High Fidelity Sensor Simulations for the Virtual Autonomous Navigation Environment

Chris Goodin, Phillip J. Durst, Burhman Gates,
Chris Cummins, and Jody Priddy

U.S.Army Engineer Research and Development Center

Abstract. The Virtual Autonomous Navigation Environment (VANE) is a high-fidelity simulation environment for ground robotics. Physics-based realism is the primary goal of the VANE. The VANE simulation incorporates realistic lighting, vehicle-terrain interaction, environmental attributions, and sensors. The sensor models, including camera, laser ranging, and GPS, are the focus of this work. These sensor models were designed to incorporate both internal (electronic) and external (environment) noise in order to produce a sensor output that closely matches that produced in real-world environments. This sensor output will allow roboticists to use simulation further into the development and debugging process before exposing robots to field conditions.

1 Introduction

The Virtual Autonomous Navigation Environment (VANE) is a high-fidelity simulation environment that is being developed by the U.S. Army Engineer Research and Development Center to facilitate virtual testing and algorithm development for ground robotics. The VANE incorporates physics-based vehicle dynamics, radiation transfer, environmental attributions, and sensor models into a closed-loop simulation.

An overview of robotics simulations was recently completed by [1], who found that most commercial simulations were of medium to low fidelity. Lower fidelity, real-time simulation environments such as Real/Player, Microsoft Robotics Studio, or the National Institute of Standard's Unified System for Automation and Robot Simulation (USARSim) [2] have proven useful in aiding the development and debugging of robotic algorithms. However, these environments often present a sanitized, noiseless environment that may not expose certain algorithmic problems that are later discovered in field tests. The VANE is intended to provide an unsanitized simulation environment that incorporates the noise and unpredictability that is often encountered in the real world, both in the environment and the sensors.

The VANE simulation concept is depicted in Fig. 1, which shows the critical components of the VANE. The first component, shown as the large yellow block, is the VANE Run Time Interface (RTI). The VANE RTI is an event driven simulation environment that manages the passage of simulation time. Because the VANE's focus is on realism and high-fidelity, it is not intended to be a "real time" simulation. Instead, care was taken to create an environment with flexible time management that allows each physics-based model to operate at its characteristic frequency.

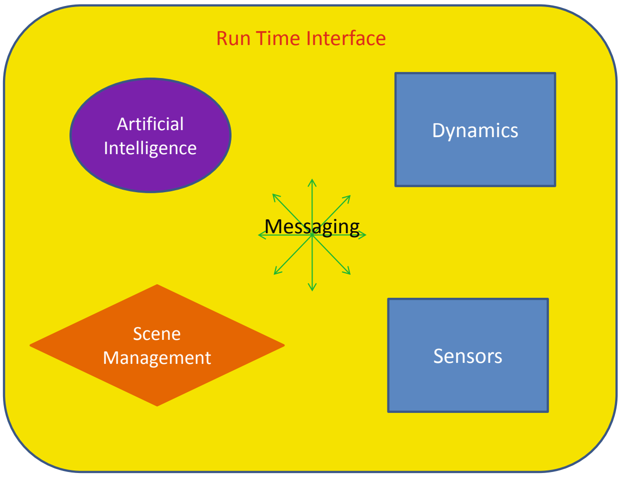

Fig. 1. A visualization of the VANE simulation architecture. The yellow rounded box containing the other boxes is the RTI that provides a general framework for setting up the simulation. The details of the environment are controlled by the Scene Manager (orange diamond), and the physical models are represented by the blue rectangles. The messaging, depicted by the green arrows, is a separate but integral part of the RTI that communicates between the components of the simulation.

The second part of the VANE simulation is the messaging, depicted by the green arrows in Fig. 1. The messaging defines protocols for sending messages of a particular type, including text, sensor data, or execution instructions to the other models. The messaging is independent of the RTI both for the sake of simplicity and flexibility. While the details of the physical models (the blue boxes in Fig. 1) may change from one simulation to the next, the message types will remain the same. Furthermore, the messaging system allows information to be passed within a processor, between processors, or over a network.

The orange diamond in Fig. 1 is the scene manager, which controls the virtual representation of the physical environment being simulated. Each of the physical models (blue boxes) may require a unique representation of the virtual environment, and for a dynamic simulation the physical environment may change in an unpredictable way as the simulation proceeds. Furthermore, the models may have different requirements for the level of fidelity used to represent the environment, and the required level of detail may itself be dynamic. For example, a typical Laser Detection and Ranging (LADAR) sensor may require the scene to be represented at the millimeter scale to a range of 40 meters but a centimeter scale beyond. This varying level of detail would require different representations of the scene for each sensor (and implicitly each robot) in the simulation. The scene manager tracks the dynamic environment and makes the necessary updates to the scene during the simulation.

The purple oval in Fig. 1 represents the Artificial Intelligence (AI) in the simulation. In VANE simulations, "AI" primarily refers to the Autonomous Driver (ADr) on the robot being tested. However, "AI" may also refer to any other intelligence algorithms that control some aspect of the simulation. For example, for a simulation of riotous behavior in groups of people within the VANE, the algorithms controlling the mob's behavior would be included in the "AI" box. As mentioned above, the simulation is not real time, meaning an AI interacting with these scenes must be time managed.

The final components of the VANE are the blue boxes in Fig. 1, which represent the physical models of the VANE. The models depicted are "Dynamics" and "Sensors." The Dynamics models include the Vehicle-Terrain Interface (VTI), which models the forces on wheeled vehicles based on ground and vehicle properties, as well as the forces on any other dynamic objects in the scene. The Sensors section includes LADAR, cameras, Global Positioning System (GPS), accelerometers, and gyroscopes.

The three main types of sensor models currently used in the VANE are GPS, inertial sensors (gyroscope, accelerometer), and electromagnetic sensors like LADAR and cameras. The electromagnetic sensors were introduced in a previous work [3]. Both the camera and LADAR sensors use a raycaster, called the QuickCaster (QC), to input the appropriate radiance values into the sensor models. The QC therefore incorporates the attributions and fidelity of the environment with detailed ray casting as a first step in creating realistic sensor output.

2 The Quick Caster

The QC is a parallelized ray-casting code that is optimized to render images of very detailed scenes. Scene geometries are represented in the QC as meshes of triangular facets. A typical desktop computer will only be able to render scenes with a few million facets, perhaps up to ten million, but the QC is capable of rendering scenes with over 300 million facets. This allows the scene geometries to be very detailed and encompass a large physical extent in the simulation.

The QC is run on a Cray XT4, called Jade, which is available through the DoD Supercomputing Resource Center (DSRC). Jade has 2,146 2.1-GHz AMD Opteron 64-bit quad-core processors with 8 GBytes of dedicated memory each. Most parallel ray casters are parallelized on rays, pixels, or parallelized by object or spatial location. However, in order to take advantage of the large number of processors available through the DSRC while also ensuring load balancing, the QC was parallelized by facets. The facets comprising a scene are distributed round-robin style across all the available processors. Each processor then performs the ray-casting routines (triangle intersections) on its facets. The results are combined at the end to determine which facets are in view, which are blocked, and which are closest. After that, the QC can be used in different ways to provide input radiance data for the camera and LADAR sensors and ranges for the GPS.

Input for the Charge Couple Device (CCD) camera model comes from the QC as an array of radiance values based on each facet's reflective properties and the solar loading. The corresponding distances for the rays are also passed to the camera model from the QC. These array values are used by the CCD model to create an ideal image. Similar data values are also passed for the LADAR model but rays are only traced from the sensor into the scene. The pinhole camera model is used in a novel way to account for the diverging beam in the LADAR model.

3 Generic CCD Model

The CCD model used in the VANE is based on ray optics. Conceptually, the model is quite similar to that proposed by Subbarao and Nikzad [4], which models the camera as a series of algebraic and convolution operations on an initial ideal image. These operations include filtering, vignetting, spectral response, exposure time, and lens optics. Subbarao and Nikzad propose a model using the Point Spread Function (PSF) to map the rays of the ideal image to the real image, while the CCD model in the VANE uses an explicit ray tracing based on the third-order Seidel abberations as described in [5].

An example of a CCD model image is presented in the right part of Fig. 2. The corresponding ideal image is shown in the left part of the same figure. There are several important features in the model image which may have an effect on machine vision or image processing. These include pixel saturation, color shifts due to filter and CCD response functions, spherical aberration, and intensity vignetting. The CCD model used in the VANE accounts for all these effects, as shown in Fig. 2

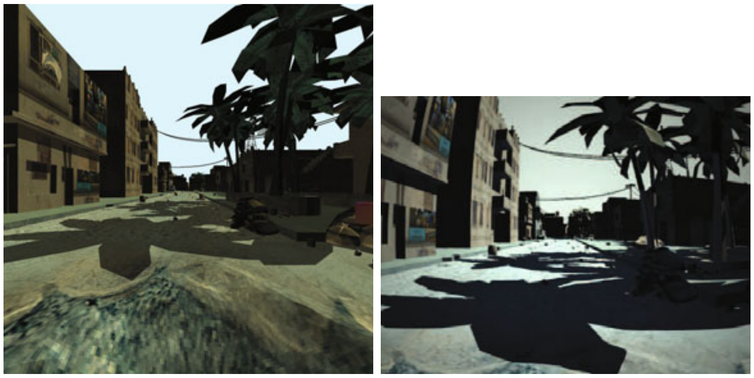

Fig. 2. A comparison of an ideal (left) and CCD model (right) image. Effects reproduced in the CCD model include vignetting and aberrations.

At the present time, the CCD model is notional and has not been validated against a particular camera. However, the model is highly parameterized, allowing several different types of CCD cameras to be simulated with the same model. More specific mathematical details of the CCD sensor model can be found in [3].

4 Scanning LADAR

The model presented here is roughly representative of the Sick LMS-291, for which mathematical models have previously been developed by [6] and [7]. Our model differs from these in that we use ray-tracing through an ideal-image plane to define the laser pulse shape and divergence by using a virtual focal length and detector dimensions.

It was shown in our previous work [3] that accounting for beam divergence effects in LADAR simulations produces more realistic results than simple pencil-ray tracing or z-buffer LADAR. Beam divergence for the SICK LMS-291 line scanning LADAR has been modeled in VANE by tracing between 100 and 10,000 rays for each pulse of the laser. As we previously determined, the necessary number of rays (N) is given by

$$N = \frac{2(r + R_{max} \tan(\frac{\Theta}{2}))}{x} \tag{1}$$

where x is the scale of the spatial variation of the target, r is the aperture radius of the receiver, R_{max} is the the maximum range of the LADAR, and Θ is the the divergence of the beam.

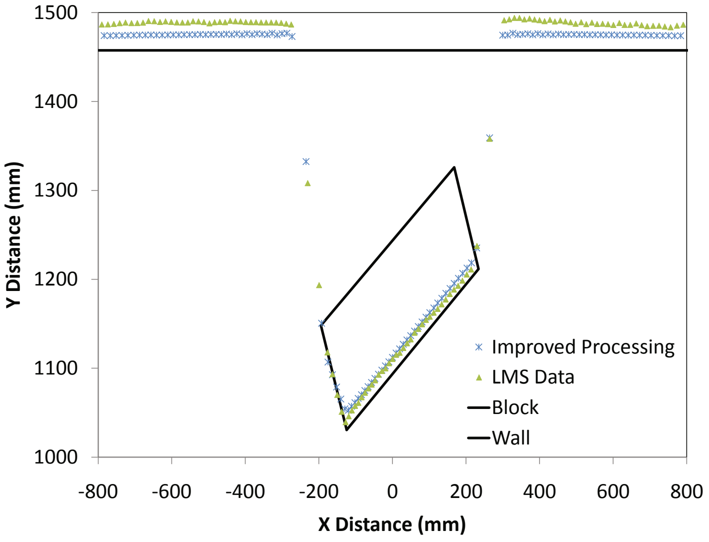

Fig. 3. A schematic of a simple experiment used to determine the signal processing algorithm for the SICK LMS-291. In this plot, the real data are represented by the green triangles, and the improved processing algorithm are the blue stars. The truth data for the wall and block are represented by the solid black lines.

The rays cover the spatial extent of the beam spot, which can be close to one meter diameter at the maximum operating distance of the LADAR, which is about 80 meters. The intensity distribution of the beam spot is modeled as a Gaussian beam profile. Each ray carries a radiant intensity corresponding to its position in the profile, and the reflected intensity is calculated based on the properties of the materials in the environment. All the return intensities are time stamped and summed to produce a return pulse.

Since our previous work, we found that the processing of this signal can have a significant impact on the distance returned. We have used a trial-and-error method with a real system in a set of controlled experiments to determine as closely as possible the correct signal processing to yield the most realistic results. A schematic of the experiment is shown in Fig. 3. A rectangular block was placed in front of a wall and scanned with the SICK LMS-291 LADAR. The key feature of the resulting data are the "floating" points or mixed pixels between the block and the wall. The figure shows how the data from the model with improved signal processing closely match the real data.

5 GPS Sensor Model

The GPS model finds the receiver's position using a geo-specific, pseudorange method that calculates position using the distances between the receiver and the GPS space vehicles (SVs). The model operates in three steps. First, it uses the QC to find the true ranges, R, between the receiver and the SVs. The initial SV positions are specific to the site at which the receiver is being simulated, and they are taken from the Scripps Orbit and Permanent Array Center (SOPAC) archive [8]. At each time step during the simulation, the SV positions are updated using Kepler's equations [9] and the ephemeris (orbit parameter) data contained in the SV broadcasted RINEX file [10]. Once the SV positions have been determined, the QC is used to check whether or not the SV is blocked by objects in the scene and to check for multipath effects.

After finding the unblocked SVs and their distances from the receiver, the model takes the true range (or multipath range) and alters it using several error functions. The effects of the Earth's atmosphere are modeled using three different error functions. Ionosphere range errors (dR_{iono}) are modeled using a Montenbruck mapping function [11] with the constants derived from the standard range error at zenith. Likewise, the effects of the upper Troposphere ($dR_{trop,dry}$) are modeled using a Neill mapping function [12]. the detailed derivation of which can be found in [13]. The range errors caused by the lower atmosphere ($dR_{trop,wet}$), or 'hydrostatic Troposphere', are modeled using the equations derived by Park and Ha in [14].

In addition to atmospheric effects, the model also accounts for range errors due to SV onboard ephemeris errors and errors in the GPS system atomic clock. As the SVs orbit the Earth, errors in their internal calculations can degrade the SV position reported to the GPS receiver. Errors in the reported SV position in turn create errors in the SV-receiver range. These range errors are modeled using an equation derived from the work of Hwang and Hwang and Parkinson [15] [16]

$$dR_{SV ephemeris} = \sqrt{(1 + \frac{1}{2}\sin\theta)(10^{\frac{2R_{ref}}{R_0}}) + c^2(10^{-18}) + 0.918} \qquad (2)$$

where θ is the SV's latitude, R_{ref} is a reference orbit radius of 26550000 meters, R_0 is the SV orbit radius, and c is the speed of light in a vacuum.

Once all of the error effects have been accounted for, a new distance between the receiver and SV is calculated

$$R' = R + dR_{iono} + dR_{trop,wet} + dR_{trop,dry} + dR_{SV ephemeris} \qquad (3)$$

The new ranges, R', are then fed into a generic trilateration algorithm, which computes the receiver's position. A detailed discussion on how GPS receivers find position using SV-receiver ranges can be found in [16]. By using this method, the receiver's returned position will suffer from errors caused by poor SV geometry, an effect known as dilution of precision (DOP). Fig. 4 shows the effects of each error source on the GPS receiver output position during a test run around a circular track.

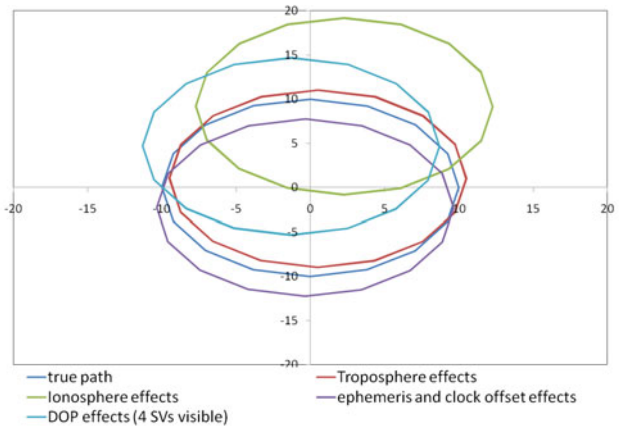

Fig. 4. Output GPS receiver position for each individual error source

6 Results

Several tests of VANE capabilities were conducted to demonstrate the capabilities of the VANE. The three key pieces of a simulation are

1. The environment. This includes the geometry of the environment, the attributions, and the trafficability and soil parameters affecting navigation.

Environments can either be geo-specific (exact representation of a particular latitude and longitude) or geo-typical (a non-specific representation of a type of place, like a forest). Environments of both types were constructed to test the VANE.

2. The sensor package. Physics-based sensors in the VANE include CCD, LADAR, GPS, gyroscope, and accelerometer. Many sensor packages on existing robotic platforms can be reproduced using only these sensors. Other sensors, such as vehicle sensors like wheel encoders, are currently being developed
3. The ADr. We tested the VANE with two simple software ADr for truck-sized robots. The first is a simple waypoint follower that issues driving commands with no regard for sensor information. The other is a slightly more advanced path planner that uses a two dimensional "go/no-go" map generated from LADAR data to navigate through a scene.

Figures 5 shows an example of a geo-typical scene that is roughly representative of a Middle-Eastern city. An initial test using this Middle-Eastern scene was conducted using the simple path planning ADr. Partial results from this test are shown in Fig. 6, which shows the map that was generated by a robot driving through the streets with two panning SICK LADAR mounted on the front and rear of the robot. The map was generated from the resulting (x,y,z) point cloud. The green line in this figure is drawn from the robot's current position to the desired waypoint.

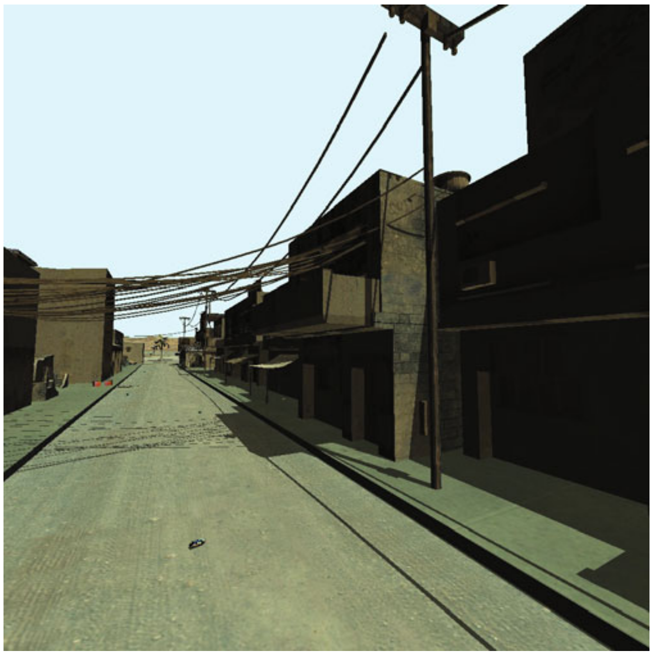

Fig. 5. A street-level view of the geo-typical Middle-Eastern city

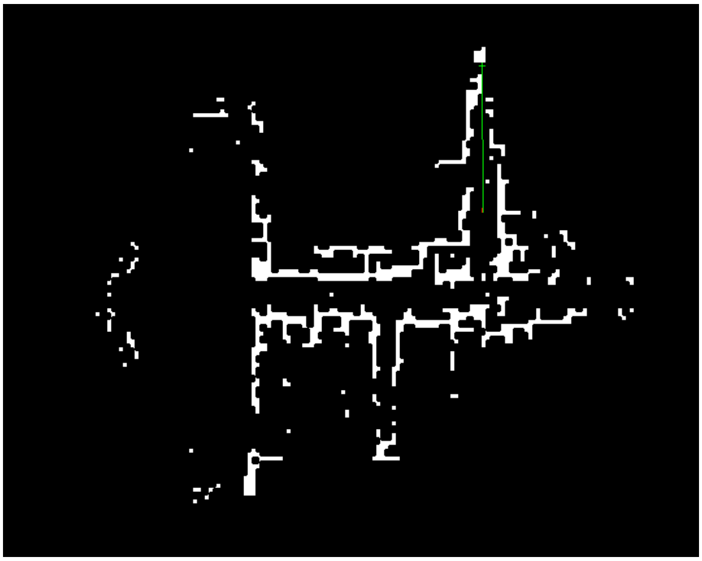

Fig. 6. A "go/no-go" map generated from a robot with a single panning SICK LADAR driving through the city

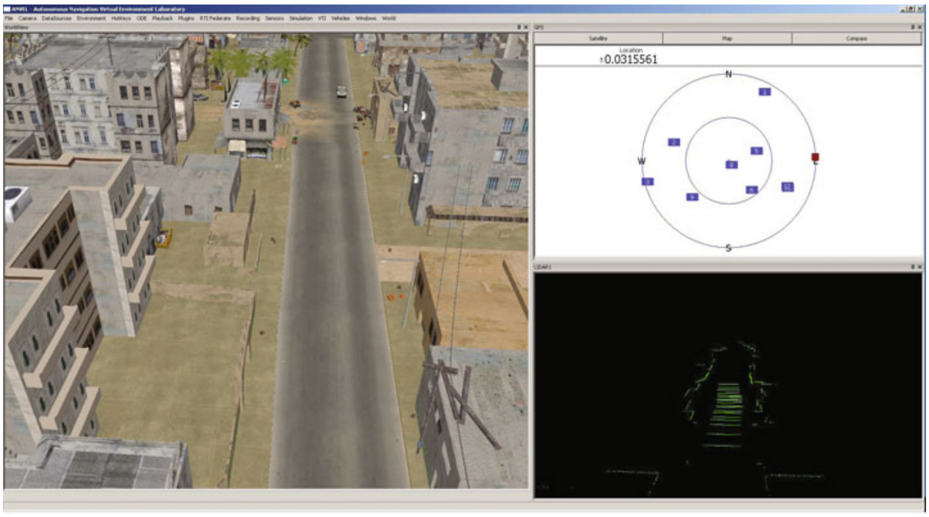

Fig. 7. VANE simulation replay in the ANVEL. The left part shows the simulation environment. Te upper right is the GPS model, and the lower right is the LADAR point cloud.

Post processing of VANE simulations, including replay, debugging, mission planning, and interactive data viewing, are accomplished with the Autonomous Navigation Virtual Environment Laboratory (ANVEL) [17], which can be used

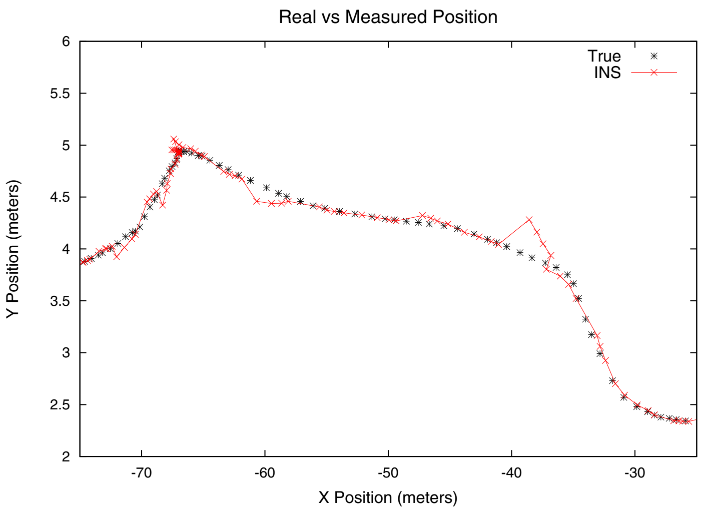

Fig. 8. Comparison of the true and perceived path as calculated by Inertial Navigation System (INS) using IMU and GPS data. The points are spaced at $\frac{1}{4}$ of a second in time. The GPS updates occurred at 1 Hz, and the IMU was used for localization between GPS updates.

to view LADAR, GPS, and image data and replay the path of the vehicle and other objects in the simulation. A screenshot of ANVEL in the Middle-Eastern city is shown in Fig. 7, which shows the world model on the left, the GPS model data on the top right, and the LADAR point cloud on the bottom right. Note that the LADAR point cloud includes intrinsic sensor noise, environmental noise, and registration error caused by GPS error.

Several initial tests of the VANE simulation process have been completed and demonstrate the results of incorporating sensor noise and realistic intrinsics in the simulation. One result of using realistic GPS and Inertial Measurement Unit (IMU) models is shown in Fig. 8, which shows the vehicles real path compared to its perceived path from IMU and GPS data. Further tests in this environment are ongoing.

7 Future Work and Conclusions

The VANE is still in the initial stages of development. Future work will focus on the simulation architecture, environmental attributions, developing new sensors, and verification and validation. In the area of attributions, we are specifically focused on adding directional spectral reflectance properties to the objects in the environment by incorporating a Bidirectional Reflectance Distribution Function to the material attributions of each object. Models for the Velodyne and Hokuyo LADAR are also currently under development, and our verification and validation efforts will focus initially on validation of these models.

In conclusion, we developed a high-fidelity physics-based simulation environment to aid in the development and testing of autonomous navigation algorithms for unmanned ground vehicles. Initial tests indicate that the environment provides a virtual proving ground for autonomous vehicles and that the sensor data used to feed the robot perception closely resembles real sensor data. Potential development partners for the VANE should contact the authors.

Disclaimer

Permission to publish was granted by Director, Geotechnical and Structures Laboratory.

References

1. Craighead, J., Murphy, R., Burke, J., Goldiez, B.: A survey of commercial & open source unmanned vehicle simulators. In: 2007 IEEE International Conference on Robotics and Automation, pp. 852–857 (2007)
2. Balakirsky, S., Scrapper, C., Carpin, S., Lewis, M.: UsarSim: providing a framework for multirobot performance evaluation. In: Proceedings of PerMIS, Citeseer, vol. 2006 (2006)
3. Goodin, C., Kala, R., Carrrillo, A., Liu, L.Y.: Sensor modeling for the virtual autonomous navigation environment. IEEE Sensors (2009)
4. Subbarao, M., Nikzad, A.: Model for image sensing and digitization in machine vision. In: Proceedings of SPIE, vol. 1385, p. 70. SPIE, San Jose (1991)
5. Born, M., Wolf, E.: Principles of Optics, 6th edn. Pergamon Press, Elmsford (1980)
6. Tuley, J., Vandapel, N., Hebert, M.: Analysis and Removal of artifacts in 3-D LADAR Data. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2005, pp. 2203–2210 (2005)
7. Rong, L., Chen, W., Yu, S., Song, W.: Mathematical model of coordinate transformations for 3D Depth-of-field collection system. In: 6th IEEE International Conference on Industrial Informatics, INDIN 2008, pp. 80–85 (2008)
8. SOPAC: Scripps orbit and permanent array center (2010), `http://sopac.ucsd.edu/cgi-bin/dbShowArraySitesMap.cgi`
9. Mehrtash, M.: Gps navigation toolbox gnt08.1.2 GNT08.1.2 (2008), `http://www.mathworks.com/matlabcentral/fileexchange/20578i`
10. RINEX: Receiver independent exchange format, ver. 2.10 (2009), `http://gps.wva.net/html.common/rinex.html`
11. Montenbruck, O., Garcia-Fernandez, M.: Ionosphere Path Delay Models for Spaceborne GPS. Deutsches Zentrum für Luft- und Raumfahrt DLR-GSOC TN 05-07 (2005)
12. Niell, A.E.: Improved atomspheric mapping functions for VLBI and GPS. Earth Planets Space 52, 699–702 (2000)
13. Niell, A.: The IMF Mapping Functions. In: GPSMet Workshop (2003)
14. Kim, H.I., Ha, J., Park, K.D., Lee, S., Kim, J.: Comparison of Tropospheric Signal Delay Models for GNSS Error Simulation. Journal of Astronomy and Space Sciences 26, 21–220 (2000)

15. Hwang, C., Hwang, L.S.: Satellite orbit error due to geopotential model error using perturbation theory: applications to ROCSAT-2 and COSMIC missions. Computers and Geosciences 28, 357–367 (2000)
16. Parkinson, B., Spilker, J.: The global positioning system: theory and applications. In: Aiaa (1996)
17. Rohde, M., Crawford, J., Toschlog, M., Iagnemma, K., Kewlani, G., Cummins, C., Jones, R., Horner, D.: An interactive physics-based unmanned ground vehicle simulator leveraging open source gaming technology: progress in the development and application of the virtual autonomous navigation environment (VANE) desktop. In: Gerhart, G.R., Gage, D.W., Shoemaker, C.M. (eds.) Unmanned Systems Technology XI. SPIE, Orlando (2009)