Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 5 (2023)

ROS based SLAM for a Gazebo Simulated Fetch Robot in Unknown Indoor Environment

[1]Smt. Thilagavathy R, [2]Dr. Sumithra Devi K A

[1] Department of ECE, DSATM, Bengaluru, India
[2]Department of Data Science, DSATM, Bengaluru, India

Email: [1]thilagavathyphd2016@gmail.com, [2]sumithraka@gmail.com

Abstract— Mapping and Navigation are the basic need in many autonomous robot applications like Search and Rescue, Mining, surveillance and Ware house management etc., The ability of the robot to make simultaneously map of its surroundings and localize itself in relation to that environment is the most important element of mobile robots. To solve Simultaneous Localization and Mapping (SLAM) many algorithms could be utilized to build up the SLAM process and SLAM is a developing area in Robotics research. This paper describes Gazebo simulation approach to SLAM based on Robot Operating System (ROS) using Fetch robot. The mapping process is done by making use of the open-source Gmapping method. The Gmapping method applies Rao-Blackwellized particle filters and laser scanner data to find the Fetch robot in unknown Indoor environment and build a map. Data visualization is done using Rviz and a robot simulation is done in Gazebo. From our research, it is shown that for indoor environments Gmapping algorithm is employed to generate 2D maps with SICK Laser scanner and RGBD sensor placed on mobile robot. The algorithm has been evaluated and maps are constructed by the Fetch robot in static unknown indoor environment.

Index Terms— SLAM, ROS, Gazebo, Rviz, Fetch Robot, Gmapping, Rao-Blackwellized Particle Filter.

1. Introduction

Robots are used in various industries like manufacturing, medicine, personal service, space exploration, search-and-rescue and warehouse logistics. Mobile robots are capable of moving around in their environment and carrying out intelligent activities autonomously. A robot cannot explore an unknown environment unless it is provided with some sensing sources to get information about the environment. There are different kinds of sensors used to make a robot capable of sensing a wide range of environments: Odometers, Laser range finders, Global Position System (GPS), Inertial Measurement Units (IMU), Sound Navigation and Ranging (Sonar) and cameras [1]. The map of environment is a basic need for a robot to perform actions like moving room to room, picking an object from one place and taking it to another one. To perform such actions, the robot should know about the environment and while it is moving it should also be aware of its own location in that environment. Simultaneous Localization and Mapping (SLAM) is one of the most widely used mapping techniques. The Robotic Operating System (ROS) platform offers the mapping technique known as SLAM. It can create or modify the map while simultaneously tracking the position of the mobile robot [2]. SLAM will reduce faults in terms of mapping accuracy by considering the location of the robot. As a result, numerous SLAM algorithms are created and employed in other studies using mobile robots [3]. Under SLAM, a number of algorithms are available, for example, GMapping, Google cartographer, Core SLAM, Graph SLAM, Karto SLAM and Hector SLAM. The robot simulated is Fetch mobile manipulator and in absence of a real robot, the simulator provided by ROS is used. The aim to achieve robot navigation in a new and unknown environment using the ROS. As it will be presented, the robot builds the map, localizes itself on it and performs navigation.

Based on the fusion of the 2D SICK Laser scanner and the RGBD camera sensor, this paper focuses on the fetch robot's navigation system, which uses the Adaptive Monte Carlo Localization (AMCL) ROS package and the Gmapping SLAM system [4]. The map created is visualized using Rviz tool in ROS.

The rest of the paper is organized as follows: Section II describes the related works; Section III covers the system requirements; Section IV gives a description of Gmapping algorithm with an overview of map generation using ProcessScan; Section V starts with the implementation of Gmapping algorithm; Section VI provides and analyzes the experimental simulation results and the simulation of fetch Robot on ROS; Section VII describes the conclusion and future work of this research.

2. Related Works

Early SLAM techniques primarily focused on the filter-based techniques, such as Extended Kalman Filter (EKF) [5], Particle filter (PF) [6] and so on. Christopher Weyers and Gilbert Peterson presents a SLAM solution combining stereo cameras, inertial measurement units and vehicle odometry into a Multiple Integrated Navigation Sensor (MINS) path based on the Rao-Blackwellized Particle Filter (RBPF), which is operated by the FastSLAM algorithm and the Kalman Filter (KF) [7]. Zhang, Ou et al. uses a depth camera, move base with odometry and gyro to perform a SLAM task for a restaurant service robot using the RBPF approach [8]. It is well-known that the development of SLAM technology is from the filter-based method to the optimization-based method. The optimization-based approach is more effective and consistent, specifically for the pose-graph SLAM framework [9]. This is widely used in the indoor navigation of the multirobot system. Lee and Park present a mobile mapping system (MMS) whose back-end uses graph-based optimization and front-end uses map-based scan matching [10].

It is simple to understand that a platform cannot receive an accurate and reliable solution from a single sensor. Therefore, one of the frequently adopted choice is the fusion of the laser and camera. Su, Zhou et al. proposes a reliable global localization technique that can handle the abducted robot problem that uses both laser and camera sensors based on the pose graph optimization [11]. Adrian Ratter and Claude Sammut deliver a unique technique for combining Laser Rangefinder and RGB-D data to precisely monitor the exact location of a robot and produce substantial 3D maps, which is based on the ICP and pose-graph optimisation [12].

To construct the map, Balasuriya et al [13] has applied the Gmapping-based SLAM technique. The robot loaded with a laser scanner with additional sonar sensor as a backup. The grid-based map will be generated using the slam_gmapping package in ROS. Based on the results, the Gmapping algorithm is capable of dealing with both static and dynamic landmarks. As the map is updated, it retains the static landmarks while removing the dynamic landmarks. Other researchers use Gmapping in their studies, making it one of the most prominent options for SLAM mapping algorithms [14-16].

3. System Requirements

A. Robotic Operating Systems (ROS)

Robotic Operating System is a free and open-source and one of the most commonly used middle wares for robotics programming. ROS comes with message passing interface, tools, package management, hardware abstraction etc., It provides different libraries, packages and several integration tools for the robot applications. ROS is a message passing interface that allows inter-process communication hence it is usually known as middleware. ROS provides several features which helps researchers to develop robot applications. In this research work, ROS is considered as the main base because it publishes messages in the form of topics in across different nodes and has a distributed parameter system. ROS also provides Inter-platform operability, Modularity, Concurrent resource handling. ROS simplifies the whole process of a system by ensuring that the threads are not actually trying to read and write to shared resources, but are rather just publishing and subscribing to messages. Instead of building the entire system in hardware, ROS allows us to establish a virtual environment, develop robot models, apply algorithms, and visualize them in the virtual world. Therefore, the system can be improved accordingly it gives us a better result when finally implemented in hardware [17].

B. Gazebo

The Gazebo is a robot simulator which enables a user to create complex environments and gives the opportunity to simulate the robot in the created environment. The user can make a robot model with Gazebo and incorporate sensors in a three-dimensional space. In the environment, the user can build a platform and assign obstacles to that. An XML (Extensible Markup Language) file called the URDF (Universal Robotic Description Format) or SDF (Simulation Description Format) is employed to explain various parts of the robot [17]. By giving the link we can give the degree of movement for each section of the robot. The robot model which is created for this research is a two-wheeled differential drive robot, Laser, and a camera on it (Fetch robot). A sample environment is created in the Gazebo for the robot to move and ma p accordingly.

C. Simultaneous Localization and Mapping (SLAM)

Autonomous robots should be capable of safely exploring their surroundings, avoiding clashing with people or slamming into obstacles. SLAM enable the robot to achieve this task by knowing how the surroundings look like (mapping) and where it stays with respect to the surrounding (localization). SLAM can be implemented using different types of 1 Dimensional, 2 Dimensional and 3 Dimensional sensors like Acoustic sensor, Laser Range Sensor, Stereo Vision Sensor and RGB-D sensor. ROS can be used to implement different SLAM techniques like Gmapping, Hector SLAM, KartoSLAM, Core SLAM, Lago SLAM [17]. KartoSLAM, Hector SLAM, and Gmapping are better in the group compared to others. These algorithms have a quite similar performance from map accuracy point of view but are actually conceptually different. That's, Hector SLAM is EKF based, Gmapping is based on RBPF occupancy grid mapping and KartoSLAM is based on the graph-based mapping. Gmapping can perform well for a less processing power robot. The mapping package in ROS supports laser-based SLAM via the slam_gmapping ROS node.

D. Rviz

Rviz stands for ROS visualization. Rviz is a simulator in which we can visualize the sensor data in the 3D environment, for example, if we fix a Kinect sensor to the robot model developed in the Gazebo, the Laser scan value can be visualized in Rviz. We can create a map using the laser scan data and use it for auto navigation. In Rviz we can access and graphically represent the values using camera image, laser scan etc. This information can be used to build the point cloud and depth image. In Rviz coordinates are known as frames. We can select many displays to be observed in Rviz they are data from different sensors. By clicking on the add button we can give any data to be displayed in Rviz. Grid display will give the ground or the reference. Laser scan display will give the display from the laser scanners. Laser scan displays will be of the type sensor messages/Laser scans. Point cloud display will display the position that is given by the program. Axes display will give the reference point. The user may troubleshoot a robot application from sensor inputs to planned (or unplanned) actions by visualising what the robot is seeing, thinking, and doing [18]. In our work Rviz is utilized to display the generated map of environment.

E. Fetch and Freight Robot

Robots have started invading factories, automating tedious and sometimes dangerous workloads once assigned to humans. But there is one area of the industry business where robots have yet to become a more efficient and more viable option i.e., warehouse logistics. The process of repeatedly picking up products from warehouse shelves, carrying them back and forth locations points to prepare for shipping. Fetch mobile manipulator as fig. 1 is designed to pick items off of warehouse shelves, while Freight, a mobile base, acts as an autonomous cargo delivery cart. The robot is made up of a differential drive mobile base, an arm, a pan and tilt head, a torso lift actuator, and a stand-alone mobile robot platform. The mobile base includes a SICK laser scanner with a 220o field of view and 25-meter range. It's base also includes a base-mounted 3D camera. Fetch includes a head-mounted PrimeSense Carmine 1.09 depth camera. The gripper is a parallel-jaw gripper capable of grasping a wide range of objects. Intel-based computers provide processing power for navigation, manipulation and perception activities. The back-drivable 7 DOF arm is capable of lifting 6 kilograms, this is large enough payload to handle the vast majority (90-95 percent) of all items in a typical warehouse [19]. The mobile base is driven by two brushless hub motors in a differential drive configuration. Each hub motor is held in a drop suspension configuration, allowing the robot to keep traction when crossing obstacles, without allowing the robot to sway side-to-side when manipulating on flat ground. Four casters provide stability during movement. Each of the Fetch arm joints are built from a harmonic drive coupled to a brushless frameless motor, mounted inside a custom cast aluminum housing. Each joint has two 14-bit absolute magnetic encoders, one coupled to the joint output shaft, and one coupled to the motor back shaft. The robot has sufficient battery power to work an 8-hour day.

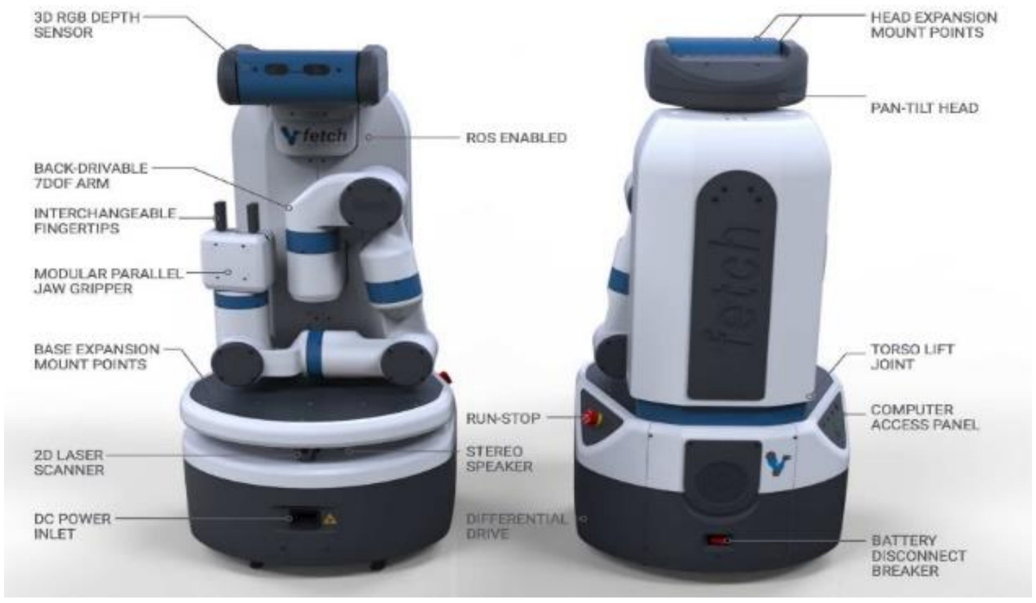

Fig. 1: Fetch Mobile Manipulator, Courtesy of Fetch Robotics Inc. San Jose CA, USA

4. GMAPPING Algorithm

Gmapping is an open-source distance measuring-based SLAM algorithm developed in 2007. It is the most popular SLAM algorithm, considered to be the most powerful algorithm for location and mapping worldwide [20]. Indoor and outdoor mapping and localization can be performed with the Gmapping algorithm. Gmapping Tool needs odometry data (wheel encoder data) and Laser data. The slam gmapping tool is used to generate a 2-D occupancy grid map, similar to a building floor plan.

The Gmapping SLAM algorithm's steps are outlined below [21].

Step 1. Landmark Subtraction:

Its purpose is to identify elements that are distinct from the surrounding environment. These objects depict the obstacles that must be observed and avoided by the robot.

Step 2. Data Association:

Data from numerous sensors is matched to make sure identical landmarks are defined. This is a re-observation process. Thus, false or meaningless landmarks can be removed.

Step 3. Situation Prediction:

Its purpose is to estimate the robot’s local position on the map created after landmark extraction and data association.

Step 4. Status Update:

The map is created in real-time through the recursive repetition of movement and location determination.

Gmapping is a SLAM algorithm, Rao-Blackwellized particle filter-based, in which each particle carries a separate map of the space. With the help of this algorithm, the robot creates a map of the space by sampling the particles with its sensor (LIDAR). The information on how far the robot has progressed in space is obtained by using the odometry sensor. SLAM problem can be expressed by using two factors: localization and mapping as given in Equation 1 below. Based on the Rao-Blackwellized particle filter for SLAM; p(x1:t, m|z1:t, u0:t-1) is the posterior of robot’s potential trajectories x1:t of the robot given its observations z1:t and robot’s odometry measurements u0:t, p(m|x1:t, z1:t) is the posterior over maps, and p(x1:t|z1:t, u0:t-1) is the posterior over maps and trajectories [17].

$$p\left(X_{1:t}, m \middle| Z_{1:t}, u_{0:t-1}\right) = p(m|X_{1:t}, Z_{1:t})\ p(X_{1:t}|Z_{1:t}, u_{0:t-1}) \qquad \ldots\ldots\ldots\ldots \qquad (1)$$

Gmapping creates a propagation pattern using probabilistic distribution methods based on the last observation site of the robot. In this way, a more accurate map is obtained by eliminating uncertain situations. With the Rao-Blackwellized particle filter used in the method, resampling is performed at each update and the sample set representing the robot's trajectory and a map is updated. Sequentially; sampling, assignment based on the weight of importance; resampling, assigning according to the results, and estimating the map corresponding to each particle observation. As a result, low-weight particles are lost and instead high-predictive particles are updated, increasing prediction accuracy.

There are various benefits to using Gmapping as the primary SLAM technique explored in this paper. First and foremost, it is a ready-to-use open-source algorithm available in ROS [22]. Second, it is well-known and has demonstrated performance in recent works. Finally, the algorithm's design lends itself particularly well to parallelization because each particle is computed individually using a single thread, eliminating the requirement for data sharing between them. When the robot acquires a new laser scan, the algorithm performs the processes depicted as a flowchart given in Fig. 2 below. Neff is calculated using equation 2, to assess the extent to which the current particle set accurately represents the desired posterior:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (\tilde{\omega}^i)^2} \quad \dots\dots\dots \quad (2)$$

where $\tilde{\omega}^i$ is the normalized weight of particle i. Finally, each time N_{eff} drops below half of N, where N is the number of Particles, the resampling phase is performed, and particles with low significant weights are replaced by samples with significant weights. Because there are fewer resampling operations during a run and they are carried out only, when necessary, this value significantly reduces the probability of replacing good particles.

The ProcessScan function's scan matching phase takes up 98.47% of its average processing time. This occurs as a result of scan matching, which compares the Laser Range Finder (LRF)'s set of data with the 2D map already acquired. This process is repeated for each of the N particles involved. Since it includes various mathematical operations and is therefore run numerous times during localisation, Furthermore, it should be emphasised that scan matching only happens when the distance travelled by the robot exceeds a predefined linear threshold γ or an angular threshold θ (cf. Fig. 2), which are essential algorithmic factors [23].

The SLAM algorithm is used in a variety of applications, including unmanned aircraft, underwater reef monitoring, self-driving vehicles, and mine exploration.

5. GMAPPING Algorithm

SLAM is concerned with the challenge of building a mobile robot map of an unknown environment while also navigating that environment using the map. To implement this, the following steps were followed.

1. Creating the Environment

- Gazebo Simulator
- Extract URDF (Unified Robot Description Format) files for Fetch Robot Model
- Create Simple Ware House World in Simulation
- Create a Launch File to Bring-up World with Fetch Robot Model

2. Install ROS Package

- Create ROS Workspace
- Compile Fetch ROS packages
- Compile ROS Navigation Stack Meta-package.

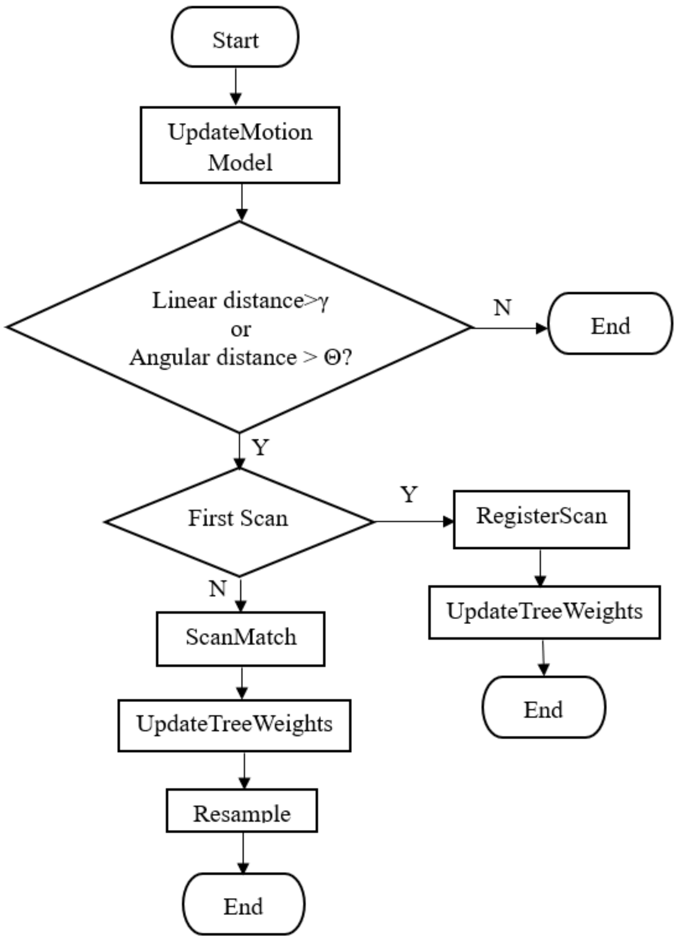

Fig. 2: ProcessScan function flowchart for the Gmapping algorithm

3. Executing Mapping Process

- Launch World with Fetch Robot
- Launch Gmapping
- Launch Rviz
- Save Map

4. Navigation with Map

- Launch Map
- AMCL
- Move_base
- Path Planning
- Navigate using Map

The Ware house environment for the robot model to perform the navigation is created in the gazebo and the Fetch robot model created is imported into the environment. The Fetch robot model consists of mobile base with SICK laser scanner, and a 3D camera is attached to the robot model. Later the plugins were incorporated into the gazebo files. SICK laser provides laser data that can be utilized for creating the map. By adding the required parameters, a map is constructed in the Rviz using the Gmapping tools.

6. Experimental Results And Discussion

In this section, we show the examples of the availability and effectiveness of the SLAM system. The implementation process is carried out using the ROS which is based on a distributed structure and the nodes communicate with each other through the message mechanism. The data exchange is implemented through publishing or subscribing topics. There are multiple kinds of sensors in Fetch robot, including a SICK TIM571 laser, a 6-axis inertial measurement unit (IMU), a Primesense Carmine 1.09 short-range RGBD sensor and gripper sensors [24]. For our application, the laser scanner and head camera are the most useful data sources on the Fetch robot. These sensors publish to the ROS topics 'head camera/depth down sampled/points' and 'base scan'. The slam_gmapping node subscribes to sensor scan data produced by SICK laser scanner model, /tf (transformation) data by /tf package and odometry data to create and publish 2D map of simulated environment on /map topic.

To acquire mapping simulation results, we must first create a simulation environment in which the robot can move around. We have created the indoor static unknown environment in Gazebo for this research work. Here the Fetch robot will move around, navigate and localize itself in the indoor static environment. The sample of the Gazebo environment created is shown in Fig. 3 given below. In this environment, several objects have been placed randomly. The map is generated along with the objects and have been considered as static obstacles.

Fig. 4 given below, shows the initial generation of the map when launched. Initially, the robot is moved to every corner of the unknown environment until a full map is created using the "teleop_key" package where the fetch robot is controlled using the keyboard. The Fig. 5 shown below gives the final generated map in the Rviz which is very much similar to the created environment in the gazebo. For visualization in Rviz, necessary topics have been selected and added. The SICK laser sensor in this robot model publishes the laser data in the form of the topic "/scan" which is selected as a topic of laser scan in rviz. In a similar way for creating the map, "/map" topic has been added. The map_server package, which is part of ROS, has been used to save the created map. The robot is now ready for the integration of navigation stack packages after the map has been created and saved.

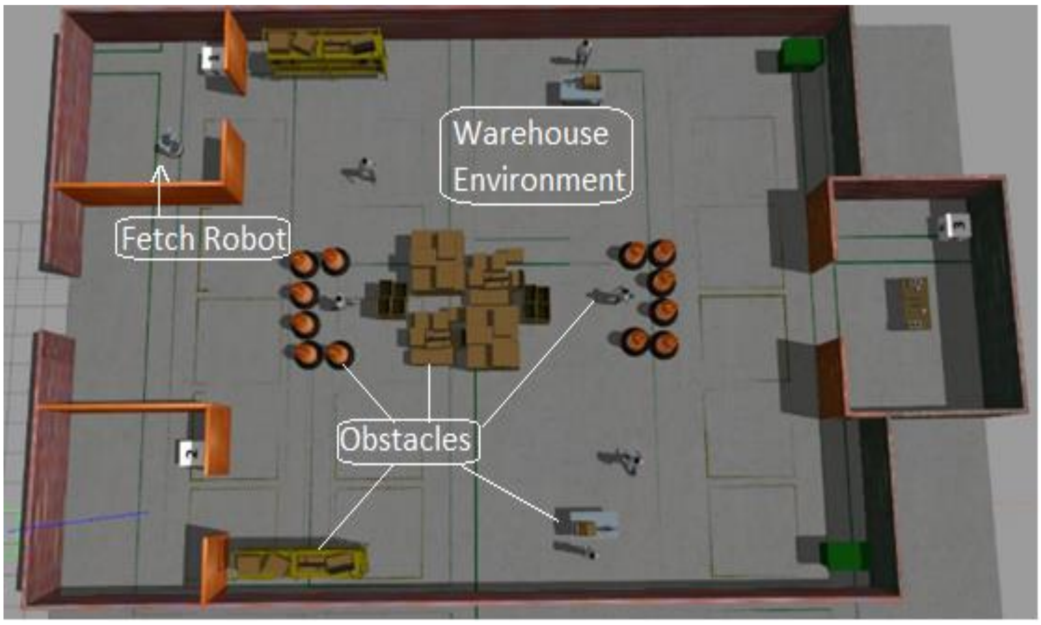

Fig. 3: Sample Warehouse Environment created in Gazebo Simulator

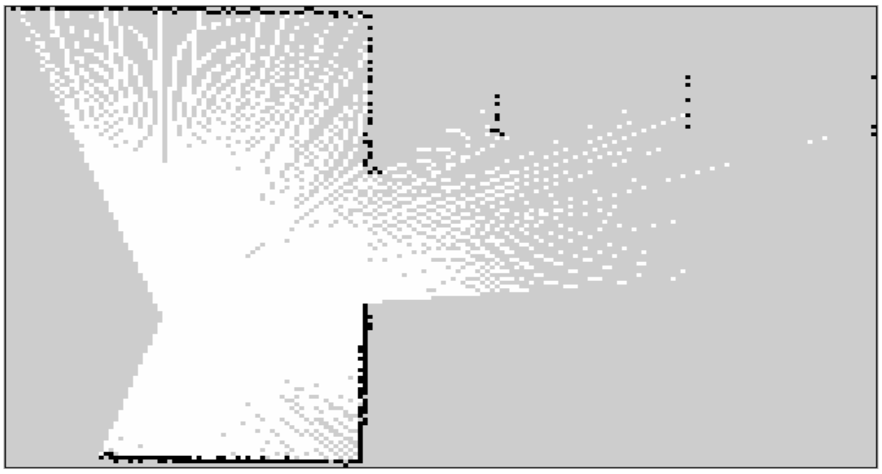
Fig. 4: Initial generation of the map

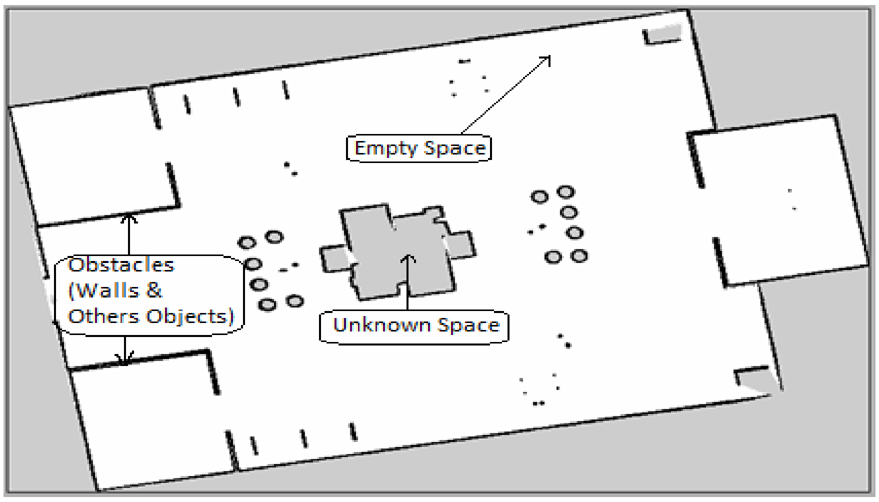

Fig. 5: The final map of the warehouse environment created in Rviz

In the above simulation experiments and field experiments for GMapping, it performs better in a flat indoor environment. Therefore, in a relatively flat indoor ware house environment, it is more appropriate to choose the Gmapping algorithm as the basis of the SLAM system. Gmapping generated the map with lowest error and is an extremely optimized PF algorithm with an improved resampling process and this justifies the quality of the resulting map. Gmapping algorithm presents exceptional results, which reveal the accuracy of PF approaches.

It is essential to note that a robot cannot be navigated without feeding the map to it. Navigation stack packages by using Adaptive Monto Carlo (AMCL) have been used [25]. AMCL is a 2D probabilistic localization system for a mobile robot, which implements the KLD-sampling Monte Carlo localization approach. The algorithm uses a known map of the environment, range sensor data, and odometry sensor data. To localize the robot, the algorithm uses a particle filter to estimate its position. The particles represent the distribution of the likely states for the robot. Each particle represents the state of the robot. After the robot moves forward the generation of new sample starts that will predict the robot's position after the command. Random uniformly distributed samples can be added as the robot recovers if it loses track of its position. Initially, for both the Robots (Fetch Robot & Robot with a bin), particles are very dispersed indicating great uncertainty in the robot position. At this point, sensors have not yet provided any information regarding the location. Fig 6 shows the great uncertainty of the robot's position represented by the particles denoted by red arrows.

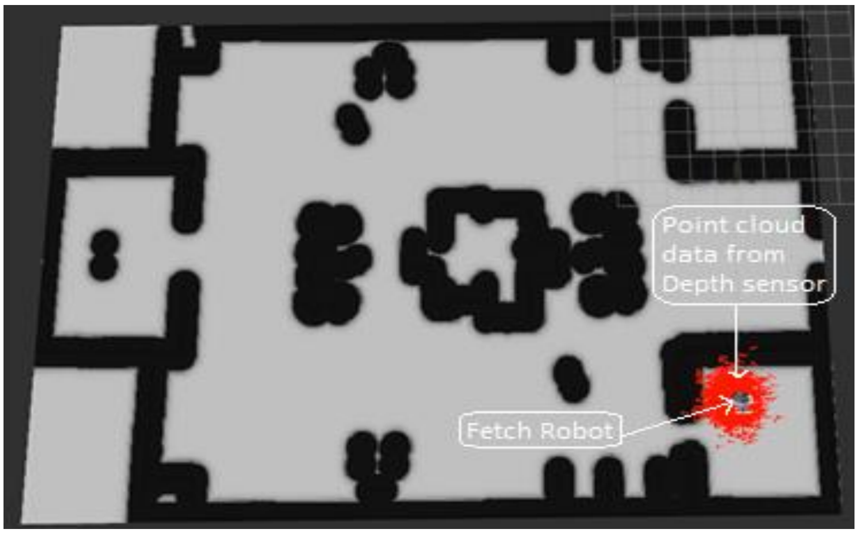

Fig 6: High Uncertainty in the Robot Position

After the simulation is started, the localization process starts from taking the sensor measurements and gradually improve. After the algorithm converges, the particles effectively depict the pose of the robot in the map, thus making the robot successfully navigate through the maze and reach the goal state.

The Fetch Robot reaches the goal within approximately one minute. So, the localization results are pretty decent considering the time taken for the localization and reaching the goal. However, it does not follow a smooth path for reaching the goal. Initially, the robot heads towards the north as it was unable to add the obstacle over there in its local cost-map and hence it followed the shortest path to the goal. But soon, it discovered the presence of the obstacle and it changed its strategy to the next shortest route to reach the goal, i.e., head south-east, turn around where the obstacle ends and reach the goal. Fig. 7 depicts Fetch Robot moving towards the goal position.

Now, the Robot is ready to navigate anywhere in the created map. The destination for the robot can be given using the 2D nav goal option in the Rviz. The user has to click on the desired area in the map and should also point out the orientation of the Robot which it must be in. The robot may not follow the exact path that is given to it because of some of the parameters, but it always tries to follow it by rerouting itself constantly.

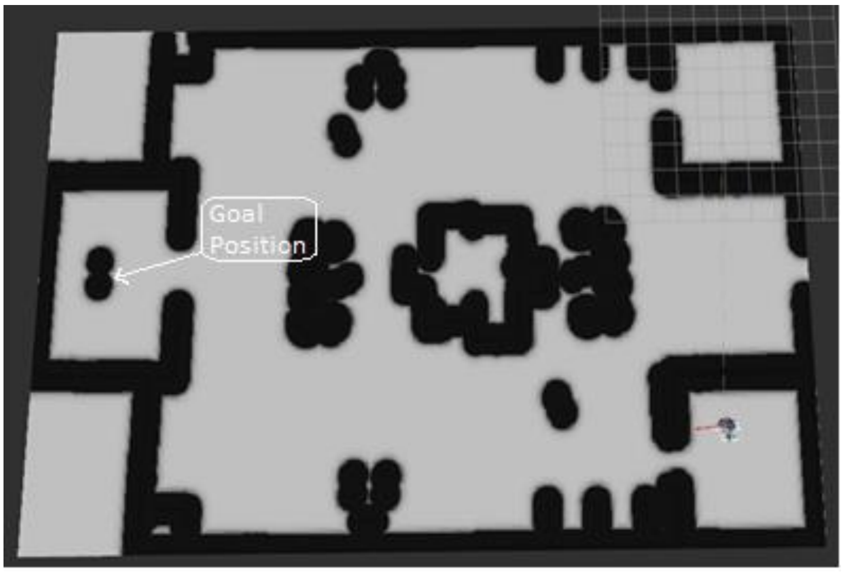

Fig. 7: Fetch Robot moving towards the goal position.

A GUI plugin called Rqt graph is offered by the Rqt tool package in ROS. The rqt graph is a tool for visualising all the active nodes and processes as well as the interaction among them. An overall perspective of the system can be obtained via the rqt graph. Rqt can be utilized as a system debugging tool. Fig. 8 given below shows the Rqt graph of Gmapping. This lists the many topics that are published and subscribed to by the various nodes. The /move_base node is subscribed to several topics like odometry, velocity commands, map, goal, these topics gives the necessary data for the base of the Robot to navigate in the environment.

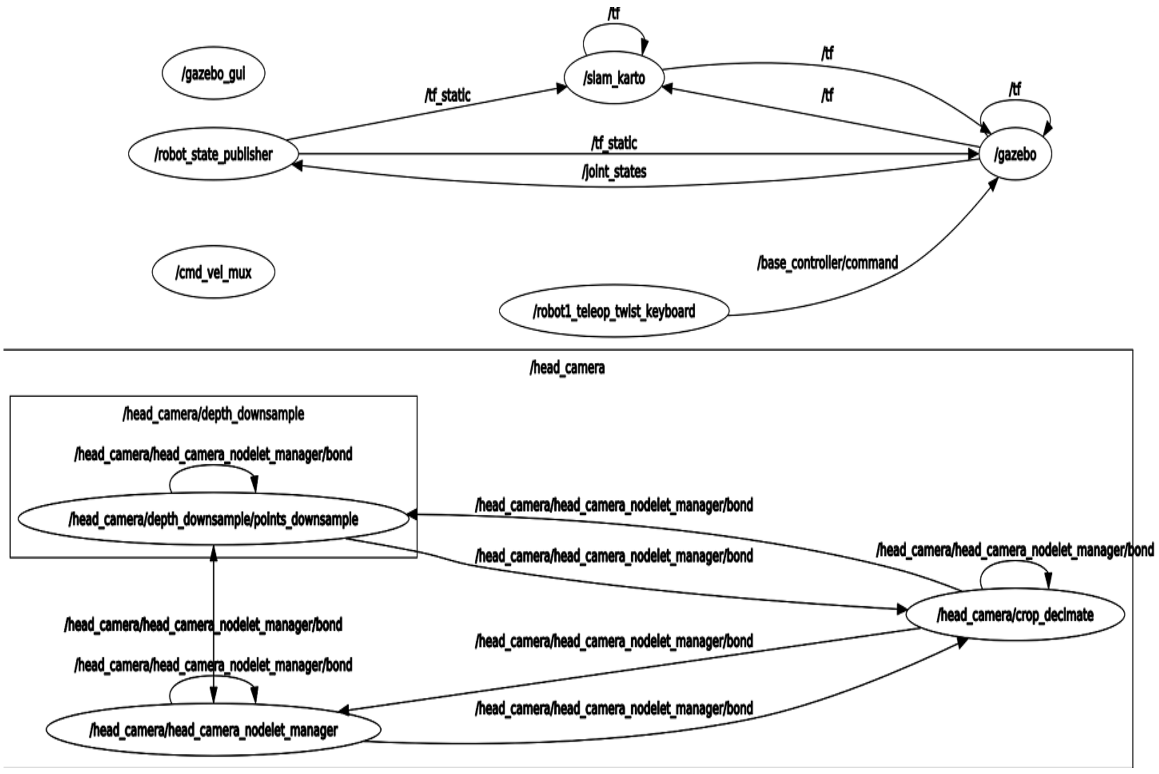
/tf
/gazebo_gui
/slam_karto
/tf
/tf_static
/tf
/tf
/robot_state_publisher
/tf_static
/gazebo
/joint_states
/cmd_vel_mux
/base_controller/command
/robot1_teleop_twist_keyboard
/head_camera
/head_camera/depth_downsample
/head_camera/head_camera_nodelet_manager/bond
/head_camera/depth_downsample/points_downsample
/head_camera/head_camera_nodelet_manager/bond
/head_camera/head_camera_nodelet_manager/bond
/head_camera/head_camera_nodelet_manager/bond
/head_camera/crop_decimate
/head_camera/head_camera_nodelet_manager/bond
/head_camera/head_camera_nodelet_manager/bond
/head_camera/head_camera_nodelet_manager/bond
/head_camera/head_camera_nodelet_manager/bond
/head_camera/head_camera_nodelet_manager

Fig. 8: Node graph or Rqt graph for Mapping

To check the performance of Gmapping SLAM algorithm, criteria like the map quality during mapping, map accuracy, CPU Load and the time taken for the robot to navigate and get to its destination for the warehouse environment, were recorded. The visual inspection of map quality is used to describe it. The effectiveness of the SLAM algorithm is primarily influenced by the accuracy of the maps. Utilising Structure Similarity Index (SSIM) metrics, the Accuracy is evaluated. The SSIM is a metric that compares two maps that are identical. The SSIM value may vary between -1 and +1. The +1 indicates that both maps are identical. The SSIM metric creates the variations in the visual in a structural data format, resulting in greater accuracy than other standard approaches. The map quality is mostly determined by the sensors, processors, and environment. The Sensors used are 2D SICK laser scanner, 3D camera and PrimeSense Carmine 1.09 depth camera. The map quality of Gmapping is relatively better (Fig. 6) than Hector mapping and Karto SLAM mapping. The GMapping is heavily influenced by factors like resampling threshold, linear update, and angular update. The performance was ensured by setting these parameters to a modest level. The main drawback of Gmapping is its reliance on Odometry. A small error in the odometry data will impair the accuracy and quality of the map and result in a structure error. When the simulation maps are compared to the ground truth, the SSIM index value for Gmapping is 0.81, 0.78 for Hector mapping & 0.72 for Karto SLAM. Table I lists the SSIM index value for simulation maps employing ground truth and the graphical representation is as shown in Fig. 9.

TABLE I: SSIM index value of the simulation map applying ground truth values for maps created with all the three algorithms

Sl. No.	**SLAM**	**SSIM (Simulation map Vs. Ground truth)**
1	GMapping	0.81
2	Hector SLAM	0.78
3	Karto SLAM	0.72

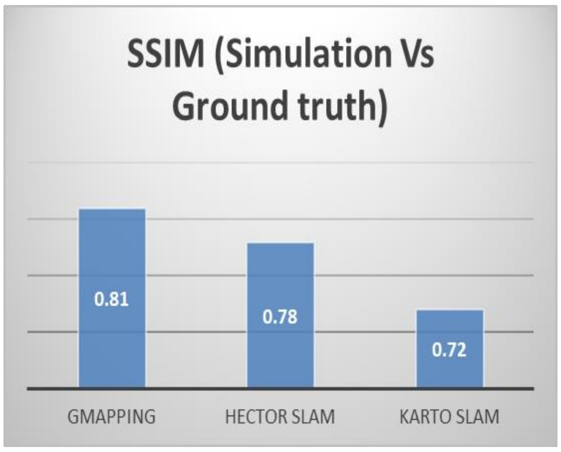

Fig. 9: The graphical representation of the SSIM Index

According to the comparison analysis, Gmapping is the most efficient method, followed by Hector SLAM. The Karto SLAM algorithm is remarkably inaccurate of the SLAM algorithms in use. Figure 10 compares the CPU usage of each algorithm on a desktop computer with an Intel Core i7-4770 and 16GB of RAM. The results of this CPU analysis show that all three algorithms are quite resource-efficient and that they may be used to map general 2D scenarios or in real-world settings.

Our experiments indicate that certain concepts can be kept. On the one hand, HectorSLAM depends exclusively on scan matching and does not employ odometry, this could be advantageous or disadvantageous depending on the qualities of the robot and the environment. Gmapping demonstrated its robustness by keeping error and CPU load to a minimum. It reduces the amount of particles by combining odometry with scan matching. KartoSLAM is Graph based SLAM approach, but its results is distinctively different.

The efficiency of each SLAM algorithm is evaluated using the time taken to reach the destination. The average value of the time taken is recorded for both the conditions (With Obstacles and without Obstacles). With static obstacles Fetch robot navigate from its source to various destinations, and each of this navigation is carried out using a map generated by each SLAM algorithm. Fig.11 shows the time taken for the robot to reach its destination for maps created with GMapping, Hector SLAM and Karto SLAM. They describe the navigation of the robot the global cost map gets updated every time the robot navigates in this environment, and when new obstacles detected via laser scan the map gets updated with the help of AMCL, and the Robot reroutes itself to prevent the new obstacles. Fig.12 shows the time taken by the Robot to reach its destination for maps created with GMapping, Hector SLAM and Karto SLAM.

7. Conclusion

In this study, we used the Fetch Robot in a static indoor environment to analyze and assess the ROS-based SLAM framework. The goal is to provide the robot a precise map of its surroundings so that it can navigate them safely. The Gmapping algorithm, an open-source library, has been implemented on the Fetch

Robot, which is outfitted with a SICK Laser scanner, RGBD Camera, and Inertial Measurement Unit (IMU). This independently explores the environment and creates the map. A more precise and accurate map is obtained from the simulation in ROS-Gazebo. The Fetch Robot is successfully localized with the support of AMCL algorithm and reached the goal state within a decent time-limit (within one second). The mapping quality and accuracy were calculated using the SSIM index to validate these algorithms. Results may differ when implemented in different specification systems because all of the implementations were carried out on a lower level GPU and CPU. It can be inferred that GMapping is the ideal algorithm for creating maps in simulations. The validation results agree with the results published in earlier literature as well. For the future aspects, the comparison of 2D mapping techniques can be done for the different environment. The next step is to extend the following research with 3D SLAM maps.

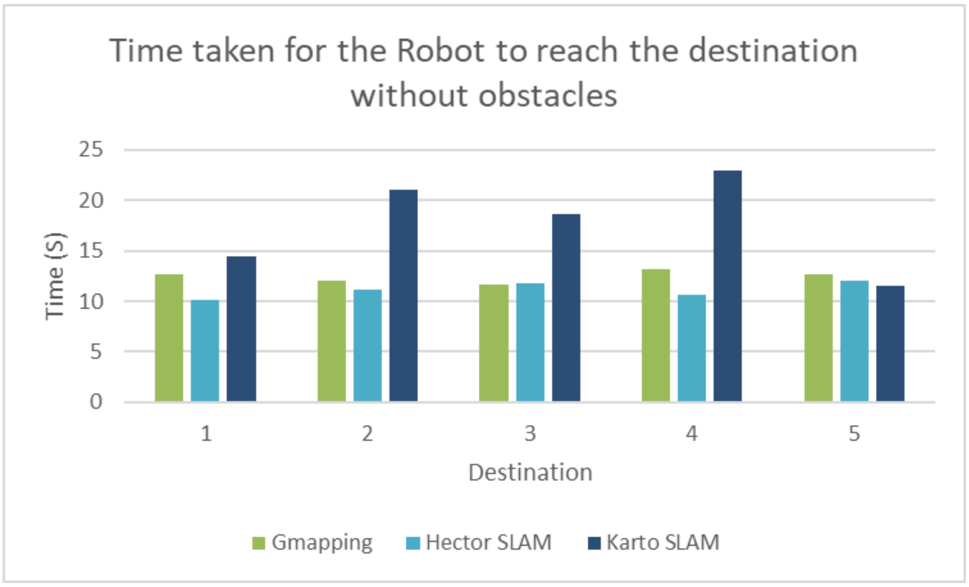

Fig. 11: The graphical representation of time taken to reach the destination without obstacles in the environment

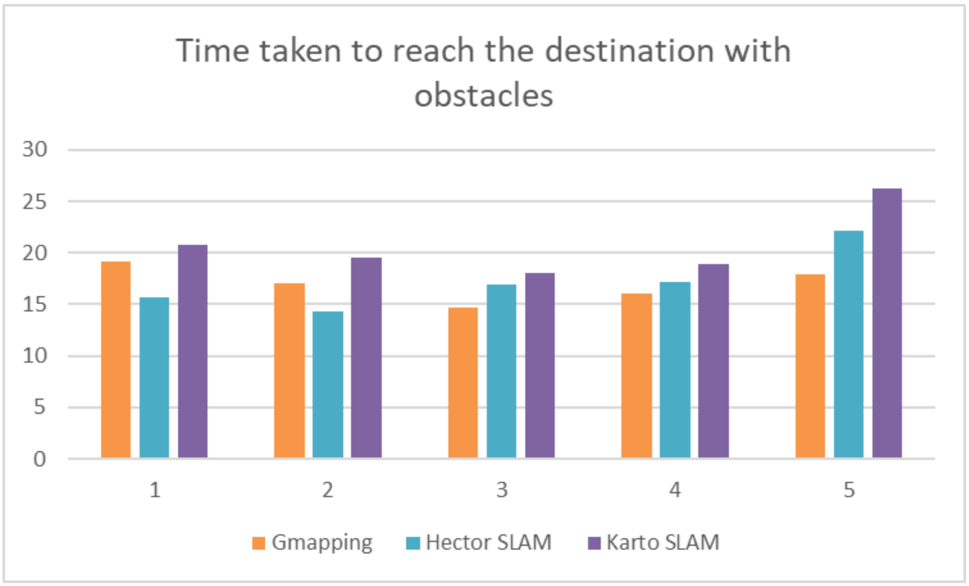

Fig. 12: The graphical representation of time taken to reach the destination with obstacles in the environment

REFERENCES

[1] "The free dictionary". [Online]. Available at http://www.thefreedictionary.com

[2] Dong Shen, Yuhang Xu, Yakun Huang, “Research on 2D-SLAM of Indoor Mobile Robot based on Laser Radar”, Proceedings of the 2019 4th international Conference on Automation, Control and Robotics Engineering (CACRE 2019), Article No.64, pp. 1-7, 2019, https://doi.org/10.1145/3351917.3351966

[3] Julio A. Placed, Jared Strader, Henry Carrillo, Nikolay Atanasov, Vadim Indelman, Luca Carlone, José A. Castellanos, “A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers”, IEEE Transactions on Robotics,pp.1-20, https://doi.org/10.48550/arXiv.2207.00254

[4] A. R. Khairuddin, M. S. Talib and H. Haron, "Review on simultaneous localization and mapping (SLAM)," 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 2015, pp. 85-90, **DOI:** 10.1109/ICCSCE.2015.7482163.

[5] Huishen Zhu, Brenton Leighton, Yongbo Chen , Xijun Ke, Songtao Liu, and Liang Zhao, “Indoor Navigation System Using the Fetch Robot”, ICIRA 2019, LNAI 11743, pp. 686–696, 2019, **DOI:10.1007/978-3-030-27538-9_59.**

[6] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba, "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, June 2001, **DOI:** 10.1109/70.938381

[7] Thrun S, Fox D, Burgard W, Dellaert F, “Robust monte carlo localization for mobile robots”, Artificial Intelligence, Volume 128, Issues 1–2, 99–141 (2001), https://doi.org/10.1016/S0004-3702(01)00069-8

[8] C. Weyers and G. Peterson, "Improving occupancy grid FastSLAM by integrating navigation sensors," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 2011, pp. 859-864, doi: 10.1109/IROS.2011.6094514

[9] J. Zhang, Y. Ou, G. Jiang and Y. Zhou, "An approach to restaurant service robot SLAM," 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), Qingdao, China, 2016, pp. 2122-2127, doi: 10.1109/ROBIO.2016.7866643

[10] G. Hu, S. Huang and G. Dissanayake, "Evaluation of Pose Only SLAM," 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 2010, pp. 3732-3737, doi: 10.1109/IROS.2010.5649825.

[11] Yu-Cheol Lee and Seung-Hwan Park, "3D map building method with mobile mapping system in indoor environments," 2013 16th International Conference on Advanced Robotics (ICAR), Montevideo, Uruguay, 2013, pp. 1-7, doi: 10.1109/ICAR.2013.6766588.

[12] Z. Su, X. Zhou, T. Cheng, H. Zhang, B. Xu and W. Chen, "Global localization of a mobile robot using lidar and visual features," 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, Macao, 2017, pp. 2377-2383, doi: 10.1109/ROBIO.2017.8324775.

[13] A. Ratter and C. Sammut, "Fused 2D/3D position tracking for robust SLAM on mobile robots," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 2015, pp. 1962-1969, doi: 10.1109/IROS.2015.7353635.

[14] B. L. E. A. Balasuriya et al., "Outdoor robot navigation using Gmapping based SLAM algorithm," 2016 Moratuwa Engineering Research Conference (MERCon), Moratuwa, Sri Lanka, 2016, pp. 403-408, doi: 10.1109/MERCon.2016.7480175.

[15] H. I. M. A. Omara and K. S. M. Sahari, "Indoor mapping using kinect and ROS," 2015 International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR), Putrajaya, Malaysia, 2015, pp. 110-116, doi: 10.1109/ISAMSR.2015.7379780.

[16] Darmanin, R. and Bugeja, M. (2016). Autonomous Exploration and Mapping using a Mobile Robot Running ROS. In Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO; ISBN 978-989-758-198-4; ISSN 2184-2809, SciTePress, pages 208-215. DOI: 10.5220/0005962102080215

[17] Darmanin, R. and Bugeja, M. (2016). Autonomous Exploration and Mapping using a Mobile Robot Running ROS. In Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO; ISBN 978-989-758-198-4; ISSN 2184-2809, SciTePress, pages 208-215. DOI: 10.5220/0005962102080215

[18] D. Shen, Y. Huang, Y. Wang and C. Zhao, "Research and Implementation of SLAM Based on LIDAR for Four-Wheeled Mobile Robot," 2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE), Lanzhou, China, 2018, pp. 19-23, doi: 10.1109/IRCE.2018.8492968.

[19] Rajesh Kannan Megalingam, Chinta Ravi Teja, Sarath Sreekanth, Akhil Raj, "ROS based Autonomous Indoor Navigation Simulation Using SLAM Algorithm", International Journal of Pure and Applied Mathematics, Volume 118, No. 7, 2018, 199-205, Corpus ID: 201685925.

[20] Sumegh Pramod Thale, Mihir Mangesh Prabhu, Pranjali Vinod Thakur, Pratik kadam, "ROS based SLAM implementation for Autonomous navigation using Turtlebot", International Conference on Automation, Computing and Communication 2020 (ICACC-2020), ITM Web of Conferences 32, 01011 (2020), DOI: https://doi.org/10.1051/itmconf/20203201011

[21] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr and David Dymesich M, "Fetch & Freight: Standard Platforms for Service Robot Applications," Workshop on autonomous mobile service robots, 1-6, 2016, Corpus ID: 42886148

[22] Santos, J. M., Portugal, D. and Rocha, R. P., An evaluation of 2D SLAM techniques available in robot operating system, IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Linkoping, Sweden, (2013).

[23] Singh, D., Trivedi, E., Sharma, Y. and Niranjan, V., TurtleBot: Design and hardware component selection, International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, (2018).

[24] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, R.: ROS: an open-source Robot Operating System. In: International Conference on Robotics and Automation, WS on Open Source Software, Kobe, Japan (2009)

[25] Bruno D. Gouveia, David Portugal, Daniel C. Silva and Lino Marques, "Computation Sharing in Distributed Robotic Systems: a Case Study on SLAM", IEEE Transactions on Automation Science and Engineering, Volume: 12 Issue: 2, 2014, 410-422, DOI: 10.1109/TASE.2014.2357216 .