Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

SIMULATION RESULTS FOR LOCALIZATION AND MAPPING ALGORITHMS

Doris M. Turnage
U. S. Army Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg, MS 39180, USA

ABSTRACT

The main goal of this research was to use simulation to compare the performances of three simultaneous localization and mapping (SLAM) algorithms and show the superiority of one algorithm's performance over the performance of the other two algorithms. The superior algorithm from the simulation experiments may be used to test an unmanned ground vehicle's (UGV's) capability to explore the complex subterranean environment for various Department of Defense (DOD) missions.

Using simulation shows the performance of these algorithms and aids in the development of a robotic platform that has the capability to perform the localization and mapping of subterranean environments in a cost effect manner. Simulation, using the robotic simulator STAGE, provided a platform to implement multiple algorithms easily in multiple topologies and to compare the performance of three algorithms: CoreSLAM, Gmapping, and HectorSLAM, in a cost effective manner without an actual robot.

1 INTRODUCTION

This paper uses simulation results to perform a quantitative evaluation of three laser-based SLAM algorithms, i.e., CoreSLAM, Gmapping, and HectorSLAM, implemented in the 2-D Stage simulator. The Stage simulator models sensors, cameras, and many features of an actual mobile robot. The simulated tele-operated robot explored three ground truth map images, and each produced a generated map image. The 2-D simulation environment, Stage, provides users with the capabilities of simulating a robot or a variety of robots in an environment or a variety of environments.

The three SLAM algorithms were tested and compared on three distinct topologies. Each algorithm required laser-based inputs as data for the simulation. The tele-operated robot explored three ground truth map topologies, and each produced a generated map image.

2 EVALUATED SLAM ALGORITHMS

Many types of SLAM algorithms exist. For example, those that are vision-based or laser-based, and those that are 2-D or 3-D. The three algorithms that were evaluated, i.e., CoreSLAM, Gmapping, and HectorSLAM, were available at www.ros.org. Each algorithm requires laser-based inputs as data for the simulation. The three SLAM algorithms collect data via the tele-operated robot within the robotic STAGE simulation, each outputting a map image of the navigated environment.

Each algorithm was used as a black box in this research; however, the underlying details of the implementation differ in the following aspects, i.e.,: (1) Both HectorSLAM and CoreSLAM rely on scan matching, while Gmapping uses particle filters, (2) CoreSLAM may produce a different map each time with the same input dataset, and (3) CoreSLAM requires loop closing while Hector SLAM does not.

2.1 CoreSLAM

CoreSLAM is a version of SLAM that implements tinySlam, requires a mobile robot that provides odometry data, and is equipped with a horizontally mounted, fixed, laser range-finder. The slam_CoreSLAM node will attempt to transform each incoming scan into the Odom (odometry) "tf" frame.

CoreSLAM relies on a simple Monte Carlo algorithm for scan matching and was developed by Steux and El Hamzaoui with the goal of producing a SLAM algorithm with no more the 200 lines of codes. CoreSLAM has a particle filter routine, ts_distance_scan_to_map, and a map update function. The ts_distance_scan_to_map routine tests each state position, and the map update function updates the map as the robot navigates its environment.

The slam_CoreSLAM node takes as input laser data and pose data collected from the laser range finder and outputs a low-quality map, yet a recognizable one.

Overall, CoreSLAM performed better on a slow robot.

2.2 Gmapping

Gmapping is a highly efficient Rao-Blackwellized particle filter that develops grid maps from laser range data (Grisetti, Stachniss, and Burgard 2007). Implementation requires a mobile robot equipped with a mounted, fixed, laser range finder. Loop closure is the hardest part; when closing a loop, the robot must be driven another 5 to 10 meters to get plenty of overlap between the start and end of the loop.

This package contains Gmapping, from OpenSlam, and a ROS wrapper. The Gmapping package provides laser-based SLAM as a ROS node called slam_Gmapping. Using slam_Gmapping, it creates a 2-D occupancy grid map (like a building floor plan) from laser range finder and pose data collected by a mobile robot (Gerkey 2016).

The slam_Gmapping node takes as input laser data and poses data collected from the laser range finder. It outputs a high-quality map.

2.3 Hector SLAM

Hector SLAM relies on scan matching, uses a Gauss-Newton Approach, and is accurate enough that it doesn't require loop closure. The Hector SLAM package consists of three main packages, i.e., hector_mapping, hector_geotiff, and hector_trajectory_server.

The Hector_mapping node is a SLAM approach used with or without odometry on platforms that exhibit roll/pitch motion (of the sensor, the platform, or both). It leverages the high update rate of modern LIDAR systems like the Hokuyo UTM-30LX and provides 2-D pose estimates at the scan rate of the sensors (40 Hz for the UTM-30LX). Although the system does not provide an explicit loop closing ability, it is sufficiently accurate for many real-world scenarios. The system has been used successfully on Unmanned Ground Robots and Unmanned Surface Vehicles and Handheld Mapping Devices and logged data from quadrotor UAVs (Kohlbrecher 2014).

Hector_geotiff saves the map and robot trajectory to geotiff image files. The hector_trajectory_server saves tf-based trajectories files as output. The hector_mapping node's main input is scan data on the /scan topic. The data are then transformed via the /tf topic.

Overall, Hector SLAM outputs a high-quality map that is recognizable.

3 GROUND TRUTH MAPS

Figure 1 displays the three maps chosen as the ground truth maps for the simulation, labeled as Map One, Map Two, and Map Three. Each is simple in design and simple to navigate. All three maps were downloaded from the web and used with the STAGE simulator as the ground truth map for the simulated robot to navigate.

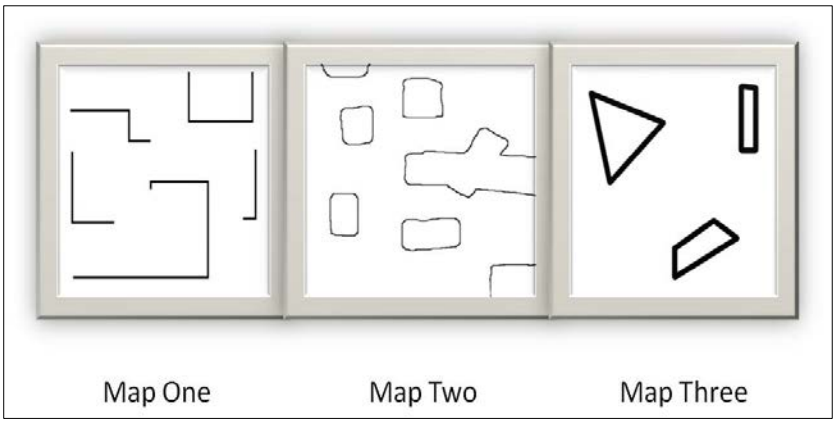

Figure 1: Ground Truth Maps.

4 SIMULATION RESULTS

Each of the three SLAM algorithms previously discussed was tested using Stage. Stage, an open-source software, provides multiple physics-based models for robot sensors and actuators. Some of the currently supported models are sonar and infrared rangers, 2-D scanning laser rangefinder, color-blob tracking, fiducial tracking, bumpers, grippers, and mobile robot bases with odometric and global localization (Kohlbrecher 2014).

One advantage Player/Stage provides is the ability to move from simulation to the robot by changing a few parameters (Staranowicz and Mariottini 2011). The learning curve on the Stage software is a disadvantage.

Stage, used standalone or with ROS, has many versions. This research implements Stage 4.1.1, the most recent version and requires ROS Fuerte for implementation.

The simulated robot completely navigated each of the three ground truth maps with each of the three algorithms to produce the three generated maps. Figure 2 shows the ground truth map, labeled Fixed, and the three generated maps produced by each algorithm, labeled CoreSLAM, Gmapping, and Hector Slam, after navigating the ground truth of Map One. Figure 3 shows the ground truth map, labeled Fixed, and the three generated maps produced by each algorithm, labeled CoreSLAM, Gmapping, and Hector Slam, after navigating the ground truth of Map Two. Figure 4 shows the ground truth map, labeled Fixed, and the three generated maps produced by each algorithm, labeled CoreSLAM, Gmapping, and Hector Slam, after navigating the ground truth of Map Three.

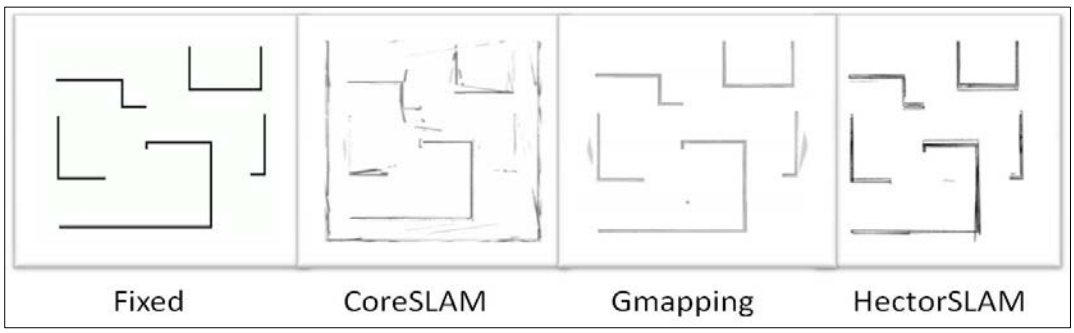

Figure 2: Map One and Generated Map of Each Algorithm.

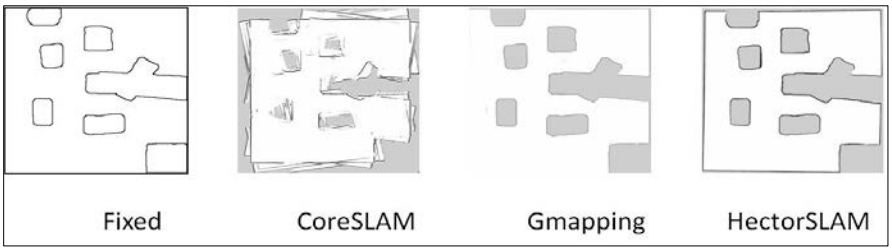

Figure 3: Map Two and Generated Map of Each Algorithm.

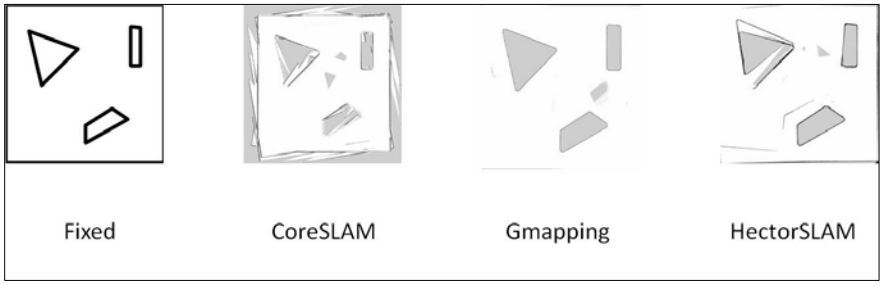

Figure 4: Map Three and Generated Map of Each Algorithm.

5 IMAGE REGISTRATION

MATLAB (**mat**rix **lab**oratory) is a multi-paradigm numerical computing environment and fourth-generation programming language with many functions and libraries (MathWorks 2012). This research uses the image registration tool to compare the generated maps to the ground truth map.

Image registration, the process of aligning two images of the same scene, is then used to align the ground truth map image and the generated map image for a multimodal comparison of the two images. An intensity-based automatic image registration process requires a pair of images, a metric, an optimizer, and a transformation type to align one image with another (MathWorks 2012). The pair of images is the ground truth map image and the generated map image.

The metric defines the image similarity metric for evaluating the accuracy of the registration. The optimizer defines the methodology for minimizing or maximizing the similarity metric.

The transformation type defines the type of 2-D transformation that brings the misaligned image (called the moving image or the generated image) into alignment with the reference image (called the fixed image or the ground truth image). Four transform types exist, i.e., affine, rigid, similar, and translation.

The image registration process begins with the transform type specified and an internally determined transformation matrix. Together, they determine the specific image transformation that is applied to the moving image with bilinear interpolation.

Next, the metric compares the transformed moving image to the fixed image and a metric value is computed.

Finally, the optimizer checks for a stop condition. A stop condition is anything that warrants the termination of the process. In most cases, the process has reached a point of diminishing returns or it has reached the specified maximum number of iterations (MathWorks 2012). If there is no stop condition, the optimizer adjusts the transformation matrix to begin the next iteration (MathWorks 2012).

The following sections discuss the resulting aligned images produced with the four transformations, i.e., affine, rigid, similar, and translation, and the three algorithms. The maps will contain three colors, i.e., magenta, green, and black. The magenta represents the intensity of the ground truth image, the green

represents the intensity of the SLAM generated map, and the black represents where both images align or are the same.

5.1 **CoreSLAM**

Figures 5, 6, and 7 were produced using two images, (1) the ground truth map images, i.e., Map One, Map Two, and Map Three, respectively, and (2) the respective Stage simulated generated map images from the CoreSLAM algorithm, as inputs to MATLAB's image registration function, which was executed with the four different transform types, i.e., affine (upper right), rigid (upper left), similar (lower left), and translation (lower right). In general, the image registration overlays or aligns the generated map on to the ground truth map image to compare the two images. The image registration tool is executed with the four different transform types: affine, translation, rigid, and similar.

The generated maps show up with more intensity than the ground truth maps due to CoreSLAM producing multiple edges (green shaded areas) along the exterior portion of the map. While the ground truth and generated maps are similar in nature, there are few overlapping points, because there is very little black, which shows points where the two images are identical.

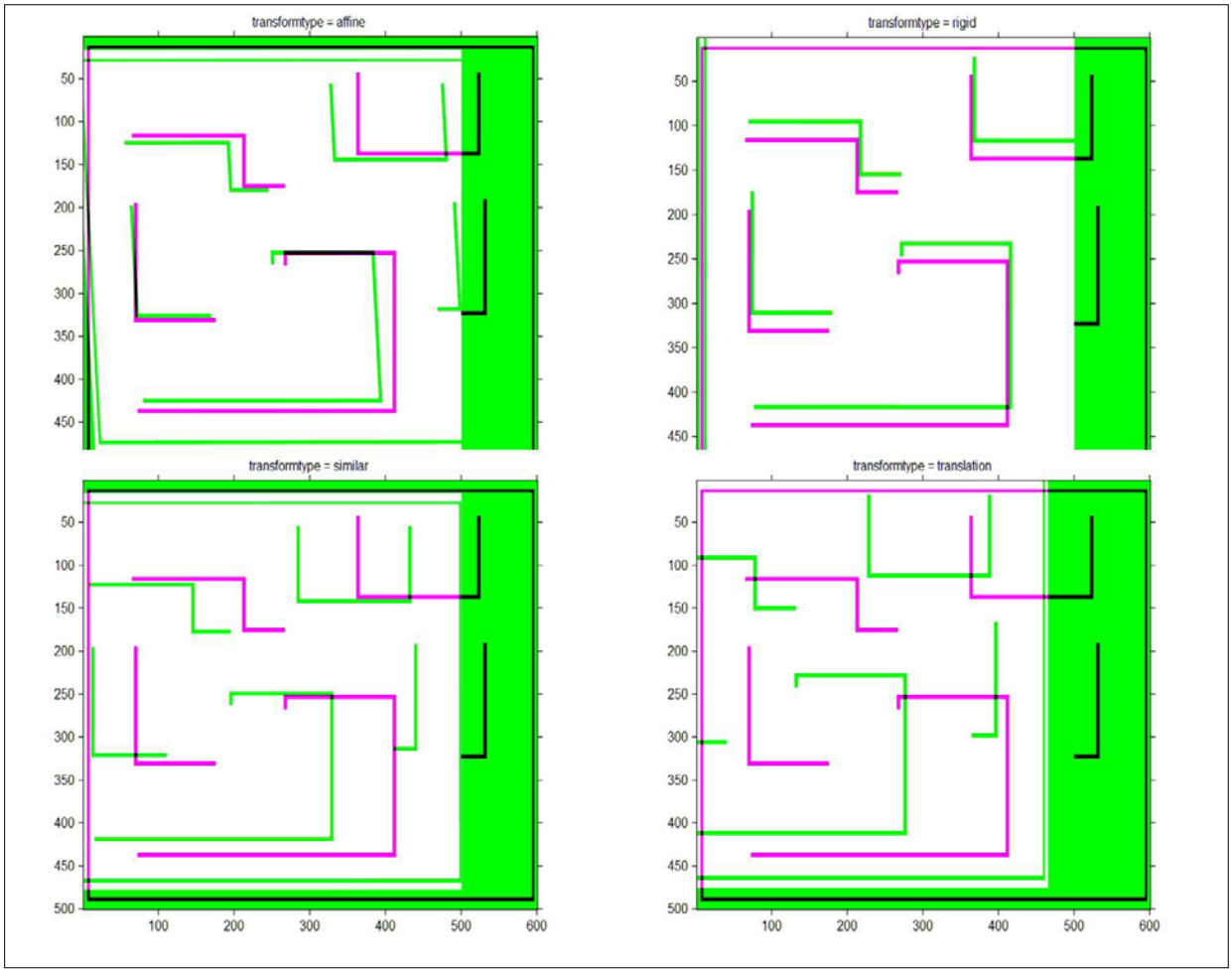

Figure 5: Image Registration of Map One and CoreSLAM Generated Map.

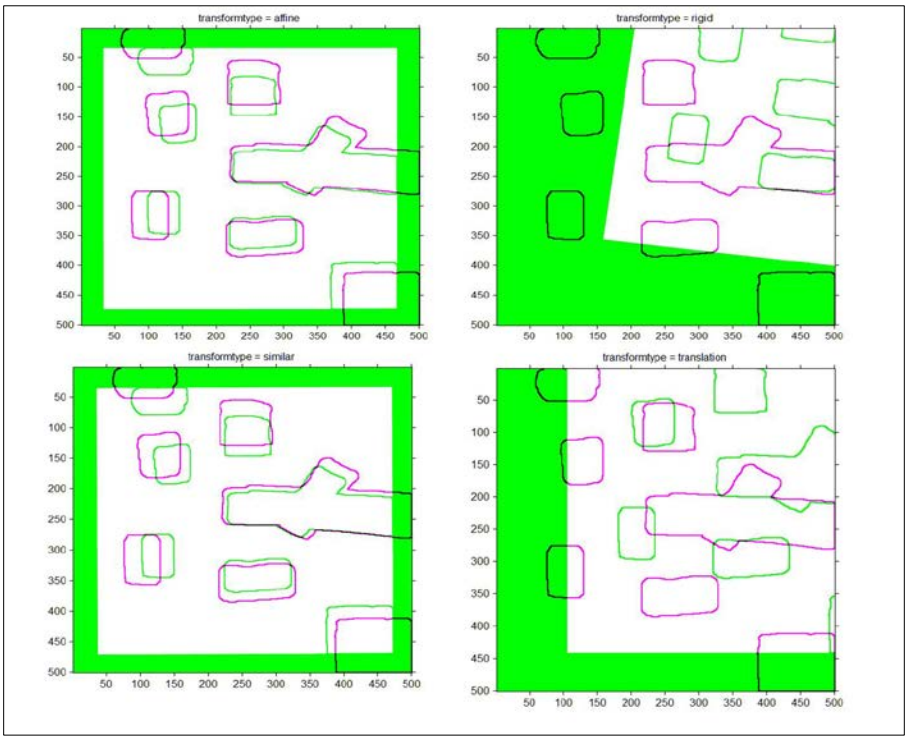

Figure 6: Image Registration of Map Two and CoreSLAM Generated Map.

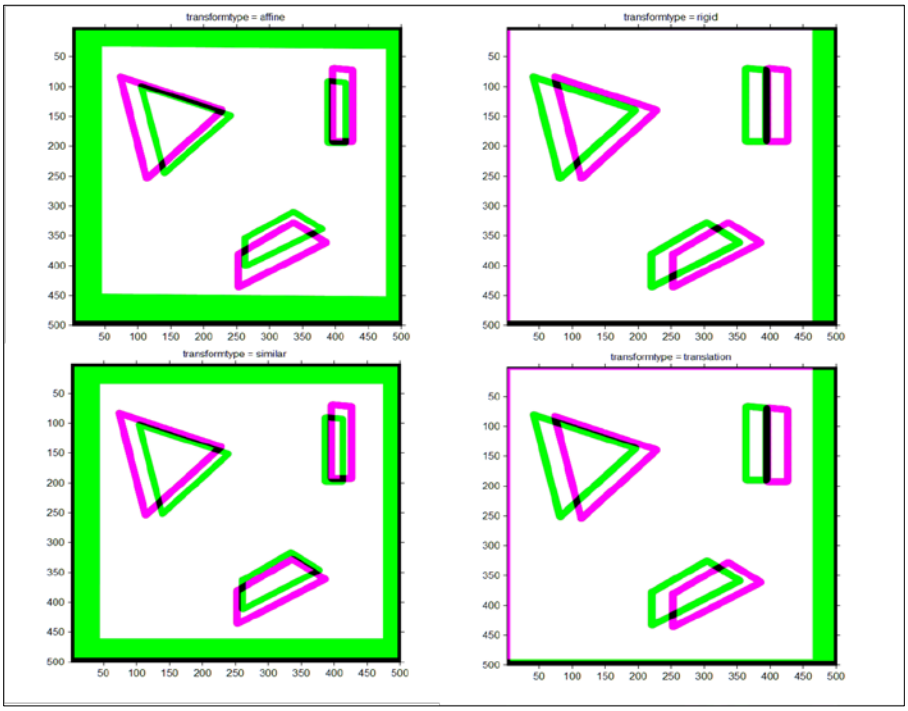

Figure 7: Image Registration of Map Three and CoreSLAM Generated Map.

5.2 **Gmapping**

Figures 8, 9, and 10 were produced using two images, i.e., the ground truth map images and the Stage simulated generated map images from the Gmapping algorithm in MATLAB's image registration function. The magenta in the image of Figure 8 represents the intensity of the ground truth image, the green represents the intensity of the SLAM generated map, and the black represents where the images overlap. In Figure 9, all transform types align all most perfectly. Transform type affine (upper left) has a small amount of green in the bottom indicating that the ground truth map of Map Two has a slightly higher intensity than the generated map. Transform type translation (bottom right) has a small amount of green on the left and the bottom of the map, indicating the ground truth map of Map Two has a slightly higher intensity than the generated map. In Figure 10, the magenta is stronger in all transformations, indicating that the generated map has higher intensity than the ground truth of Map Three. The black shows where the images align.

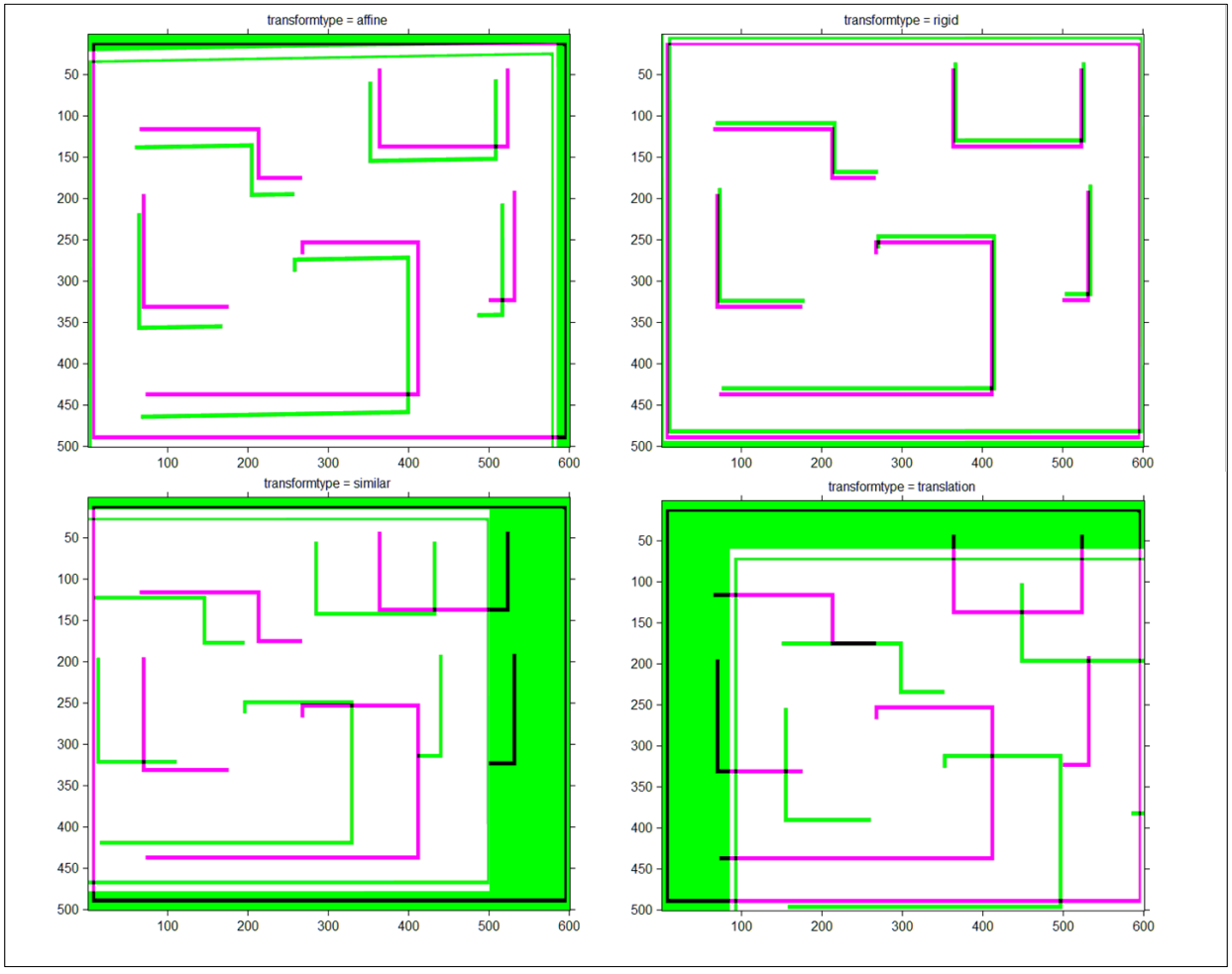

Figure 8: Image Registration of Map One and Gmapping Generated Map.

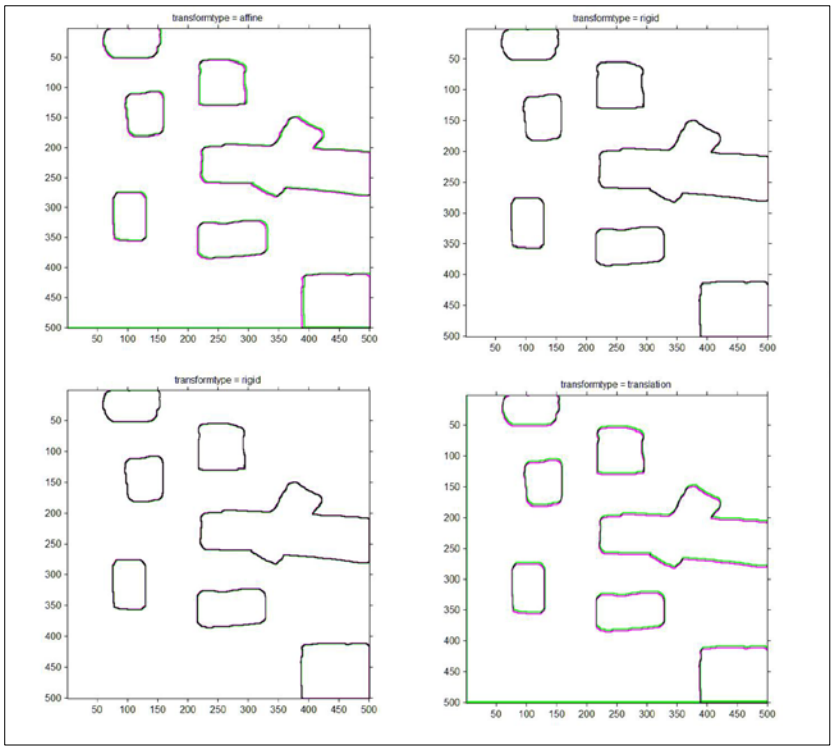

Figure 9: Image Registration of Map Two and Gmapping Generated Map.

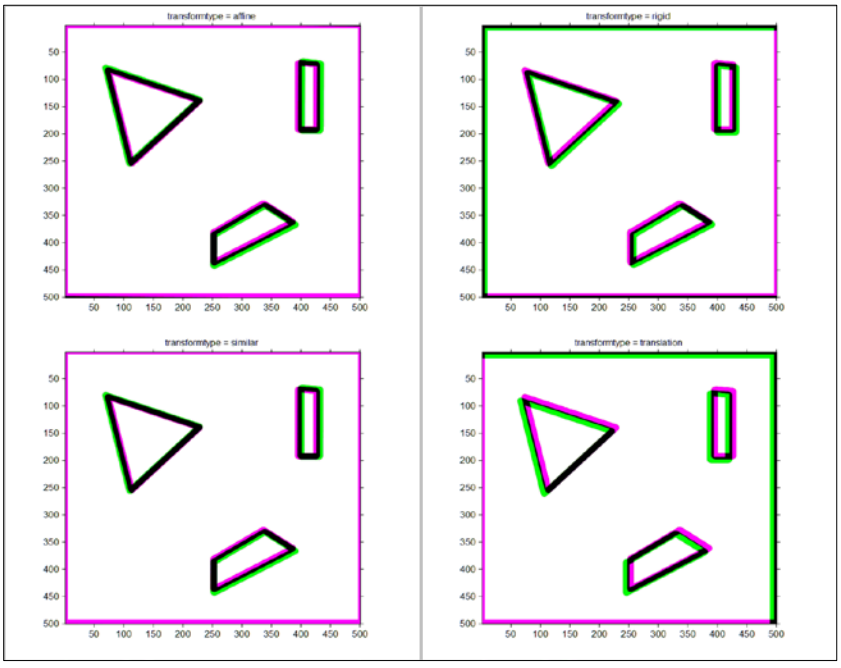

Figure 10: Image Registration of Map Three and Gmapping Generated Map.

5.3 **Hector SLAM**

Figures 11, 12, and 13 were produced by using the ground truth maps and the Stage simulated generated maps with the Hector Slam algorithm as input in MATLAB's image registration function. The image registration tool was again processed with the four different transform types, i.e., affine, translation, rigid and similar. Figure 11 shows the generated map (green) has more intensity than the ground truth map due to the alignment being off and the degree of difference in the two maps. While the maps are similar in nature, they have few overlapping points. Figures 12 and 13 shows black more than magenta and green, indicating that the two images have very little differences. Figure 11 has more green with all transform types. Figure 12 shows green around the exterior due to the processing of the image. In Figures 12 and 13, the rigid (upper right) and translation (lower right) transforms produce near perfect alignments.

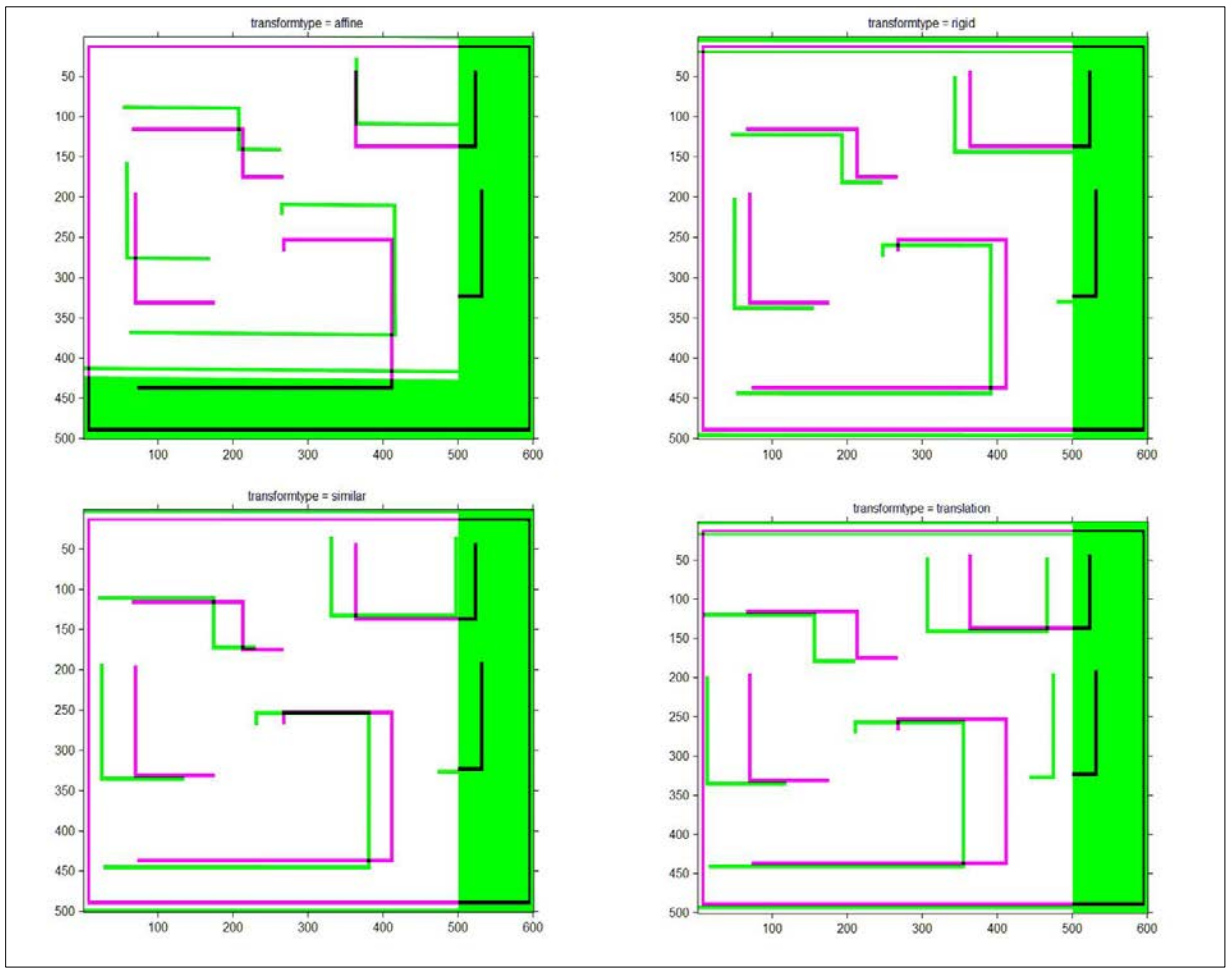

Figure 11: Image Registration of Map One and
Hector SLAM Generated Map.

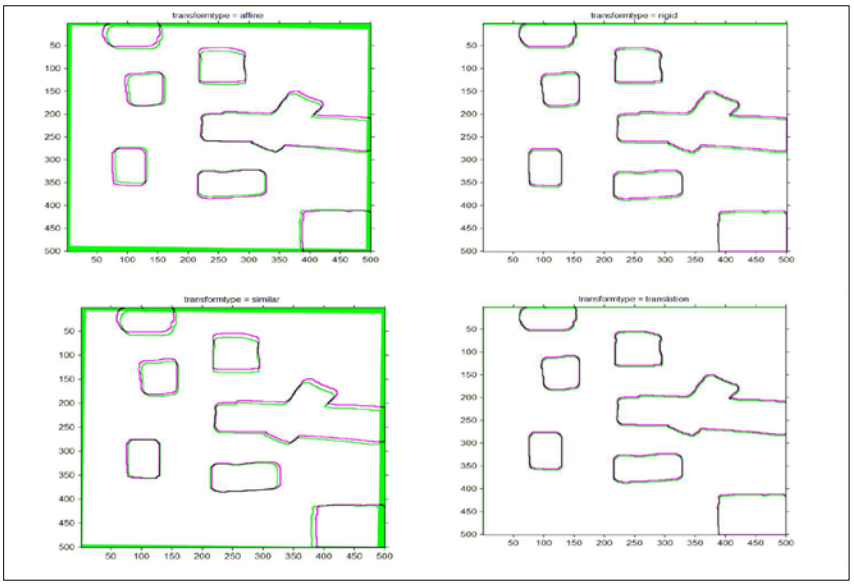

Figure 12: Image Registration Map Two and
Hector SLAM Generated Map.

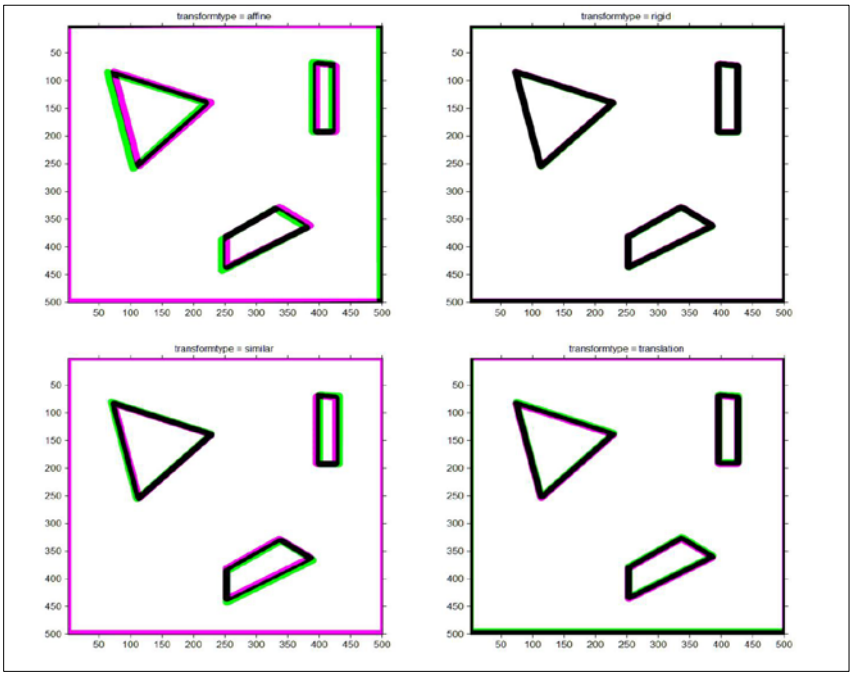

Figure 13: Image Registration of Map Three and
Hector SLAM Generated Map.

6 COMPARISON

For a quantitative measure between the ground truth maps and the generated simulator maps, Hausdorff Distance is calculated. The Hausdorff distance, by definition, is as follows: Given two finite sets A = $(a_1 \ldots a_p)$ and B = $(b_1 \ldots b_p)$, the distance is calculated as

$$H(A,B) = \max(h(A,B), h(B,A)) \quad (1)$$

where

$$h(A,B) = \sup_{a \in A} \inf_{b \in B} \|a - b\| \quad (2)$$

$\| \ \|$ represents some underlying norm defined in the space of the two point sets, which is generally required to be an L_p norm, usually the L_2 or Euclidean norm. The function h(A, B) is called the directed Hausdorff distance from A to B. If A and B are compact sets, then

$$h(A,B) = \max_{a \in A} \min_{b \in B} \|a - b\| \quad (3)$$

The Hausdorff Distance is calculated with the function h(A,B), which returns the distance of matrix A from matrix B. It identifies the point an element of A that is the farthest from any point in B and measures the distance from A to its nearest neighbor in B (Zhang, Han, and Wo 2005).

Tables 1, 2, and 3 show the values returned from using the Hausdorff function for comparing the two images, the ground truth maps (Map One, Map Two, and Map Three), and the generated maps from CoreSLAM, Gmapping, and HectorSLAM. The lower the Hausdorff Distance, the better the alignment of the two maps.

Table 1 shows the HectorSLAM algorithm has the lower values of 15.5885, in the Hausdorff Distance Column. CoreSLAM has the second lowest value of 16.55. Gmapping has the highest value if 22.7156.

Table 2 shows the CoreSLAM algorithm has the lower value of 15.5563, in the Hausdorff Distance Column. HectorSLAM and Gmapping have an equal value of 16.8523.

Table 3 shows the HectorSLAM algorithm has the lower values of 14.1067, in the Hausdorff Distance Column. CoreSLAM having the second lowest value of 18.13857. Gmapping has the highest value of 19.2354.

Based on all the results, HectorSLAM outperforms both CoreSLAM and Gmapping for Map One and Map Three. CoreSLAM outperforms HectorSLAM and Gmapping for Map Two while HectorSLAM and Gmapping are tied for Map Two.

Table 1: Hausdorff Distance for Map One.

	Map Size	Hausdorff Distance
CoreSLAM	600 × 500	16.55
Gmapping	600 × 500	22.7156
Hector SLAM	600 × 500	15.5885

Table 2: Hausdorff Distance for Map Two.

	Map Size	Hausdorff Distance
CoreSLAM	500 × 500	15.5563
Gmapping	500 × 500	16.8523
Hector SLAM	500 × 500	16.8523

Table 3: Hausdorff Distance for Map Three.

	Map Size	Hausdorff Distance
CoreSLAM	500 × 500	18.13847
Gmapping	500 × 500	19.2354
Hector SLAM	500 × 500	14.1067

7 CONCLUSIONS

In conclusion, the robotic simulator STAGE explored three topologies to produce three generated maps which provided a repeatable, cost-effective method to evaluate topologies without an actual robot. Simulation provided a means to evaluate and compare three SLAM algorithms in a cost-efficient and cost-effective manner with three different topologies. HectorSLAM performed best with Map One and Map Three when using the Hausdorff Distance, and CoreSLAM performed better with Map Two.

REFERENCES

Gerkey, B. 2015. Gmapping. Accessed June 24, 2016. http://wiki.ros.org/gmapping.

Grisetti, G., C. Stachniss, and W. Burgard. 2007. "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters." *IEEE Transactions on Robotics* 23:34-46.

Kohlbrecher, S. 2014. Accessed June 24, 2016. http://wiki.ros.org/hector_mapping.

MathWorks. 2012. "Image Registration Toolbox: User's Guide (R2012a)." Accessed June 23, 2013. www.mathworks.com.

Staranowicz, A., and G. Mariottini. 2011. "Robotic Simulation Guide." ASTRA Robotics Lab, Department of Computer Science and Engineering, University of Texas, Arlington, Texas. Accessed April 23, 2015. http://ranger.uta.edu.

Zhang, J., G. Han, and Y. Wo. 2005. "Image Registration Based On Generalized and Mean Hausdorff Distances." In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics* 1:5117-5121.

AUTHOR BIOGRAPHY

DORIS TURNAGE is a Computer Scientist at the U. S. Army Engineer Research and Development Center in Vicksburg, MS. Her background includes 31 years of varied experience in computing, logistics, robotics, and software engineering. Doris.M.Turnage@usace.army.mil. She has a Ph.D. in Engineering Science from the University of Mississippi.