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

Thomas J. Laverghetta, James F. Leathrum, Jr., Nathan Gonda
Department of Modeling, Simulation and Visualization Engineering
Old Dominion University
Norfolk, Virginia
tlave002@odu.edu, jleathru@odu.edu, ngond002@odu.edu

ABSTRACT

Test and evaluation (T&E) of autonomous vehicles presents a challenge as the vehicles may have emergent behavior and it is frequently difficult/impossible to ascertain the reason for software decisions. This creates the need for T&E during the complete lifecycle of a vehicle's development and fielding, requiring T&E after deployment due to emergent behavior and software updates. To support these requirements, a software framework has been developed to support development of autonomous software such that it can be tested during the complete lifecycle, migrating through the virtuality-reality spectrum. The paper demonstrates the application of simulated/virtual reality/augmentd reality/physical environments to test and evaluate the sense, plan, and act phases of an autonomous system to ensure proper operation with little to no physical activity from the system allowing for safe testing of the system without endangering either the system or the environment. The demonstrations consist of the processes of constructing models for simulating the physical world through various sensors to transpose onto the autonomous software from a virtual environment. The autonomous software cannot distinguish between physical and simulated data; allowing for more fidelity in the testing of the autonomous software in a simulated environment. Then a process for various levels of simulation can be introduced, such that, the testing of the software goes from a fully virtual environment to a fully physical environment ensure proficient results without compromising safety.

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, simulation-based test and evaluation, and simulation education.

Thomas Laverghetta is an undergraduate researcher working towards a Bachelor of Science degree in Modeling, Simulation, and Visualization Engineering at Old Dominion University; prospectively graduating in the May of 2021. His research interest includes simulation-based test and evaluation, big data analytics, systems and industrial engineering, and human behavioral modeling.

Nathan Gonda is a graduate student working toward a Masters degree in Modeling and Simulation Engineering. He has earned a Bachelor of Science 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.

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

Thomas J. Laverghetta, James F. Leathrum, Jr., Nathan Gonda
Department of Modeling, Simulation and Visualization Engineering
Old Dominion University
Norfolk, Virginia
tlave002@odu.edu, jleathru@odu.edu, ngond002@odu.edu

INTRODUCTION

Despite the obvious benefits of autonomous vehicles (e.g., environment, economy, and safety (Fagnant and Kockelman, 2015)), 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. This paper demonstrates the use of a software framework to allow autonomous software to be testing during the development process by immersing it in a virtual environment and then gradually adding reality, but augmented with virtuality, until finally the system is testing in full reality.

In (Koopmann and Wagner, 2016), (Menzies and Pecheur, 2005), and (Schumann and Visser, 2006) the challenges of testing autonomous software are discussed. 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, autonomous 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 approaches for enhancing the capability to test and evaluate autonomous software (Brat and Denney, 2006) (Mullins and Stankiewicz, 2017). 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.

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 (Jonette M Stecklein, 2010). Figure 1 illustrates how simulated and physical testing in virtual and augmented reality can be integrated into the development lifecycle of an autonomous vehicle. This paper demonstrates how utilizing a software framework introduced in (Leathrum et al. 2018a) and (Leathrum et al. 2018) for the development of the autonomous software allows seamless testing in VR and AR, finally resulting in physical testing. The software framework is described, and the architecture used to integrate the autonomous software with the physical and vehicle, allowing operation in both virtual and physical worlds, or a hybrid. The use of the framework to integrate both virtual and physical sensor data is described. Then an example using a range finder and compass is provided to illustrate the process.

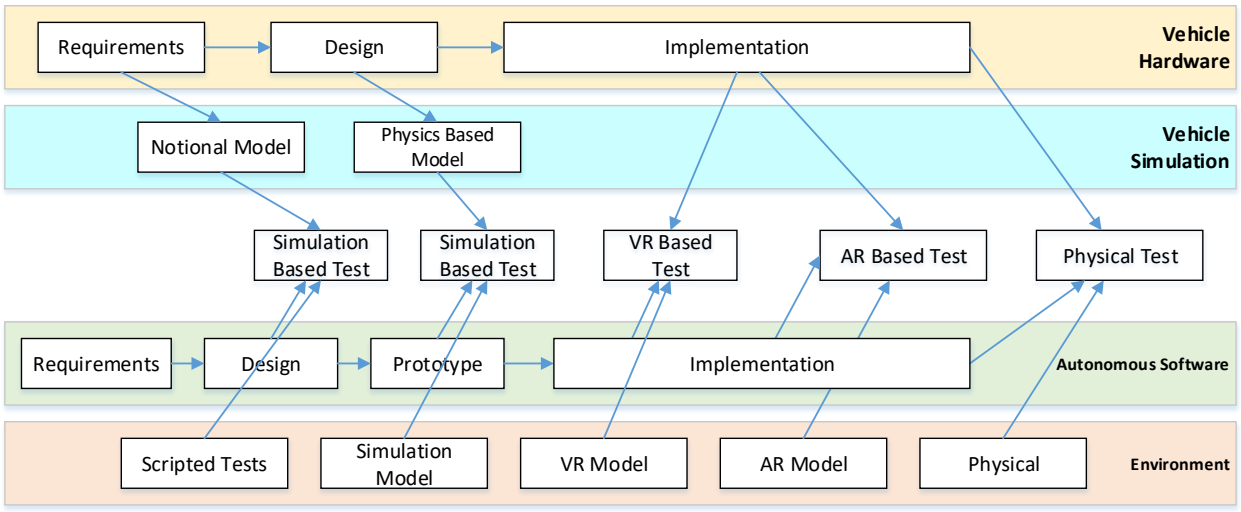

Figure 1. Integrating the virtuality-reality testing spectrum into the lifecycle of autonomous system development.

CONCEPT: APPLYING VR AND AR TO TESTING DURING DESIGN AND DEVELOPMENT

Promoting hardware/software codesign and development, a simulation-based approach is presented. The approach relies 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 as shown in Figure 1. 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 (Milgram et al., 1994) (Davis and Lane, 2010) are introduced. 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. A 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. The simulation of the physical platform progresses from a behavioral model to a fully functional model as the specifications and design of the system progress, allowing greater and greater detail in the testing process.

Augmented Virtuality

As development progresses to allow the computing platform to be mounted on a physical robot, it is beneficial to start introducing 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 vehicles 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.

AUTONOMOUS SOFTWARE TEST & EVALUATION FRAMEWORK

This section presents the test & evaluation framework first described in (Leathrum et al. 2018a) and (Leathrum et al. 2018b). The framework facilitates building an effective test harness for different scenarios in the virtuality-reality spectrum. This includes:

- isolating components of the autonomous software for testing purposes,
- isolating the autonomous software from its' operating environment, and
- isolating the autonomous software from knowledge of the source or use of information

A general model for an autonomous system is first introduced. The general model identifies the major processes of an autonomous system that need to be considered for test & evaluation and how they interact together under general circumstances. Autonomous systems require the ability to utilize various sensors to gain information about the external environment. Autonomous systems must also be able to interface with a physical system's actuators to instruct the system to act. Finally, autonomous 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.

Autonomous Software Model

One approach is to model the control system using a pipeline of functional modules – sense, plan and act (Gat 1998). 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.

This model is employed to illustrate the ensuing work; however, the work is not solely relegated to this model and may include other stage decompositions such as to include a perception stage.

Framework Structure

The focus of the framework is then to provide the ability to test each module in isolation as well as in integration with the external system(s) without the need to reconfigure or alter the autonomous model for different scenarios. Figure 2 illustrates a high-level view of this architecture. Four major sections are highlighted. They are the:

- Physical Systems (Physical Vehicle, Physical Environment)
    - The *Physical Vehicle* is composed of *Sensors* (which provide information to the autonomous software), *Actuators* (which accept information as control signals to alter the vehicle's operation) and the physical plant providing power and motion.
    - The *Physical Environment* represents all the external factors and stimuli that can influence and be influenced by the physical vehicle.
- Virtual Systems (Virtual Environment and Test Scenario, Simulated Vehicle) - representing generated or simulated versions of the physical counterparts.
    - The *Simulated Vehicle* is composed of *Virtual Sensor Models* and *Virtual Actuator Models* that represent the behavior of the physical counterparts found in the *Physical Vehicle*.
    - The virtual environment represents a generated version of the environment external to the vehicle.
- Autonomous Software – software implementing the system autonomy
    - For this work, decomposed into *Sense*, *Plan* (including *World Representation*, an internal representation of the world on which *Plan* can act), and *Act*.
- Test Harness
    - A software framework isolating the autonomous software from its operating environment, and modules of the autonomous software from each other, to allow a control of information for testing purposes.

Publish-Subscribe and Robotic Operating System (ROS)

Underlying the *Test Harness* is a communication layer that connects the separate parts of the framework architecture together. The layer primarily needs to handle the communication of different types of information without knowledge of specific senders or recipients. A pattern that matches these requirements is the Publish-Subscribe pattern (Eugster 2003). Publish-Subscribe is a loosely coupled, message-oriented pattern for communicating in a network. The pattern does not involve direct communication between message sender and recipients (publishers and subscribers) and allows for network entities to be replaced without impacting the whole network.

In this way, several mappings can be made from the architecture to the pattern. The main entities that compose a Publish-Subscribe system are nodes and topics. Nodes are defined as the functional elements that produce and consume information. Topics are defined as logical channels that associate a type of data to the channel such that nodes can communicate by interacting with the topic through publication or subscription rather than directly with each other (Eugster 2003). When a node publishes information, it is to a named topic that accepts the message content the publisher provides. A node that subscribes to information indicates a named topic from which to receive data and registers to be notified as messages are made available through the topic. This also assumes the node can readily accept the message content from the named topic.

ROS (*Robotic Operating System*) (ROS, 2018) is the communication layer utilized int this work and provides the facilities for the types of communication that the framework requires. *ROS* is a collection of libraries that provide Publish-Subscribe service on top of the normal communication layers for communicating between separate nodes in

a peer-to-peer network (ROS). These nodes can be either on the same machine or different machines, for instance allowing the virtual environment to be hosted on a different computational platform that the autonomous vehicle.

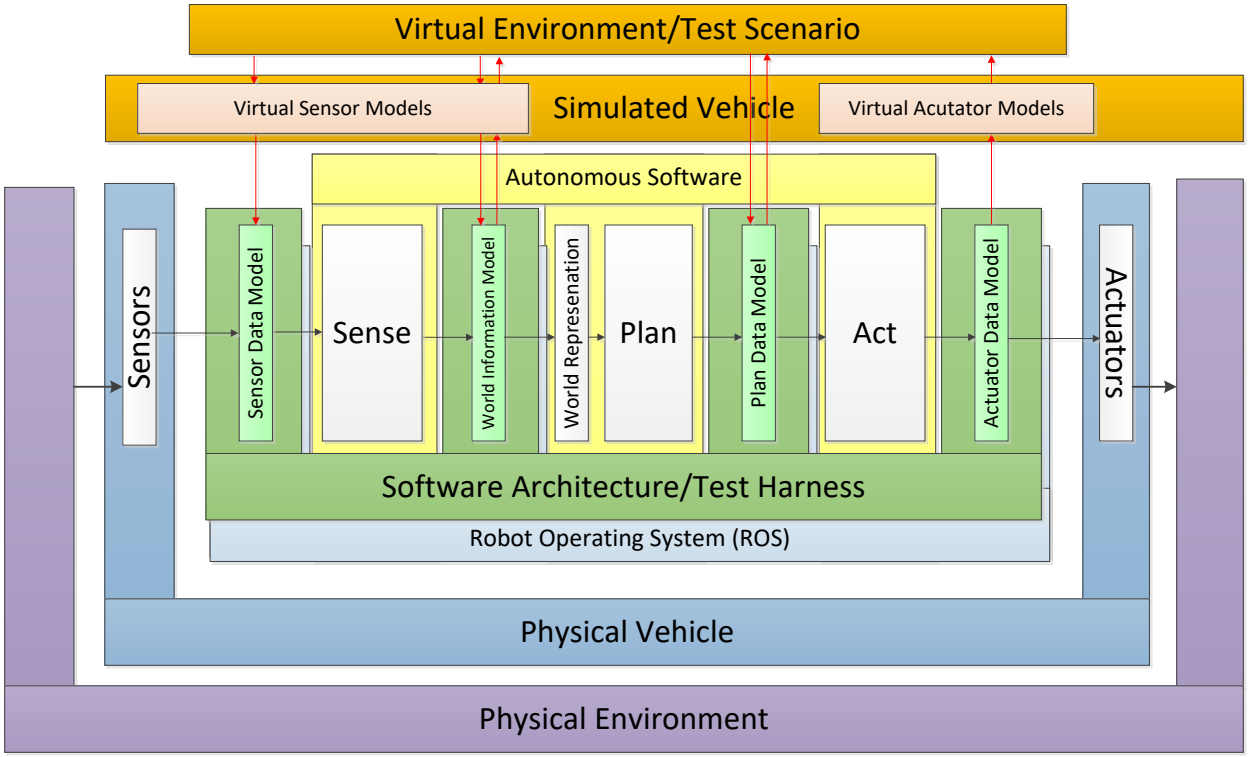

Figure 2. Test & Evaluation Architecture.

INTEGRATING THE VIRTUAL/AUGMENTED REALITY-SPECTRUM FOR T&E

In this section, integrating the virtual/augmented reality spectrum into the test harness is discussed. The virtual/augmented reality spectrum requires the ability to choose from multiple information sources external to the autonomous software to manage different stages of the spectrum. Sources of information to provide to the autonomous software include: virtual data, real data, and a hybrid of the two. The mechanism for choosing between the sources is determined by a set of reality-modes, one for each individual node defined within the test harness components: Sensor Data Models, World Information Model, Plan Data Model, and Actuator Model. A node is created for each data being communicated to control its flow. The process required to integrate VR and AR for testing is discussed for the virtual sensor models and the world information models shown in Figure 2. Note, we will only be discussing Sensor Data Models in this paper because it is relevant to our example.

Sensor Data Model

The Sensor Data Model accepts sensor data from either physical sensor sensing the real world or virtual sensors sensing the virtual environment as shown in Figure 3. The architecture supporting this is composed of a set of ROS nodes. ROS nodes are defined individually for the virtual environment, the virtual sensor model, the sensor interface, the sense stage of the autonomous software, and each individual block within the data sensor model, each controlling the flow/structure of an individual class of data.

Physical sensor information is handled by the information being obtained by the sensor interface and then published to the appropriate topic for consumption by the appropriate node in the sensor data model. Conversely, virtual data is published as truth to the appropriate topic for consumption by the virtual sensor model. The virtual sensor model takes this truth and generates a “distorted” version of the data, introducing error models, etc., and then publishing the resulting information to topics that relate to corresponding information from physical sensors within the sensor data models.

Once real and virtual sensor data is produced at the appropriate topics for the sensor data model, the data is consumed to produce the appropriate data to the sense stage. At each sensor data model node, shown in Figure 4, the real and virtual (if appropriate) data is manipulated to produce data to a topic for use by the sense stage of the autonomous software. Each node of the sensor data model operates in one of three modes: physical, virtual or augmented. In the physical mode, the data from the associated physical sensor is passed through to the sense stage, allowing testing on real data. In the virtual mode, sensor data from virtual sensors is passed through to the sense stage, allowing testing in the virtual world. In the augmented mode, the two data sources are manipulated to generate a mixed real-virtual form of the data.

Development, if the physical sensor interface, virtual environment and sense stage are available, primarily involves connecting ROS nodes and topics to properly route information. The only functional development involves creation of the virtual sensor model and the augmentation mode for the sensor data model nodes, complexity of which varies greatly between different sensors, simple for range finders, extremely complex for video.

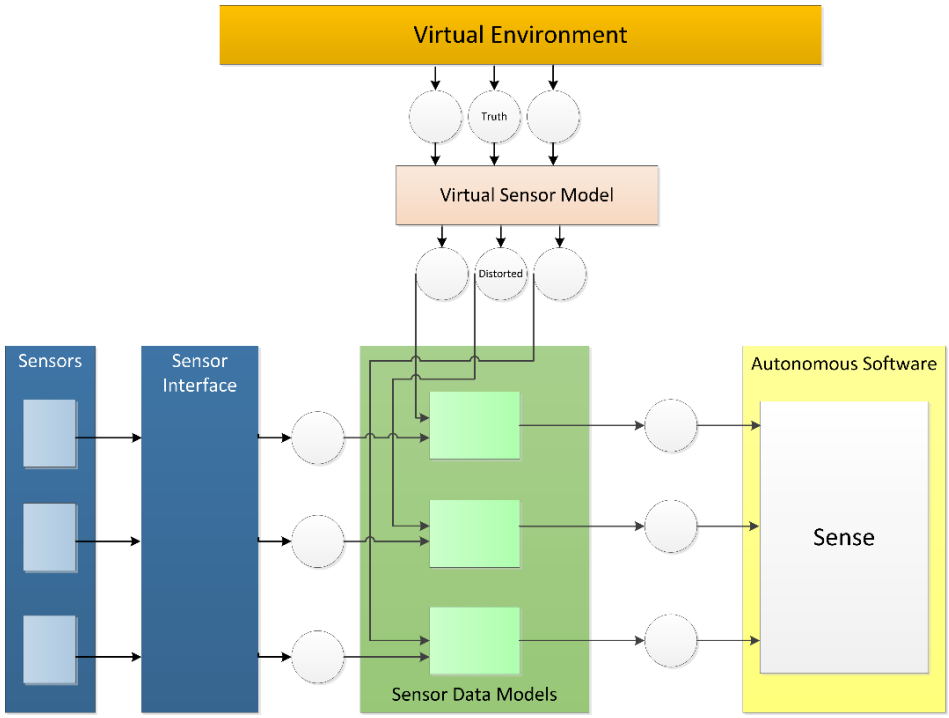

Figure 3. Transaction between Sensor Data to Sense-Node

World Information Model

Figure 5 shows the transaction between the sense stage, the virtual environment and the plan stage through the world information model. This is very similar to the sensor data model, though the form of the information is radically different. The information being passed from sense to plan is in an internal representation of the autonomous software defined as the world representation. The representation assumes that the external environment, whether real or virtual, has been processed into a useful form. Thus, it is likely in a software object-based form. Therefore, the primary development in Figure 5 involves the virtual sensor model where the internal representation of information in the virtual environment must be transformed into the world representation for use by the plan state.

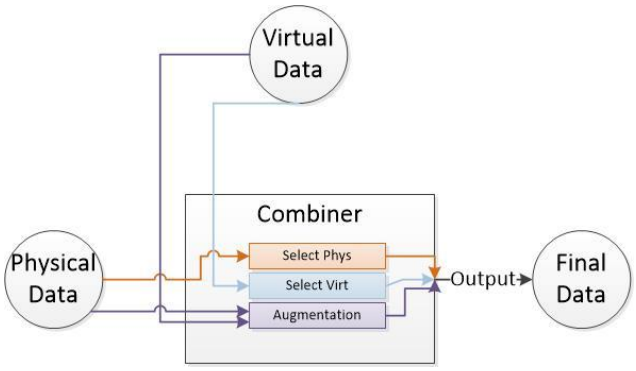

Figure 4. Reality Modes

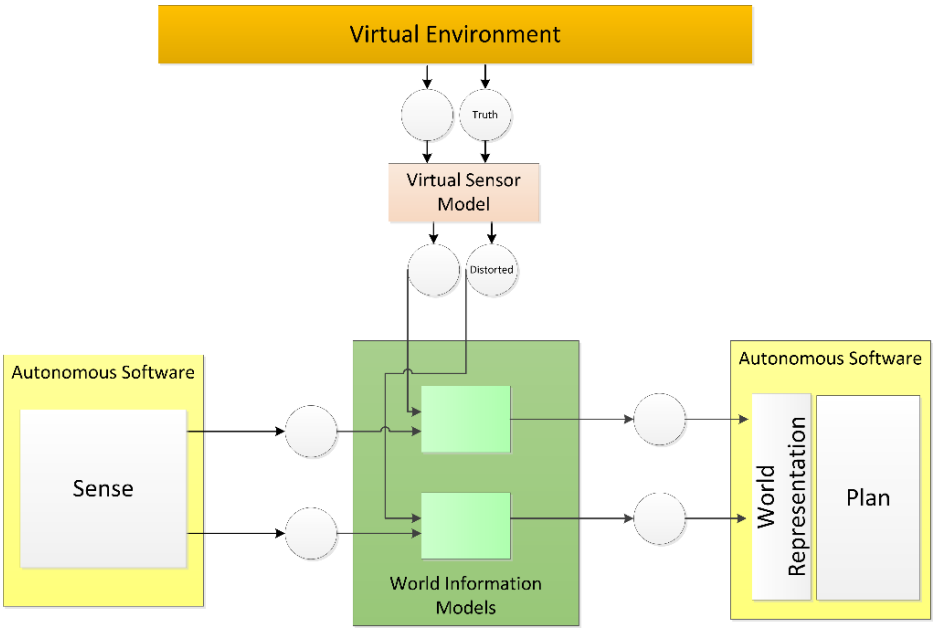

Figure 5. Transaction between Sense-Node Data & World Representation Node

EXAMPLE APPLICATION – RANGE FINDER INTEGRATION

In this section, a demonstration of the T&E software framework is presented. The demonstration involves testing of a range finding sensor for use on a robot platform in the absence of a complete system. The range finding sensor is paired with a compass providing heading to plot cartesian coordinates of detected points in the environment. Physical sensors are mounted on a turntable, shown in Figure 6, so they can rotate to sense the environment (translation is not supported as there is no location or motion sensor involved). The compass data is always produced as real, while the range finder data can be either real or virtual.

Data from the physical range finder is provided as a distance in centimeters, and rotating the turntable allows distances from one or more objects to be detected and plotted. Data from a virtual world containing virtual objects involves defining the location and orientation, location being fixed, and orientation being provided from the compass through the software framework to the virtual environment to modify the state of an avatar representation of the platform. Given location and orientation of the avatar, the distance to any objects in the given heading are computed and returned as truth.

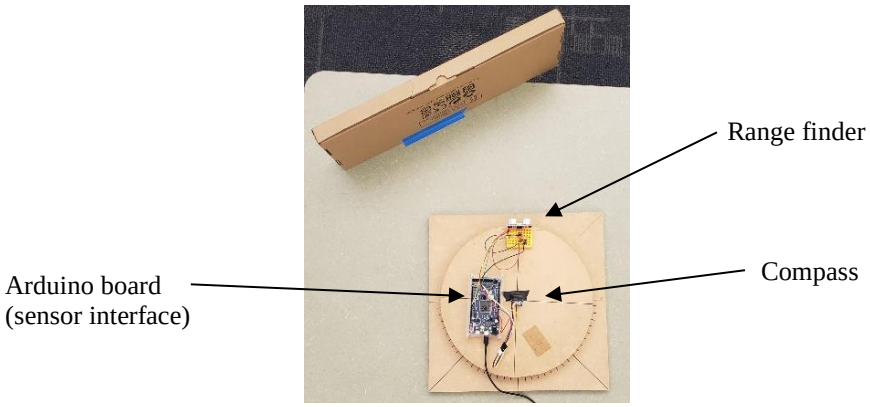

Figure 6. Setup for Example

T&E Software Framework Example

Figure 7 provides the structure of the framework used for the example demonstration. Physical sensor data is obtained using an Arduino board configured as a ROS node and published to topics for the sensor data model. In addition, virtual sensor data is obtained from the virtual environment by the virtual range finder model to produce virtual data. The range data is then manipulated based on the following modes of operation:

- 0 (physical range finder data) – sensor data from the physical range finder is passed through.
- 1 (virtual range finder data) – sensor data from the virtual range finder is passed through.
- 2 (augmented range finder data) – sensor data from the physical range finder is augmented with data from the virtual range finder. This simple demonstration involves just taking the minimum of the two data. Other sensors would involve much more complicated augmentation.

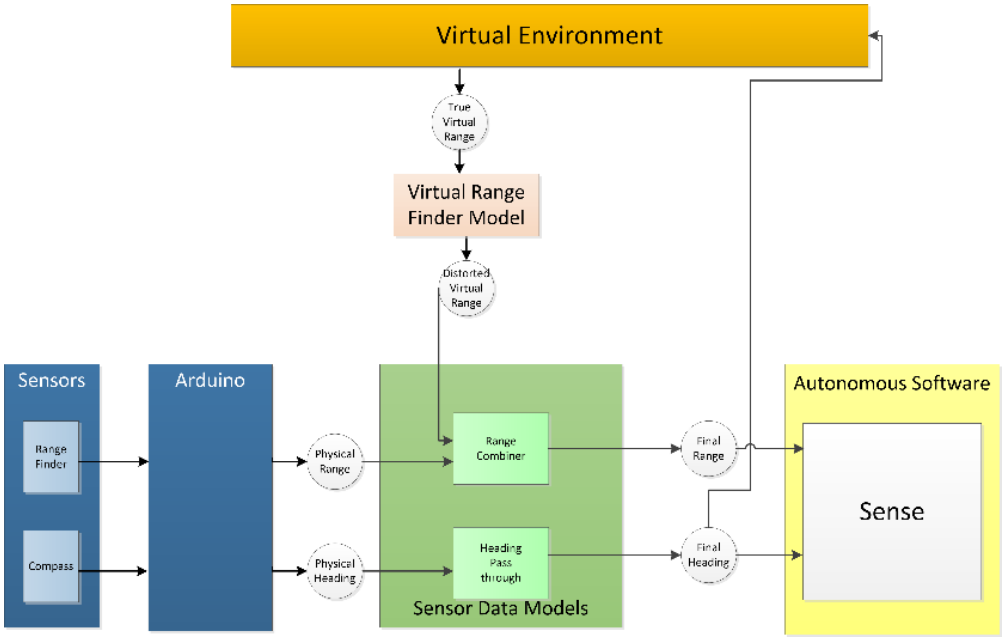

Figure 7. Example Transaction between

Creating the Virtual Sensor Model

As mentioned previously in integrating the virtuality-reality spectrum, virtual environment sensor data only displays truth, and as such, models of physical-sensor data must be created. For the example, virtual distance data should

emulate physical distance data. Data was collected from the physical range-finder to capture sensed and real distances from various object shapes and materials. The data was used to create behavior and error distributions used to modify truth data provided from the virtual environment. Ideally the resulting model would prevent the autonomous software from being able to detect the source of data by the form of the data. This would require more complete sensor modeling that done for demonstration purposes here. The process can be very complex, for instance the simple ultrasonic range finder employed may produce no data if the angle of the object being detected is to steep, or even worse, greatly chaotic results.

Results with different Reality Modes

Once all the parameters have been set (i.e., virtual sensor model behavior, reality modes, etc.), then runs can be performed to observe the sense node data. Figure 8 shows the various reality modes and virtual sensor modeling. Each plot shows data points representing coordinates derived from the heading and distance data. Virtual sensor data models include showing truth from the virtual environment and virtual sensor data with an error model applied. Virtuality-reality modes include real, virtual, and augmented. Plots in Figure 8 are shown in the order of introduction of fidelity and movement across the virtuality-reality spectrum. The plots show:

- Virtual sensor data, no distortion – robot operating in a virtual environment with no error models. The data points show a straight-line representation of an object in the virtual world.
- Virtual sensor, distortion – error model has been introduced to the virtual environment. The data points show a "fuzzy" representation of an object in the virtual world due to sensor error.
- Augmented sensor, no distortion – Both a virtual object (truth) and a physical object are detected. A physical object has been placed in front of the virtual object.
- Augmented sensor, distortion – same as augmented sensor, no distortion except that an error model has been applied to the virtual data.
- Physical sensor data – the data associated with only detecting a physical object is presented.

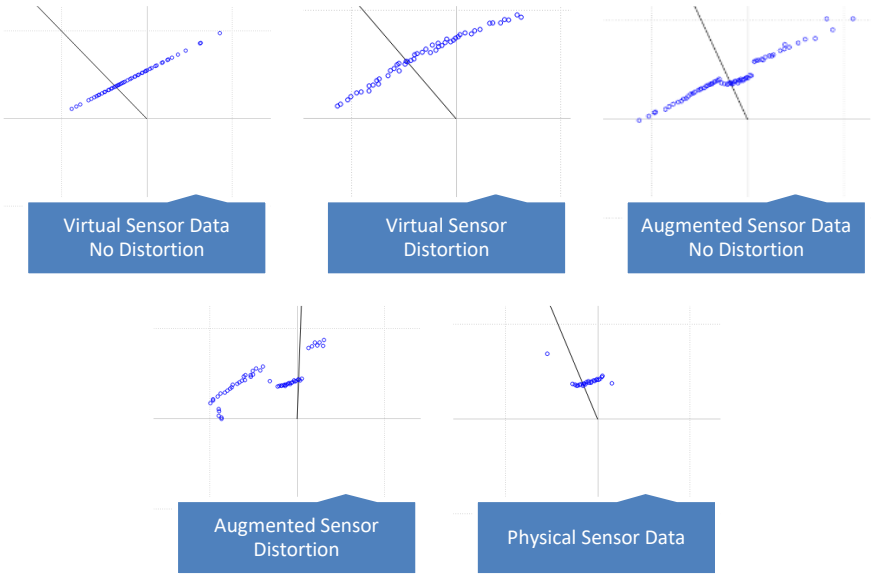

Figure 8. Reality Modes from Fully-Virtual to Fully-Physical

CONCLUSIONS

The increasing complexity and rigorous requirements placed on autonomous software present new challenges for testing and evaluation. The software must be able to perform in unexpected conditions and meet high standards for safety for acceptance in the field. Virtual Reality (VR) and Augmented Reality (AR) provide methods for testing autonomous subsystems using realistic stimuli in a less hazardous and more cost-effective environment. To facilitate seamless testing across the spectrum of the virtuality-reality spectrum, this paper describes a software architecture for design and development that allows the autonomous software to operate in both virtual and physical worlds, without

direct knowledge of either. The example showcases putting a simple Sense stage through the Virtuality-Reality spectrum using sensor data from a range finder and compass. This forms a first step in test and evaluation of the entire autonomous system.

While the example presented is simple for demonstration purposes, the concept has been extended to a complete robot system in which a physical robot can operate in virtual, augmented, and physical environments, detecting both real and virtual objects and avoiding them. Future research will focus on the virtual environment to allow realistic modeling of camera data and to allow improved human observation of the robot's perception of its environment. In addition, an autonomous system design case study will be performed to address the process of utilizing the T&E framework during the design and development lifecycle. Lastly, the framework will be considered to testing of already fielded systems, allowing testing as emergent behavior and software updates potentially change behavior.

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.

Eugster, P.Th., P.A. Felber, R. Guerraoui, A.-M. Kermarrec. (2003). The Many Faces of Publish/Subscribe. ACM Computing Surveys, Vol. 35, No. 2, p.114-131.

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.

Leathrum, J., Shen, Y., Mielke, R., & Gonda, N. (2018). Integrating Virtual and Augmented Reality Based Testing into the Development of Autonomous Vehicles. MODSIM World 2018, Norfolk, VA, April 24-26.

Leathrum, J., Mielke, R., Shen, Y., and Johnson, H. (2018). Academic/Industry Educational Lab for Simulation-Based Test & Evaluation of Autonomous Vehicles. WinterSim 2018, Gothenburg, Sweden, December 9-12.

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.

Fagnant, D. J., & Kockelman, K. (2015). Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, *77*, 167-181. https://doi.org/10.1016/j.tra.2015.04.003

Jonette M Stecklein, J. D. (2010). Error Cost Escalation Through the Project Life Cycle. *14th Annual International Symposium* (p. 11). Toulouse, France: NASA Technical Report Server.