Integrating Virtual and Augmented Reality Based Testing into the Development of Autonomous Vehicles

James F. Leathrum, Jr., Yuzhong Shen, Roland R. Mielke, Nathan Gonda
Department of Modeling, Simulation and Visualization Engineering
Old Dominion University
Norfolk, Virginia
jleathru@odu.edu, yshen@odu.edu, rmielke@odu.edu, ngond002@odu.edu

ABSTRACT

Test and evaluation of autonomous vehicle software using classical testing techniques is difficult because the software often is complex and designed to exceedingly high safety standards, making thorough testing infeasible. In addition, autonomous software generally contains non-deterministic components and statistical algorithms, making test results non-repeatable and difficult to interpret. Current test and evaluation approaches for autonomous systems place the vehicles in various operating scenarios to observe their behavior. However, this means that extensive testing must be delayed until a vehicle prototype is ready to operate in a physical environment. The purpose of this paper is to examine the benefits of utilizing virtual reality (VR) and augmented reality (AR) to facilitate full system testing throughout all phases of the design and development process. Early in the design process, system testing is conducted with virtual system components. As the design progresses, real system components are augmented with virtual components until final testing is possible using all real components. The approach is based on a new software framework and use of modeling and simulation to seamlessly incorporate VR and AR components with real components. The paper describes the software framework and explores requirements and constraints associated with this testing approach.

ABOUT THE AUTHORS

James Leathrum is an Associate Professor in the Department of Modeling, Simulation and Visualization Engineering at Old Dominion University. He earned the Ph.D. in Electrical Engineering from Duke University. His research interests include simulation software design, distributed simulation, and simulation education. His e-mail address is jleathru@odu.edu.

Yuzhong Shen is an Associate Professor in the Department of Modeling, Simulation and Visualization Engineering at Old Dominion University. He earned the Ph.D. in Electrical Engineering from the University of Delaware. His research interests include visualization and computer graphics, modeling and simulation, and signal and image processing. His e-mail address is yshen@odu.edu.

Roland Mielke is a University Professor in the Department of Modeling, Simulation and Visualization Engineering at Old Dominion University. He earned the Ph.D. in Electrical and Computer Engineering for the University of Wisconsin-Madison. His research interests include system theory, the theory and application of simulation, and simulation education. His e-mail address is rmielke@odu.edu.

Nathan Gonda is a graduate student working toward a Master Degree in Modeling and Simulation Engineering. He has earned a Bachelor Degree in Modeling and Simulation Engineering from Old Dominion University and has experience in computer programming and simulation software design for about 5 years. His research interests are in computer game design, virtual reality, and cybersecurity. His email address is ngond002@odu.edu.

Integrating Virtual and Augmented Reality Based Testing into the Development of Autonomous Vehicles

James F. Leathrum, Jr., Yuzhong Shen, Roland R. Mielke, Nathan Gonda
Department of Modeling, Simulation and Visualization Engineering
Old Dominion University
Norfolk, Virginia
jleathru@odu.edu, yshen@odu.edu, rmielke@odu.edu, ngond002@odu.edu

INTRODUCTION

Autonomous vehicles, whether air, land, or sea, clearly present a major business opportunity in the near term. Goldman Sachs Research (2015) predicted a $100B market just for drones from 2016-2020. This includes $70B for military applications, $17B for retail, and $13B for commercial/civil). The top commercial/civil sectors include construction, agriculture, insurance claims, and public safety (police, fire, coast guard). Example applications of autonomous vehicles include driverless cars and drone package delivery systems.

Autonomous vehicles provide numerous benefits over systems requiring human control. Driverless cars could be safer since the system does not suffer from lack of attention. Roads could handle more traffic as the distance between vehicles could reduce due to faster reaction times. They could be cheaper to operate, potentially more fuel efficient and resulting in lower insurance rates. Autonomous vehicles can also handle situations that classically place a human in danger. Even remote controlled vehicles may require autonomous capability due to communication delays such as inherent with a Mars rover.

However, autonomous vehicles present new issues. Acceptance of the autonomous systems is of critical concern as people may not trust the systems to operate correctly. Related, the ethical behavior of the systems is crucial when the autonomy is making life critical decisions. This is particularly important for the military where the system needs to interpret rules of engagement. But it is also important for systems such as driverless cars where the vehicle may be placed in a situation where it must select the lesser of two evils.

Despite the obvious benefits of autonomous vehicles, the development, in particular the testing, of autonomous vehicles is not in a mature state. Classical software techniques often are not applicable as it may not be possible to determine why the software made a decision. This results in a requirement for a higher level of black box testing. But it may be difficult to test subcomponents of the system in the absence of the complete system and the ability to perceive the environment. This presents the opportunity for system and environment simulations to drive the black box testing, providing the stimuli to the subcomponents to allow the observation of their behavior.

To provide realistic stimuli for subsystem testing, virtual reality (VR) and augmented reality (AR) can gradually increase the resolution of the stimuli until real world testing is possible. It also presents the opportunity to test early in the design and development system, known to reduce development costs. To achieve this, it is desirable to have clearly identified points to which virtual and augmented reality capabilities can be developed. To do this, this work utilizes a common autonomous system model and then a software architecture to support injecting testing stimuli to the autonomous software, gradually increasing the realism of the vehicle being operated and the environment in which it operates. A continuum defining the range of VR and AR operations is selected and how they can be utilized is described.

The paper is composed of six sections. The first describes the autonomous system model utilized. The second presents related work, broken down to testing of autonomous systems and the use of VR and AR in autonomous systems. The third section presents how VR and AR can be applied across the continuum from a totally virtual world to a totally real world. System requirements necessary to support introducing VR and AR to support generalized testing is presented. The benefits of utilizing VR and AR for testing are then enumerated followed by conclusions.

AUTONOMOUS SYSTEM MODEL

In this section, a model for an autonomous control system is described to identify key features of the system when developing and testing autonomous software. It is important to understand these features and how they interact before placing the system within a virtual or augmented reality setting. Control systems require the ability to utilize various sensors to gain information about the external environment. Control systems must also be able to interface with a physical system's actuators to instruct the system to act. Finally, control systems must be robust enough to adapt to changes in the environment, maintaining consistent feedback and behaving sensibly to a wide variety of possible situations (Brooks 1985). To this end, the purpose of the control system is to generate a plan based on knowledge of the external environment and execute the plan based on actions made available by hardware actuators.

The modeling of autonomous control systems for mobile robots has an extensive history of research and development. One approach is to model the control system using a pipeline of functional modules – sense, plan and act (Gat 1998) as illustrated by Figure 1. It is composed, at a high-level, of the autonomous software, the hardware, and the external environment. The hardware can be broken down into sensors, actuators, and vehicle dynamics/state information. The vehicle dynamics and state information include physical attributes of the vehicle such as velocity, orientation, and fuel level. The software can be further decomposed into the functional modules that generate and execute the plan for the system and a world representation, an internal representation of the external environment and internal vehicle state. The world representation could include a panoramic view of distances to boundaries, sets of recognized objects and their computed attributes, or a map of the environment based on past experiences of the robot. The modules are:

- Sense – Computes a perception of the environment based on incoming raw data from the hardware sensors. This involves mapping the raw sensor data to the world representation.
- Plan – Generates a plan composed of actions based on the system's current world representation, operational goals, and past experiences.
- Act – Executes the plan by converting actions to control signals to send to the actuators.

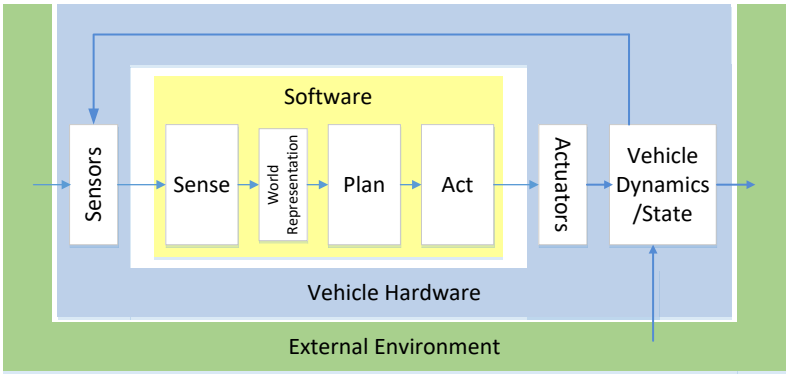


Figure 1. Autonomous System Model.

This model will be the basis for system testing. The focus in this paper is the testing of the autonomous software as found in the sense and plan modules. This involves testing each module in isolation as well as their integration with the complete system. However, similar testing can be done for the act module and the hardware components.

RELATED WORK

In this section, we conduct a brief literature review with a focus on three objectives. The first objective is to explain why traditional software testing and evaluation techniques often are insufficient for autonomous software. The second objective is to identify new test and evaluation procedures that have been developed especially for autonomous software. Finally, the third objective is to explain how virtual and augmented reality techniques can be used to positively impact both the design and testing of autonomous software.

Testing of Autonomous Systems

Three recent papers explore the challenges of testing autonomous software. In (Koopmann and Wagner, 2016), testing and evaluation of autonomous software for unmanned land vehicles is discussed. In (Menzies and Pecheur, 2005),

challenges in verifying and validating artificial intelligence software for use in autonomous vehicles for conducting deep space missions is explored. And in (Schumann and Visser, 2006), the results of a survey of NASA autonomy experts and software engineers to identify the challenges of verifying and validating autonomy software for applications in unmanned air vehicles are reported. All three papers identify very similar reasons why testing and evaluation of autonomous software is challenging. First, in fully autonomous vehicles, there is no human backup to address faults, malfunctions, and unexpected operating conditions. The autonomy system must assume the role of the primary exception handler. Thus, the autonomy software must have significant additional complexity to address all potential contingencies, making testing more difficult. Second, autonomous software often utilizes non-deterministic components and statistical algorithms. The planning component of autonomous software often is based on ranking the performance of randomly generated alternatives. Additionally, common sensing algorithms are based on stochastic models for noise resulting in probabilistic test results. This makes it difficult to evaluate the results of testing because there is no uniquely correct result for a given test scenario and the tests are non-repeatable. Third, autonomy software for unmanned vehicles must meet extremely high standards for safety. A failure of the software could result in the destruction of property and loss of life. Thus, the software system must be tested extensively to demonstrate that failure rates do not exceed an acceptable safety threshold. Such vehicle testing is time consuming and expensive; often it simply is not feasible to conduct enough tests with the physical vehicle to ensure desired safety levels.

Several papers have suggested approaches for enhancing the capability to test and evaluate autonomous software. In (Brat and Denney, 2006), a modular compositional approach to designing autonomous software is proposed. Using this approach, adaptation to different applications and features is done incrementally. Then a testing and validation methodology that limits certification of new adaptations to components and relations that are modified is proposed. In (Mullins and Stankiewicz, 2017), a procedure to identify test scenarios that are close to performance boundaries in the configuration space is proposed. A performance boundary is a location in the configuration space where large changes in performance are observed for small changes in system input. This approach in some cases can greatly reduce the amount of testing that must be conducted. In (Hodicky, 2015), it is suggested that modeling and simulation should play a larger role in integrating autonomous systems into the operational field. It is conjectured that testing autonomous software in physical and synthetic domains using modeling and simulation holds the potential to greatly reduce the cost of autonomous system deployment.

VR and AR Applied to Autonomous Vehicles

In (Milgram and Takemura, 1994), augmented reality is described as being a member of a larger class of mixed reality displays. Mixed reality displays are described as existing in a *reality-virtuality continuum*. This continuum is bounded on one end by *reality* and on the other end by *virtuality*. In between these end points live a continuum of mixed reality displays that includes *augmented reality* displays and *augmented virtuality* displays. An augmented reality display consists of a display of mostly real objects but augmented by one or more virtual objects. An augmented virtuality display consists of a display of mostly virtual objects but augmented by one or more real objects. The application of this mixed reality perspective to the design, testing, and evaluation of autonomous systems quickly becomes apparent. During initial phases of design, virtual models of autonomy strategies, vehicle components, and vehicle environment can be utilized. As the design continues, a few system components are prototyped in software or hardware and used to augment the virtual system representation. As design and prototyping continue, a majority of system components are realized in software and hardware and are augmented by virtual components whose realization are not yet complete. In the final stages of design and testing, the autonomy software and vehicle hardware are complete and tested in the real environment. An example of using this approach applied to the design and testing of autonomous underwater vehicles is described in (Davis and Lane, 2010). A virtual reality framework is developed to model the environment and interfaced to physical vehicle hardware and software via an Ethernet-based communication network. This system is used primarily for vehicle testing and evaluation.

The purpose of this paper is to present a more detailed look at the development of a system architecture that incorporates the use of augmented and virtual reality for designing and testing autonomous vehicles.

APPLYING VR AND AR TO TESTING DURING DESIGN AND DEVELOPMENT

Attempting to promote hardware/software codesign and development, a simulation-based approach is presented, relying on VR and AR to supply external stimuli in the absence of available real stimuli early in the process. To

achieve this, there is a parallel development process for a simulation of the physical vehicle and for a virtual environment in which the vehicle can operate. Given initial requirements for the physical vehicle, a behavioral model is developed and simulated, allowing initial testing of the autonomous software. As the vehicle is designed, the details of the vehicle are included in the model, increasing the realism of the simulation, until a full functional simulation is available. Likewise, the virtual environment must be developed in parallel, appropriately representing the world as required for testing. Failure to do so would hinder development as system testing would/should delay future development.

To facilitate the development/testing process, a software framework is required to enable seamless integration of the autonomous software into the test environment. The software under test should be oblivious of whether operating in a virtual/test environment or in physical operating conditions. The framework should support isolation of the different levels of the autonomous system model software for testing purposes. The framework should also support replacing or augmenting the physical environment with the virtual environment at varying levels of detail as required for testing, i.e. the level of detail required for testing the plan stage may be lower in detail than the sense stage.

Isolating the levels of autonomous software is assisted by the world representation. The world representation acts as a well-defined interface between the sense and plan stages. The representation could be as simple as a panoramic representation of distances to nearest obstacles, could increase in sophistication to have a memory of obstacles seen as the vehicle moves, or be as advanced as categorizing or recognizing obstacles. For instance, for ethical purposes it may be necessary for a driverless car to recognize the difference between a child and a senior citizen when responding to an emergency avoidance situation.

During the design process, the four phases of the reality-virtuality continuum are introduced as illustrated in Figure 2. This allows testing to increase in realism, gradually moving from a fully simulated test to a fully real world test, by slowly integrating actual computational, sensing, and motion capabilities as the hardware becomes available. Each phase is described as to its contribution to the testing of the design and development.

Virtuality

The process starts with a fully simulated system operating in virtual reality. Note that while the hardware system is fully simulated, the current state of the autonomous software is being tested, not a simulated version. The current state can progress from behavioral to algorithmic to functional as defined by the autonomous software development lifecycle. The software architecture enables this by isolating the autonomous software from direct interaction with the vehicle hardware, allowing information passing from the hardware to the software to by replaced/augmented. To test the sense layer, a virtual environment is required, representing truth. Sensor models are created and simulated to provide inputs to sense. The sensor models can start by providing truth themselves, and then migrate to realistic models to include appropriate error. To test the plan layer, a high level representation of the sensors and sense layer can be created to map directly from the virtual environment to the computed world view. If representing entity recognition in the computed world view, the entity can be directly translated from the virtual environment to the computed world view with no error for testing purposes.

The next step in the virtual reality phase is to integrate the sense and plan layers. As the vehicle's motion simulation is developed, the act layer can be included as well. While this completes a fully simulated development, this does not complete the use of the virtual reality phase. The simulated version should be maintained as future development is done, requiring further testing prior to introducing reality. The simulated system provides a first pass at rapid testing. In addition, the physical computing platform can be introduced while still in virtual reality. By operating the software on the actual computing platform but in virtual reality with a simulated vehicle, timing requirements can be tested.

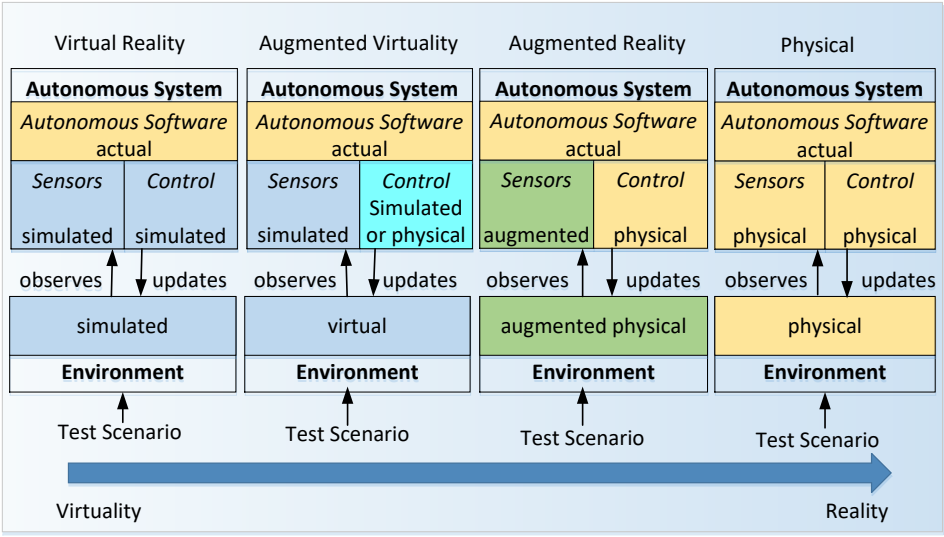


Figure 2. Reality-Virtuality Continuum.

Augmented Virtuality

As development progresses to allow the computing platform to be mounted on a physical robot, it is beneficial to introduce some reality. This begins by allowing the physical robot to maneuver in the virtual world. All sensed information is provided from the virtual environment. The autonomous software can react to this information and physically move the vehicle. By maintaining an avatar in the virtual world to represent the vehicle's state information in the physical world (position, etc.), the virtual environment can be appropriately sensed. This provides a safe environment to observe the vehicles response to various scenarios represented in the virtual world without risk of injury to people, the environment, or the vehicle itself. It also allows testing of individual sensors by imposing parts of the real world on the virtual world.

Augmented Reality

The continuum now introduces more reality and less virtuality. The vehicle now fully senses its physical environment and responds to it. Physical objects can now be placed in the environment. However, for safety reasons it may be undesirable to place all objects in the real environment. For instance, people or other autonomous vehicle may be represented in virtual reality and then imposed on the real environment. Now virtual information is imposed on the real sensed information, requiring a stage prior to the sense or plan stages where real information can be augmented.

Reality

Finally, testing must be completed in the real world. The integration of actual sensors with the computational platform and then the physical actuators and vehicle response is necessary, both to ensure proper operation and for public perception.

SYSTEM REQUIREMENTS

This section discusses the requirements for integrating VR/AR for testing and development, including elements and attributes of the virtual environment, the communication between the virtual environment and the autonomous vehicle, sensor data synthesis, and test harness software and hardware requirements.

Virtual Environment

The real-world environment in which autonomous vehicles operate is highly complex with numerous different types of objects, for a driverless car this could include other vehicles, roads, pedestrians, and traffic signals. A real-world environment can be described by its attributes as follows.

- **Geometry** is the geometrical shape of the objects in the real world, e.g., terrain, roads, buildings, water surface. The geometry of the objects is used for different purposes depending on the types of autonomous vehicles, e.g., while a driverless car should not collide with a building, a drone can possibly safely land on the top of the building.
- **Visual material property** mainly represents the color (either emitted or reflected) of the objects in the environment. For example, while street name signs have rectangular shapes, it is the text on the signs (visual property) that convey critical information. Many objects exhibit different colors depending on the time of the day. Autonomous vehicles should be able to reliably identify these objects under varying lighting and whether conditions.
- **Physics property** represents the physical characteristics of the objects with which the autonomous vehicles interact, e.g., the velocity and viscosity of the water in which an autonomous ship navigates, the friction of the road or terrain on which an autonomous vehicle travels. An object's physics property can vary with other factors, such as weather conditions.
- **Static and dynamic objects** are classification of objects based on various properties. Buildings, terrain, and roads are considered static in most scenarios. For traffic lights and other similar signal devices, while their geometry does not change, their varying visual properties (color) make them dynamic for most applications. Autonomous systems are dynamic objects.

The virtual environment serves the purpose of providing the input information necessary to model sensors as an input to the sense stage or to model the sensed information provided to the world representation. This enables the system to operate in a complete virtual reality or to augment reality with virtual information to varying degrees for augmented virtuality and augmented reality. The reality of the virtual environment must be sufficient to allow the models used to map the information to the format used by the autonomous software such that the software does not perceive a difference between virtuality and reality.

A virtual environment is an abstraction and representation of the real-world environment or fictional space. A virtual environment can represent an existing real-world environment, e.g., New York City, a modified real-world environment, e.g., New York city with new roads or modified traffic controls, or a future or fictional environment, e.g., a future city with the state-of-the-art infrastructure. To support the required realism through the virtuality/reality continuum, a multi-resolution, multi-purpose virtual environment is needed. Multi-resolution representation enables modeling and simulation of objects at different levels of details. For example, the same object can be represented by polygonal meshes with different number of triangles. While a high-resolution representation is necessary for objects near the autonomous vehicle, a low-resolution model can be more than sufficient for objects that are distant from the autonomous vehicle. Multi-resolution representation is not limited to geometry only, but also includes lighting, material property, and physics property as well. It reduces the testing cost and improves testing efficiency. The virtual environment should afford means for automatic and manual selection of resolution levels.

Virtual Reality

In virtual reality (VR) configuration, everything begins virtual, including environment, autonomous vehicle hardware and software. The autonomous software is tested across its full lifecycle in this configuration. The software originally executes on the host computing platform, but can migrate to an emulated computing platform. The highest level of testing or the final stage of testing that can be achieved using only virtual reality is to run real autonomous software on the autonomous vehicle's physical computing platform, and to this end, a communication path between the VR testbed and the autonomous vehicle must be established, which requires additional hardware and software that are not integral part of the autonomous system. The communication between the VR testbed and the autonomous system is handled by the software architecture and can be wired or wireless (Wi-Fi, Bluetooth), depending on the type, size, and movement range of the autonomous system. The VR testbed and the autonomous system must be synchronized in time, e.g., the dynamic objects in the VR testbed should be updated in real time using wall clock, the VR testbed should be paused during communication between the VR testbed and the autonomous vehicle as this time does not exist in real autonomous vehicle operation.

The sensor data are synthesized based on the virtual environment. The virtual environment should be able to generate different types of sensor data, e.g., lidar data, optical image data, and radar data, with programmable resolutions and accuracies, corresponding to various types and qualities of real sensors. The virtual environment should provide an array of various selectable virtual objects with configurable properties. It is worth to note that although the real

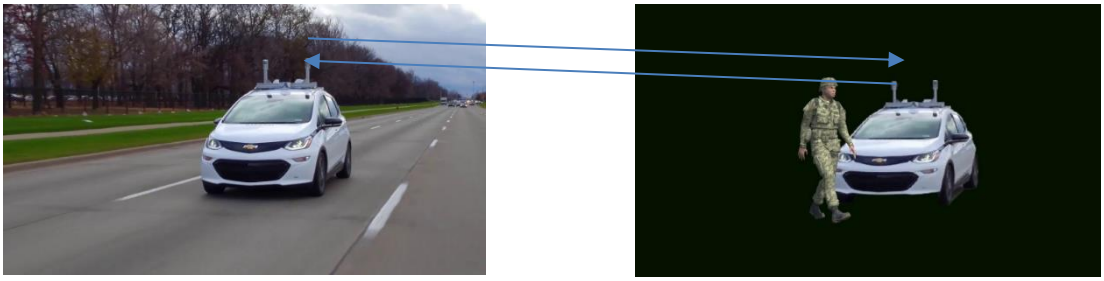
autonomous software can be used in virtual reality, the real autonomous hardware inputs (sensors) and outputs (actuators) are not used, the autonomous vehicle does not move physically.

Augmented Reality

A physical real-world environment is now utilized in the testing. However, the real autonomous system is tested in a real environment augmented with virtual objects. Figure 3 provides an illustration of the augmentation. The sensor data are synthesized based on the real-world environment and virtual objects. Testing in AR can be performed at two levels depending on the approach of sensor data generation.

1. **Real sensors are not used.** At this level of testing, a virtual environment that contains a realistic representation of the real-world environment and the virtual objects and the autonomous is placed in the real-world environment. The virtual environment receives real-time update of the autonomous system position and orientation and generates the sensor data accordingly. A communication path between the virtual environment and the autonomous system must be established, with the considerations discussed in the previous section. As the autonomous system operates in a real-world environment, the sensor data synthesis process must be fast enough to meet real-time requirements, i.e., comparable to the sampling rate of the real sensors. For dynamic objects in the real-world environment, their behavior must be known in the virtual environment beforehand. This testing is suitable for relatively simple real-world environments, e.g., an empty room. As the real-world environment is relatively simple, it poses less risk to the autonomous vehicle.
2. **Real sensors are used**. Real sensors on the autonomous systems are used to generate the preliminary sensor data, which are further modified based on the virtual objects. This approach is more complex and computationally intensive as it requires additional changes to the autonomous system hardware, i.e., intercepting the sensor data, modifying them, and send them back to the autonomous system. This is done at the sampling rate of the sensor, e.g., 30 Hz, 60 Hz or even higher. Comparing with the previous approach using one-way communication, this approach requires two-way communication. This testing is suitable for complex real-world environments and should be performed following the test described above. For example, assuming we want to include two obstacles in the environment. The first test includes the two obstacles only in the virtual environment and the sensor data are completed generated from the virtual environment. The second test puts one obstacle in the virtual environment and the other in the real-world environment and the final sensor data are generated based on both real and virtual environments.

Both approaches require the position of the autonomous system in the real-world environment be known and passed into the virtual environment to support an avatar representation of the physical vehicle in the virtual environment. This configuration cannot test all scenarios. For example, it can test the accuracy of obstacle avoidance of the autonomous system with a virtual object. However, it cannot test the collision between the autonomous system and a virtual object as the virtual object does not exist in the real-world environment.



(a) (b)

Figure 3. Illustration of augmented reality for autonomous vehicle testing. (a) A vehicle is moving in a real environment. (b) A virtual obstacle is added to the real environment. The sensor information processed in (a) is a combination of real objects in (a) and virtual objects in (b).

Human Observation

For both VR and AR configurations, multiple perspectives can be created to gain a better understanding of the behavior of autonomous systems, including first-person view, third-person view, bystander view, and bird's-eye view.

- **First-person view** is the perception of the environment from the point of view of the autonomous system in the forms of sensor data, including lidar data, optical image data, and radar data. The autonomous system itself is not visible in the first-person view.
- **Third-person view** follows the autonomous system and observes the interactions between the autonomous system and its surrounding environment. The relative distance and orientation between the autonomous system and the third-person camera can be adjusted, e.g., the camera can follow the vehicle from behind, looking in the forward direction of the vehicle; or the camera can follow the vehicle from front, looking in the backward direction of the vehicle.
- **Bystander view**. While both first-person view and third-person view follow the autonomous vehicle, bystander view offers observation of the autonomous vehicle from a (selected) fixed location in the environment, such as a pedestrian at an intersection observing a turning autonomous vehicle.
- **Bird's-eye view** provides understanding of the behavior of the autonomous system and the environment at a higher level or a larger scale. The bird's-eye view helps the user or developer gain immediate insight of the vehicle and the overall environment with just a glance. The level of the bird's-eye view can be adjusted and the camera can follow the vehicle as well.

All the views can be made available at the same time or one at a time based on what is selected by the user. For real-world testing, we have bystander view and bird's-eye view, although first-person view can be made available by transmitting sensor information to a server in real-time or stored in an on-board computer in the autonomous vehicle for after-action review.

Software Architecture

A primary goal of the testing process is to provide a seamless transition from testing to fielding, covering the complete continuum, to facilitate its use. To this end, the software should be unaware of its current environment, requiring no modifications to allow it to operate in a test environment. The goal is to avoid impeding autonomous software development for the purpose of testing. The process requires information to be inserted/replaced within the autonomous system pipeline to simulate the input of sensor data or sensed objects without the knowledge of the autonomous software. After processing, it is desirable to observe outputs to evaluate the behavior of the autonomous software and the ability for the vehicle to enact the plan. The architecture must support a consistent format between virtual and real data such that the system can readily accept both types.

A proposed high-level architecture is shown in Figure 4. The architecture is composed of the main stages of the autonomous system model, but includes as part of a test harness a software interface between the stages to define locations where information can be injected/replaced without the knowledge of the autonomous software. The software architecture isolates each of the main stages (Sense, Plan, and Act) of the autonomous software from each other and the hardware of the autonomous system (sensors and actuators) for modular testing. At each interface between stages or between a stage and hardware, a model of the information being passed is inserted. Each model operates in one of three modes for each piece of information being passed: it can pass the information through as is, it can augment the information with information from the virtual environment, or it can replace the information with a representation from the virtual environment. This allows each interface to address the continuum. VR and AR are applied for the sensor models and the information model.

Building the architecture on the Robot Operating System (ROS) (2018) facilitates use of previously built models and interfaces, especially for sensors and actuators. In addition, it enables testing during the virtual reality phase on a single computing system and then mapping the architecture onto multiple processors as is common on current computing platforms found on autonomous vehicles (Wei, 2016). It also allows the virtual environment to be moved off vehicle to support augmented virtuality and augmented reality with wireless communication providing the necessary transfer of information. Finally, the architecture must be lightweight in terms of processor intensive tasks to keep up with real-time communication with a physical vehicle. Building onto ROS allows for simplification of the top layers of the architecture by taking advantage of ROS's existing and efficient infrastructure for data mapping and communication.

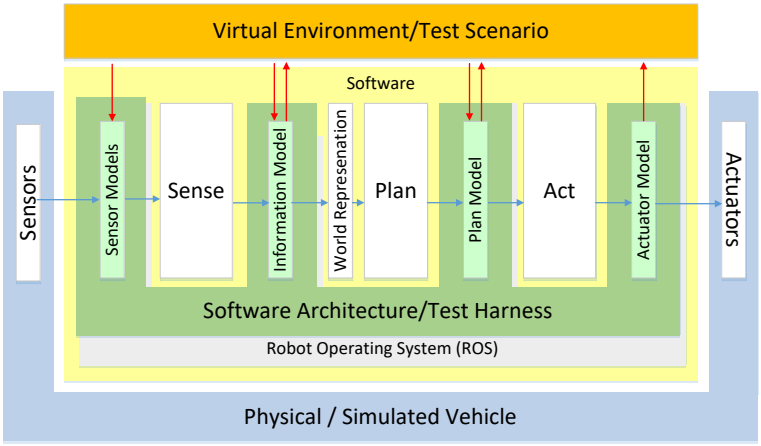


Figure 4. Software Architecture Supporting Integration of VR/AR.

BENEFITS

This paper presents a high-level view of a highly structured test environment for autonomous software. Knowledgeable practitioners quickly will see that there are significant costs associated with a realization of this environment. These costs include: development of detailed component interface specifications; implementation of a real-time communication infrastructure to facilitate exchange of information among system components; and design of appropriate test input generators/simulators and test output observers. Thus, it is important that there be corresponding benefits that justify these costs.

The authors believe that the following list identifies the most significant benefits of utilizing the proposed test environment.

- Economics of Structured Test Environment – Many of the costs associated with an implementation of this test environment are one-time costs; however, the test environment can be reused thus distributing the costs over a number of test and evaluation studies. Implementation of a real-time communication infrastructure and the design of test generators/simulators and output observers can be used throughout the life cycle of an autonomous software design and implementation. With minor modification, these test environment components may be applicable to the testing of other autonomous vehicles. Other costs, such as the development of component interface specifications, would occur as part of any testing process and thus are not additional costs that can be attributed to the use of this test environment. The capability to reuse this test environment also makes it practicable to make a more significant initial investment that includes system documentation, training materials for users, and maintenance and upkeep of the initial system implementation. Such an investment is almost certain to result in faster and more efficient testing using the test environment.
- Flexibility of Test Environment – The proposed test environment architecture is very flexible from the point of view of the user. The architecture allows the user to select various system components, or combinations of components, as the system under test while other system components can be included as part of the test harness. This makes it possible to test isolated system components or capabilities without the need to develop new testing procedures or infrastructure. In addition, the presence of a well-defined interface for communication means the autonomous software can be tested utilizing real, virtual, or augmented system components, for the vehicle and the vehicle environment.
- Autonomous Software Design – Not only is the test environment useful for testing autonomous software, it also is useful in the design of autonomous software. In the early stages of design, behavioral models of the autonomous software can be implemented as virtual system components and evaluated through simulation testing. As autonomous software components are developed, they may be used to augment the remaining virtual autonomous components while retaining the capability to test the entire autonomous vehicle system. This capability facilitates a spiral design process for the autonomous software as well as the vehicle hardware and software realization. This is especially useful in situations where testing of the complete real system is extremely difficult or highly dangerous.
- Early Integration of Design/Testing – The proposed test environment enables designers to conduct full-system testing throughout the design and development phases of an autonomous vehicle. The test environment facilitates the use of real, virtual, or augmented system components to assemble the entire system for test and

evaluation. This approach not only results in better component design, it also results in components that are much more likely to work properly when they are interfaced to form a complete system. It is well known that identifying and correcting errors in component designs, or component designs that will not interface properly, early in the design and development process dramatically reduces system develop time and cost.

- Seamless Test Integration – The software architecture provides a mechanism to support the testing of the autonomous software. The lightweight software supporting the architecture is present whether being used for testing or being fielded. It simply selects the source of information being provided to each stage based on the current mode of operation. This also supports testing after a system failure/error during actual operation.
- Autonomous Software Training – Autonomous systems often require training prior to deployment. The development of the virtual environment and its ability to drive the autonomous software allows training early in the lifecycle prior to introduction of reality.

CONCLUSIONS

This paper examines the benefits of utilizing VR and AR techniques to facilitate testing and evaluation of autonomous vehicles during all phases of the design and development process. The approach addresses many of the difficulties that autonomous vehicle software presents to classical software testing methods. A new software architecture for seamlessly incorporating VR and AR system components in the system test process is described, and requirements and constraints for this testing approach are examined. The authors presently are implementing a limited version of the software architecture. This implementation will support testing of a real autonomous land vehicle in a real environment that is augmented with virtual obstacles. In addition, the authors are working with several local companies to plan an enhanced version of the architecture to design and develop a test capability for small aerial drones.

REFERENCES

Brat, G., Denney, E., Farrell, K., Giannakopoulos, D., & Jonsson, A. (2006). A Robust Compositional Architecture for Autonomous Systems. Aerospace Conference, Big Sky, MT, March 4-11.

Brooks, Rodney A. (1985). A Robust Layered Control System for a Mobile Robot. IEEE JOURNAL OF ROBOTICS AND AUTOMATION, Vol. RA-2, No. 1, p. 14-23.

Davis, B., & Lane, D. (2010). Guided Construction of Testing Scenarios for Autonomous Underwater Vehicles Using the Augmented-Reality Framework and JavaBeans. IMechE, Vol. 224 Part M: Journal of Engineering for the Maritime Environment, p. 173-191.

Erann, Gat. (1998). On Three-Layer Architectures. Artificial Intelligence and Mobile Robots. AAAI Press.

Goldman Sachs Report. (2015) Drones: Reporting for Duty. Retrieved February 1, 2018, from http://www.goldmansachs.com/our-thinking/technology-driving-innovation/drones/.

Hodicky, J. (2015). Modelling and Simulation in the Autonomous Systems' Domain – Current Status and Way Ahead. Modelling and Simulation for Autonomous Systems: Second International Workshop, MESAS 2015, Prague, Czech Republic, April 29-30, Revised Selected Papers, Springer.

Koopman, P. & Wagner, M. (2016). Challenges in Autonomous Vehicle Testing and Validation. SAE Journal on Transportation Safety, Vol. 4, No. 1, p. 15-24.

Menzies, T. & Pecheur, C. (2005). Verification and Validation and Artificial Intelligence. Advances in Computers, Vol. 65, p. 153-201.

Milgram, P., Takemura, H., Utsumi, A., & Kishino, F. (1994). Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum. SPIE Telemanipulator and Telepresence Technologies, Vol. 2351, p. 282-292.

Mullins, G., Stankiewicz, P, Hawthorne, R., Appler, J., Biggins, M., Chiou, K., Huntley, M., Stewart, J. & Waktkins, A. (2017). Delivering Test and Evaluation Tools for Autonomous Unmanned Vehicles to the Fleet. John Hopkins APLTechnical Digest, Vol. 33, No. 4, p. 279-288.

Robot Operating System (ROS). Retrieved January 31, 2018, from http://www.ros.org/.

Schumann, J. & Visser, W. (2006). Autonomy Software: V&V Challenges and Characteristics. Retrieved January 23, 2018, from https://archive.org/details/nasa.tech.doc_20060015099.

Wei, Hongxing, Zhenzhou Shao, Zhen Huang, Renhai Chen, Yong Guan, Jindong Tan, & Zili Shao. (2016). RT-ROS: A real-time ROS architecture on multi-core processors. Future Generation Computer Systems, Vol. 56, p. 171-178.