Towards Development of Performance Metrics for Benchmarking SLAM Algorithms

Mudit Bhargava, Rushad Mehta, Chandan Das Adhikari, K Sivanathan

Autonomous Systems Lab Department of Mechatronics Engineering SRM Institute of Science and Technology, Chennai, India
Email: *sivanatk@srmist.edu.in

Abstract. The true autonomy of mobile robots cannot be achieved without Simultaneous Localization and Mapping (SLAM). With this capability, mobile robot could concurrently build a map of the environment and locate itself with respect to the map. Although there are several variants of SLAM algorithms contributed by researchers so far, only a very few works were aimed at comparing their performances with appropriate metrics and providing detailed directions and insights to the user on selection criteria and indicative use cases. In this work, we presented a comparative study of three popular SLAM algorithms and provide some significant quantitative performance measures of the same by using our novel | R | and | S | performance metrics as well as conventional metrics. The comparative study was carried out in ROS (Robot Operating System) using Turtlebot3 robot model on three SLAM packages viz G-mapping, Karto SLAM, and Frontier Exploration SLAM. Furthermore, the results show that the proposed metrics are very efficient and compact in comparing and quantifying the performance of SLAM algorithms.

Keywords: Performance metrics; Benchmarking SLAM; ROS; G-mapping; Karto; Frontier

1. Introduction
Mobile Robotics is ordinarily considered to be a subfield of Information and Robotics Engineering. Autonomous Mobile Robots [1] which is coined as AMR are competent in exploring an uncontrolled and unknown environment without the need of direction gadgets [26]. Mobile robots have gotten to be more commonplace in industrial, military, commercial, and security settings [2]. Moreover, mobile robots have found their way into regular household applications with the innovation of mobile vacuum cleansers [3], which can operate nearly with complete autonomy. Mobile robots are moreover a major center of current research and nearly every major college has one or more Lab that focuses on mobile robot research. Despite mobile robotics being at a moderately early stage of advancement, prototype models and a few serial products are beginning to be found all through a wide run of segments. For a long time, advancements in robotics were essentially centered on the parameters of speed, performance, accuracy, and repetition [4]. The basic prerequisites of present-day robots in all divisions, especially fabrication, is that they must show adaptability, versatility, and, above all, autonomy. This autonomy is directly dependent on navigation. Navigation in mobile robots has been a major issue over the years and was put to extensive research [5][12]. All of these as a whole were answered by an algorithm called Simultaneous Localization and Mapping (SLAM). These questions when asked individually are quite easy to answer, but when asked altogether pose a great hole to dwell into.
SLAM answers all of these questions at once serving the need for navigation quite effectively [22]. For that, we need to understand the terms “Localization” and “Mapping” individually. Broadly speaking, localization [11] is a method by which we can predict the pose of or mobile robots in the environment in which it resides. Mapping refers to the process of computing the position or the location of our mobile robot with respect to its environment in terms of gathering odometry data when the pose of the robot is known beforehand. Now since we have no clue of either the environment or the robot's pose, we have to estimate both from the data simultaneously. This is where the SLAM algorithm comes into the picture.

In this project report, we will be discussing three different SLAM methods and compare them with the help of some performance metrics that will help us identify some significant differences and compare them accordingly.

Simulations are being performed on Linux and Robot Operating System (ROS) platform using the R-Viz and Gazebo. ROS is a special framework that was initially developed by Stanford Artificial Intelligence Laboratory back in the year 2007 for the development of robots widely. Though ROS is not an operating system but sure does provide all the facilities that an operating system should. It is accepted and adopted widely and has a wide network for open-source developer’s community which strives to make it better each rolling day. Starting from single board platforms to more complex systems, it can be easily mounted and interactively used. The version of ROS that has been primarily used is Kinetic. Since ROS-Melodic is a newly developed version, all the simulations are done on it. Although Kinetic is a more stable version than Melodic, it is picking up pace. The support for Melodic queries is immense these days which led us to use Melodic for this work.

2. Problem Statement

Consider a mobile robot present in a static and unknown environment and does it have access to its own pose. The robot executes control and collects the perception of highlights within the world. Both the controls and the perceptions are adulterated by noise, which increases the difficulty level. SLAM is the method of re-coupling an outline of the environment and the robots’ set of noisy controls and observations [17]. SLAM can be also termed as Concurrent Mapping and localization (CML) [18]. For attaining true autonomy in a robot, SLAM is a fundamental problem. The majority of SLAM approaches are introduced to now use probabilistic approaches for solving two types of the SLAM problem. Firstly, is the Online SLAM problem and second one is the Full SLAM problem [18][23]. The Online SLAM [23] problem is based on including the posterior over the transient pose alongside the map. The estimation of the variables at time $t$: 

$$p (x_t, m \parallel z_{1:t}, u_{1:t}) \quad (1.1)$$

In the above equation 1.1, $p$ denotes the probability, $m$ is the map, $x_t$ is posterior at the time $t$, and $u_{1:t}$ & $z_{1:t}$ is the controls and measurements respectively. In the Full SLAM problem, rather than computing the current posture $x$ shown in equation 1.2, to calculate a posterior for $x_{1:t}$ path:

$$p (x_{1:t}, m \parallel z_{1:t}, u_{1:t}) = \int \cdots \int p (x_{1:t}, m \parallel z_{1:t}, u_{1:t}) \, dx_1 \cdots dx_2 \cdots dx_t \quad (1.2)$$

Dependent variables are recursively integrated individually. The subtle distinction within the definition of both the major problems has repercussions within the type of algorithms that can be brought to bear [16]. A second key issue is concerned with estimation; the continuous estimation issue relates to all the robot’s pose variables and the objects within the map [10]. Objects represent the landmarks that are re-observed and distinguished with the other anomalies in explored environment using the feature-based representation [24]. They could also be identified by range finders. The extracted landmarks from the laser scan are the information associated which further proceeds for calculating the odometry changes. SLAM framework must collect information from the robot’s own sensors and any other sources to create what sums to an internal GPS for its own or other sensors like lasers, cameras, auditory sensors, or a sort of pre-installed or wire-guided framework [14]. In most cases, SLAM collects data in the absence of light. Mapping empowers the mobile robot to use sensor input to create a virtual environment for analyzing the landmarks. Various SLAM methods introduced
for the Turtlebot 3 includes G-Mapping, Karto, Hector, Google Cartographer, and Frontier Exploration.

A. **G-Mapping:**

G-Mapping is based on the Rao-Blackwellized Particle Filter [7][8][11]. This is to learn grid maps from laser run data. Unlike, the Karto SLAM method, it utilizes Particle Filter (PF) which uses model-based estimation [11]. As in SLAM, we are trying to achieve two things: the map and also the robot’s position as well as its orientation. Each and every particle present contribute to finding the solution on approximating the genuine probability disposition of the attained environment’s map and the mobile robot’s pose given the sensor readings (for example - LiDAR) and control inputs (for example – motor encoder counts) [6]. Particle Filter is more flexible because it does not accept linearity and Gaussian nature of noise in data, but is more computationally expensive. Particle Filter represents the distribution by weighting and creating random samples instead of mean and covariance matrix as in Gaussian distribution. Rao – Blackwellized particle filter [7] uses a particle filtering method to drastically reduce the vulnerability of the robot’s position and orientation in the prediction step of filtering [8]. Being the most used SLAM method, this approach of re-sampling operation truly decreases the problem of particle depletion.

B. **Karto:**

Karto SLAM package was developed in 2010 by an organization of Stanford University, which was categorized as the highest performance and full-featured SLAM. Karto SLAM is known to provide high-accuracy navigation, mapping, and exploration functionality for a huge range of mobile robots. Karto SLAM includes GPS integration, because of which the localization problem [11] can be simplified by obtaining the GPS, coordinates, and user-stored data. The SLAM Karto package essentially uses open karto to make and keep up the pose-graph and SBA (Sparse Bundle Adjustment) package. The SBA package helps in providing a vertex and a constraint to the pose-graph. As the slam_karto.cpp bundle record handles approaching laser scans utilizing a laser callback function (internally used for calling add Scan). The add Scan begins with acquiring the odometry pose, to look for any change from Odom frame to base_link by default. In case no odometry pose transform is found, the add Scan returns false and if any change in transformation is found, then add Scan continues and converts the ROS laser scan function to a karto localized range scan and places it at the udometric pose. Then, the range scan is processed by the Karto Mapper, and the adjusted pose of the recently added vertex is recovered, these corrections more often happen when the loop closure is detected.

C. **Frontier Exploration:**

The idea of Autonomous exploration was proposed in 1997, for exploring an unknown environment using an autonomous mobile robot while forming the map of the environment. Autonomous exploration is considered to be the backbone as well as the fundamental approach for solving the SLAM problem. Frontiers can be classified as the region present between the explored and unexplored space. While moving, the robot explores various frontiers and the continuous process leads to the development of the mapped territory by reducing the unknown region. By eliminating more and more frontiers till there is are none left? This is used to decrease the time complexity of the whole algorithm which is based on Breadth-First Search. Once frontiers have been recognized inside a particular evidence grid, the robot endeavors to explore to the nearest accessible, unexplored frontier [25]. The path planner employs a Breadth-First Search or BFS [13] on the grid, starting from the current cell of the robot and attempting to acquire the shortest obstacle-free path to the goal cell location [25]. The robot performs an all-round area sweep of 360-degree using laser-limited sonar [25] and updates the evidence grid with all the new information [19], and further attempts to navigate to unvisited frontiers.
3. Implementation

A. Hardware Specifications:

To implement and verify the simulations of different SLAM algorithms, the Robot Operating System (ROS) platform was an ideal choice. To make use of this platform, the system with the following configuration had to be dual booted:

Table 1: Hardware Specifications of the PC

| Configuration | Specifications |
|---------------|---------------|
| Operating System | Linux Platform [Ubuntu 18.04.4 LTS] |
| CPU           | Intel i7 [7th Generation] |
| RAM           | 8GB |
| GPU           | Nvidia GTX 1050 Ti [4GB] |

B. Software Installation and Building:

The ROS Melodic installation started after the setup of Ubuntu with all the required drivers. The ROS Melodic installation guide and documentation was followed. On its completion, the necessary ROS software packages for TurtleBot3, Gazebo, and R-Viz were installed. Gazebo is a 3D Robotics simulation software that was used to create different environments to run our algorithms on. R-Viz or the ROS-Visualization is a three-dimensional visualization tool for ROS. Through the R-Viz the user can view the robot simulation, robot’s sensor log data, and replay the robots logged sensor information. The TurtleBot package enables the user to configure and interact with the simulated robot. In this project the TurtleBot3 model = "Burger" was used.

Now that all the software and ROS packages were collected and installed, a file called "catkin_ws" was created on the home directory. Now, this was the base directory from which all the operations took place. To able to work from this directory, "catkin_make" had to be used to build the directory, so that ROS would know it was going to be called from there. From this directory, the files of Karto SLAM and Frontier Exploration had to be built separately.

C. Testing and Experimentation:

Testing of the different algorithms were started once the build process described in the subsection above was complete. For the initial tests, the three different SLAM algorithms i.e., G-Mapping SLAM, Karto SLAM, and Frontier Exploration were implemented on the default inbuilt virtual environment provided by Gazebo. Once the above task was successfully completed, the Autonomous Systems Lab (ASL), SRMIST was recreated as a virtual environment using Gazebo. Now, this Gazebo virtual environment launch file would be used for all our future analysis and comparison. Using the methods described above, each SLAM algorithm was implemented in the ASL’s virtual environment. The CPU load and run time of the three different algorithms were recorded as a measure of performance.

D. Quantification of Metrics:

To concretely compare the different algorithms, there was a need to convert the recorded data to suitable metrics [20]. Firstly, the derivatives of the recorded data were calculated. Since the derivative would have units: CPU load per unit time, a derivative close to zero would suggest optimal performance. Since there is no predefined map, there is no way by which the absolute efficiency of these algorithms could be calculated. Hence, to get a relative measure of these three algorithms’ efficiency, the following metric is used:

\[ \%\text{Efficiency} = 100 \times \left( \frac{d\text{CPU}}{dt} - 1 \right) \]  \hspace{1cm} (3.1)

In the above equation 3.1, \( \frac{d\text{CPU}}{dt} \) is the time derivative of the CPU usage with respect to time. Since this way of calculating the efficiency gives information about CPU load per usage run time, and not the accuracy [21], there exists the need to introduce two other metrics to calculate the relative
similarity and error between the generated maps. To calculate the similarity between the generated maps, the Structural Similarity Index (SSIM) was used [12]. SSIM is calculated as follows:

$$\text{SSIM}(x, y) = \frac{\left(2\mu_x\mu_y + C_1\right) \left(2\sigma_{xy} + C_2\right)}{\left(\mu_x^2 + \mu_y^2 + C_1\right) \left(\sigma_x^2 + \sigma_y^2 + C_2\right)}$$ (3.2)

In equation 3.2, $\mu_x$ is the mean of image data $x$, $\mu_y$ is the mean of image data $y$, $\sigma_x^2$ is the variance of image data $x$, $\sigma_y^2$ is the variance of image data $y$, $\sigma_{xy}$ is the covariance of image data $x$ and $y$, and $C_1 = (k_1L)^2$, $C_2 = (k_2L)^2$ (3.3)

In equation 3.3, $L$ is the dynamic range of the image data [27] which is given by $(2^{\text{no. of bits per pixel}} - 1)$; $k_1$ and $k_2$ are constants that have the values 0.01 and 0.03 respectively by default.

The output of SSIM ranges from -1 to 1; where 1 represents there is a perfect similarity between the two images being compared, and -1 represents that there is no similarity at all. Thus, to get a relative measure of accuracy, the images of the generated were compared using SSIM [1].

To calculate the error, the Root Mean Squared Error (RMSE) [1][5][6] was used. RMSE is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - y_i}{d_i}\right)^2}$$ (3.4)

Equation 3.4 provides an understanding of the error amongst the three maps. A low value of RMSE indicates a low error while a higher value of RMSE indicates a high error and more instability in the algorithm.

4. Results and Discussions

The experiments and tests were carried out on the Replica of the Autonomous System Lab (ASL) map created in the Gazebo. And the CPU usage data and generated map images in R-Viz were recorded and saved. Figures 1, Figures 2, and Figures 3 compare the images of the maps generated in the R-Viz.

Figure 1: Map generated by G-Mapping SLAM

Figure 2: Map generated by Karto SLAM

Figure 3: Map generated by Frontier Exploration SLAM
Figure 4: shows the variation of CPU usage of three methods with respect to time.

CPU data was obtained and stored in the form of Comma Separated Values (CSV) file for each SLAM Method and the combined plot is shown in Figure 4 [15]. Final Computational Time was recorded for each SLAM method performed on the ASL map and further, calculating the derivative of the CPU usage with respect to time and also, the relative efficiency for each of the following three methods [5], the results obtained are shown in Table 2. Since the derivative of G-Mapping SLAM is the smallest, it is clear that its relative efficiency is the best, and turns out to be 90.61% [9]. The derivate for Karto SLAM is the largest, making it the least efficient algorithm, with an efficiency of 73.01%. Frontier Exploration’s derivative sits between both these, having an efficiency of 86.6%. Thus, it is deduced that G-Mapping SLAM is the most efficient in terms of CPU load and run time.

Table 2: Comparison of Various SLAM methods is showcased based on Computation Time, derivatives, and Relative Percentage Efficiency.

| METRIC PARAMETERS | SLAM Methods           |
|-------------------|------------------------|
| Computation Time  | G-Mapping 177 (Fast)   |
|                   | Karto 260+ (Slow)      |
|                   | Frontier Exploration 257 (Moderate) |
| Derivative        | 0.0939                 |
|                   | 0.2699                 |
|                   | 0.1340                 |
| Relative Percentage Efficiency | 90.61% | 73.01% | 86.6% |

Since this metric does not provide any insight into the accuracy of the algorithm, different metrics are calculated as in the previous section. Therefore, for similarity accuracy, SSIM was calculated amongst the three images of the generated maps, and for calculating the error, the RMSE was calculated. These results are shown in Table 3 and Table 4.

Table 3: Metric comparison of Various SLAM methods was generated while calculating the Similarity Accuracy using the SSIM method.

| SLAM Methods | G-Mapping | Karto | Frontier Exploration |
|--------------|-----------|-------|----------------------|
| G-Mapping    | 1         | 0.75  | 0.76                 |
| Karto        | 0.75      | 1     | 0.76                 |
| Frontier Exploration | 0.76     | 0.76  | 1                    |
From Table 3, ignoring the similarity indices of the images with themself, it is evident that the Karto SLAM - Frontier Exploration SLAM, and G-Mapping SLAM - Frontier Exploration SLAM pairs have the highest similarity index scores of 0.76 each. The Karto - G-Mapping pair has a slightly lower similarity index of 0.75. From this, it can be deduced that Frontier Exploration SLAM has the highest similarity index amongst all three, and thus recreates the map with the highest accuracy.

Table 4: Metric comparison of Various SLAM methods were generated while calculating the Root Mean Square Error

| SLAM Methods | G-Mapping | Karto | Frontier Exploration |
|--------------|-----------|-------|----------------------|
| G-Mapping    | 0         | 75.17 | 70.08                |
| Karto        | 75.17     | 0     | 77.4                 |
| Frontier Exploration | 70.08 | 77.4 | 0                     |

From Table 4, ignoring the error of the images with themself, it is evident that the G-Mapping SLAM - Frontier Exploration SLAM pair has the least RMSE error of 70.08. The Frontier Exploration - Karto pair has the highest RMSE error of 77.4. The Karto - G-Mapping pair has an error of 75.17 which sits between both these values.

Comparing two parameters for three algorithms is a tedious process and does not provide a holistic picture. To avoid this, consider Table 1 and Table 2 described above as Matrices defined as:

\[
S = \begin{bmatrix}
1 & S_{GK} & S_{GF} \\
S_{GK} & 1 & S_{KF} \\
S_{GF} & S_{KF} & 1
\end{bmatrix}
\]

In equation 4.1, \(S_{GK}, S_{GF}, \) and \(S_{KF}\) are the SSIM indices between G-Mapping - Karto, G-Mapping - Frontier Exploration, and Karto - Frontier Exploration respectively.

\[
R = \begin{bmatrix}
0 & R_{GK} & R_{GF} \\
R_{GK} & 0 & R_{KF} \\
R_{GF} & R_{KF} & 0
\end{bmatrix}
\]

In equation 4.2, \(R_{GK}, R_{GF}, \) and \(R_{KF}\) are the RMSE between G-Mapping - Karto, G-Mapping - Frontier Exploration, and Karto - Frontier Exploration respectively.

For ideal conditions, \(S_{GK} = S_{GF} = S_{KF} = 1\) and \(R_{GK} = R_{GF} = R_{KF} = C,\) where \(C\) is a Constant.

Ideally, \(|S| = 0\) and \(|R| = 2C^3\).

On Expanding \(|S|\) we get:

\[
|S| = (1 - S_{KF}^2) - S_{GK}^2(S_{GK} - S_{KF}) + S_{GF}^2(S_{GK} - S_{GK}) - S_{GF}^2 = 0
\]

\[
\Rightarrow 1 - S_{KF}^2 - S_{GK}^2 + 2S_{GF}^2 - 2S_{GK}S_{KF} = 0
\]

\[
\Rightarrow 1 - S_{KF}^2 - S_{GK}^2 + 2S_{GF}^2 = 0 \quad (4.3)
\]

Similarly, expanding \(|R|\) we get:

\[
|R| = 0 - R_{GK}^2(0 - R_{GF}^2R_{KF}) + R_{GF}^2(R_{GK}^2R_{KF} - 0)
\]

\[
\Rightarrow (R_{GK}^2R_{GF}^2R_{KF}) + (R_{GF}^2R_{GK}^2R_{KF}) = 2C^3
\]

\[
\Rightarrow (R_{GK}^2R_{GF}^2R_{KF}) - C^3 = 0 \quad (4.4)
\]

Substituting in the above equation 4.3 and 4.4, we get:

\[
1 - S_{KF}^2 - S_{GK}^2 + 2S_{GF}^2 = 0 \quad (4.5)
\]

The above equation describes the process in ideal conditions. Since ideally, \(|S| = 0\) and \(|R| = 2C^3\) the determinant for the two Matrices \(S\) and \(R\) were calculated and shown in Table 5. A low non-negative determinant of matrix \(S\) indicates high accuracy, and a non-negative determinant of matrix \(R\), which is close to the value of 2C^3, indicates a low error. For calculation purposes, the value of \(C\)
(constant) was selected as the average of all the RMSE values.

Table 5: Shows the determinant values of Matrices S and R, as a measure of the overall average accuracy.

| Metric | Value | Difference % |
|--------|-------|--------------|
| | Ideal | Calculated | |
| | | 0 | 0.1487 | - 0.1487 |
| | 2C^3 ⇒ 2*(74.21)^3 | 815473.0252 | 0.2317 |

It should be noted that all three algorithms compared in this paper use similar mapping techniques. If in case the algorithms being compared use different mapping techniques, the values of \( R_{GK} \), \( R_{GF} \), \( R_{KF} \) will be lower than expected, and the values of \( S_{GK} \), \( S_{GF} \), and \( S_{KF} \) will be higher than expected. Thus, a high \(| S |\) score (>0.7) suggests high similarity amongst maps, whereas a mediocre \(| S |\) score (~0.6) suggests that the maps were generated using different mapping techniques. A low \(| S |\) score (<0.6) suggests that the maps have high variance and do not represent the given environment well enough.

5. Conclusions
The simulations were carried out on ASL’s virtual environment using G-Mapping, Karto, and Frontier Exploration SLAM methods. The data of the TurtleBot3 model = “Burger” robot was logged and analyzed. Based on the analysis, we derived a few performance metrics as discussed above. Using these metrics, we deduced that the G-Mapping SLAM algorithm is the fastest and hence computationally least expensive. It has the highest computational efficiency amongst all the algorithms. Based on SSIM and RMSE metrics, we can also conclude that even though G-Mapping is the most computationally efficient algorithm, Frontier Exploration SLAM has better accuracy compared to the other two algorithms. The determinant of \(| R |\) and \(| S |\) metrics provide us with valuable information about the overall accuracy of the three algorithms combined. Since these metrics are well within the permissible limits of allowable error, we can conclude, that all the maps generated using different SLAM algorithms are usable at industrial standards, each suiting a specific task. For example, if low computational power is a constraint, but good speed is a requirement; G-Mapping is the SLAM method that is best suited for the task. Similarly, if there is more computational power to spare, and map accuracy is the primary concern of the project, Frontier Exploration is better suited.

6. Scope and Future Work
The objective of this paper is to create a novel method to quantify some performance metrics and parameters that facilitate us to make concrete comparisons using different SLAM methods presented using a mobile robot to map all the unknown regions present in an unknown indoor environment. One can choose a specific SLAM method based on the performance metrics provided for the autonomous mobile robots they are trying to use for a specialized task. The most necessary top performing metrics change with the task the robot is trying to accomplish. Thus, we discuss several metrics such as computation time, efficiency, accuracy, SSIM, RMSE, \(| S |\), and \(| R |\) for each of the SLAM methods and try to provide insight into which methods are best suited for which applications.

The comparative performance metric study developed during this work will benefit in applying for the various maps with Multi-Robot System Environment which is also known as MR-SLAM (Multi-Robot SLAM). With the determinant of \(| R |\) and \(| S |\) metrics more comparative studies can be produced with different other SLAM methods available. Emerging autonomous mobile robots because SLAM will be an alternative of the User-built maps, as there is a constant need for research needed for achieving more robust navigation and perception with respect to the task-driven framework for autonomous robots [22]. Implementation and comparison of the simulated results in the real-time mobile robot will be carried out as the extension of this work.
References

[1] Filatov, A. Filatov, K. Krinkin, B. Chen and D. Molodan, “2D SLAM quality evaluation methods”, (2017) 21st Conference of Open Innovations Association (FRUCT), Helsinki, 2017, pp. 120-126.

[2] Caesar, F., et al. (2019), “A Novel SLAM Quality Evaluation Method”.

[3] Wang et al., “Autonomous mobile robot navigation in uneven and unstructured indoor environments”, International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, (2017), pp. 109-116.

[4] Chen L., Tang W., John N. W., Wan, T. R., & Zhang J.J., “SLAM-based dense surface reconstruction in monocular Minimally Invasive Surgery and its application to Augmented Reality”, Computer Methods and Programs in Biomedicine (2018), 158, 135–146.

[5] Chen, Yuwei, et al. “The Accuracy Comparison of Three Simultaneous Localization and Mapping (SLAM)-Based Indoor Mapping Technologies”, Sensors (Basel, Switzerland) vol. 18,10 3228. (2018).

[6] Filipenko, Maksim & Afanasyev, Ilya, “Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment”, (2018).

[7] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard (2005), “Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling”, In Proc. of the IEEE International Conference on Robotics and Automation (ICRA).

[8] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard (2007), “Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters”, IEEE Transactions on Robotics, Volume 23, pages 34-46.

[9] Gomez, Clara; Hernandez, Alejandra C.; Barber, Ramon. (2019), “Topological Frontier-Based Exploration and Map-Building Using Semantic Information”, Sensors 19, no. 20: 4595.

[10] J. Leonard and H. Durrant-Whyte, “Simultaneous map building and localization for an autonomous mobile robot”, Proceedings IROS’91: IEEE/RSJ International Workshop on Intelligent Robots and Systems’91, no. 91, pp. 1442–1447, (1991).

[11] Leonard, John J., et al. “Dynamic Map Building for an Autonomous Mobile Robot”, The International Journal of Robotics Research, vol. 11, no. 4, (1992), pp. 286–298.

[12] Le, Xuan & Fabresse, Luc & Bouraqa, Noury & Lozenguez, Guillaume. (2018), “Evaluation of Out-of-the-Box ROS 2D SLAMs for Autonomous Exploration of Unknown Indoor Environments”, 11th International Conference, ICIRA 2018, Newcastle, NSW, Australia, August 9–11, 2018, Proceedings, Part II.

[13] M. Bosse, P.M. Newman, J.J. Leonard, and S. Teller, “An ALTAS framework for scalable mapping”, In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), pages 1899–1906, Taipei, Taiwan, (2003).

[14] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csurba, “A solution to the simultaneous localization and map building (SLAM) problem”, in IEEE Transactions on Robotics and Automation, vol. 17, no. 3, pp. 229-241, (2001).

[15] Pålsson, Albin, and Markus Smedberg. “Investigating Simultaneous Localization and Mapping for AGV systems”, (2017).

[16] S. Se, D. Lowe, and J. Little, “Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks”, The International Journal of Robotics Research, vol. 21, no. 8, pp. 735–758, (2002).

[17] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (slam): Part ii”, IEEE Robotics & Automation Magazine, vol. 13, no. 3, pp. 108–117, (2006).

[18] Thrun, S., Burgard, W. and Fox, D. (2006), “Probabilistic Robotics”, Kybernetes, Vol. 35 No. 7/8, pp.

[19] Topiwala, Anirudh & Inani, Pranav & Kathpal, Abhishek, (2018), “Frontier Based Exploration for Autonomous Robot”.
[20] W. Burgard et al., “A comparison of SLAM algorithms based on a graph of relations”, (2009) IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, 2009, pp. 2089-2095.

[21] Yagfarov, Rauf & Ivanou, Mikhail & Afanasyev, Ilya. (2018), “Map Comparison of Lidar-based 2D SLAM Algorithms Using Precise Ground Truth”.

[22] Yasir Latif, Cesar Cadena, Luca Carlone, Henry Carrillo, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard., “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age”, Trans. Rob. 32, 6 (December 2016), 1309–1332.

[23] Schlegel, Christian & Hochdorfer, Siegfried. (2008), “Localization and Mapping for Service Robots: Bearing-Only SLAM with an Omnicam”, 10.5772/5952.

[24] Zhang, Wenjun & Zhang, Qiao & Sun, Kai & Guo, Sheng (2016), “A LASER-SLAM ALGORITHM FOR INDOOR MOBILE MAPPING”, ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLI-B4. 351-355. 10.5194/isprs-archives-XLI-B4-351-2016.

[25] Rajesh M., Gana Rani Jose, Sudarshan T.S.B.. "Multi robot exploration and mapping using frontier cell concept", 2014 Annual IEEE India Conference (INDICON), 2014

[26] Hampton, Brian & Al-Hourani, Akram & Risti, Branko & Moran, Bill (2017), “RFS-SLAM robot: An experimental platform for RFS based occupancy-grid SLAM”, 10.23919/ICIF.2017.8009744.

[27] Yongxing Jia, Chuanzhen Rong, Ying Zhu, Yu Yang, Yuan Wang. "Multi-focus Image Fusion Scheme Using Adaptive Dual-Channel Pulse Coupled Neural Network", 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2016