e-ISSN: 2146 - 9067

International Journal of Automotive Engineering and Technologies

journal homepage:

https://dergipark.org.tr/en/pub/ijaet

Original Research Article

Performance analysis of different SLAM packages in autonomous robots using structural similarity index measure

Mehmet Zeki Sildir [1], Koray Erhan [2, *]

[1, 2 *] Department of Mechatronics Engineering, Faculty of Technology, Marmara University, Istanbul, Türkiye.

ARTICLE INFO

Orcid Numbers

1. 0009-0006-5993-2744
2. 0000-0003-0505-9389

Doi: 10.18245/ijaet.1634390

* Corresponding author
korayerhan@gmail.com

Received: Feb 14, 2025
Accepted: Jul 29, 2025

Published: 30 Sep 2025

Published by Editorial Board Members of IJAET

To cite this paper: Sildir, M., Z., Erhan, E., Performance analysis of different SLAM packages in autonomous robots using structural similarity index measure, International Journal of Automotive Engineering and Technologies. 2025, 14 (3), 181 – 189. http://dx.doi.org/10.18245/ijaet.1634390

ABSTRACT

In this study, a performance analysis is conducted on robot localization and mapping in the context of differential-drive autonomous mobile robots, a field commonly referred to as Simultaneous Localization and Mapping (SLAM). For a mobile robot to move from a starting point to a target location and complete assigned tasks effectively, it requires a reliable environmental representation namely, a map. The creation of such a map is managed through various sensor inputs and algorithmic processes, predominantly implemented within Robot Operating Systems (ROS). With the transition from ROS1 to the more advanced ROS2 framework, SLAM software packages have also evolved. This study evaluates and compares the performance of three SLAM packages Gmapping, Cartographer, and SLAM Toolbox within the ROS2 environment using the Create-3 mobile robot platform, which is specifically developed for ROS2 applications. The experiments are conducted in three different indoor environments with varying characteristics, including a narrow, obstacle-dense room; a corridor; and a wide room with sparse features. All trials are performed using a consistent speed profile and sensor setup. Performance is assessed based on the Structural Similarity Index Measure (SSIM) between the generated maps and manually created reference maps, as well as mapping speed. The results indicate that SLAM Toolbox consistently outperformed the other packages, achieving the highest SSIM values (particularly 0.72 in the wide-room scenario) and delivering more accurate maps in complex environments. Cartographer SLAM performed well in corridor-like spaces but exhibited decreased accuracy in larger, obstacle-rich environments. Gmapping provided relatively stable results but lagged behind in terms of fine-grained mapping precision. These findings demonstrate that SLAM Toolbox is better suited for applications requiring high-accuracy 2D mapping in ROS2-based systems. The study offers a comprehensive comparison framework and concrete performance data, guiding researchers and developers in selecting the most suitable SLAM approach based on environmental conditions and system requirements.

Keywords: Mobile Robot, Robot Operating System, Simultaneous Localization and Mapping, Kalman Filter, Particle Filter, Structural Similarity Index Measure.

1. Introduction

Autonomous mobile robots are widely deployed in diverse environments, performing tasks that require reliable navigation and mapping capabilities. The core functionality that enables this is known as Simultaneous Localization and Mapping (SLAM), wherein the robot concurrently builds a map of the environment and estimates its position within it [1,2].

Earlier studies have extensively addressed SLAM in outdoor and indoor scenarios. Outdoor applications, for example, tackle dynamic challenges such as lane keeping and intersection handling under unpredictable traffic [3], whereas indoor scenarios emphasize accuracy in confined, obstacle-rich spaces, such as warehouses and service robots [4].

SLAM research has evolved significantly in recent years, with the emergence of the Robot Operating System 2 (ROS2) providing improved communication infrastructure, real-time processing, and multi-robot support [5,6]. This has led to renewed interest in evaluating SLAM software specifically within ROS2 environments [7,8]. However, comparative studies focusing on ROS2-based SLAM packages remain sparse, and most evaluations still rely on ROS1 frameworks.

Moreover, while standard SLAM evaluation metrics such as Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) are common, they are often insufficient to assess the quality of 2D map outputs. The Structural Similarity Index Measure (SSIM) has emerged as an alternative metric, providing perceptually meaningful comparisons between generated and ground-truth maps [9]. Recent work has demonstrated SSIM’s sensitivity in detecting structural differences in SLAM-generated maps, particularly in dense indoor environments [6,10].

Against this backdrop, our study investigates the performance of three widely used SLAM packages Gmapping, Cartographer, and SLAM Toolbox within the ROS2 framework. We specifically evaluate these packages in three distinct indoor environments, leveraging SSIM and mapping speed as primary metrics. This focused evaluation provides updated insights into the relative strengths and weaknesses of the selected SLAM software and serves as a practical guide for researchers and developers working with ROS2-based robotic systems.

2. Literature Review

Simultaneous Localization and Mapping (SLAM) is a fundamental capability for autonomous mobile robots, enabling them to build a map of the environment while simultaneously estimating their position within it. Over the years, various SLAM software packages have been developed and evaluated using different metrics, primarily within the ROS (Robot Operating System) framework.

Early comparative studies focused on ROS1-based SLAM packages, evaluating aspects such as CPU load, mapping accuracy, and execution time. Bettencourt et al. [5] evaluated the CPU performance of five SLAM techniques in urban search and rescue robots, showing that Gmapping and Karto SLAM produced more efficient results than others. Kumar et al. [11] compared three SLAM software options using the Adaptive Directional Nearest Neighbor (ADNN) metric and found Google Cartographer achieved the lowest error rates in indoor environments. Similarly, Bhushan et al. [12] analyzed map creation times of four SLAM software packages, demonstrating that Google Cartographer outperformed others in creating maps more quickly. Mertyuz et al. [9] used the SSIM metric to evaluate SLAM packages implemented on the Turtlebot3 Burger platform, showing Hector SLAM produced better structural similarity scores.

In more recent studies, the focus has shifted to ROS2-based SLAM packages, which leverage improved communication protocols and multisensor capabilities. Qiu et al. [7] evaluated ROS2-based SLAM algorithms in highly dynamic indoor environments, concluding that SLAM Toolbox maintained robust performance despite frequent environmental changes. Li et al. [10] analyzed real-time SLAM techniques for service robots and highlighted the superior mapping fidelity of SLAM Toolbox while maintaining acceptable CPU performance. Kim et al. [6]

benchmarked ROS2 SLAM packages with multisensor fusion, finding that combining LiDAR, odometry, and camera data improved both SSIM and mapping speed, with Cartographer performing well in corridors and SLAM Toolbox excelling in dense environments. Gonzalez and Pereira [8] evaluated SLAM performance on resource-constrained robots and recommended SLAM Toolbox for its balanced computational load and high accuracy.

Collectively, these studies demonstrate that the choice of SLAM package should depend on the environmental conditions, sensor configuration, and computational resources. Our study builds on this body of work by experimentally comparing Gmapping, Cartographer, and SLAM Toolbox in ROS2 environments, using SSIM and mapping speed metrics across three distinct room configurations, and providing updated insights for researchers and practitioners.

3. Materials and Methods

3.1. Optimization based SLAM

Optimization-based SLAM approaches formulate the simultaneous localization and mapping problem as a global optimization task, where the entire trajectory of the robot and the positions of the landmarks are estimated by minimizing a cost function that measures the discrepancy between predicted and observed measurements. Unlike filter-based methods that incrementally update the state estimate based on probabilistic prediction and correction steps, optimization-based methods consider all past poses and observations to find the most likely configuration that best explains the data. This is typically done through iterative optimization techniques, such as least squares minimization, applied on a pose graph or factor graph representation of the SLAM problem.

In the Optimization-Based SLAM approach, a state vector containing all robot poses and all observed features is used. Linearization is performed in the iteration step. Optimization Based SLAM is also referred to as GraphSLAM in the literature [13]. Figure 1 shows the block diagram of the Optimization Based SLAM algorithm. If we look at the operation of the Graph SLAM algorithm, let's say we have a measurement set 'z1:t'. Correspondence variables associated with ‘z1:t’ are ‘c1:t’ and an array of controls ‘u1:t’. The notation c1:t refers to the sequence of correspondence variables between the measurements collected up to time t and the associated environmental features. These variables determine which observed measurements correspond to which landmarks in the map, and accurate estimation of c1:t is crucial for both optimization- and filter-based SLAM solutions. GraphSLAM transforms this data into a graph. The nodes of this graph are the robot poses ‘x1:t’ and the features in the map ‘m = {m_j}’. Each edge in the graph corresponds to an event: a movement event creates an edge between two robot poses, and a measurement event creates a connection between a pose and a feature in the map. Edges represent soft constraints between poses and features in GraphSLAM [13].

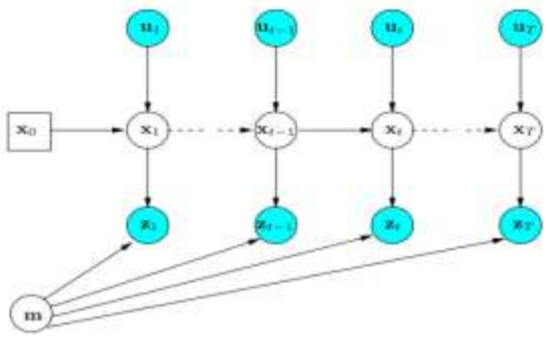


Figure 1. Optimization Based Approach Diagram [13]

3.2. Filter Based SLAM

In the Filter-Based SLAM approach, commonly used filters are Extended Kalman Filter (EKF) and Particle Filter (MCL). Although there are singular uses in the literature, the more effective SLAM approach is the fusion of the two. In the literature, using both as fusion is called FastSLAM [14].

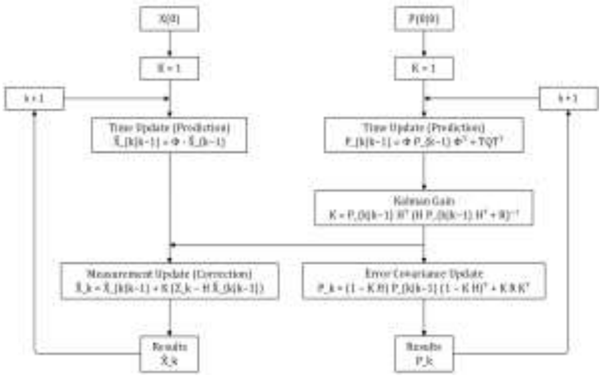
3.2.1. Kalman Filter

The Kalman Filter algorithm is a recursive filter based on Bayes Theorem and belongs to the Bayes Filter family. Bayes Theorem is used to calculate the probabilities of dependent events [15]. The equation (1) given below expresses Bayes' Theorem.

$$P\left(\frac{B}{A}\right) = \frac{P(A/B).P(A)}{P(B)} \quad (1)$$

Kalman Filter is based on an article written by

Rudolf Kalman in 1960. The KF algorithm was used for the first time in NASA's Apollo program. This filter is a powerful tool for estimating and predicting system states in the presence of uncertainty, but these systems are useful specifically for the Gaussian distribution in linear models [16]. The Kalman Filter algorithm basically starts by making an initial state estimate and uncertainty estimate while trying to estimate the system states. After the initial step, sensor measurement is performed. After the measurement, the system status and uncertainty are updated. After the update phase, the system state estimation step begins, and this cycle continues until the specified iteration or predetermined threshold value [17]. The flow chart of the Kalman Filter is shown in Figure 2.


X(0)
K = 1
k + 1
Time Update (Prediction)
X_(k|k-1) = Φ · X_(k-1)
Measurement Update (Correction)
X_k = X_(k|k-1) + K (Z_k − H X_(k|k-1))
Results
X_k
P(0|0)
K = 1
k + 1
Time Update (Prediction)
P_(k|k-1) = Φ P_(k-1) Φ^T + TQT^T
Kalman Gain
K = P_(k|k-1) H^T (H P_(k|k-1) H^T + R)^-1
Error Covariance Update
P_k = (1 − K H) P_(k|k-1) (1 − K H)^T + K R K^T
Results
P_k
Figure 2. Kalman Filter Flow Chart [16,17]

3.2.2. EKF

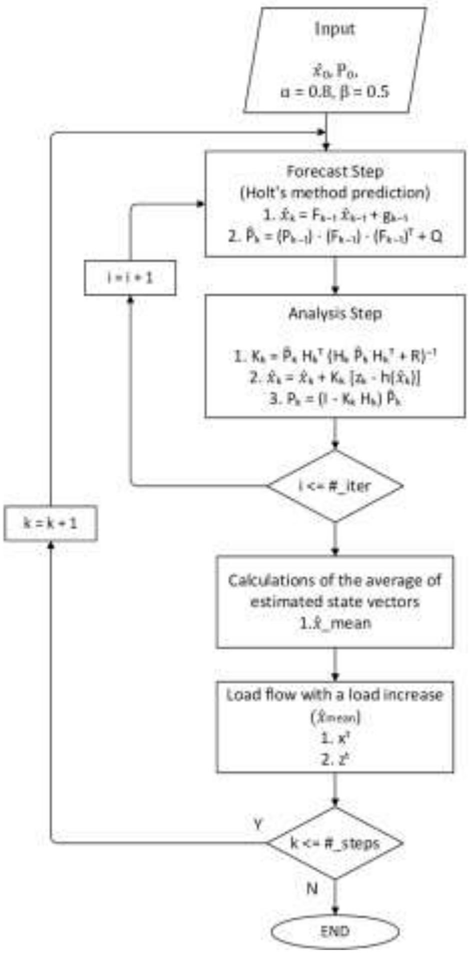
EKF is one of the very popular Kalman filter variations for estimating system states. It has a wide range of uses, from estimating the system states of engines to estimating robot poses. Unlike the basic Kalman filter, it is used in non-linear measurement and motion models. The systems around us show nonlinear behavior in terms of movement model and measurement model. It is valid for systems that can be modeled with a Gaussian distribution. Unlike a basic Kalman filter, linearization is also included [18]. The flow chart of the Extended Kalman Filter is shown in Figure 3.

3.2.3. MCL (Particle Filter)

Particle Filter, particles are the basis of the algorithm. While the robot is at any location on the map, the particles are randomly distributed in each location of the map. Each particle has x, y, and w components. These components; x (position on the X axis), y (position on the Y axis), orientation and w (weight of the particle) [19]. The general stages of the particle filter are:

- Initial State Estimate
- Motion Update
- Measurement Update
- Resampling
- State Predict

The reference for robot localization of the particle filter and the representation representing the robot position after filtering are given in Figure 4.


Input
x̂0, P0,
α = 0.8, β = 0.5
Forecast Step
(Holt's method prediction)
1. x̂k = Fk-1 x̂k-1 + gk-1
2. P̂k = (Pk-1) · (Fk-1) · (Fk-1)^T + Q
i = i + 1
Analysis Step
1. Kk = P̂k Hk^T (Hk P̂k Hk^T + R)^-1
2. x̂k = x̂k + Kk [zk − h(x̂k)]
3. Pk = (I − Kk Hk) P̂k
i <= #_iter
k = k + 1
Calculations of the average of estimated state vectors
1. x̂_mean
Load flow with a load increase
(x̂mean)
1. x^t
2. z^t
Y
k <= #_steps
N
END
Figure 3. EKF Flow Chart [18]

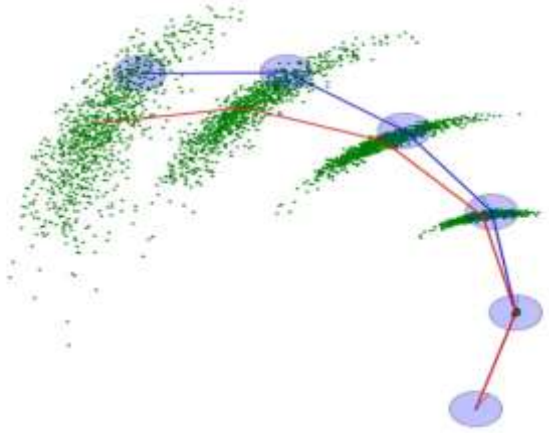


Figure 4. Robot Localization Using Particle Filter

3.3. SLAM packages and SSIM

The aim of this study is to measure the Structural Similarity Index by using the SLAM Toolbox package, known as the most up-to-date SLAM package used in the ROS2 system, and the Cartographer Slam and Gmapping packages, which are other SLAM packages configured for ROS2. Experimentally, to compare the performance of the methods and analyze their success in creating maps using the Create-3 training robot. A similar study is available in the literature for the ROS1 version and is carried out in the reference map simulation environment [20]. In this study as a novelty, the reference map is created using real environment data and sensors. Since there is no comparison in the literature for ROS2, especially based on the SLAM Toolbox package, compared to ROS1, this study presents concrete data by using SSIM in the performance analysis of the packages used in ROS2. SSIM is an image quality index used in many fields. It is a metric used to measure the perceptual similarity between two images (or signals). It is especially common for quality assessment in image processing and compression algorithms. The general definition of SSIM is given in Equation (2). SSIM is a statistical measure that is a product of three local difference factors: brightness, variance, and correlation. Table 1 shows package features. SLAM packages used in ROS2 provide “pgm” extension files as output, which is an image file. Therefore, it is appropriate to compare the reference image and package outputs using the SSIM metric. For two images A and B, the formula for SSIM per pixel is given by [21]:

$$\mathrm{SSIM}\,(x,y) = (l(x,y))^{\alpha}.\,(c(x,y))^{\beta}.\,(s(x,y))^{\gamma} \tag{2}$$

3.4. Sensor Fusion and Odometry Calculation

The scope of this study is to compare and analyze SLAM packages using the SSIM metric on the Create-3 robot, a robot developed for educational purposes. Here, instead of building a robot itself, odometry data is taken from the robot using a ready-made mobile robot. What is referred to as odometry here is wheel odometry. The wheel odometry method relies on wheel encoders mounted on a robot to monitor the number of revolutions made by each wheel. Sensor fusion integrates data from multiple sensing modalities to reduce the number of sensing uncertainties and overcome the shortcomings of individual sensors operating independently [22,23]. Create-3 robot fuses using IMU, Optical Mouse and Wheel Encoders to obtain odometry data and uses Extended Kalman Filter (EKF) for a good odometry estimation. Obtaining odometry data is shown in the diagram given in Figure 5.

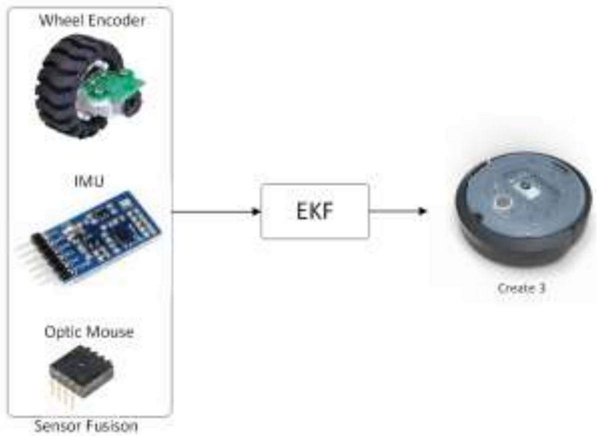


Figure 5. Odometry Calculation

Table 1. Package Features

Package Name	Sensor	Approach	Literature Name	Odometry
Gmapping SLAM	2D Lidar	Filter Approach (Particle Filter + EKF)	FastSLAM	YES
Cartographer SLAM	2D Lidar/3D Lidar/Camera	Optimization Based Approach	GraphSLAM	YES
SLAM Toolbox	2D Lidar/3D Lidar/ Camera	Optimization Based Approach	PoseGraph SLAM	YES

3.5. Hardware and Software Components of the Robot

The method part should be discussed under two main headings. The first part consists of the Create-3 robot, hardware connection and communication. The second part consists of the software layer, explaining the workspace created by ROS2 and the packages included in it.

3.5.1. Hardware Layer

In the hardware layer, Create-3 robot is used as robot, RPLidar-A1, Monster Abra A5 Mainframe and Raspberry Pi 4 are used as 2D Lidar. The connection on the Create-3 robot is given in Figure 6. The necessary condition for the Create-3 robot, Raspberry Pi single board computer and our main computer to be able to share topics, services, actions and parameters with each other is to be on the same network. In the next stages, the LIDAR sensor will be connected to the Raspberry Pi, and the host computer will connect to the Raspberry Pi using the SSH communication protocol and control it remotely. By connecting with the Raspberry Pi, it enables the LIDAR to run and, accordingly, remote data transfer and the robot to perform the necessary mapping operations via remote control.

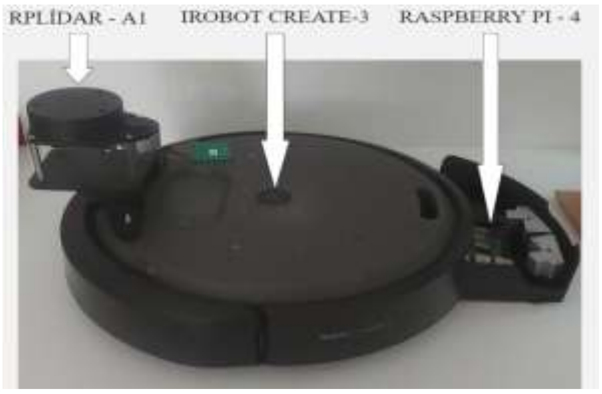


Figure 6. Create-3 and Connections

The connection diagram for this is given in Figure 7. The core system components consist of one PC, two antennas, a power converter, an RFID reader, a battery, and the Create-3 robot. The experimental system is shown in Figure 8.

3.5.2. Software layer

ROS2, the second version of the robot operating system (ROS) framework used for robots, is used in the software layer. Although ROS2 has many differences from ROS1, the main difference is that DDS is introduced as a protocol to the communication layer and, as in ROS1, the need for a master to run a node in ROS2 is eliminated. Detailed information about ROS2 can be found in ROS2 documents [24]. The workspace called “slam_ws” created on the ROS2 framework and the packages included in it are given in Figure 9. The functions of these packages are shown in Table 2.

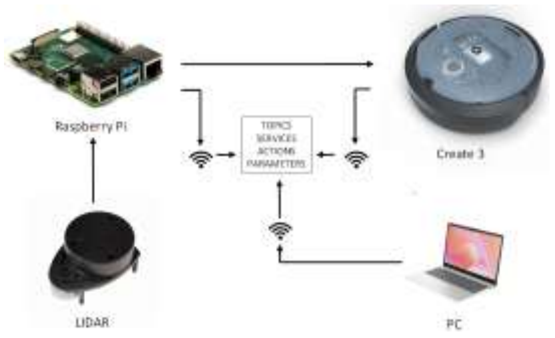


Figure 7. General Wiring Diagram

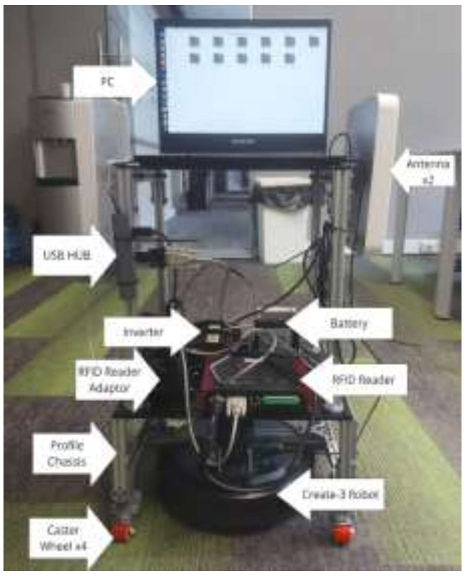


Figure 8. Robot and Hardware Components

Table 2. Created Packages and Their Functions

Package Name	Package Function
carto_slam	There are initialization files for Cartographer SLAM
gmapping_slam	There are initialization files for Gmapping SLAM.
slam_toolbox	There are initialization files for SLAM Toolbox.
lidar_drive	The lidar sensor's driver file is located.
create3_messages	Create-3 robot has message files.
maps	Map files are available.
manuel_drive_records	Manual driving speed information for the rooms is available.

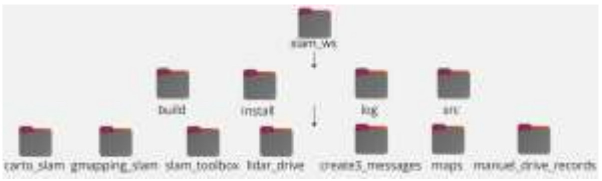


Figure 9. Workspace

Part work of the robot is carried out in 3 different rooms. Regardless of the general characteristics of these rooms, room 1 was chosen as a small place where obstacles are used extensively. In the 2nd room, a place chosen as the wall nearby, just from the corridor, is preferred. Finally, the third room, which is larger and has fewer products in terms of variety, is preferred. Images of these rooms are shown in Figure 10.

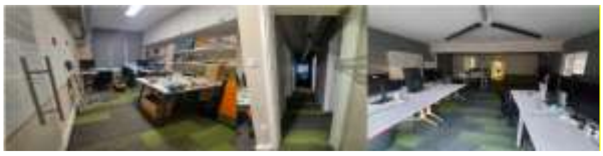


Figure 10. Rooms (Left room-1, Middle room-2, Right room-3)

While the robot is manually moved at the same speed profile, minimal changes in speed and environmental factors of the room (temperature, dust, etc.) are ignored. Here, the previously created and recorded speed data for each room is played for all 3 packages and the maps are produced under these conditions as output.

3.6. Experimental Study

The corrected reference map and package outputs for each room with the speed profile published via "cmd_vel", taken from the manual recording, are shown in the images below. The resolution of each map is kept constant at 300x300. Reference and package outputs for Room-1 are given in Figure 11. Reference and package outputs for Room-2 are given in Figure 12. Reference and package outputs for Room-3 are shown in Figure 13.

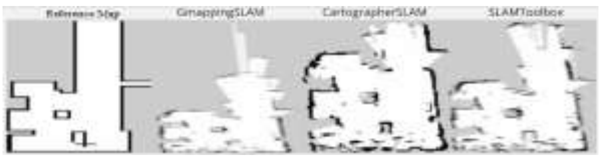


Figure 11. Room-1 Map Outputs

4. Results and Discussion

After the mapping process for each room, the similarity ratio was obtained by comparing it with the reference map for each room. The results regarding the similarity ratio are given in Table 3.

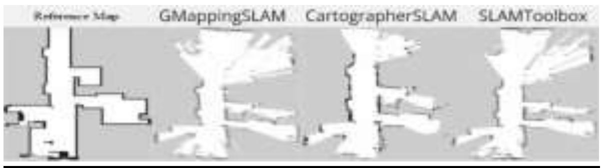


Figure 12. Room-2 Map Outputs

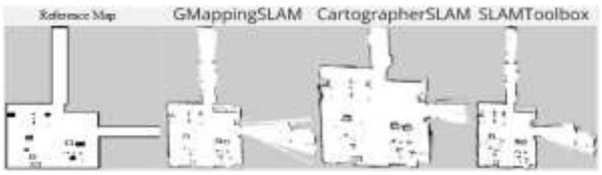


Figure 13. Room-3 Map Outputs

Table 3. SSIM Comparison Table

	GMappingSLAM	**Cartographer SLAM**	**SLAM Toolbox**
Room-1	0.63	0.53	0.66
Room-2	0.61	0.66	0.63
Room-3	0.71	0.66	0.72
Average	0.64	0.62	0.67

The results analyzed in the table show that while Cartographer SLAM is simpler and offers high performance, it can provide low accuracy values in large areas with many obstacles. It can be seen that the SSIM solutions Gmapping SLAM in Room-3 has a similarity value of 0.71. However, it is seen that the rate of developments in Room-2 decreased to 0.61. The results of our Cartographer SLAM package generally provide successful results in narrow areas such as corridors, but it has been observed that this performance decreases in wide areas. SLAM Toolbox, which is taken as the basis, performs well compared to other packages in large areas where obstacles are dense, offering a high SSIM value of 0.72. It is concluded that the SLAM Toolbox program can perform a better mapping, as the higher SSIM means we produce an image closer to the reference.

When the results are examined, the differences in SSIM values of the SLAM packages in different environments should be taken into consideration. The main factors that are important when mapping SLAM packages include the size of the environment, obstacle density and the use of odometry data of the package. If mapping speed is considered a metric, the Cartographer SLAM package can extract obstacles more smoothly and sharply during mapping. On the other hand, higher rates slow down the mapping speed. Gmapping SLAM package and SLAM Toolbox package are important factors in choosing both SSIM value and mapping speed over GmappingSLAM.

5. Conclusions

In this study, the performances of three distinct SLAM packages currently supported in the latest version of the Robot Operating System (ROS2) were comparatively evaluated under varying indoor conditions — including obstacle density and room size — using the SSIM metric and mapping speed as primary performance indicators. The experiments were conducted on the Create-3 differential-drive mobile robot, a platform specifically designed for ROS development, with particular attention paid to the reliance of each SLAM approach on odometry data. Based on the findings, the study provides a concrete analytical comparison of SLAM performance under real-world experimental conditions.

The results reveal that SLAM_Toolbox performs more favorably under the evaluated scenarios, particularly in terms of SSIM similarity, making it a strong candidate for applications requiring high-fidelity map generation. The use of the SSIM metric is especially appropriate given that SLAM outputs are 2D visual maps, and SSIM offers sharper, pixel-level similarity comparisons compared to alternative metrics. Furthermore, the existence of a readily available SSIM library in Python increases its practical utility for implementation.

One of the key contributions of this study is its comparison of SLAM algorithms under three distinct environmental configurations: a narrow room, a corridor, and a wide room. This highlights the strengths and weaknesses of each package in different real-world scenarios, providing valuable guidance for both researchers and practitioners. However, the study is not without limitations. The use of a specific robot platform, a limited test environment, and reliance on a single evaluation metric are potential constraints that should be addressed in future research.

Nomenclature

SLAM	**:** Simultaneous Localization and Mapping
ROS	**:** Robot Operating System
EKF	**:** Extended Kalman Filter
MCL	**:** Monte Carlo Localization
SSIM	**:** Structural Similarity Index Measure
ADNN	**:** Adaptive Directional Nearest Neighbor

6. References

1. Panigrahi, P. K., & Bisoy, S. K., “Localization Strategies for Autonomous Mobile Robots: A Review”, Journal of King Saud University-Computer and Information Sciences, Vol. 34, No. 8, pp. 6019-6039, 2022.
2. Leonard, J. J., & Durrant-Whyte, H. F., “Simultaneous Map Building and Localization for an Autonomous Mobile Robot”, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vol. 3, pp. 1442-1447, 1991.
3. Taheri, H., & Xia, Z. C., “SLAM; Definition and Evolution”, Engineering Applications of Artificial Intelligence, 97, 1–25, 2021.
4. Leonard, J. J., & Durrant-Whyte, H. F., “Mobile robot localization by tracking geometric beacons”, IEEE Transactions on Robotics and Automation, 7(3), 376–382, 1991.
5. Bettencourt, R., Serra, R., Lima, P. U., & Vale, A., “Comparison of Different LiDAR Sensors in SLAM: Criteria Formulation and Results”, 2025 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 1-6, April 2025.
6. Kim, H., Park, D., & Lee, S., “Benchmarking SLAM Toolboxes on ROS2 with Multisensor Fusion for Indoor Navigation”, Sensors, 24(9), Article ID 3981, 2024.
7. Qiu, Z., Wang, H., & Chen, Y., “Comparative Evaluation of ROS2-based SLAM Algorithms in Highly Dynamic Environments”, IEEE Transactions on Robotics, 41(3), 580–592, 2025.
8. Gonzalez, L., & Pereira, A., “Evaluating SLAM Performance on Resource-Constrained Robots Using ROS2”,

International Journal of Advanced Robotic Systems, 22(1), 45–59, 2025.

9. Mertyuz, İ., Yakut, O., & Taşar, B., “Mobil Robotlar için ROS Kullanılarak 2B SLAM Algoritmalarının Karşılaştırılması”, Trakya Üniversitesi Mühendislik Bilimleri Dergisi, Vol. 24, No. 2, pp. 29-38, 2023.

10. Li, F., Zhang, X., & Sun, J., “Performance Analysis of Real-Time SLAM Techniques for Autonomous Service Robots”, Robotics and Autonomous Systems, 172, Article ID 104546, 2024.

11. Kumar, N. S. P., Chandra, G. N., Sekhar, P., Sola, R., & KG, M., “Comparison of 2D & 3D LiDAR SLAM Algorithms Based on Performance Metrics”, International Conference on Innovation in Computing and Engineering (ICE), 1–6, 2025.

12. Bhushan, U. P., Innocent, N., & Laddi, A., “A Comparative Study of 2D SLAM Algorithms for ROS-Based Mapping”, International Conference on Signal Processing, Computing and Control (ISPCC), 900–906, 2025.

13. Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W., “A tutorial on graph-based SLAM”, IEEE Intelligent Transportation Systems Magazine, Vol. 2, No. 4, pp. 31-43, 2010.

14. Montemerlo, M., & Thrun, S., “FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics ”, Springer Science & Business Medi, 2007.

15. Le Roux, J., “An Introduction to Kalman Filter”, ASME Journal of Basic Engineering, Vol. 82, pp. 34–45, 1960.

16. Kalman, Rudolph Emil., “A new approach to linear filtering and prediction problems”, J. Fluids Eng, Vol. 82, No. 1, pp. 35-45, 1960.

17. Becker, Alex., “Kalman Filter from the Ground Up”, NET, 2023.

18. Khodarahmi, Masoud, and Vafa Maihami., “A Review on Kalman Filter Models.”, Archives of Computational Methods in Engineering, Vol. 30, No. 1, pp. 727-747, 2023.

19. Fox, Dieter, et al., “Particle filters for mobile robot localization. Sequential Monte Carlo methods in practice”, Springer, 2001.

20. Sankalprajan, P., Sharma, T., Perur, H. D., & Pagala, P. S., “Comparative Analysis of ROS Based 2D and 3D SLAM Algorithms for Autonomous Ground Vehicles”, 2020 International Conference for Emerging Technology (INCET), IEEE, pp. 1-6, 2020.

21. Nilsson, Jim, and Tomas Akenine-Möller., "Understanding ssim", Cornell University, arXiv preprint, Vol. 2006.13846, pp. 1-8, 2020.

22. Yeong, D. J., Velasco-Hernandez, G., Barry, J., & Walsh, J., “Sensor and sensor fusion technology in autonomous vehicles: A review”, Sensors, Vol. 21, No. 6, 2021.

23. Alatise, M. B., & Hancke, G. P., “A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods”, IEEE Access, Vol. 8, 2020.

24. https://docs.ros.org/en/humble/index.html, 15/9/2024.