nodes.

In many cases it is favorable to control the vehicle using more realistic peripherals such as an actual steering wheel, throttle pedal, and brake pedal in a real vehicle cab. This can also be done in ROS by inputting information from sensors such as encoders or potentiometers into the computer via serial or TCP/IP connection, CAN or serial inputs/outputs, etc. As with using a joystick, the inputs are subscribed to, processed, and published as an input to the vehicle. In this way, virtually any physical object could be used as a peripheral.

2) Simulated Dynamics: Modeling vehicle dynamics is completed in Gazebo with the Open Dynamics Engine (ODE) physics engine [31]. With ODE, the user is able to create a series of bodies with individual properties called a "link" and specified connections and physical interactions with other bodies called a "joint." In the driving simulator, the simplest model is created by links for the chassis, suspension, and wheels as rigid bodies, and the joints between them. The suspension is assigned linear motion with respect to the body and can be assigned spring and damping constants to model real suspension. The wheels are given a rotational relationship with the suspension rods. In order to drive the car a torque is applied to the wheels as output by the ROS node that converts a peripheral input into a torque command. Again, in the simplest implementation this can be done by simply applying a gain to the joystick commands and outputting that as torque, but in the vehicle dynamics model ROS is able to process the throttle and brake inputs through a full driveline model to send an accurate torque output to the wheels.

3) Outputs: Gazebo is capable of outputting states of all bodies in the physics environment which is important for any experimentation outside of simply driving a vehicle. In addition to monitoring vehicle states for experimentation, these states can be used to control hardware like a motion base and force-feedback steering. In the case of this simulator, cab actuation is controlled by the MOOG 6DOF2000E motion base. While ROS and Gazebo are capable of controlling this motion base, the intricacies of accurately outputting vehicle motion is not within the scope of this paper. Force feedback is done with pre-equipped force feedback in the Logitech wheel, but it can also be achieved by sending an output to a separate motor, for example one on the steering column in a full vehicle cab application.

4) Environment: Creating the visual and physical environment was accomplished using Google SketchUp [32]. Within Google SketchUp there is a function to "Add Location…" which will import an image from Google Maps including the texture of the landscape. In this study, this image was saved as a ".dae" file and imported into Gazebo as both the physical and visual environment. Fig. 3 shows a terrain constructed by importing images from Google Maps.

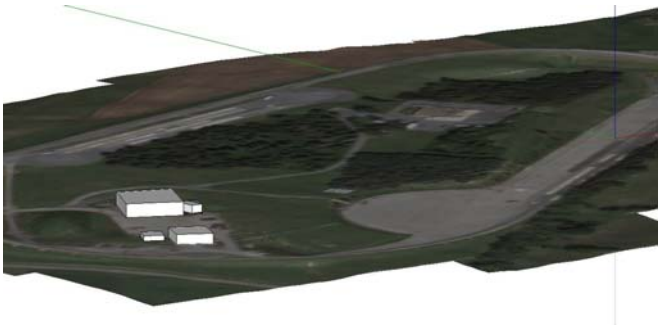

Fig. 3. Terrain imported via Google Maps in Google SketchUp

An additional benefit to using Gazebo is that it readily enables multiple environments to be loaded on top of each other. This allows the operator to quickly and easily change road elements, for example removing a single road sign to adding an entire additional road surface. The ability of Gazebo to load the physical and visual environment separately also provides a diversity of scenes. For example, an environment could be imported from Google Maps and used for the physical surface, but a snowy visual surface could imported to simulate the visual difficulties of winter driving. Gazebo also includes some preinstalled "skins" that increase the realism of the scene, for example adding clouds to the sky.

Fig. 4 demonstrates the ability to customize scenes in Gazebo. The scene includes a terrain imported from Google SketchUp, augmented with a speed limit sign as an additional map layer, along with the preloaded cloud skin for the sky. For reference, the rendered image is shown next to an actual picture of the same scene.

Fig. 4. 3rd person view of imported terrain rendered in Gazebo next to the actual scene

Additionally, Gazebo includes the ability to change friction for different bodies in the environment. This provides the ability to give different surfaces like pavement

and grass varied friction properties, which will more realistically simulate a situation such as driving off the road.

B. *Advanced Capabilties of ROS and Gazebo*

Because both software packages are fundamentally designed as "middleware", ROS and Gazebo provide the ability to create some unique features in the driving simulator. These capabilities come from a combination of the modular nature of ROS and the built-in physics platform of Gazebo. One of the greatest benefits of the ROS platform is the ability to easily communicate across many devices via TCP/IP connection, allowing different CPUs to control different functions. Some specific benefits of this include:

- The ability to display large numbers of camera views across multiple CPUs.
- The ability to separate CPUs for specific purposes, for example a dedicated DAQ CPU.
- Allowing multiple-drivers-in-the-loop.
- The ability to interface with real vehicles.

We have observed that these capabilities, inherent in Gazebo, are often missing in other simulation software packages. Gazebo includes these because it was designed to simulate sensors in a closed-loop testing environment. This provides the ability to integrate user-defined surfaces with LiDAR point cloud data. Further, a vehicle driving remotely with RADAR, LiDAR, etc can update the scene using live measurements streaming into the simulated environment. Having accurate LiDAR scans can also greatly increase the accuracy of the physical road surface, further increasing the realism of the simulation.

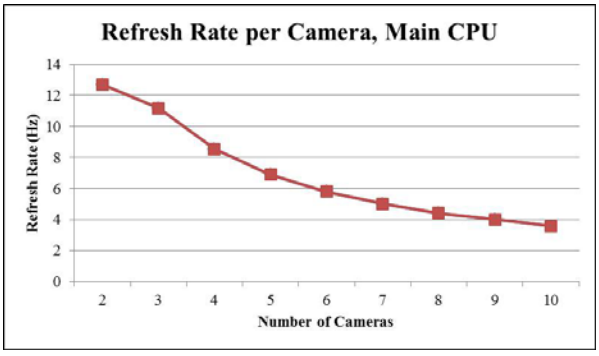

Fig. 5. Refresh rate of scene renderings vs number of scenes rendered on the main CPU

C. *Quantification of Simulator Performance*

A primary concern for designing driving simulators, especially in HIL applications, is to ensure real-time performance. Not only will lag disrupt the realism of the driving environment, but it will also lessen the fidelity of the hardware integrated with the system. A series of tests examining the performance of the system was done using an Asus Essentio Desktop with an Intel® Core i7 Processor, 8 GB RAM, and a NVIDIA GT 530 graphics card as the main simulation computer. A primary purpose of these tests was to determine the causes of latency in the system, so the number of cameras, vehicles, nodes, and linked CPUs were chosen as the variables. For each test, the camera resolution is 640x480 and the desired camera update rate is set to 20 Hz. The actual refresh rate is a calculated average over 500 samples.

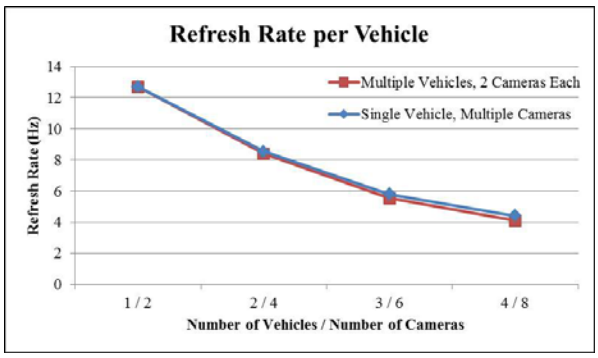

Fig. 6. Refresh rate of cameras per number of vehcles running on the main CPU, compared to number of cameras for one vehicle

The first test examines the effect of the number of rendered scenes on latency by varying the number of virtual cameras running on the main computer. The results are shown in Fig. 5. This clearly shows that the number of rendered scenes has a large effect on latency in the system. Additionally, the refresh rate is not as high as some professional simulators, but these simulations are intentionally being run on a sub-standard computer to isolate latency effects and trends.

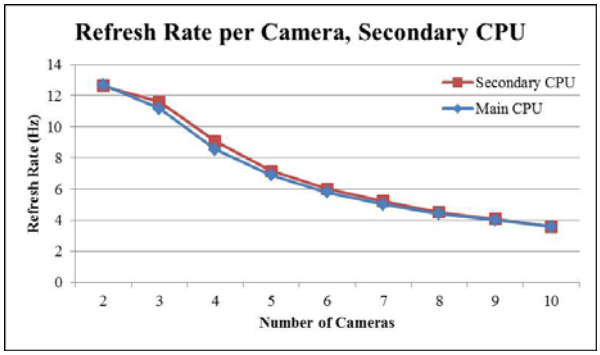

Fig. 7. Refresh rate of cameras per number of cameras running on the secondary CPU, compared to number of cameras on the main CPU

The second test examines how latency increases with increasing number of human-driven simulators connected together, assuming each vehicle is one joy node and displays two scenes. For comparison, the results are plotted next to the same number of cameras being displayed by a single vehicle. The results, displayed in Fig. 6, show that the number of vehicles in the scene and nodes running has negligible effect on latency compared to the number of cameras, e.g. that additional scene rendering requires far more computation than adding additional driver inputs.

The final test examines the effect of remotely linking the driver simulation to other computers, specifically by adding additional displays on a secondary computer connected via

TCP/IP. The secondary computer is a Gateway MT3705 laptop with an Intel® Pentium T2060 processor, 2 GB RAM, and an ATI Radeon Xpress 200M graphics card. The results in Fig. 7 show that, while there is a slight increase in performance over running displays on a single computer, the effect is minimal. This secondary computer was again chosen to have very low processing power. In future tests, it is desirable to examine the effect of both a higher power CPUs along with that of more than two connected CPUs.

V. CONCLUSIONS

This paper has presented a novel driving simulator platform with the focus on HIL applications. To maintain high fidelity, it is necessary for the system to do the following: 1) Meet the requirements for traditional driving simulators by creating an accurate feeling of the world through basic inputs and outputs. 2) Meet the requirements shared by traditional and HIL simulators by incorporating real world data including real-world maps and advanced sensor data (RADAR, LiDAR, etc.). 3) Meet the requirements of a HIL simulator by allowing multiple-drivers-in-the-loop and real time data recording and playback. The software examined in this study based on ROS/Gazebo appears to meet all of these requirements to some degree. The results indicate that interconnection of different inputs and computers together has little effect on increasing latency, whereas adding additional scene renderings has a large effect. Work is ongoing on the effect of CPU choices on decreasing latency and increasing system fidelity.

REFERENCES

[1] P. Hancock and T. Sheridan. "The future of driving simulators" in *Handbook of Driving Simulation for Engineering, Medicine, and Psychology*, D. Fisher, M. Rizzo, J. Caird, and J. Lee, Ed. Boca Raton, FL: CRC Press, 2011, 4-1 – 4-11.

[2] J. Olstam and S. Espié. "Combination of autonomous and controlled vehicles in driving simulator scenarios." *International Conference Road Safety and Simulation*, 2007, pp. 1-6.

[3] G. Wilde. "Risk homeostasis theory: An overview." *Injury Prevention*, vol. 4, pp. 89-91, 1998.

[4] S. Wright, N. Ward, and A. Cohn. "Enhanced presence in driving simulators using autonomous traffic with virtual personalities." *Presence*, vol. 11, pp. 578-590, 2002.

[5] S. Brennan. "Modeling and control issues associated with scaled vehicles." M.S. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.

[6] S. Brennan, "On size and control: The use of dimensional analysis in controller design." Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2002.

[7] S. Lapapong, "Vehicle similitude modeling and validation of the Pennsylvania State University rolling roadway simulator." M.S. thesis, The Pennsylvania State University, University Park, PA, 2007.

[8] T. Fong and C. Thorpe. "Vehicle Teleoperation Interfaces." *Autonomous Robots*, vol. 11, pp. 9-18, 2001.

[9] A. Kelly et al. "Real-time photorealistic virtualized reality interface for remote mobile robot control." *The International Journal of Robotics Research*, vol. 30, pp. 384-404, 2010.

[10] P. Onesti. "Combination of Human- and Hardware-in-the-Loop Testing Reduces Development Time." *ATZautotechnology*, vol. 9, pp. 32-37, 2009.

[11] H. Fathy, Z. Filipi, J. Hagena, and J. Stein. "Review of hardware-in-the-loop simulation and its prospects in the automotie area." *Proc. of SPIE*, 2006, pp. 62280E-1-62280E-20

[12] "The National Advanced Driving Simulator, The University of Iowa." Internet: http://www.nads-sc.uiowa.edu/, Feb. 26, 2013 [Feb. 26, 2013].

[13] P. McNamara. "Inside Toyota's $15m driving simulator." Internet: http://www.carmagazine.co.uk/Community/Car-Magazines-Blogs/Phil-McNamara-Blog/Inside-Toyotas-15m-driving-simulator/, Feb. 26, 2013 [Feb. 26, 2013].

[14] "Driving simulators (CarSim and TruckSim)." Internet: http://www.carsim.com/products/ds/index.php, Feb. 26, 2013 [Feb. 26, 2013].

[15] B. Güvenç and E. Kural. "Adaptive cruise control simulator: A low-cost, multiple-driver-in-the-loop simulator." *IEEE Control Systems Magazine*, pp. 42-55, June 2006.

[16] L. Nehaoua, A. Amouri, and H. Arioui. "Classical and adaptive washout comparison for a low cost driving simulator." *Proc. of the 13th Mediterranean Conference on Control and Automation*, 2005, pp. 586-591.

[17] A. Huang and C. Chen, "A low-cost driving simulator for full vehicle dynamics simulation." *IEEE Transaction on Vehicular Technology*, vol. 52, pp. 162-172, Jan. 2003.

[18] "On-road simulators." Internet: http://www.simcreator.com/simulators/onroadsim.htm, Feb. 13, 2013 [Feb. 26, 2013]

[19] G. Weinberg and B. Harsham. "Developing a low-cost driving simultor for the evaluation of in-vehicle technologies." *Proc. of the 1st International Conference on Automotive User Interfaces and Interactive Vehicle Applications*, 2009, pp. 51-54.

[20] "PreScan: ADAS development." Internet: http://www.tass-safe.com/en/products/prescan, April. 4, 2013 [April 4, 2013]

[21] "CarMaker: Your open integration and testing platform for virtual test driving." Internet: http://www.ipg.de/index.php?id=carmaker, April. 4, 2013 [April 4, 2013]

[22] A. Benoit, et al. "Multimodal signal processing and interation for a driving simulator: Component-based architecture." *Journal of Multimodal User Interfaces*, vol. 1, pp. 49-58, March 2007.

[23] "R. Darken, P. McDowell, and E. Johnson. "Projects in VR: The Delta3D open source game engine." *Computer Graphics and Applications, IEEE*, pp. 10-12, May/June 2005.

[24] R. Mangharam, D. Weller, D. Stancil, R. Rajkumar, and J. Parikh. "GrooveSim: A topograpy-accurate simulator for geographic routing in vehicular networks." *Proc. of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks*, 2005, pp. 59-68.

[25] J. Kuhl, D. Evans, Y. Papelis, R. Romano, and G. Watson. "The Iowa Driving Simulator: An immersive research environment." *Computer Magazine*, pp. 35-41, July 1995.

[26] J. Lawson, A. Al-Akkad, J. Vanderdonckt, and B. Macq. "An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components." *Proc. of the 1st ACM Symposium on Engineering Interactive Computing Systems*, 2009, pp. 245-254.

[27] A. Gregory, "FLECT augments training with technology." Internet: http://www.fletc.gov/news/press-clips/fletc-augments-training-with-technology.html, Feb. 26, 2013 [Feb. 26, 2013].

[28] "ROS Wiki." Internet: http://www.ros.org/wiki/, Feb. 26, 2013 [Feb. 26, 2013].

[29] "Gazebo." Internet: http://www.gazebosim.org/, Feb. 26, 2013 [Feb. 26, 2013].

[30] L. Fernandes, J. Souza, P. Shinzato, and G. Pessin. "Intelligent robotic car for autonomous navigation: Platform and system architecture." *2012 Second Brazilian Conference on Critical Embedded Systems*, 2012, pp. 12-17.

[31] "ODE Wiki." Internet: http://ode-wiki.org/wiki, June 29, 2012 [Feb. 26, 2013].

[32] "Trimble SketchUp." Internet: http://www.sketchup.com/, Nov. 13, 2012 [Feb. 26, 2013].