Validating The Search and Rescue Game Environment As A Robot Simulator By Performing A Simulated Anomaly Detection Task

Jeff Craighead, Rodrigo Gutierrez, Jennifer Burke, and Robin Murphy

***Abstract*— This paper presents the results from experiments validating the physics and environmental accuracy of a new robot simulation environment, the Search and Rescue Game Environment (SARGE), which is the foundation for series of robot-operator training games. An ATRV-Jr. outfitted with a SICK laser, GPS, and compass was used both in the real-world and in a simulated environment modeled after the real-world testing location in a simulated anomaly detection task. The ARTV-Jr., controlled by the Distributed Field Robotics Architecture, navigated through a series of waypoints in the environment. The simulated ATRV-Jr. matched the actions of the real ATRV-Jr. in both velocity and path similarity within 0.08m/s and 0.7m respectively.**

I. Introduction

This paper presents the results from experiments which validate the physical accuracy of a simulated ATRV-Jr. in the Search and Rescue Game Environment (SARGE) as well as the method of constructing the virtual environments in SARGE. The primary goal of the SARGE project is to create open source single and multiplayer video game that can be used for training individuals and teams of robot operators. SARGE can also be used as a simulation environment that can interface with any network capable robot architecture. In the last few years there has been a renewed interest in robot simulators. There are many commercial and open source simulators available such as Webots [1], USARSim [2], SimRobot [3], Player/Stage/Gazebo [4], Microsoft's Robotics Studio [5] and many others. However none of these simulators target robot operator training as SARGE does, nor do they attempt to bring the field of serious games into HRI research. Of these only USARSim has had published results for validation experiments.

Jeff Craighead, Rodrigo Gutierrez, Jennifer Burke, and Robin Murphy are with the Institute for Safety, Security and Rescue Technology at the University of South Florida. 4202 E. Fowler Avenue, Tampa, FL, USA. {craighea,rgutier2,jlburke4,murphy}@cse.usf.edu.
This work was sponsored, in part, by the US Army Research Laboratory under Cooperative Agreement W911NF-06-2-0041. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ARL or the US Government. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. All aerial photography provided by GoogleEarth.

The general consensus among researchers in the field of serious games is that video games have great potential for use in education and training beyond the simple reading, writing, and mathematics of the past. Not only can games be used as an entertaining teaching tool for young students, but for adults as well in a variety of non-education fields [6], [7], [8], [9]. SARGE will take the next step in this area by combining robot simulation, action games, and an intelligent tutoring system to facilitate increased robot operator performance. This necessitates the validation of the accuracy of the environments and the capabilities of the robots within SARGE.

The remainder of this paper is divided into six sections. Section II discusses the validation of other robot simulators. Section III discusses the basic implementation and features of SARGE. Section IV describes the plan for the validation experiment. Section V describes the scripts used to analyze the data collected during the experiment. Section VI describes the results of the analysis. Finally Section VII provides a summary of this article.

II. Related Work

Previous work [10] surveyed many of the available commercial and open source robot simulators. Aside from the work on the USARSim project there does not seem to be much work on validating robot simulators themselves but instead the literature focuses on validation of robot prototypes in simulation. One cannot simply assume that a simulator will work as described in marketing literature which lacks empirical evidence. As Wang states in [11], "To draw valid conclusions from robotic simulations it is important to know the metrics which are consistent with the operation of the actual robot and those which are not." The developers of USARSim have performed a rather thorough series of tests in order to validate the similarity between performance of a robot or sensor in USARSim vs the performance of the real-world counterpart.

In [11] Wang, et al. performed a series of experiments to validate USARSim. The first stage of their experiments was focused on validating the physical accuracy of the simulator. The experiment compared velocities and sensor readings for a real and simulated robot to minimize the error of the simulated robot. Additionally, the paper describes the use of a laser range finder to verify the mapping capabilities of the simulated robot [12]. The second stage was focused on verifying the HRI capabilities of USARSim and involved participants performing various tasks in a simulated version and the real world NIST Orange arena.

While the goal of the SARGE project is to create a training game and is not intended to be a simulator per se, it is being used as such and for this reason it must be validated. This validation also aids the goal of gaming using physically accurate representations of the real world and of the robots. Our experiment falls between the two USARSim validation experiments in terms of complexity. We compare velocities and pose in a simulated version of a real world outdoor area while the robot autonomously performs a simulated anomaly detection task. The experiment and anomaly detection task are described in greater detail in Section IV.

III. Implementation

This section describes the implementation of SARGE including a discussion of the Unity engine and the communication protocol used to interact with external architectures such as the Distributed Field Robotics Architecture (DFRA).

A. Architecture

SARGE is built using the Unity game engine, which is developed by Unity Technologies. Unity was chosen after abandoning UnrealEngine 2 because of the aging Karma physics engine, buggy world editor, lack of documentation and distribution issues. Newer physics engines such as ODE, Havok, and PhysX provide higher fidelity physics than Karma and the primary concern with distributing an UnrealEngine mod is that the user must own a copy of UnrealTournament 2004 and have the experience necessary to install the mod on top of Unreal. The UnrealEd world editor application is also very buggy, crashing often, and supports only a limited set of media file types. Because the SARGE project was going to include environments and robots that were not available in any existing simulator, switching engines did not increase development time. In fact moving to Unity increased productivity because of the well written scripting documentation.

Unity uses the PhysX engine provides the best development environment and licensing terms when compared to many game engines including Torque, Source, Ogre + ODE, CryEngine, and others. Unity is designed to be easy to use for professional developers and novice users alike, using JavaScript and C# along with the Mono framework for scripting objects and interaction with other applications. Content can be created in a variety of formats which Unity reads natively. These features allow SARGE to have a simple architecture that make developing new simulator content easy for even novice users. Additionally the Unity allows a license holder to freely distribute any applications created to an unlimited number of users who will not have to purchase any additional applications to run the content. This is unlike USARSim or other mod based simulators which require the end user to own a copy of the game on which the mod is based. Unity also provides a web browser plugin, similar to Adobe's Flash player, which allows applications to be run in the browser for users who are unable or do not wish to install applications on their PCs.

Communication with external controllers such as the Distributed Field Robotics Architecture is implemented using the Mono Sockets library. A controller can simply establish a TCP connection on a LAN or over the Internet to SARGE and use a simple text based protocol that identifies the target robot and service to issue a command to. While the ASCII protocol does incur at least a 25% overhead vs a pure binary protocol it makes debugging and rapid prototyping much simpler.

Content creation relies heavily on Unity's prefab concept. Textured objects, modeled in an external application, are imported into the SARGE project workspace. These objects are then set up with all the necessary physics properties and, control and sensor scripts. The objects are then turned into Unity prefabs. Prefabs can be instantiated at any time, in any location in the game world.

B. Sensors and Effectors

SARGE includes the typical sensors found on a robot including a laser range finder, compass, global positioning system, inertial measurement unit, and cameras. Sensors in SARGE are implemented without a defined class hierarchy because they are not normally spawned at runtime. Instead sensors are scripts or prefab objects which are placed on the robot's prefab by the creator.

SARGE implements several laser range finders based on the SICK LMS 200 series range finder, error is added to the range reading to be consistent with the SICK LMS 200 documentation, that is +/- up to 4mm for each range reading. The compass and GPS in SARGE return readings based on a reference object that has been geo-referenced using GoogleEarth. The reference object is assigned real world GPS coordinates and placed with its Z-axis facing north. SARGE calculates the offset distance and rotation from the reference point to determine the real world location and heading of a robot equipped with these sensors. The inertial measurement unit (IMU) in SARGE is a 6-degree-of-freedom IMU that reports 7 measurements. Values provided by the IMU include both linear and angular velocities on 3 axes, both linear and angular accelerations on 3 axes, as well as pose (roll, pitch, and yaw) of the unit.

SARGE supports a virtually unlimited number of cameras on each vehicle. It is possible to create an interface that displays some or all of a robots cameras simultaneously. Additionally images from simulated cameras can be sent over a network connection for display on another device such as a physical mock up of a robot controller. Cameras can be placed at any position on a robot or in an environment. However each additional camera requires the entire scene to be rerendered, which can cause significant performance loss in terms of display frame rate. This can be avoided by switching cameras on and off as needed.

C. Robots

The current version of SARGE include three vehicles: an ATRV-Jr., the USF Sea-RAI (a portable surface vehicle for search and rescue and security operations), and the iSensys IP3 (a small R/C helicopter based UAV). The ATRV-Jr. carries a typical complement of sensors including a compass, IMU, GPS, camera, and laser range finder. The Sea-RAI carries a similar sensor suite. The IP3 has a limited carrying capacity, as such it carries only two cameras. The simulated motors of the robots work by applying forces and torques on the vehicle. The motors are tuned so that the performance of the simulated vehicles in terms of linear and angular velocities are as close as possible to measured values from corresponding real world vehicles.

D. Environments

SARGE includes four environments, both indoor and outdoor. One environment is a model of the robot testing and training facility being constructed at the University of South Florida. The facility includes a 3 story tower, a rubble-crushed car, and underground sewer pipes. The second environment is based on a pier located in Pensacola, Florida and covers several square kilometers of the surrounding ocean. The third environment is a 100m x 100m x 100m empty room with yellow grid lines spaced at roughly 1m intervals and was inspired by the Star Trek "holodeck". The fourth environment was constructed to perform the analysis for this paper and is a duplication of an area on the University of South Florida campus near the Research Park.

IV. Experimental Design

This section describes the experiment performed to compare real world performance of an ATRV-Jr. to that of a simulated ATRV-Jr. in a virtual version of the test environment. The robot was tasked to autonomously follow a series of waypoints through an open but weighted area on its a priori map. The real environment contains an area with many trees, to test the anomaly detection behavior the trees were not included in the a priori map. The robot would then report any anomalies found in the area and reactively respond in a safe manner. The robot should behave more cautiously (slow down) in areas in which the map was marked as less traversable as well as in areas that contain anomalies. This terrain traversability measure would ideally be measured on the fly using a sensor and the map updated accordingly, however the sensor that was to be used for this was not complete at the time of the experiment. This scenario forces multiple unplanned deviations from the given route which would significantly affect the results if the simulator does not accurately represent the real world.

As described in Section III, the SARGE environments are constructed using GoogleEarth images, which are typically accurate to less than 1 meter, as templates for accurate placement of objects. While GoogleEarth has been show to have positioning errors far larger in some cases, the GoogleEarth reported GPS coordinates were verified with a hand held GPS unit. Additionally since buildings are placed visually by hand, even environments that cover several square-kilometers will have buildings placed correctly relative to one another. SARGE uses a single reference point in the simulated environment which is tagged with a real-world GPS location to calculate the GPS coordinates and heading of robots. It is this design method that the experiment was designed to validate by showing that the simulated robot travels along the same path as the real robot in roughly the same amount of time. Both the real robot and the simulated robot run the Distributed Field Robotics Architecture [13], which allows the same navigation behaviors to be used on both.

The first step in validating the design method was to duplicate our outdoor testing environment in the simulator. The environment is located on the University of South Florida campus next to the Research Park parking lot. Figure 1 shows the reference point in GoogleEarth. The interested reader can point GoogleEarth or other mapping software to 28.05711 N by 82.41460 W to examine the area in greater detail.

The second step in the validation process was to determine the path the robot should take. We chose a path that would activate both the obstacle avoidance and caution behaviors on the robot to force reactive course corrections. These course corrections should happen at roughly the same point in the both paths if the virtual environment is a reasonable facsimile of the real environment. Figure 1 shows the path the robot will be directed to follow. The MoveToWaypoint (MTW) behavior in DFRA will be responsible for moving the robot to each of the given waypoints. MTW uses a threshold value of 5m for stopping the robot at a given waypoint. This may seem like a rather large value, however this is to allow for slight drift in the waypoint location due to GPS error and to prevent the robot from wandering if a waypoint is partially inside an obstacle.

The next step was to conduct the experiment in the real-world as well as in simulation while logging the robots heading and position along with a timestamp. This data allows us to later calculate the difference in position and velocity between the real robot and the simulated robot. To conduct the experiment the robot was placed in the environment and instructed to drive to the starting point. A script was then executed to command the robot to travel to each waypoint in the path in order. As the robot travels along the route it marks anomalies on its map, slows down and reactively avoids the anomaly. The data was logged using a DFRA service that listens for messages from the sensor services on the robot. The sensor data is then stored in a mySQL database. A set of PHP scripts is then used to create a GoogleEarth KML file from the sensor logs for later analysis.

V. Data Analysis

This section describes how the pose data logged during the experiment was analyzed. Using a Perl script each pose point and corresponding time stamp was extracted from the KML files used by GoogleEarth. The pose points were saved into a comma separated file and passed to another Perl script which computed the average velocities of each robot and the similarity between path taken by the simulated robot and that of the real robot.

To compute the average velocity of a robot the velocity between each pose point along the path[1] was calculated. The average of these velocities was taken to be the average robot velocity. The distance between each point along the path must be known for this calculation, however the raw pose data contains the robots position in decimal degrees and thus the difference between these points is also in decimal degrees. This difference was converted to a distance for easy readability using the following formula.

$$distance = 6378137.0 * \cos(lat) * \delta \tag{1}$$

$$\delta = \sqrt{(p1_{lat} - p2_{lat})^2 + (p1_{lon} - p2_{lon})^2} \tag{2}$$

Equation 1 calculates the distance between GPS coordinates in meters as follows: 6378137.0 is the radius of the earth at the equator in meters, which is multiplied by the cosine of the latitude of the reading to get the nominal radius of the earth at the current latitude. This is then multiplied by the angle between points measured in radians δ, which is shown in 2. This gives the distance in meters.

To compare the similarity of the two paths we again use Equation 1. For each point in the real robot path the script identifies the closest point in the simulated robot path based the distances found using Equation 1. From this list of matching points we find the maximum, minimum, and average distances along with the variance and standard deviation. The standard deviation is used to then recalculate the average after pruning distances which are greater than 2σ. This pruning was done to eliminate a set of points which were known to be caused by GPS error. These points can be seen in Figure 2 Pose 158 through 166 are caused by GPS error as during this time the robot was stopped under the tree at the "Under Tree" waypoint.

VI. Results

The analysis of the logged sensor data show that the performance of the simulated robot in the simulated environment matches that of the real vehicle within the 3m error tolerance of GPS. This experiment was repeated 5 times. During each run, the real robot maintained the same path within 3m, which is the expected performance based on the use of GPS for positioning. The simulated robot maintained the same path within 0.5m because the simulated GPS reports perfect position information every time step. For the following analysis we focus the run corresponding to Figure 2. These results are consistent with the results from the other runs.

The maximum deviation between the two paths as measured between closest points is less than 1.51m (σ=1.35) and if we remove several outliers caused by GPS drift when the robot was stopped under trees the average deviation is less reduced by more than half to

[1]Pose points that were closer than 0.2m were ignored to eliminate points during times when the robot was stopped.

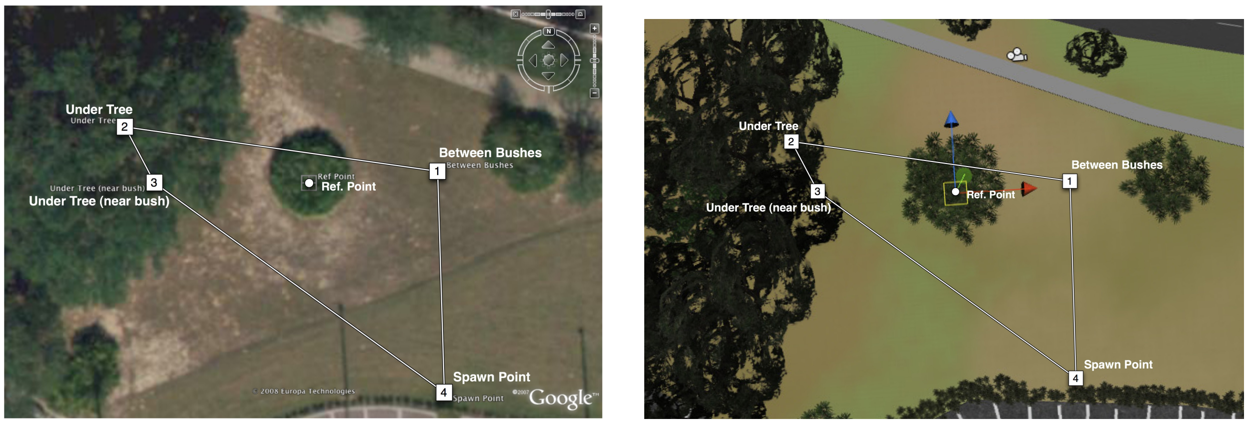

Fig. 1. This image shows the linear path between waypoints the real and simulated robots should follow as well as the reference point used in SARGE.

0.71m. The velocity of the simulated robot matched that of the real robot within 7% with the simulated robot traveling at an average of 1.06m/s while the real robot traveled at an average of 0.99m/s. Figure 2 shows the recorded positions of the robots. The red markers represent the real robot while the blue markers represent the simulated robot. Note that all distances are measured using the GoogleEarth measurement tool and the positions logged from the experiment.

At the start of the experiment the robots were commanded to drive to the starting location, this is labeled "Spawn Point" in Figure 1 and Figure 2. After driving autonomously to the starting point the simulated robot began 2.1m from the intended position which is within the 5m threshold of DFRAs MoveToWaypoint behavior. After driving autonomously to the starting point the real robot began 6.6m from the intended postion, which is outside the 5m threshold of DFRAs MoveToWaypoint behavior. This indicates that the GPS error in the field during the experiment was between 1.6m and 4.5m.

As the script was executed the robots headed toward the "Between Bushes" waypoint. The robots stopped between the bushes, both within 5m of the waypoint. The simulated robot stopped 2.08m from the waypoint and the real robot stopped 2.88m from the waypoint. The next waypoint the robots were sent to is the "Under Tree" waypoint. The direct path would send the robot through the middle of a large palmetto bush. The robots initially head straight towards the bush and then reactively avoid the bush as shown in Figure 2. The real robot maintains a closer distance to the bush which may indicate that the real laser is reporting that the robot is farther away from the bush than it actually is due to low reflectivity of the bush and the filtering algorithm that ignores any non-consecutive points. In simulation this is not a problem because the laser returns the exact distance to hit point on the object for each ray plus or minus a few millimeters of random error. As the robots round the bush they head for but do not successfully reach the "Under Tree" waypoint due to the caution behavior being activated. As the robot enters the tree line it reaches the less traversable area as marked on its map and the robot slows down significantly as the caution behavior activates.

The third waypoint, "Under Tree (near bush)" is 7m from the previous waypoint. The robots are commanded to head for this waypoint just before they reach "Under Tree". Because of the very slow speed both robots stop just inside the 5m boundary. Finally the robots are sent back to the initial location "Spawn Point".

Again the real robot stays closer to the bush while the simulated robot veers around it. The simulated robot stops on exactly the same spot it started in (which is 2.1m from the waypoint). The real robot stopped 6.8m from its initial starting point. This may have been caused by the timer expiring a few seconds before it reached the stopping location or because of GPS drift or a combination of both. This is consistent with the data analysis that shows the simulated robot is slightly faster than the real robot.

VII. Discussion and Summary

This paper presented the analysis of a validation experiment for the Search and Rescue Game Environment (SARGE), a new simulation environment that forms the basis for a multiplayer robot operator training video game. The validation experiment consisted of running a real ATRV-Jr. in an outdoor environment and a simulated ATRV-Jr. in a simulated version of that environment. The robots were tasked to move around a series of

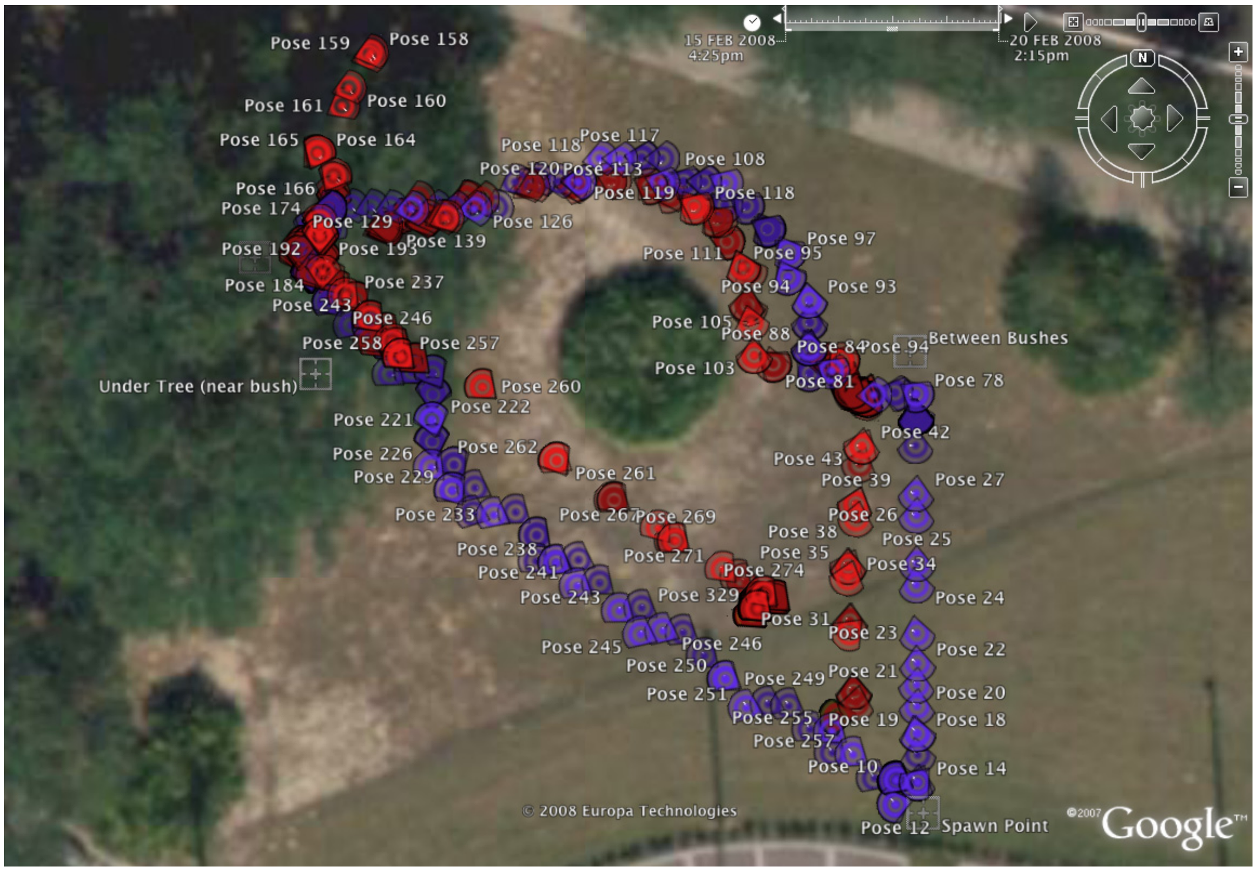

Fig. 2. A comparison of poses. The simulated robot's positions are marked in blue while the real robot's positions are marked in red.

waypoints in an anomaly detection task. The anomaly detection task consisted of the robot following a pre-planned path using what it understood to be a good map; anything that was not on the map was considered an anomaly. When the robot encountered an anomaly, it would change its behavior to be more cautious (slow down), which is shown as the robot approaches the group of trees (anomalies) to the west of the test area. The similarity between the real and simulated paths was computed along with the average velocity of the robot based on the pose data recorded during the run. The analysis showed that the simulated robot traced the same path as the real robot with an average deviation of 0.7m. The simulated robot was 0.08m/s faster (7%) than the real robot based on the calculated average velocity. These results are very promising despite only performing 5 runs.

In all five runs the robots consistently performed as expected. Due to the setup time needed to perform outdoor experiments, coupled with Florida's weather we determined the results from the first five experiments were good enough for our purpose of validating our world construction method and the physics model of the ATRV-Jr. The buildings and other objects within the simulated environment were placed within 1m of the real world location using our visual construction method using georeferenced GoogleEarth images. While validation of SARGE's other environments would be beneficial to showing that these environments also are a good representation of their real world counterparts, the environments either do not exist yet as in the case of the USF Testbed, or are very far from our location as in the case of the Pensacola pier. As well, since the main purpose of SARGE is to be a video game, perfect accuracy is not necessary, buildings that are placed "close enough" to their real locations are sufficient.

Now to focus on the path disparity. Figure 2 shows the starting location of the robots in the bottom right corner of the image. As mentioned in Section IV DFRA uses a 5m radius around a target waypoint for a stopping condition. The simulated robot begins and ends almost on top of the location marker. On the other hand the real robot is several meters away. This caused the path of the robots as they travel due North from the starting point to be 3m apart. Once the mid point is reached the paths begin to converge. The initial discrepancy may be due to wheel slippage caused by the sandy environment. The sand causes unpredictable wheel slippage which cannot be simulated to the degree that it occurs in the real world. This will affect the average speed of the vehicle.

Greater than 3m of error is likely when objects interfere with a GPS signal as can be seen in the top left of the image where the trees interfere with reception, however typical accuracy with an open view of the sky is about 1m. During the test runs we noted the predicted GPS error as reported by a handheld device to be less than 2m. The GPS error combined with DFRA's 5m stopping radius explain the discrepancy between the staring and stopping locations of the real vs. simulated vehicle. Up to 1m of error may be attributed to the use of GoogleEarth images due to the lack of resolution in the images. This does not appear to be the major cause of error however.

It appears that the largest cause for the path deviation is the obstacle avoidance behavior and its input sensor the SICK LMS 200 planar laser. The bush in the center of the image required the robot to deviate from its planned path while traveling both North-West (NW) and South-East (SE). The simulated robot avoided the bush when it approached within 5m while traveling NW and avoided the bush when approaching with 6.7m while traveling SE. The real robot approached within 1.7m of the edge of the bush before avoiding it when traveling NW and did not attempt to avoid the bush when traveling SE, despite approaching within 2.6m. This caused the real robot to take a path that was roughly 3m away from the simulated robots path while traveling to the SE. This error is caused by the lower reflectivity of the plant material which is not simulated in SARGE.

This shows that SARGE does accurately represent the environment and that it can duplicate the performance of a real vehicle within a small tolerance as measured by GPS. One may argue that GPS is not ideal for this type of measurement and that odometry may be better suited. However we used GPS because that is what is used in DFRA for robot localization at the time the experiments were conducted. While a fused GPS + Odometry system may be more accurate for localizing robots we were comparing DFRA + GPS running on a real robot to DFRA + GPS running in simulation and believe this is a fair comparison. Odometry alone would not be any better than GPS, instead of a bounded error around each point we would have an accumulated error, which due to the slippage in the sand would be extremely large.

Future work will continue to test SARGE's physics system with additional robots in both simulated indoor and outdoor environments. However, our main focus with SARGE will be on enhancing the feature set related to gaming and intelligent tutoring, then using it to conduct experiments with human operators to identify the utility of training with games for distributed, team-based robot operation.

References

[1] "Webots," http://www.cyberbotics.com/products/webots/. [Online]. Available: http://www.cyberbotics.com/products/webots/

[2] "Usarsim," http://usarsim.sourceforge.net/. [Online]. Available: http://usarsim.sourceforge.net/

[3] "Simrobot," http://www.informatik.uni-bremen.de/simrobot/. [Online]. Available: http://www.informatik.uni-bremen.de/simrobot/

[4] "Player/stage/gazebo," http://sourceforge.net/projects/playerstage. [Online]. Available: http://sourceforge.net/projects/playerstage

[5] "Microsoft robotics studio," http://msdn.microsoft.com/robotics/. [Online]. Available: http://msdn.microsoft.com/robotics/

[6] C. Fabricatore, "Learning and videogames: An unexplored synergy," in *The International Conference of the Association for Educational Communicataions and Technology (AECT)*, 2000.

[7] M. Pivec, O. Dziabenko, and I. Schinnerl, "Aspects of game-based learning," in *Third International Conference on Knowledge Management (I-KNOW 2003)*, 2003, pp. 216–225.

[8] B. Chamberlin, "Creating entertaining games with educational content: Case studies of user experiences with the children's website, food detectives fight bac!" Ph.D. dissertation, University of Virginia, May 2003.

[9] A. L. Aitkin, "Playing at reality: Exploring the potential of the digital game as a medium for science communicaiton," Ph.D. dissertation, The Australian National University, October 2004.

[10] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A survey of commercial and open source unmanned vehicle simulators," in *Proceedings of the 2007 International Conference on Robotics and Automation (ICRA)*, April 2007, p. 852.

[11] J. Wang, M. Lewis, S. Hughes, M. Koes, and S. Carpin, "Validating usarsim for use in hri research," *Proceedings of the 49th Annual Meeting of the Human Factors and Ergonomics Society, September*, 2005.

[12] S. Carpin, J. Wang, M. Lewis, A. Birk, and A. Jacoff, "High fidelity tools for rescue robotics: Results and perspectives," *Proceedings of the Robocup 2005 Symposium*, 2005.

[13] M. Long, "Creating a Distributed Field Robot Architecture for Multiple Robots," Master's thesis, University of South Florida, 2004.