Comparative analysis of ROS based 2D and 3D SLAM algorithms for Autonomous Ground Vehicles

Sankalprajan P*
(Intern)
KPIT Technologies Ltd., Bangalore, India

Thrilochan Sharma**
(Intern)
KPIT Technologies Ltd., Bangalore, India

Hamsa Datta Perur
KPIT Technologies Ltd., Bangalore, India

Dr. Prithvi Sekhar Pagala
KPIT Tech Ltd., Bangalore, India

(MTech, Mechatronics Engineering, Vellore Institute of Technology, Chennai, India)*
*(** MTech, Automation and Robotics Engineering, University of Petroleum and Energy Studies, Dehradun, India)*

***Abstract*— Simultaneous localization and mapping algorithm (SLAM) is a technique for estimating sensor motion and reconstructing the structure in an unknown environment. SLAM is extensively used in the concept of autonomous driving or navigation which helps robotic vehicles to move autonomously. This paper presents a comparative analysis of different ROS based 2D SLAM algorithms such as GMapping, Hector SLAM, Karto SLAM and 3D SLAM algorithm such as Real-Time Appearance-Based Mapping. Scenarios with a scaling factor resembling an underground parking was built in real-time and simulation for analysis and validation purposes. The maps generated by applying the respective SLAM algorithms in real-time and simulation was analyzed and validated quantitatively based on quality and accuracy using the Structure Similarity Index.**

Keywords—Simultaneous localization and mapping, robotic vehicle, GMapping, Hector SLAM, Karto SLAM, RTAB-Map, Structure Similarity Index.

I. INTRODUCTION

Robotics is a field of engineering that deal with design and application of robots and the use of computer for their manipulation and processing. They are used in various fields such as industrial automation, nuclear science, sea-exploration, medical surgeries, etc. Robotics requires the application of computer integrated manufacturing, mechanical engineering, electrical engineering, biological mechanics, software engineering. Robots are also used in industries for speeding up the manufacturing process by transferring resources and moving people from one place to other autonomously. These systems can be called as robotic vehicles, autonomous vehicles, robo-car, etc.

The concept of the movement of robotic vehicles is known as autonomous driving or autonomous navigation. Autonomous Driving is amongst one of the most researched concepts in the automobile sector. The concept of autonomous driving comprises 6 levels according to the Society of Automotive Engineers (SAE) [1] from level 0 (fully manual) to level 5 (fully automated). The autonomous driving feature majorly ensures the safety of the user by avoiding accidents and reducing fatalities along with the increase in comfort [2] which is a major concern amongst consumers. The cost of building a framework with all the necessary tools that provide autonomous features for the consumers is a huge investment for the industries and this paves way for the possible usage of open-source framework and tools in the automobile industry.

Robot Operating System (ROS) is an opensource middleware framework which has a high number of tools and packages that can be applied for developing new applications. The major advantage of the usage of ROS in the OEMs and Industries is that it simplifies the activities to the task level [3]. The ROS is a framework which offers a communication channel to different algorithm nodes and Autonomous Ground Vehicle (AGV) nodes along with advantage of existing features. Automobile industries are also slowly moving towards the usage of ROS with the development of ROS Industrial and ROS 2. Companies like BMW have started using ROS [4] in their systems for the application of autonomous driving.

One of the algorithms for autonomous driving is the Simultaneous Localization and Mapping (SLAM). SLAM is the feature applied in autonomous vehicles to generate a map of the environment it explores (mapping) and simultaneously find the vehicle's position (localization) using necessary sensors. This functionality of the SLAM is the major factors that make it a suitable method in the field of autonomous driving. The classification of SLAM can be also based on the types of sensors used majorly [5], laser (2D or 3D), vision based (monocular, stereo, omnidirectional and RGBD) and radar. As shown in Figure-1, many SLAM algorithms have been implemented with the Bayes filters. According to the surveys [6][7], The Kalman filters (KF) is one of the applications based on the Bayes Filter. The KF has two parts which are the Prediction and Updating stage. The prediction stage uses the previous values and iterations to predict the current state's state. The update phase uses this predicted state and combines its current data collected from the sensor nodes. This is called the posterior. The Extended Kalman Filter (EKF) [5], is developed on top of the KF to mainly eliminate the Non-Linearity problem of the pose model of the vehicle (AGV). In the EKF method, the current state is estimated by a first-order Taylor expansion. In most of the cases, the estimates are nearly close to the ground truth. In cases where the map is continuously growing, the EKF is unable to support the large-scale SLAM as the update time depends on the size of the state vector. To counter this problem, the concept of creating sub-maps was introduced. When the map scale and size become large, a blank map (sub-map) is created and the existing map is replaced. To track the links between the sub-maps and not lose data, a higher-level map is employed. This increases the need for computational power and is a major disadvantage of the EKF method.

The second branch of the Filter-based SLAM is the Particle Filter (PF) [6] in which a set of particles which have individual weights are used to represent the posterior probability. PF usually correlates the value of the current state with the next state. The PF uses multi-modal distribution to predict uncertainty and can handle the non-gaussian noise. The FastSLAM [5] is developed based on the PF and is mainly utilized for the estimation of the posterior. The landmark locations are mapped to KFs present in each particle in the FastSLAM. The computational complexity of the FastSLAM is lesser than the EFK even when the number of landmarks is increased. Both KF and PF has a long-term inconsistency problem.

Optimization based SLAM [5] consists of two subsystems. The first subsystem estimates the constraints based on the sensor data by calculating the relation for the new observations with and the map. The second subsystem estimates the vehicle posterior and the map with the constraints. Based on Filters, they are further divided into Bundle adjustment and Graph SLAM. The bundle adjustment is a method which uses vision to combinedly optimizes a 3D structure and the parameters of the camera (pose). The Graph-based SLAM builds a map based on the data obtained. Arcs and Nodes mainly compose the graph. Each Arc in the graph represents a constraint between two successive poses. The arc depicts the possibility of a measurement event or a motion event. The constraints are linearized to obtain a sparse matrix to create a sparse graph map. A reduction process is used to remove the redundant map variables while the optimization process makes it efficient for even large scenarios. The Graph-based SLAM is a method developed to overcome the disadvantages present in PF and EKF.

The next major step after constructing a map from a SLAM algorithm is the evaluation of the results obtained. Prior research has investigated several factors related to the efficiency of the SLAM algorithms in real-time or simulation by applying various metrics. Existing surveys on the analysis of the SLAM algorithms evaluate the trajectories obtained [8][9][10], processing time and computational power [11][12][13] either in simulation or real time. While a few concentrates on the map accuracy and quality [14][15]. The validation of SLAM algorithms based on the comparison of real-time and simulation outcomes, either on comparison with the ground truth or between each other is an essential study needed to estimate its efficiency.

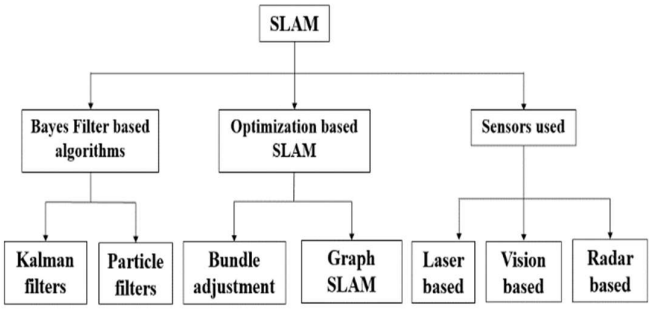


Fig. 1. Classification of SLAM based on Algorithm and Target Generation

To extend the quantitative analysis demonstrated by the previous studies; a comparative analysis of the SLAM algorithms on the accuracy and quality of the maps generated is investigated using the Structural Similarity Index Method (SSIM) [16]. Existing studies [17][18] highlight that the SSIM is better than other conventional image quality measurement methods. In this paper, the maps generated in real time and simulation are compared to each other using the SSIM metric. Additionally, the maps generated are compared to their respective ground truths for the validation purposes.

The outline of the paper is as follows. The system setup section describes the AGV configurations, the real scenario, the scaled down environments and the SLAM algorithms used for evaluation. The results and analysis section describe the implementations of the SLAM algorithms and the output obtained from them along with comparative analysis based on the generated Map's accuracy and Quality. Finally, the conclusions are drawn, and the future work is outlined.

II. SYSTEM SETUP

A. Real-Time AGV

The Real-Time AGV shown in Figure-2, is a Scaled down prototype of a commercial on road vehicle. The dimensions of the AGV are equated with the dimensions of a real car by a scaling factor of 8, that is, the dimensions of a real Sports Utility Vehicle (SUV) are measured relative to the distance to the obstacles (pillars and boundaries) and then a scaling factor is obtained. The prototype is designed using this scaling factor. The hardware components are a Master PC, Single Board Computer (SBC) for the host PC, Microcontroller, Depth perception camera, LIDAR, Inertial Measurement Unit (IMU), Motor Driver, Motors and Power source. The master PC also runs the master node of the software for of the real time AGV. For the purpose of experiment and to decrease complexity. The real time scaled down AGV model is treated similar to a standard Electric Vehicle due to its similarity of components. The features such as the vehicle dynamics, powertrain, etc. are not recreated in the AGV prototype and focus is on the SLAM related aspects. The master PC runs Ubuntu Operating System and ROS along with the necessary packages installed in it. The SBC is the Remote Host which has ROS installed in it. The Master PC and the Remote Host works like a Master and Slave. The SBC is the node that is responsible for the transfer of data from the sensor nodes to the Master PC and vice versa. It is also responsible for the movement of the vehicle or the AGV. The detailed specifications of the real time AGV are tabulated and given in Table 1.


Fig. 2. The hardware setup of the AGV.

TABLE I. DETAILED SPECIFICATIONS OF THE DIFFERENT COMPONENTS

Node	Specifications
Master PC	RAM: 8GB DDR4 SSD: Samsung EVO CPU: Intel i5 GPU: Nvidia MX150-4GB
SBC (Remote PC)	RAM: 1 GB Storage : SD/MMC/SDIO Processor: 1.2GHz Quad-Core ARM Cortex-A53 (64Bit)
LIDAR	Distance Range: 0.2 – 6m Angular Range:0-360° Distance Resolution: <0.5mm Angular Resolution: =<1° Sample Duration: 0.5 Mili Seconds Sample Frequency: >=2000Hz Scan Rate: 5.5Hz
Camera	Specifications: Horizontal field of view: 57 degrees Vertical field of view: 43 degrees Tilt Motor Range: -27- 27 degrees vertical (from base)

B. Simulation model of the AGV

The Simulation model shown in Figure-3, is scaled down model of the real SUV car implemented in the simulation platform. The dimensions of a real Sports Utility Vehicle (SUV) are measured relative to the distance to the obstacles (pillars and boundaries) and then the scaling factor is obtained. The dimensions of the Simulation model are equated with the dimensions of a real car by the same scaling factor of 8. The prototype is designed using this scaling factor. The simulation model is described in the Unified Robot Description Format (URDF) form. The URDF file contains complete description of the hardware part of simulation model. The .urdf file contains the description features like the sensor plugins, joints, links, meshes, etc. This simulation model can be used for performing SLAM algorithms in any simulation environment designed that supports the file format.

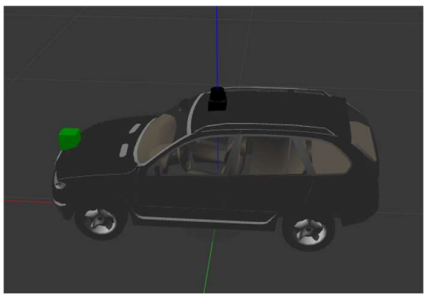
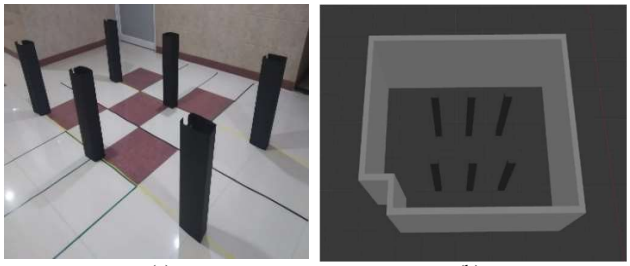


Fig. 3. Simulation model of the AGV

C. Scenario and Environment

For the validation of the SLAM algorithms, the underground parking scenario shown in Figure-4 is used. Since, the experiments either in real time or simulation needs a scenario for the validation, a scaled down version of the underground parking scenario must be reconstructed in the real time as shown in Figure-5(a) simulation platform as shown in Figure-5(b). The simulation or real time environment is constructed using the same scaling factor, that is the distance of the AGV model to the obstacles (pillars and boundaries) in the simulation platform or real time environment are of the same ratio as it was in the underground parking scenario. The real time environment is constructed by using the same scaling factor and constructing using the same method, i.e., measuring the distance from the real time AGV and the obstacle. The Simulation involves a platform called gazebo [19] in which the underground parking scenario can be replicated and saved as a world. The gazebo simulator has the capability to replicate the physical properties like gravity, friction, etc. The simulation environment is created as a .world file which contains all the descriptions of the obstacles. The same scaling factor and method is considered to construct a world in the simulator also. The .urdf file is put in the simulation environment to complete the simulation setup. The Gazebo publishes the topics from the sensor plugins that can be used to generate maps of the environment and be visualized using visualization tools. Similarly, the Real-Time AGV and the Real-Time environment together completes the Real-Time setup. The necessary binaries installed in the Host PC helps in subscribing to and sending data to Master PC. This data acquired is essential to generate maps for the real-time environment

Fig. 4. The underground parking area



(a) (b)

Fig. 5. Real time environment (a) and Gazebo .world Scenario (b).

D. Metrics for SLAM Analysis and Validation

Structure Similarity Index (SSIM): The SSIM metric gives us the measure of structural similarity (local) that compare the cell intensities pattern (local) as shown in equation 1.

$$SSIM = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x{}^2+\mu_y{}^2+c_1)(\sigma_x{}^2+\sigma_y{}^2+c_2)} \quad (1)$$

Where, μ_x is the average of x, μ_y is the average of y, $\sigma_x{}^2$ is the variance of x, $\sigma_y{}^2$ is the variance of y, σ_{xy} is the covariance of x and y, c_1 and c_2 are two variables to stabilize the division the weak denominator.

The SSIM is a metric that will compare the two maps which are of the same size, i.e., length and breadth. The SSIM is a metric which calculates the mean value, the variance and the covariance of the intensities of the cells obtained from the image provided. The SSIM value can range from -1 to +1. The +1

proffer that both the maps are the same. The SSIM metric constructs the difference in the image in a structural information format and thus the accuracy is more than other conventional methods.

E. SLAM algorithms used

GMapping is a SLAM algorithm which is based on the Rao Blackwellised Particle Filter approach [20]. The RBPF uses adaptive resampling technique. This helps to decrease the particle-depletion problem and computational complexity. The resampling process is not needed much and is only done when it is necessary. The GMapping algorithm is designed in such a way that it integrates the current sensor data with the odometry motion model. This helps the particle filter's prediction step of uncertainty about AGV's location to decrease. Thus, the particles needed has decreased. The GMapping is a lidar based SLAM.

Hector SLAM is a SLAM algorithm which integrates laser scan matching feature with the 3D navigation method by the help of the inertial system which employs the EKF [21]. Hector SLAM is designed in a way that it can compute the 6 degrees of freedom of the AGVs position and orientation during motion and parallelly the high update rate laser-based 2D map. The Gaussian-Newton optimization approach is used to find the alignment of the laser scan's endpoints with which the scan matching is done. It is a SLAM algorithm which does not use the odometry data. Thus, the Hector SLAM has an advantage when used in environments which exhibit the pitch and roll characteristics. The algorithm does not have a non-erroneous loop closure feature.

Karto SLAM algorithm is one of the algorithms that works based on the graph-based method to localize the vehicle or the AGV and to perform mapping [22]. The Karto SLAM utilizes the current pose and the complete map along with the adding of updating the pose estimate. This makes the Karto SLAM an efficient SLAM than the others. The major disadvantage with the Kato SLAM is that the complexity of computation is proportional to the number of landmarks. As the number of landmarks increases, the computational complexity also increases. The matrix updating method is similar to the matrix updating of the EKF-SLAM.

RTAB-Map is a SLAM algorithm which can be implemented with any type of sensor that provides depth information or three-dimensional information [23]. The RTAB-Map is based on the RGB-D Graph SLAM. It can be implemented with 3D lasers, Stereo vision and even with RGB-Depth cameras. RTAB-MAP is based on the graph-based method for Stereo and RGB-D. This method has a global loop closing detection technique [24] in-built to compute the vehicle pose and an effective memory managing system. The map is saved as a structured graph (textured RGB dense point clouds) with AGV's position and orientation and the captured image at that pose as nodes. The Map also contains the odometry data or the loop closing matrices as its edges. The major disadvantage is that the complexity of computation increases with the increase in the size of the map. The RTAB-Map SLAM can also be implemented without the need for odometry data.

III. RESULTS AND COMPARATIVE ANALYSIS OF SLAM ALGORITHMS

The Simulation needs two main components. One, the AGV Simulation model spawned in the Gazebo Scenario and one the SLAM node. The SLAM node gets these lidar data and then construct the Map. The Map is saved by using a Map server in the case of GMapping, Hector and Karto SLAM. The RTAB-Map has an internal server that saves the Map by itself. Similarly, every node in the real time is individual, unlike the Simulation where all nodes are interdependent. All the SLAM algorithms were implemented and tested under the same conditions. The teleoperation of the AGV was recorded in a bag file and played for all the algorithms to replicate the movement of the AGV in the environment while mapping.

The Comparative Analysis of the Real-Time and the Simulation results of each SLAM algorithms are based on two major factors. They are the Map Quality and Map accuracy. Map accuracy is the major factor that determines the efficiency of the SLAM Algorithm. The Accuracy is measured using the SSIM metrics. Map Quality is described based on visual observation. The Map Quality mainly depends on the sensors, the processors and the Environment. The Sensors used are LIDAR and a Depth camera. Analysis based on these factors is discussed below. The basic elements of the SLAM algorithms are summed up ad tabulated and given in Table 2.

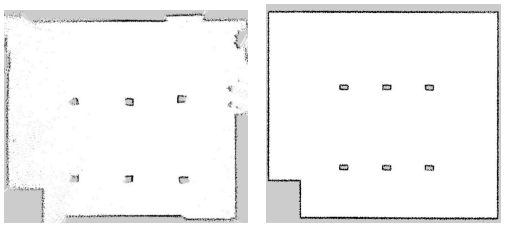
The Comparative Analysis of the Real-Time and the Simulation results of each SLAM algorithms are based on two major factors. They are the Map Quality and Map accuracy. Map accuracy is the major factor that determines the efficiency of the SLAM Algorithm. The Accuracy is measured using the SSIM metrics. Map Quality is described based on visual observation. The Map Quality mainly depends on the sensors, the processors and the Environment. The Sensors used in this research are LIDAR and a Depth camera. Analysis based on these factors is discussed below.

The maps of all the SLAM algorithms were obtained by implementing all the algorithms in the same conditions. All the Maps were generated by teleoperating the vehicle the same way. The maps generated by both the real-time and the simulation sessions were oriented and a python code using the Structural Similarity Index which internally depends on OpenCV for measuring the similarity between the Maps.

TABLE II. BASIC COMPARISON TABLE FOR 2D AND 3D SLAM ALGORITHMS

SLAM	Sensors	Classification	Based On	Odometry needed	Loop Closure Present
GMapping	2D Lidar	Particle filter	FastSLAM	YES	YES
Hector SLAM	2D Lidar	Kalman filter	EKF for 3D state estimation	NO	YES
Karto SLAM	2D Lidar	Optimization based SLAM	Graph SLAM	YES	YES
RTAB-Map	3D Lidar/ Stereo camera/ RGB-D camera	Optimization based SLAM	RGB-D Graph SLAM	NO	YES

GMapping: The map created by the GMapping Node, both in Real-Time and the Simulation is shown in Figure-6.



(a) (b)

Fig. 6. GMapping maps generated in Real-Time (a) and Simulation (b)

The SSIM index value obtained when the maps generated by the Simulation is compared to the ground truth is 0.84. Similarly, when the map generated by the Real-Time is compared with its respective Ground truth, the index value is observed to be 0.83. On comparison between maps generated by Real-Time and Simulation, the index value obtained is 0.86. This indicates that the Maps generated in the Real-Time and the Simulation are very similar. The generated Map's Quality of the Simulation and the Real-Time sessions are nearly the same. The GMapping majorly depends on parameters such as number of parameters, resampling threshold, linear update and angular update [25]. These parameters were set to a moderate level which ensures performance. The major disadvantage of GMapping is its dependency on Odometry. If the odometry values have a slight error, it will cause an error in the map structure and affect accuracy and quality. The GMapping has a higher CPU usage as well as memory usage than other types of SLAM. The GMapping takes more time to process the data received, thus the AGV had to be slowly teleoperated during the mapping session. This also influenced the overall session time

Hector SLAM: The map created by the Hector SLAM Node, both in Real-Time and the Simulation is shown in Figure-7.

(a) (b)

Fig. 7. Hector SLAM maps generated in Real-Time (a) and Simulation (b)

The SSIM index value obtained when the maps generated by the Simulation is compared to the ground truth is 0.81. Similarly, when the map generated by the Real-Time is compared with its respective Ground truth, the index value is observed to be 0.78. On comparison between maps generated by Real-Time and Simulation, the index value obtained is 0.75. This indicates that the Maps are nearly similar. The Quality of the Map generated by the Simulation sessions is better than the Real-Time sessions. The Hector also depends on factors such as angle threshold, linear threshold and update factors. The Edges of the maps for the Hector SLAM had errors as the Hector SLAM has a slow map update rate. The Hector SLAM, when simulated in a large and complex Scenario gives out errors like the angle of inclination. The Hector doesn't depend on odometry but if there is any change in the angle, the accuracy of the map is decreased.

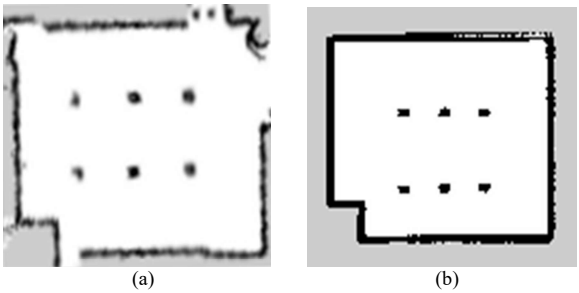
Karto SLAM: The map created by the Karto SLAM Node, both in Real-Time and the Simulation is shown in Figure-8.

(a) (b)

Fig. 8. Karto SLAM maps generated in Real-Time (a) and Simulation (b)

The SSIM index value obtained when the maps generated by the Simulation is compared to the ground truth is 0.75. Similarly, when the map generated by the Real-Time is compared with its respective Ground truth, the index value is observed to be 0.77. On comparison between maps generated by Real-Time and Simulation, the index value obtained is 0.76. This indicates that the Maps generated by the Karto SLAM Real-Time session and the Simulation are little erroneous, but they are similar. The Quality of the Map generated by the Simulation session is better than the Real-Time session. All the parameters in the Karto SLAM are set to the default values. The Karto SLAM performs faster mapping than others with lesser errors. The Karto SLAM is a little less accurate compared to other SLAM algorithms and is very similar to the working of the GMapping SLAM.

RTAB-Map: The map created by the RTAB-Map Node, both in Real-Time and the Simulation is shown in Figure-9.



(a) (b)

Fig. 9. RTAB-Map maps generated in Real-Time (a) and Simulation (b)

The Projection of the 3D map obtained is used for the validation The SSIM index value obtained when the maps generated by the Simulation is compared to the ground truth is 0.81. Similarly, when the map generated by the Real-Time is compared with its respective Ground truth, the index value is observed to be 0.76. On comparison between maps generated by Real-Time and Simulation, the index value obtained is 0.75. This indicates that the Maps generated by the RTAB-Map Real-Time session and the Simulation are little erroneous, but they are similar. The Simulation session map generated is better than the Real-Time session generated map. The parameters in RTAB-Map were set to the default values. The RTAB-Map is nearly the same as Hector SLAM inaccuracy. RTAB also has the disadvantage of dependency on Odometry. The RTAB map was

very slow in the processing of the point cloud data received from the camera. The point clouds were not properly generated if the AGV was teleoperated fast. The total mapping session time taken was high. The SSIM index of Simulation and Real-Time maps using Ground truth are tabulated under Table 3 and the SSIM index of Simulation and Real-Time maps using Ground truth are tabulated under Table 4.

TABLE III. SSIM INDEX AND QUALITY COMPARISON OF SIMULATION VS REAL-TIME MAPS

SLAM	SSIM (Real-Time vs Simulation)	Quality Comparison
GMapping	0.86	Almost equal
Hector SLAM	0.75	Simulation is better
Karto SLAM	0.76	Simulation is better
RTAB-Map	0.75	Simulation is better

TABLE IV. SSIM INDEX OF SIMULATION AND REAL-TIME MAPS USING GROUND TRUTH.

SLAM	SSIM (Simulation vs Ground truth)	SSIM (Real-Time vs Ground truth)
GMapping	0.84	0.83
Hector SLAM	0.81	0.78
Karto SLAM	0.75	0.77
RTAB-Map	0.81	0.76

The comparative analysis derives that GMapping is very efficient when it comes to both Real-Time mapping and Simulation Mapping., followed by Hector SLAM. Then follows the RTAB-Map. The RTAB-Map and the Hector SLAM are nearly same with respect to mapping quality and accuracy. The Karto SLAM is the most erroneous among the used SLAM algorithms.

Additionally, the analysis also derives that the map's generated in Real-Time mapping session has more errors than the Simulation sessions. The factors such as wall inclination, odometry readings, surface unevenness of the ground plane and walls tend to influence the results obtained in the Real-Time more than the Simulation sessions. These factors are considered non-erroneous in the Simulation setup. Thus, the maps generated in Real-Time session has errors in the edges of the maps and the quality.

IV. CONCLUSIONS

In this research work, the comparative analysis of ROS based 2D and 3D SLAM Algorithms which plays a major role in autonomous driving were done through simulations and Real-Time Experiments. A discussion of the working of each Algorithm been done. The Validation of these Algorithms was done by deriving the mapping quality and accuracy using the SSIM index. All the Implementations were done on a lower level GPU and CPU and results may vary when implemented in other specification systems. It can be concluded that the best Algorithm to generate maps both in Real-Time, as well as Simulation, is GMapping. The results obtained from the validation also concur with the results produced in the previous literatures. In future work, the researchers are planning to compare 3D SLAM techniques Real-Time and their Simulation.

REFERENCES

[1] "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles", SAE International, 2016.

[2] Amna Chaudhry, Peng Liu, Amjad Hussain, Safety Perceptions of Self-driving Cars: A Survey Study in China and Pakistan, vol. 786. Springer International Publishing, 2019.

[3] A. N. Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, "ROS: an open-source Robot Operating System," ICRA Work. open source Softw., vol. 3, p. 5, 2009.

[4] M. (Bmw) Aeberhard, "Automated Driving with ROS at BMW," ROSCon 2015, 2015.

[5] T. J. Chong, X. J. Tang, C. H. Leng, M. Yogeswaran, O. E. Ng, and Y. Z. Chong, "Sensor Technologies and Simultaneous Localization and Mapping (SLAM)," Procedia Comput. Sci., vol. 76, no. January 2016, pp. 174–179, 2015.

[6] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," IEEE Trans. Intell. Veh., vol. 2, no. 3, pp. 194–220, 2017.

[7] C. Montella, "The Kalman Filter and Related Algorithms A Literature Review," Res. Gate, no. May, pp. 1–17, 2014.

[8] M. Santos and R. P. Rocha, "An Evaluation of 2D SLAM Techniques Available in Robot Operating System An Evaluation of 2D SLAM Techniques Available in Robot Operating System," no. October 2015, 2013.

[9] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ROS-based visual SLAM methods in homogeneous indoor environment," 2017 14th Work. Positioning, Navig. Commun. WPNC 2017, vol. 2018-Janua, no. October, pp. 1–6, 2018.

[10] R. Giubilato, S. Chiodini, M. Pertile, and S. Debei, "An experimental comparison of ROS-compatible stereo visual SLAM methods for planetary rovers," 5th IEEE Int. Work. Metrol. AeroSpace, Metroaerosp. 2018 - Proc., pp. 386–391, 2018.

[11] G. Tuna, K. Gulez, V. Cagri Gungor, and T. Veli Mumcu, "Evaluations of different Simultaneous Localization and Mapping (SLAM) algorithms," IECON Proc. (Industrial Electron. Conf., pp. 2693–2698, 2012.

[12] B. M. F. Da Silva, R. S. Xavier, T. P. Do Nascimento, and L. M. G. Gonsalves, "Experimental evaluation of ROS compatible SLAM algorithms for RGB-D sensors," Proc. - 2017 LARS 14th Lat. Am. Robot. Symp. 2017 5th SBR Brazilian Symp. Robot. LARS-SBR 2017 - Part Robot. Conf. 2017, vol. 2017-Decem, no. August 2018, pp. 1–6, 2017.

[13] P Thrilochan Sharma, P Sankalprajan, Ashish Joel Muppidi, Prithvi Sekhr Pagala, "Analysis of computational need of 2D-SLAM Algorithms for an Unmanned Ground Vehicle." (accepted in ICICCS 2020).

[14] R. Yagfarov, M. Ivanou, and I. Afanasyev, "Map Comparison of Lidar-based 2D SLAM Algorithms Using Precise Ground Truth," 2018 15th Int. Conf. Control. Autom. Robot. Vision, ICARCV 2018, pp. 1979–1983, 2018.

[15] M. Rojas-Fernandez, D. Mujica-Vargas, M. Matuz-Cruz, and D. Lopez-Borreguero, "Performance comparison of 2D SLAM techniques available in ROS using a differential drive robot," 2018 28th Int. Conf. Electron. Commun. Comput. CONIELECOMP 2018, vol. 2018-January, no. April, pp. 50–58, 2018.

[16] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural imilarity", IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, 2004.

[17] U. Sara, M. Akter, and M. S. Uddin, "Image Quality Assessment through FSIM , SSIM , MSE and PSNR — A Comparative Study," pp. 8–18, 2019.

[18] A. Wadhokar, K. Sakharikar, S. Wadhokar, and G. Salunke, "SSIM Technique for Comparison of Images," Int. J. Innov. Res. Sci. Eng. Technol., vol. 03, no. 09, pp. 16281–16286, 2014.

[19] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," 2004.

[20] R. P. Guan, B. Ristic, and L. Wang, "Combining KLD-Sampling with Gmapping Proposal for Grid-Based Monte Carlo Localization of a Moving Robot," 2017.

[21] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. Von Stryk, "Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robots," pp. 624–631, 2014.

[22] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2D mapping," IEEE/RSJ 2010 Int. Conf. Intell. Robot. Syst. IROS 2010 - Conf. Proc., no. October, pp. 22–29, 2010.

[23] M. Labb, "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation," January, 2016.

[24] M. Labbé and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based SLAM," IEEE Int. Conf. Intell. Robot. Syst., pp. 2661–2666, 2014.

[25] Y. Abdelrasoul, A. Bakar, S. Hm, and P. Sebastian, "A Quantitative Study of Tuning ROS Gmapping Parameters and Their Effect on Performing Indoor 2D SLAM