2020-01-1021 Published 14 Apr 2020

Benchmarking the Localization Accuracy of 2D SLAM Algorithms on Mobile Robotic Platforms

Mugdha Basu Thakur, Matthias Schmid, and Venkat N Krovi Clemson University

Citation: Basu Thakur, M., Schmid, M. and Krovi, V.N., "Benchmarking the Localization Accuracy of 2D SLAM Algorithms on Mobile Robotic Platforms," SAE Technical Paper 2020-01-1021, 2020, doi:10.4271/2020-01-1021.

Abstract

Simultaneous Localization and Mapping (SLAM) algorithms are extensively utilized within the field of autonomous navigation. In particular, numerous open-source Robot Operating System (ROS) based SLAM solutions, such as Gmapping, Hector, Cartographer etc., have simplified deployments in application. However, establishing the accuracy and precision of these 'out-of-the-box' SLAM algorithms is necessary for improving the accuracy and precision of further applications such as planning, navigation, controls. Existing benchmarking literature largely focused on validating SLAM algorithms based upon the quality of the generated maps. In this paper, however, we focus on examining the localization accuracy of existing 2-dimensional LiDAR based indoor SLAM algorithms. The fidelity of these implementations is compared against the OptiTrack motion capture system which is capable of tracking moving objects at sub-millimeter level precision. Finally, the error statistics for each of the algorithm was determined.

1. Introduction

The quintessential questions to be answered for successful autonomous agents are: "Where am I? Where am I going? What is around me? How should I get there?"[1, 2]. Locating an autonomous ego-vehicle within its environment is very critical for a multitude of applications such as trajectory planning, navigation, controls, etc. While ego-vehicle localization is nominally easier when a map of the environment is provided, many times an accurate map of the surrounding is not provisioned and localization can get especially challenging, e.g. inside unknown enclosed structure or buildings. In such situations, SLAM can generate the map of the unknown environment while simultaneously localizing the agent in it.

Over the past three decades, significant advances in sensing/actuation technologies (e.g. radar, LiDAR), algorithms (e.g. probabilistic methods) and system implementations (e.g. ROS) have transformed the landscape [3, 4]. The availability of reference 2D SLAM implementations in ROS [5] has promoted tremendous growth in research and educational settings [6]. However, these SLAM algorithms feature significant variety and diversity and are often customized to application. Other sources of variability arise from the sensor configuration (laser scanner, camera, wheel encoder, IMU sensors etc.) and approximation methods (particle filter, extended Kalman filter etc.) [4]. Hence, it is likely that a particular implementation of SLAM outperforms other methods for specific conditions. This motivates the need for establishing both a common test environment as well as a common test protocol when benchmarking the performance of these SLAM solutions. Additionally, quite a variety of performance evaluation criteria can be considered (e.g. accuracy and computation cost) as will be discussed in section 2. Finally, the depth of the comparative analyses can also be varied; e.g., cross-evaluation of generated maps.

The objective of this paper is to create a comparison and decision-making framework aiding the selection of a SLAM algorithm (available in ROS), depending on the type of hardware available and the desired final application. We will focus on determining the location accuracy in a given map by different SLAM algorithms - GMapping, Hector, and Cartographer. While some past efforts focused on evaluating the quality of the map, we examine the end-results against the ground truth data. The trajectory of the robot, as returned by SLAM, is compared against an advanced motion capture and tracking system, OptiTrack, manufactured by Natural Point Inc [7]. All experiments were performed on the latest TurtleBot3 Waffle Pi robotic hardware platform, manufactured by ROBOTIS [8].

In this work, Section 2 provides on overview of related works comparing SLAM accuracy, Section 3 details the hardware components as well as the testing conditions for our experiments, Section 4 presents a comparison of the experimental results for the different SLAM implementations, and finally, Section 5 concludes this study with a discussion on potential future enhancements.

2. Related Work

There are substantial past efforts towards establishing the accuracy of SLAM algorithms. Yet, a significant challenge is the unavailability of the true location of the robot in the environment. Therefore, the majority of existing work focused on evaluating the quality of the generated maps. Although, various authors have made experimental datasets available to the public in order to aid further innovation, (as identified by Santos et al. [6]) these datasets are not compatible with ROS. In their work, Santos et al. analyzed maps generated by the Gmapping, Hector, Cartographer, Core and Lago SLAM algorithms. Here, the map was first binarizing and aligned with the ground truth using MATLAB's Image Processing Toolbox. Subsequently, the authors determined the quality of the resulting 2D occupancy grid employing a k-nearest neighbor search. Their analysis was performed in simulation as well as in real world settings. In [9], Anton et al. used the MIT data sequence to evaluate the 2D occupancy grid maps rendered by Gmapping, Cartographer, Hector, Tiny and Viny SLAM algorithms against ground truth. In their study, the authors evaluated the map based upon the ratio of occupied and free cells, the number of corners, and the number of enclosed areas in the generated map. In [10], Kummerle et al. determined the accuracy of 3 different SLAM algorithms - a scan matching based algorithm, a Rao-Blackwellized particle filter and a graph based SLAM algorithm - by comparing the relative translation and rotation of the robot with the true relative displacement in its pose between two instances of time. The ground truth information was once again manually calculated. The authors of [11] studied the error in a robot's trajectory for 3 different SLAM algorithms: Tiny SLAM, Gmapping and Cartographer. Their evaluation employed 11 sequences of the MIT Stata Center dataset providing the ground truth information of the robot's location. The comparison entailed the root-mean-square-error between the output of the SLAM algorithm and the ground truth value.

3. Methods and Implementation

In this study, we evaluated the performance of three 2D LiDAR based indoor SLAM algorithms: Gmapping, Hector and Cartographer. The objective is to determine the accuracy of the localization output of the SLAM implementations on identical hardware. Therefore, this section details the employed hardware and sensor suite together with the test environment and test protocols applied. Thus, this comparative study should provide researchers and developers with a proper reference for SLAM accuracy to be expected from different algorithms.

3.1. Hardware Specification

For an unbiased comparison, it is necessary to ensure that each algorithm is provided with identical sensor inputs from the mobile agent. For this study, a mobile agent, that could easily be used for prototyping with ROS and that is off-the-shelf equipped with a standard suite of onboard sensors such as LiDARs, IMU and wheel encoders, was required. The differential drive robot TurtleBot3 Waffle Pi [8] (which is manufactured by ROBOTIS and is the 3rd generation of the TurtleBot robots) was chosen for our analysis. The robot employs a Raspberry Pi 3B processor along with an OpenCR microcontroller board (32-bit ARM cortex-M7). The on-board LiDAR utilized for the SLAM benchmarking process is a 360degree laser distance sensor LDS-1 with 1-degree angular resolution. The LiDAR, operating on a 5V DC voltage, is capable of detecting object distances in the range of 120 mm to approximately 3,500 mm at a sampling rate of 1.8 kHz. The odometry information was obtained from the Dynamixel actuators with 32-bit microcontrollers and 12-bit, 360-degree, contactless absolute encoders. The Raspberry Pi processor on the TurtleBot3 Waffle Pi was set up with Ubuntu 16.04, running ROS Kinetic, and acted as a slave in the ROS network. During the evaluation, the robot was teleoperated from a separate computer (also operating on Ubuntu 16.04 with ROS Kinetic) that ran the ROS master. System time on both machines was synchronized with the help of the chrony tool. The remote computer was configured to be the chrony time server while the Raspberry Pi on the robot was the client.

3.2. Choice of SLAM Algorithms

GMapping, Hector and Cartographer were selected because of the diversity in their underlying working principle and their popularity amongst researchers and developers in ROS.

GMapping consists of a Rao-Blackwellized particle filter-based SLAM algorithm[12, 13] that maintains a distinct number of particles at all times. Each of these 'particles' carry an individual map of the environment. The algorithm processes exteroceptive sensor measurement together with proprioceptive odometry information. Here, the sensor measurement comes from LiDAR scans with the odometry provided by wheel encoders. Maintaining adequate number of particles is complex yet essential to generate a precise map of the environment. Reduction of the number of particles would compromise the accuracy of the map. One method for maintaining the desired number of particles is iterative resampling. However, the downside of this approach lies in the possible loss of correct particles and consequently may result in poor generation of the map.

The Hector SLAM algorithm [14] aims at minimizing computational cost. The current availability of high quality LiDARs with rapid update rates has been leveraged to enhance a robust scan matching approach, thereby ensuring an accurate perception subsystem in the SLAM algorithm. The planar map of the environment, generated by the perception unit, is subsequently combined with an attitude approximation system that uses inertial measurement from an IMU sensor. This renders the solution apt for implementations in which the need for estimating the 3-dimensional pose of the mobile agent or the need for pose estimation of fastmoving aerial vehicles such as drones arises.

Google's Cartographer is a graph-based SLAM solution that generates 2D probability grid maps of a resolution of 5 cm [15]. Here, the algorithm utilizes recent measurements from LiDAR sensors to create local submaps of the environment with the objective that these recent measurements contribute towards the accuracy of the local map. Since scan matching is only executed on recent scans, an accumulation of global error in the pose estimates is expected. To overcome this error, the algorithm runs a pose optimizer in parallel. Once the submap is generated, the incoming sensor measurements as well as the generated submaps participate in the loop closing aspect of the algorithm. The loop closure optimization problem is posed as a non-linear least square problem and is solved using the Ceres solver [16].

3.3. Test Environment

The controlled test environment in which the experiments were performed consists of a 3x6 m^2 area at CU-ICAR's ARMLab as shown in figure (1). To compare the accuracy of the different SLAM algorithms, the actual location of the robot in the environment, i.e. the ground truth of the robot position in the environment was required. To obtain this ground truth data, we employed the OptiTrack motion capture system manufactured by NaturalPoint, Inc. OptiTrack's Prime 13 hardware platform is an advanced positioning system capable of highspeed tracking. The hardware was operated by OptiTrack's Motive 2.1.0 software. For this study, 12 tracking cameras were used. The Prime13 cameras have a high resolution of 1.3MP and a horizontal field of view of 56 degrees. Operating at a frame rate of 120 frames per second, the system is capable of tracking IR reflective markers mounted on the mobile agent at a 3D precision of 0.001 m. The field of view of the cameras was adjusted to track only the above defined area. The mobile agent was equipped with 4 IR reflective markers

FIGURE 1 The figure shows the test area tracked by Optitrack. The coordinate system is a left hand coordinate system. In this figure, the -ve X axis and the Y axis is shown.

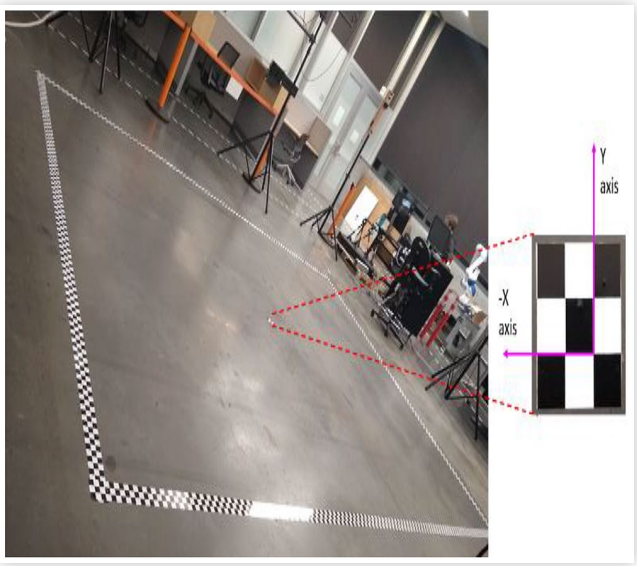
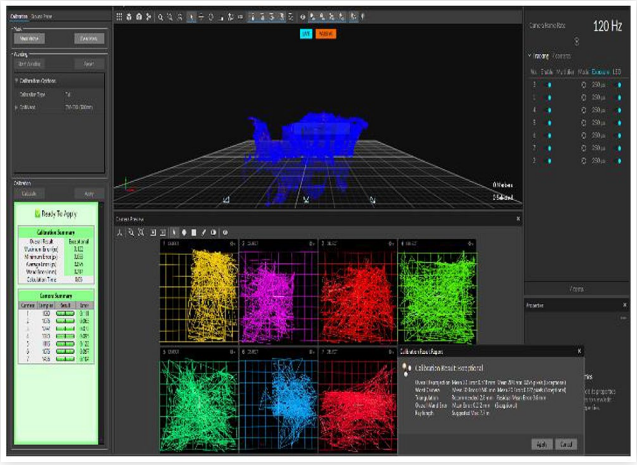


FIGURE 2 A screenshot of the OptiTrack window during the calibration process. The figure shows the field of view coverage for 8 out of the 12 cameras used for tracking.



to reflect the footprint of the base and to obtain its centroid. It was ensured that the robot was visible to at least 4 cameras from any location within the tracked area. The Motive software determines the 3D pose by first calculating the 2D position from the 2D images captured by each camera followed by overlapping each 2D position result in order to obtain the 3D pose via triangulation.

3.4. Calibration

Calibration of the OptiTrack system comprised of two procedures: wanding and setting the ground plane. The first was achieved by moving a wand with 3 IR reflectors in the area tracked by the cameras. In this process, each camera captured data samples to determine their relative position. Post wanding, the ground plane with its origin was fixed manually to complete the calibration process. During the wanding stage, over 6000 data samples were collected per camera for calibration, as shown in figure (2). This ensured an exceptional calibration result with 0.095 pixels average error in determining 2D position. The average 3D error after reprojection was determined to be 0.518 mm. Since our experiments focused on localization accuracy rather than the quality of the generated map, there was no requirement to introduce obstacles in the test environment. This also ensured that the robot was never occluded from the field of view of the Optitrack system and thus consistently provided ground truth data throughout the experiments.

3.5. Test Protocol

In this OptiTrack controlled environment, we teleoperated the agent under different speed conditions for different trajectories. While the mobile agent was teleoperated, the '*rosbag*' tool in ROS was used to record the raw LiDAR scans and odometry data from the wheel encoders (from Turtlebot3) along with the ground truth for the agent's location (from OptiTrack) on the system running the ROS Master.

The mobile agent was driven under 4 different trajectory conditions. The first 2 of consisted of simple fundamental geometric shapes, whereas the latter 2 represented more complex shapes generated from these fundamental shapes:

i. At a speed of 0.08m/s, in a straight line
ii. At a linear speed of 0.08m/s and angular speed of 0.1rad/s, in a circle
iii. At a speed of 0.08m/s, in a rectangular circuit
iv. At a speed of 0.08m/s, in a figure 8 circuit

For test trajectory (i), we have simulated a "corridor with doors" (as shown on the left-hand side of figure (3)) within the trackable area to ensure that SLAM algorithms experience sufficient features to localize the agent. In the absence of adequate unique landmarks, it is probable that the algorithms exhibit the "lost robot" behavior and are unable to localize the mobile agent accurately. The mobile agent was driven up and down the corridor multiple times. At each end, it was provided with a gradually increasing linear velocity saturating at a constant 0.08 m/s until it reached the other end of the corridor. Here, the linear velocity was terminated, followed by a constant angular velocity of 0.2 rad/s until the mobile agent executed a 180 degree turn. Then, the angular velocity was again terminated before the mobile agent was driven back to the other in a similar fashion as before. For the remaining test settings, we have simulated a "room with multiple doors". An example of the map of this environment is provided on the right-hand side of figure (3). The "doors" were strategically placed to ensure that the algorithms have adequate unique landmarks to identify the mobile agent within the environment. For trajectory (ii), the mobile agent was provided with a constant linear velocity of 0.08 m/s and an angular speed of 0.1 rad/s. In contrast, the Waffle Pi was teleoperated for trajectories (iii) and (iv) to ensure that maximum area within the trackable region was utilized.

To evaluate the performance of the SLAM algorithms the sensor data obtained and recorded from the mobile agent (for each trajectory) was run through Gmapping, Hector and Cartographer. A ROS transform listener was implemented to fetch the 2D pose of the robot in the map at a rate of 1Hz. This

FIGURE 3 The map on the left shows the environment and features created for test trajectory (i). The remaining test trajectories were executed within the environment shown in the right.

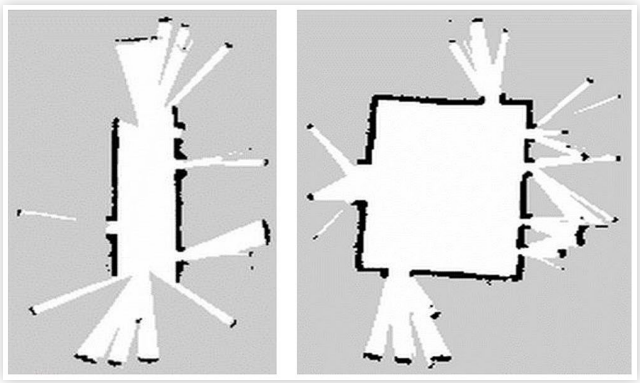
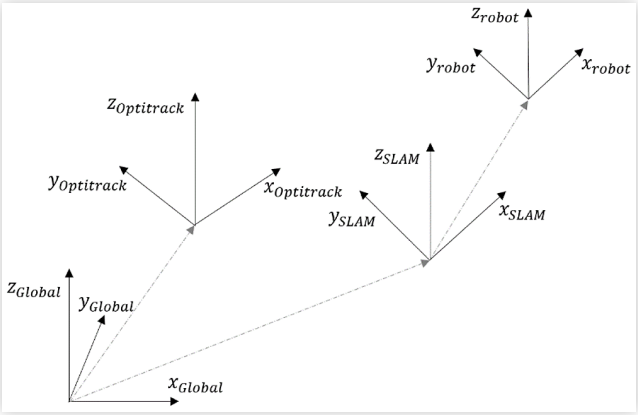


FIGURE 4 The relationship between the coordinate frames of the mobile agent, SLAM algorithms and OptiTrack



data was registered in a separate readable file for future analysis. The ground truth data was extracted from the *rosbag* to a spreadsheet in python. Finally, comparison between the localization from SLAM algorithms and ground truth was performed in MATLAB.

To compare the performance of the different SLAM implementations, results need to be comparable in a common coordinate system. Upon launch (in ROS), the SLAM algorithms establish their own coordinate frame within which the algorithms localize the mobile agent and generate a map of the environment. The ground truth data from Optitrack is independent of the coordinate frame established by SLAM but is contingent on the coordinate system initiated during its calibration. A visual representation of this relationship has been provided in figure (4).

For this study, the origin of the coordinate system of the Optitrack was chosen to be at the center of the tracked area. Yet, the initial position of the agent was chosen at random under teleoperation. Therefore, the system dependent coordinate frames were manually aligned during post processing of the SLAM localization results.

After this alignment, the output from each of the SLAM algorithms was matched to the robot position data from OptiTrack by identifying the timestamp at which both SLAM and OptiTrack started localizing the robot. Since the SLAM result was obtained at a rate of 1Hz, whereas the data from OptiTrack was fetched at a rate of 120Hz, the OptiTrack data set was down-sampled to 1Hz. Now, the error in the SLAM localization, at every instance of time, was obtained as the Euclidean distance, i.e.

$$E_t = \sqrt{\left(x_{SLAM,t} - x_{Optitrack,t}\right)^2 + \left(y_{SLAM,t} - y_{Optitrack,t}\right)^2}$$

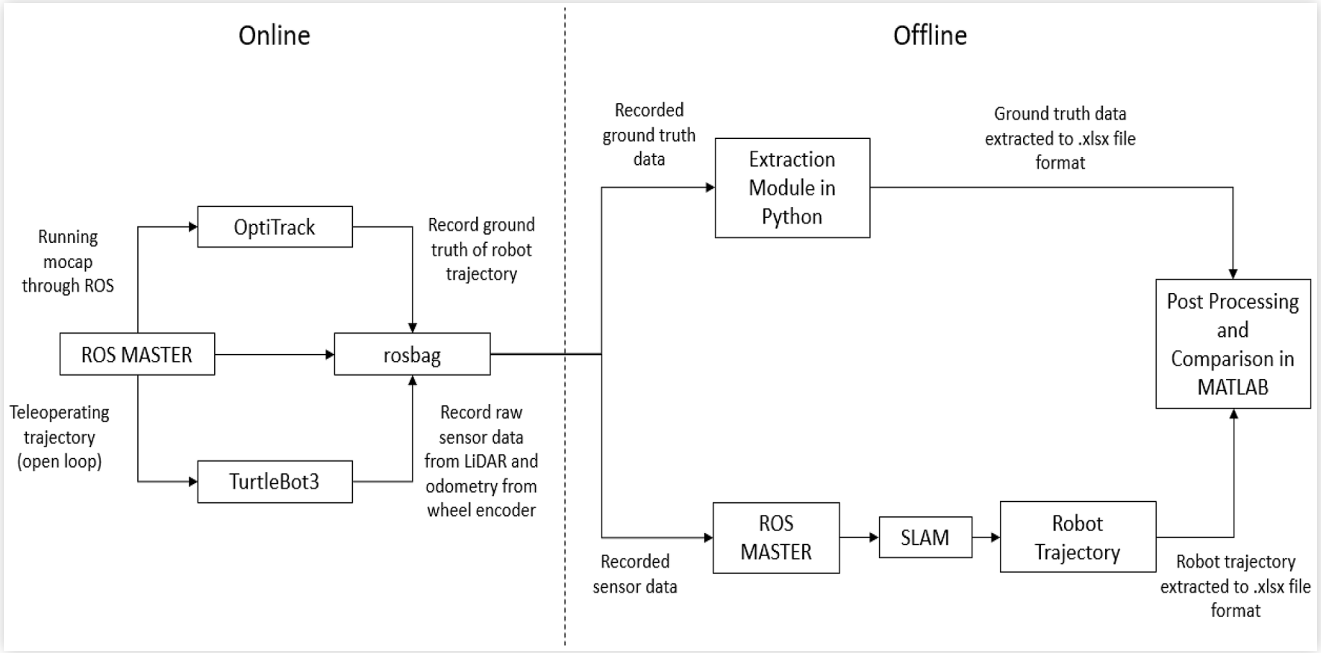
Additionally, the average error, the maximum error and the standard deviation of the localization error was determined.

Similarly, the error in the kinematic model of the mobile agent was also calculated every time step for validation purposes, yielding.

$$E_{Odom} = \sqrt{\left(x_{Odom,t} - x_{Optitrack,t}\right)^2 + \left(y_{Odom,t} - y_{Optitrack,t}\right)^2}$$

A visual representation of the described procedure is provided in Figure (5).

FIGURE 5 The end-to-end test process



4. Results

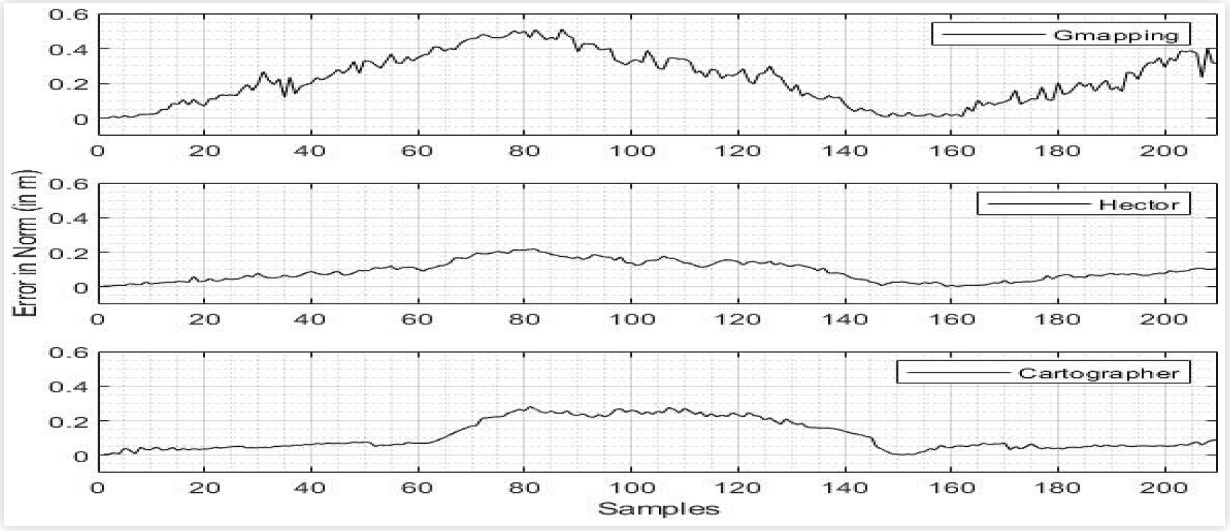
The localization accuracy of the different SLAM algorithms - Gmapping, Hector, and Cartographer - is summarized in Table [1].

Hector SLAM shows lowest standard deviation in error for the all trajectories when compared to Gmapping and Cartographer. This is thought to be caused by the inaccuracies of the manufacturer provided kinematic model of Turtlebot3. Gmapping and Cartographer incorporate the odometry data in order to improve localization results. The counter-intuitive experimental result, however, demonstrates that poor odometry information (due to limitations of physical hardware and inaccuracies of kinematic model) can indeed contribute to inferior performance of Gmapping and Cartographer localization accuracy. Indeed, this is a typical effect when proper tuning of the underlying algorithm is omitted which is often the case when off-the-shelf implementations (such as provided by ROS) are employed by users. The effects of poor tuning are expected to be even more significant for faster motion or dynamic environments. For the straight-line trajectory, the average error in the odometry of the mobile agent, compared to the ground truth, was determined to be 0.0285 m. This is because of the simplicity in the shape as well as the low speed of operation. In contrast, the average error of odometry was found to be 0.3289 m for test trajectory (ii) (circular). Hector SLAM, on the other hand, leverages the fast update rates of modern LiDAR sensors. Here, the odometry of the mobile agent is not utilized in determining the pose, therefore, the imposed uncertainties are avoided. The lower deviation in localization error of Cartographer SLAM can be attributed to the algorithm using a more accurate approach, i.e. scan-to-map matching, for generating the local submaps of the environment (in addition to the powerful Ceres solver for pose optimization). Furthermore, the localization output from Gmapping might exhibit inconsistency due to the nature of the embedded particle filter: it is possible that at times a particle reflecting the correct solution gets eliminated. To verify this expected behavior, the *rosbag* captured for test trajectory (i) was passed through GMapping three times. Here, it was observed that the localization error differed for every iteration. In test trajectory (1), GMapping generated an average error of 0.225 m, 0.2414 m, and 0.2405 m, respectively, for the same *rosbag*. Yet, the average localization errors for Hector SLAM, using the same *rosbag*, resulted as 0.0904 m and 0.0905 m. Since test trajectories (iii) and (iv) are further enhancements of the simple shapes in (i) and (ii), similar trends were observed.

TABLE 1 The table represents the average, maximum and the deviation in the localization error of each SLAM algorithm under the 4 unique test criteria.

	Test Trajectory (1)			Test Trajectory (2)			Test Trajectory (3)			Test Trajectory (4)		
SLAM	Avg Error	Max error	Std Dev	Avg Error	Max error	Std Dev	Avg Error	Max error	Std Dev	Avg Error	Max error	Std Dev
Gmapping	0.225	0.5102	0.1453	0.1161	0.3413	0.0778	0.0783	0.1549	0.0395	0.0689	0.2599	0.0626
Hector	0.0906	0.2189	0.0584	0.0767	0.1721	0.0368	0.0796	0.1472	0.0275	0.0707	0.208	0.0426
Cartographer	0.1132	0.2822	0.0861	0.1628	0.3422	0.0768	0.0779	0.1783	0.0397	0.1143	0.289	0.0528

FIGURE 6 The localization error in SLAM for a straight-line trajectory over time. The mobile agent traversed the trajectory 3 times. All units in meters.



Moreover, for each trajectory, the error in localization for Gmapping and Cartographer increased as the mobile agent moved away from its initial location (this is also the point from where the algorithms started running). Since, the mobile agent always traversed in a closed trajectory, significant reduction of error was observed every time the mobile agent returned to its initial location. This can be attributed to the loop closing modality in Gmapping and Cartographer. For trajectory (i), this can be verified in Figure (6) between samples 140 and 165. For trajectory (ii), as shown in Figure (7), the reduction in error occurred periodically as the mobile agent traversed approximately the same trajectory in the environment multiple times. This decrease in error for trajectory (iii) (Figure (8)) can be seen between samples 210 and 250 and between samples 420 and 480. Additionally, a decrease in error was observed at the corners of the trajectory (such as between samples 80 and 95) where the mobile agent was provided with zero linear velocity and a low angular velocity to make a sharp turn. As shown in Figure (9), for trajectory (iv), it was observed between samples 150 and 220. On the contrary, Hector SLAM showed minimal such variation in error since the algorithm does not implement loop closure. Nevertheless, the experimental results demonstrated that the algorithm is capable of achieving a better accuracy in localization.

FIGURE 7 The localization error in SLAM for a circular trajectory over time. The mobile agent completed the circle 7 times. All units in meters

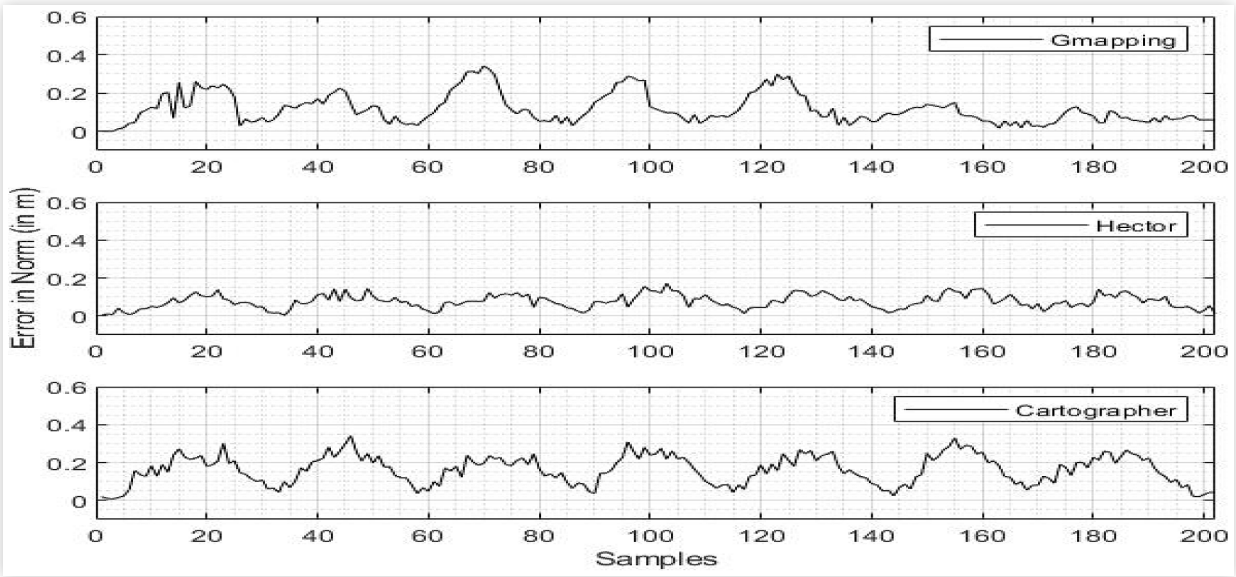


FIGURE 8 The localization error in SLAM for a rectangular trajectory over time. The mobile agent has completed the trajectory 3 times. All units in meters.

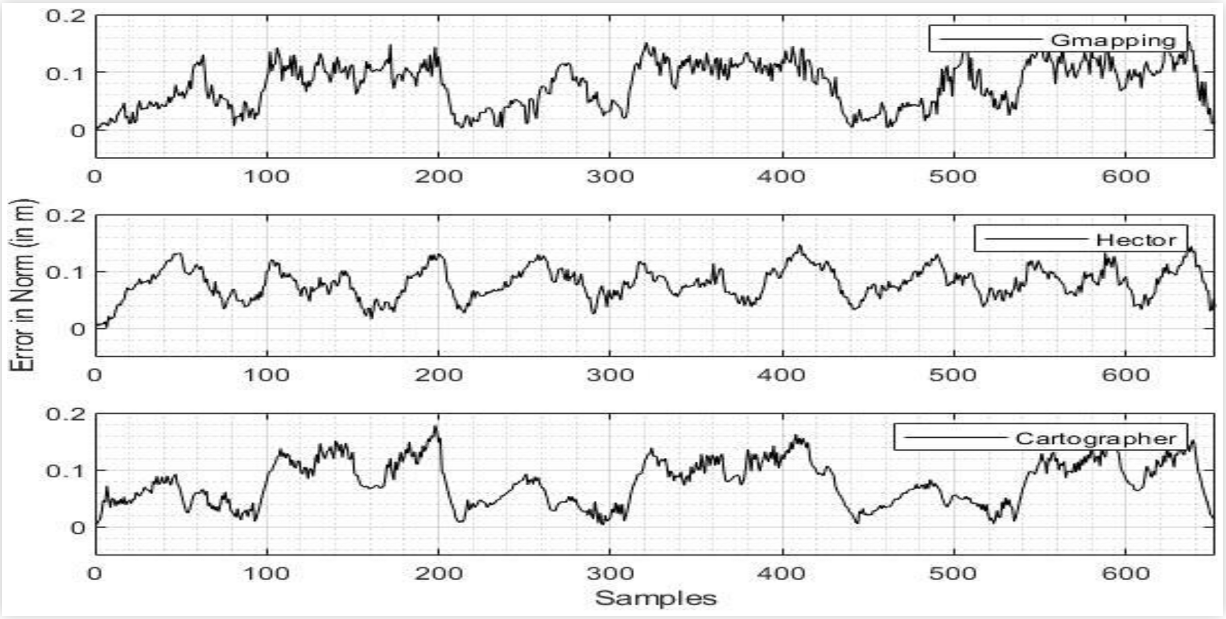
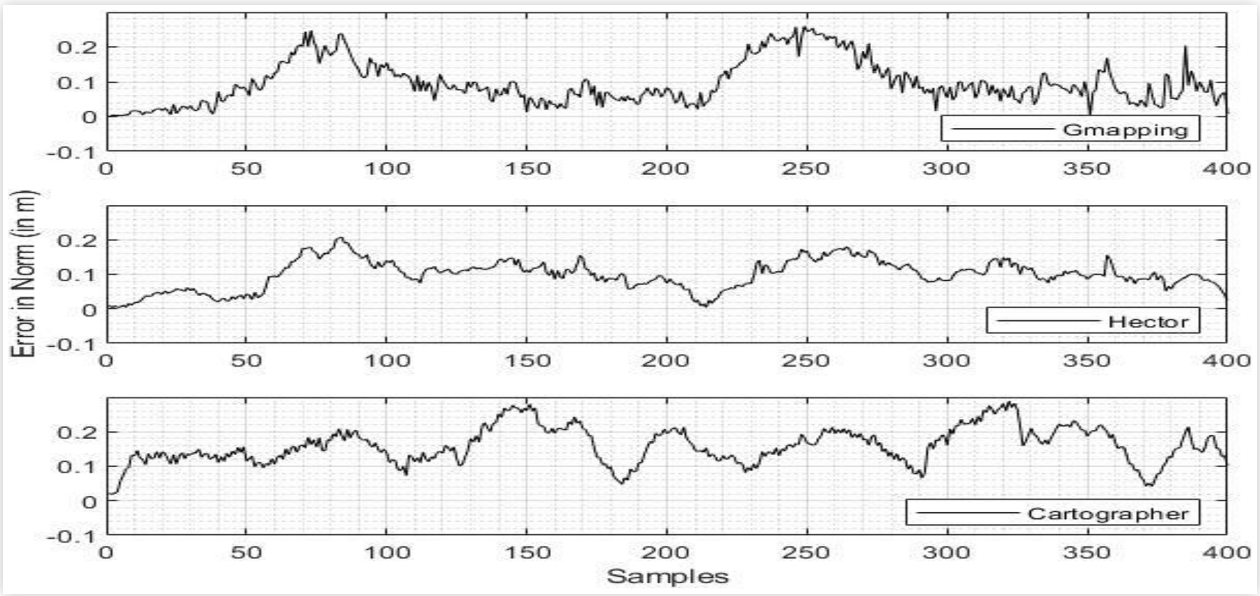


FIGURE 9 The localization error in SLAM for a figure 8 trajectory over time. The mobile agent has completed the trajectory 2 times. All units in meters.



5. Future Work

This study represents an initial effort in determining the comparative accuracy of prominently used 2D LiDAR based SLAM algorithms in ROS with respect to the localization of a mobile agent in an unknown environment. For future work, the base methods proposed here can be extended along many directions. For example, the evaluation of these SLAM algorithms for longer periods of time (long-term autonomy) can help establish the robustness and identify systematic errors in the individual implementations. Alternately, other performance criteria (e.g. computational cost) or derivative metrics could also be included to increase the depth of analyses. Further the quality of sensors and mobile platform impacts the accuracy of SLAM - hence extending this comparative analysis to other hardware platforms/sensor suites would also be critical to characterize the dependence on sensor/actuator resolution. Finally, the quality of map generation could be verified or included, respectively, by incorporating fixed landmarks in the environment that are marked for recognition in OptiTrack.

References

1. Borenstein, J., Everett, H.R., Feng, L., and Wehe, D., "Mobile Robot Positioning: Sensors and Techniques," *Journal of Robotic Systems* 14(4):231-249, 1997.
2. Leonard, J. and Durrant-Whyte, H., "Mobile Robot Localization by Tracking Geometric Beacons," *IEEE Transactions on Robotics and Automation* 7(3):376-382, 1991, doi:10.1109/70.88147.
3. Cadena, C., Carlone, L., Carrillo, H., Latif, Y. et al., "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics* 32(6):1309-1332, 2016, doi:10.1109/TRO.2016.2624754.
4. Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics* (Cambridge, MA: MIT Press, 2005).
5. Quigley, M., Conley, K., Gerkey, B.P., Faust, J. et al., "ROS: An Open-Source Robot Operating System," in ICRA Workshop on Open Source Software, 2009, http://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf.
6. Santos, J., Portugal, D., and Rocha, R., "An Evaluation of 2D SLAM Techniques Available in Robot Operating System," 2013 in *IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2013, 10.1109/SSRR.2013.6719348.
7. NaturalPoint, Inc., "OptiTrack," https://optitrack.com/products/prime-13/, accessed Aug. 2019.
8. Pyo, Y.S., Cho, H.C., Jung, R.W., and Lim, T.H., "Turtlebot3/Waffle_Pi," http://wiki.ros.org/Books/ROS_Robot_Programming_English, accessed Sep. 2019.
9. Filatov, A., Filatov, A.Y., Krinkin, K., Chen, B. et al. "2D SLAM Quality Evaluation Methods," in *21st Conference of Open Innovations Association (FRUCT)*, 2017, 10.23919/FRUCT.2017.8250173.
10. Kummerle, R., Steder, B., Dornhege, C., Ruhnke, M. et al., "On Measuring the Accuracy of SLAM Algorithms," *Autonomous Robots* 27(4):387-407, 2009.
11. Krinkin, K., Filatov, A., Filatov, A.Y., Huletski, A. et al., "Evaluation of Modern Laser Based Indoor SLAM Algorithms," in *Conference of Open Innovation Association, FRUCT*, 2018, 10.23919/FRUCT.2018.8468263..
12. Grisetti, G., Stachniss, C., and Burgard, W., "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics* 23(1):34-46, 2007, doi:10.1109/TRO.2006.889486.
13. Grisetti, G., Stachniss, C., and Burgard, W., "Improving Grid-Based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling," in *Proceedings-IEEE International Conference on Robotics and Automation*, 2005, 10.1109/ROBOT.2005.1570477.
14. Kohlbrecher, S., Stryk, O., Meyer, J., and Klingauf, U., "A Flexible and Scalable SLAM System with Full 3D Motion Estimation," in *9th IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, 10.1109/SSRR.2011.6106777.
15. Hess, W., Kohler, D., Rapp, H., and Andor, D., "Real-Time Loop Closure in 2D LIDAR SLAM," in *IEEE International Conference on Robotics and Automation*, 2016, 10.1109/ICRA.2016.7487258.
16. Agarwal, S. and Mierle, K., Ceres Solver, http://ceres-solver.org, accessed Oct. 2019.

Contact Information

Basu Thakur, Mugdha
Clemson University,
Department of Automotive Engineering,
4 Research Drive, Greenville,
SC, 29607
mbasuth@clemson.edu

Definitions/Abbreviations

SLAM - Simultaneous localization and mapping

ROS - Robotic Operating System

CU-ICAR - Clemson University-International Center for Automotive Research

ARMLab - Automation, Robotics and Mechatronics Laboratory

IMU - Inertial Measurement Unit

ISSN 0148-7191