International Journal on Interactive Design and Manufacturing (IJIDeM) (2024) 18:3799–3812
https://doi.org/10.1007/s12008-024-01781-7

ORIGINAL ARTICLE

Implementation of simultaneous localization and mapping for TurtleBot under the ROS design framework

Anish Pandey[1] · Kalapala Prasad[2] · Shrikant Zade[3] · Atul Babbar[4] · Gaurav Kumar Singh[1] · Neeraj Sharma[5,6]

Received: 29 December 2023 / Accepted: 9 March 2024 / Published online: 26 March 2024

Abstract
This work presents a comprehensive implementation of Simultaneous Localization and Mapping (SLAM) techniques on the TurtleBot robot within the Robot Operating System (ROS) framework. The study aims to advance the capabilities of the TurtleBot, a popular and cost-effective robot, by integrating hardware and software components, including laser and odometry sensors. The SLAM algorithm, specifically Gmapping, is employed for mapping while utilizing ROS visualization tools like Rviz. The robot's simulation in Gazebo enhances testing in controlled environments. Leveraging teleoperation for data collection, the research delves into the challenges and considerations specific to SLAM on the TurtleBot platform, addressing a notable research gap. The study extends the exploration by investigating potential future enhancements and benefits, showcasing the adaptability and versatility of SLAM-integrated robotic systems. Simulation results detail the successful execution of SLAM through teleoperation, providing insights into mapping accuracy, computational performance, and the overall quality of the generated maps. The work concludes with a discussion on the distance travelled, future prospects, and the profound impact of SLAM on robotic navigation.

Keywords Simultaneous localization and mapping (SLAM) · Robot operating system (ROS) · Gazebo · ROS visualization (Rviz) · GMapping

1 Introduction

SLAM is a crucial technique in robotics that enables a robot to navigate and explore its environment without human intervention autonomously. The technique allows the robot to simultaneously create a map of the surroundings and determine its location within that map. This capability is essential for various applications, including search and rescue missions, exploration of hazardous environments, and autonomous transportation. The TurtleBot is a popular and affordable robot platform that has gained widespread use in both educational and research settings. The robot features a small form factor, high-performance sensors, and a wide range of software tools and libraries, making it an ideal choice for developing and testing robotics applications. The ROS provides a robust framework for developing and implementing robotics applications. ROS is an open-source platform that provides a standardized set of tools and libraries for building and testing robotics applications. It enables developers to easily integrate hardware components, write control algorithms, and implement complex behaviors for robots.

✉ Neeraj Sharma
neerajsharma@mmumullana.org

1 Mechatronics Lab, School of Mechanical Engineering, KIIT Deemed to be University, Patia, Bhubaneswar 751024, Odisha, India

2 Department of Mechanical Engineering, University College of Engineering Kakinada, Jawaharlal Nehru Technological University Kakinada, Kakinada 533003, Andhra Pradesh, India

3 Department of Computer Science Engineering, Nagpur Institute of Technology, Nagpur 440013, Maharashtra, India

4 Department of Mechanical Engineering, Faculty of Engineering and Technology, SGT University, Gurugram 122505, Haryana, India

5 Department of Mechanical Engineering, Maharishi Markandeshwar Engineering College, Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala 133207, India

6 Production Engineering Department, National Institute of Technology Agartala, Agartala 799046, Tripura, India

In a study implementation of SLAM using a differential drive robot and laser scan data for mapping and navigation in an indoor environment was done. The authors highlighted the importance of adopting SLAM, particularly extended Kalman filter (EKF), and highlights the variations in results when SLAM is not used [1]. For autonomous robots, localization and mapping are essential tasks that are often carried out with pricey sensors. However, improvements in consumer electronics have reduced the cost of similar sensors. The SLAM technique used in their work has been modified utilizing a TurtleBot robot and a Microsoft Kinect camera [2].

The proposed hierarchical map fusion technique directs raw data to edge servers for real-time fusion and then to the cloud for global merging. A proof-of-concept prototype was created and used in actual situations to show how well it handles the difficulties of cooperating with SLAM across numerous robots with constrained computer power [3]. The authors explained the SLAM algorithm's implementation within the Robot Operating System framework for autonomous robot navigation [4]. Rviz was used for data visualization, while Gazebo runs the simulation. Laser and odometry data from various sensors were incorporated into the GMapping program for mapping purposes. In another study the usage of a Kinect sensor for robotic routing's autonomous vision and navigation was presented [5]. The ROS was optimized using the GMapping package to increase map generation and laser scan accuracy.

In a study, SLAM algorithm was implemented for mobile robots that can navigate independently. The suggested method used an RGB-D camera (Depth sensor) to identify the surroundings and create an occupancy grid, enabling the mobile robot to move around the environment independently [6]. The authors used Turtlebot3 to build an EKF-SLAM algorithm in Python using ROS. To estimate poses and map them, the program used landmark detection [7]. According to the results, the method produced a nice map with good pose estimation, but it had scaling problems and needed more landmarks. The technique can solve SLAM problems but needs to be more scalable and needs accurate landmark identification to operate at its best.

The authors Study described the use of the ROS framework and Arduino technology to create a low-cost differential drive robot for SLAM. The implementation is validated using mapping studies performed in private settings [8]. The Rao-Blackwellization particle filter algorithm utilized in the platform is theoretically explained in the study.

The authors introduced a method for simulating a multi-robot environment in the Gazebo simulator [9]. This method involved producing a launch file containing several robot attributes and specifications. A generalized MATLAB script adds a user-defined TurtleBot population and their chat topics to a Gazebo world. For MR-SLAM, SEIF with established correspondences was implemented, and the system was evaluated.

The SLAM technique for robot mapping was presented in using ROS and Arduino technologies [10]. The Rao-Blackwellization particle filter technique is theoretically explained, and information on how to build a low-cost differential drive robot for mapping home situations is provided.

The EKF-SLAM technique was implemented in this work, represented in [11], using a cloud computing architecture based on ROS. EKF-SLAM running under MATLAB, a ROS master node running in the cloud, and a TurtleBot equipped with a Kinect camera in a Gazebo environment are used in the implementation [11].

For robotic navigation and spatial awareness, investigation of a version of SLAM with lower processing costs that uses dynamic field theory (DFT) was carriedout [12]. Compared to other typical SLAM algorithms, this implementation completes SLAM tasks with a similar level of accuracy but at a memory cost of only 1/5. According to the study, a particle value of at least five and a resampling threshold parameter between 0.5 led to effective mapping and decreased robot position uncertainty [13]. To create a better map, the values of the linear step update and angular step update parameters were gradually decreased. According to the autonomous navigation stack test, the robot can go from home to the navigation target in 25 s.

The technique, as represented, keeps the real-time computing cost constant while enhancing the ICP's accuracy and resilience [14]. The paper also presents an effective RGB-D SLAM system based on planar features, trajectory, and map results from open datasets and actual robot experiments.

SLAM is the main topic of this research since it is crucial for autonomous wheeled mobile robots. While SLAM offers environment mapping and agent localization, robots are guided through user guidance, haphazard exploration, or exploration algorithms in uncharted territory. The study demonstrated that SLAM may be carried out automatically in comparable contexts without human intervention [15].

A study provided a unique method for communicating with impediments such as closed doors during autonomous navigation with the help of a cloud IoT strategy [16]. To go to the destination, a ground mobile robot such as TurtleBot3, powered by ROS, will open the entrance and navigate the entryway. The doors are left open in this study when creating the maps. The authors examined the effectiveness of map merging and multi-robot navigation in both simulated and actual environments [17]. Two TurtleBot3 Burgers are used in collaborative mapping employing SLAM techniques to blend and explore maps depending on the relative locations of the robots. A multi-robot collaborative mapping and navigation simulation is created using ROS Gazebo and virtual activities. The issues of mapping and navigating mobile robots in dynamic situations were discussed [18], which also provides a GMapping method employing slam for mapping. Unknown interior settings are mapped using the SLAM and Rviz GMapping tools, Robotic Operating System (ROS), and the RPLiDAR camera.

Despite the widespread use of SLAM techniques in robotics, there needs to be more comprehensive research focusing specifically on the implementation of SLAM with teleoperation on the Turtle Bot platform. While SLAM algorithms and hardware setups have been extensively studied and implemented in various robotic systems, the specific challenges, considerations, and performance evaluation of SLAM in conjunction with teleoperation on the Turtle Bot remain relatively unexplored.

This research gap highlights the need for a detailed investigation into the implementation of SLAM with teleoperation on the Turtle Bot, considering this specific robot platform's unique characteristics and capabilities. This suggests a need for more comprehensive research specifically addressing the challenges, considerations, and performance evaluation aspects of utilizing teleoperation for SLAM on the Turtle Bot. Identifying this research gap opens up opportunities for further research to bridge this knowledge gap and contribute to the understanding and advancement of SLAM techniques in teleoperated Turtle Bot systems.

One significant challenge is to ensure seamless integration between the teleoperation interface and the SLAM algorithm. While SLAM algorithms have been extensively studied and optimized for autonomous navigation, their performance and behavior may differ when combined with teleoperation. Understanding how teleoperation commands and movements affect SLAM performance, accuracy, and map quality is crucial. It includes investigating issues such as latency, control precision, and potential biases introduced by human operators during teleoperation.

Another consideration is the impact of the Turtle Bot's unique characteristics on SLAM performance during teleoperation. The small size and agility of the Turtle Bot can pose challenges in terms of sensor perception, motion dynamics, and environmental interactions. Investigating how these factors influence the quality and reliability of SLAM mapping and localization results is important to develop robust and efficient SLAM systems tailored explicitly for the Turtle Bot.

Performance evaluation of SLAM with teleoperation on the Turtle Bot also still needs to be explored. While existing SLAM evaluation metrics exist for autonomous navigation scenarios, they may need to fully capture teleoperation's performance nuances and limitations. Developing appropriate evaluation methodologies and metrics that consider the interactive nature of teleoperation, such as the operator's situational awareness and navigation efficiency, is crucial to assess the effectiveness of SLAM in teleoperated Turtle Bot systems.

Addressing this research gap through comprehensive investigations can yield several benefits:

1. It can provide insights into the specific challenges and considerations when implementing SLAM with teleoperation on the Turtle Bot, enabling researchers and practitioners to develop tailored solutions and methodologies.
2. It can enhance the understanding of the strengths and limitations of SLAM in teleoperated scenarios, helping optimize its performance and reliability.
3. It can contribute to the broader field of SLAM research by expanding its application domain to include teleoperated systems, potentially leading to advancements in teleoperation techniques and SLAM algorithms.
4. Precision in Localization: SLAM ensures accurate real-time self-localization, vital for navigating in unknown or dynamic environments.
5. Dynamic Mapping: The robotic system generates, and updates maps on-the-fly, enabling informed decisions based on the most current environmental data.
6. Adaptive Path Planning: With precise localization and dynamic mapping, the robot can adjust its path in real-time, enhancing adaptability in various scenarios.

In this research work, the focus will be on implementing SLAM for the TurtleBot using ROS. The goal is to enable the robot to accurately localize itself in an unknown environment while simultaneously creating a map of the surroundings. This project will require a combination of hardware and software components, including sensors, actuators, algorithms, and software tools. The implementation will involve integrating the SLAM algorithm with the robot's hardware and software components and testing the system in various environments. The successful implementation of SLAM for the TurtleBot using ROS has the potential to advance the capabilities of the robot, making it a more versatile and valuable tool for researchers, educators, and enthusiasts. The resulting system will enable the robot to navigate and explore new environments autonomously, opening up new opportunities for research and development in the field of robotics.

Furthermore, implementing SLAM for the TurtleBot under ROS can lead to the development of new and innovative robotics applications. For instance, the resulting system can be used for indoor mapping, where the robot can autonomously navigate and create a 3D map of the environment. It can be beneficial for warehouse management and indoor surveillance tasks. Additionally, implementing SLAM can help create interactive environments where the robot can navigate the environment and provide real-time feedback to users. Another benefit of implementing SLAM for TurtleBot under ROS is that it can facilitate the development of multi-robot systems. Multiple robots can be equipped with the SLAM algorithm, enabling them to collaboratively create a map of their environment and share information with each other. It can lead to the development of highly efficient and versatile multi-robot systems that can be used in various applications, such as search and rescue missions and environmental monitoring. It has tremendous potential to contribute to advancing technology in many different areas.

2 Simulation environment

2.1 ROS framework

ROS is a flexible and open-source framework for creating robotics applications. It offers robust tools, libraries, and conventions that allow developers to create highly modular and scalable systems. ROS was initially developed by researchers at Stanford University in 2007, and since then, it has gained significant traction in both academic and industrial settings.

One of the key advantages of ROS is its highly modular architecture. ROS applications are built as a set of modules, known as nodes, that communicate with each other using messages. These nodes can be written in various programming languages, including C++, Python, and MATLAB, and run on different network machines. This flexibility makes it easy for developers to distribute processing across multiple machines and create highly complex and distributed robotics systems.

Another important aspect of ROS is its support for sensors and actuators. The framework provides a standardized interface for accessing various sensors and actuators, such as cameras, LiDARs, and motors. It makes it easy for developers to integrate hardware components into their applications and provides a uniform way of accessing them, regardless of the specific hardware used.

ROS also offers robust tools and libraries for carrying out common robotics tasks, such as motion planning, perception, and localization. For example, Navigation Stack is a suite of ROS packages that provides mapping, path planning, and localization tools, making it easy for developers to create robots that can navigate autonomously in different environments.

One of the biggest strengths of ROS is its active and engaged community. The ROS community comprises developers, researchers, and enthusiasts from around the world who contribute to the development and maintenance of the framework. This community provides a wealth of resources, including documentation, tutorials, and example code, making it easier for newcomers to get started with ROS and for experienced developers to share their knowledge and expertise.

ROS also offers a powerful tool for testing and debugging robotics applications. The framework includes a built-in simulation environment, known as Gazebo, that allows developers to create virtual environments for testing their applications in a safe and controlled manner. It is beneficial for applications that involve complex and potentially dangerous environments, such as search and rescue missions or exploration of hazardous environments.

The ROS is a powerful and flexible framework that offers a wealth of tools, libraries, and conventions for creating robotics applications. Its highly modular architecture, support for sensors and actuators, and rich set of tools and libraries make it an ideal choice for building complex and scalable robotics systems. With a vibrant and engaged community, ROS is an excellent choice for both newcomers and experienced developers looking to create innovative and cutting-edge robotics applications.

2.2 Gazebo simulator

Gazebo is an open-source robotics simulation software that allows developers to simulate complex and dynamic environments for testing and debugging robotics applications.

One of the key features of Gazebo is its highly realistic physics engine. The software simulates the physical behavior of objects in a virtual environment, including gravity, collisions, and friction. It allows developers to test their applications in a highly realistic and dynamic environment, providing a more accurate representation of real-world conditions.

The software provides a standardized interface for accessing sensors and actuators, including cameras, LiDARs, and motors. This makes it easy for developers to integrate hardware components into their simulations and test their applications in a realistic and dynamic environment. Similarly, Fig. 1 represents the Turtle Bot model integrated with LiDAR and various other types of sensors in the Gazebo Simulator.

Gazebo also provides a powerful and flexible simulation environment. The software allows developers to create custom environments for testing their applications, including indoor and outdoor environments and different weather conditions. It enables developers to test their applications under various conditions and simulate scenarios that might be difficult or impossible to create in the real world.

In addition to its support for sensors and actuators, Gazebo also provides a rich set of tools and libraries for

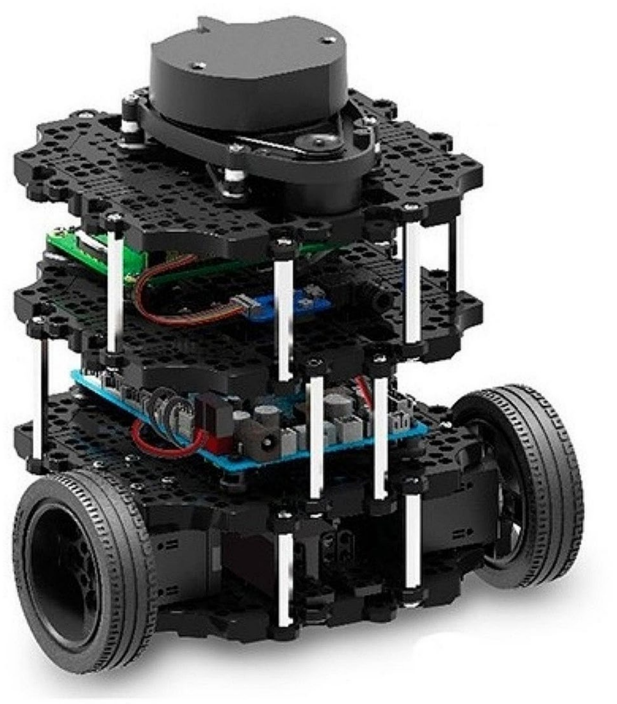


Fig. 1 Turtle Bot model in Gazebo

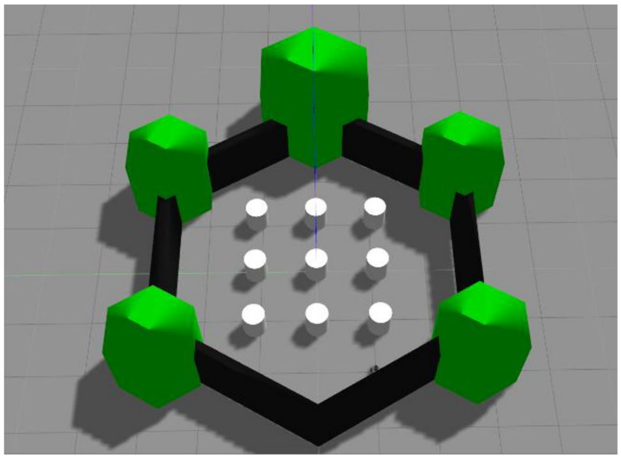


Fig. 2 Simulation environment in Gazebo

carrying out common robotics tasks. For example, the software includes a set of plugins for simulating cameras, LiDARs, and other sensors, as well as tools for path planning, localization, and control.

One of the biggest strengths of Gazebo is its integration with other robotics tools and frameworks, including the ROS. Gazebo provides an ROS interface, allowing developers to integrate their simulations with other ROS nodes and applications easily. It makes it easy for developers to test their applications in a highly realistic and dynamic environment while also integrating with other ROS tools and libraries. Figure 2 shows the simulation environment/map in the Gazebo Simulator.

Gazebo is also highly extensible, with a large and active community of developers contributing to its development and maintenance. The software provides a plugin system, allowing developers to create custom plugins for adding new sensors, actuators, or other functionality. The community also provides a wealth of resources, including documentation, tutorials, and example code, making it easier for newcomers to get started with Gazebo and for experienced developers to share their knowledge and expertise.

Gazebo is a powerful and flexible robotics simulation software that provides a highly realistic and dynamic environment for testing and debugging robotics applications. Its support for a wide range of sensors and actuators and its integration with other robotics tools and frameworks make it an ideal choice for developing and testing complex and scalable robotics systems. With a large and active community of developers, Gazebo is an excellent choice for both newcomers and experienced developers looking to create innovative and cutting-edge robotics applications.

2.3 ROS visualization

Rviz is a powerful 3D visualization tool for robotics applications. It is part of the ROS framework and is widely used in the robotics community for.

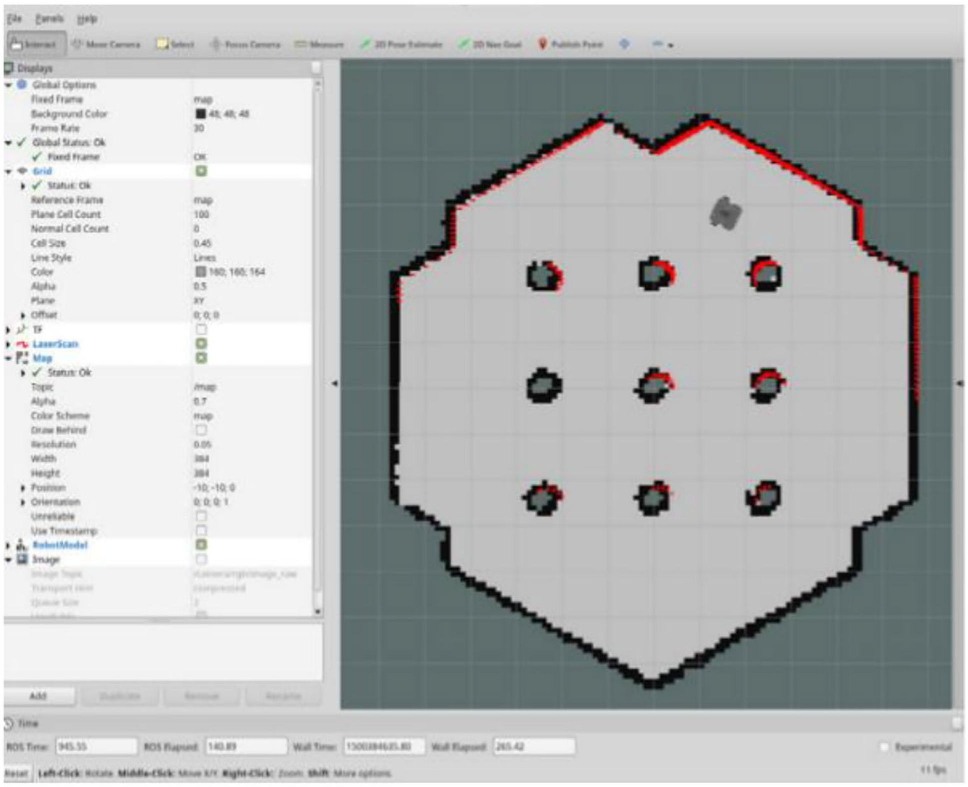
debugging and testing robots, as well as demonstrating and presenting research results.

One of the key features of Rviz is its ability to visualize data from a wide range of sensors and sources. The software can display data from cameras, LiDARs, sonar sensors, GPS, and other sensors, as well as from robot state and control topics. It makes it easy for developers to visualize and analyze the performance of their robots in real-time, providing valuable feedback for debugging and testing.

Rviz provides a highly configurable and customizable interface, allowing developers to create custom visualizations for their specific needs. The software includes a wide range of visualization tools, including 3D models, point clouds, laser scans, and heat maps. It also provides tools for displaying robot kinematics and trajectory data and creating interactive markers and menus.

Another important feature of Rviz is its support for interactive manipulation of objects in the virtual environment. Developers can use Rviz to move, rotate, and scale objects in the virtual environment, providing a powerful tool for testing and debugging robotics applications. Figure 3 gives the map visualization in Rviz for the particular simulation environment.

Rviz also provides a variety of tools for analyzing and debugging robot behavior. The software includes tools for



```
Global Options
  Fixed Frame          map
  Background Color     48; 48; 48
  Frame Rate           30
Global Status: Ok
  Fixed Frame          OK
Grid                   ✓
  Status: Ok
  Reference Frame      map
  Plane Cell Count     100
  Normal Cell Count    0
  Cell Size            0.45
  Line Style           Lines
  Color                160; 160; 164
  Alpha                0.5
  Plane                XY
  Offset               0; 0; 0
TF                     ☐
LaserScan              ✓
Map                    ✓
  Status: Ok
  Topic                /map
  Alpha                0.7
  Color Scheme         map
  Draw Behind          ☐
  Resolution           0.05
  Width                384
  Height               384
  Position             -10; -10; 0
  Orientation          0; 0; 0; 1
  Unreliable           ☐
  Use Timestamp        ☐
RobotModel             ✓
Image                  ☐

ROS Time: 945.55   ROS Elapsed: 140.89   Wall Time: 1500384635.80   Wall Elapsed: 265.42
```

Fig. 3 Map visualization in Rviz

displaying and analyzing robot path planning and localization data and for visualizing robot state and control data. This makes it easy for developers to identify and troubleshoot application problems and optimize their robots' performance.

In addition to its powerful visualization capabilities, Rviz is highly integrated with other ROS tools and frameworks. The software provides a standardized interface for accessing and visualizing data from ROS topics, making it easy to integrate with other ROS nodes and applications. It allows developers to quickly visualize and analyze data from their robots in real-time while also integrating with other ROS tools and libraries.

One of the biggest strengths of Rviz is its versatility and flexibility. The software can be used in a wide range of robotics applications, from small educational projects to large-scale industrial systems. It can be customized and extended to meet the specific needs of individual projects, making it an ideal tool for research and development.

Rviz also benefits from a large and active community of developers contributing to its development and maintenance. The community provides many resources, including documentation, tutorials, and example code. It makes it easier for newcomers to start with Rviz and for experienced developers to share their knowledge and expertise.

In conclusion, Rviz is a versatile 3D visualization tool for robotics applications. Its support for a wide range of sensors and sources, highly configurable interface, and integration with other ROS tools and frameworks make it an ideal choice for developers looking to debug, test, and optimize their robotics applications. With a large and active community of developers, Rviz is an excellent choice for both newcomers and experienced developers looking to create cutting-edge robotics applications.

3 SLAM implementation

SLAM is a technique widely used in robotics for creating maps of unknown environments while simultaneously localizing the robot within that environment. In this research paper, Fig. 4 presents the implementation of SLAM as a flowchart representation for Turtle Bot with the help of a teleoperation node in detail.

The Turtle Bot is a mobile robot that is commonly used in research and educational settings. Its small size and agility make it an ideal platform for implementing SLAM. To implement SLAM for the Turtle Bot with the help of a teleoperation node, we first need to set up the necessary hardware and software components.

The hardware components required for implementing SLAM on the Turtle Bot include a laser range finder and a computing device. The laser range finder scans the environment and generates a 2D map of the surrounding area.

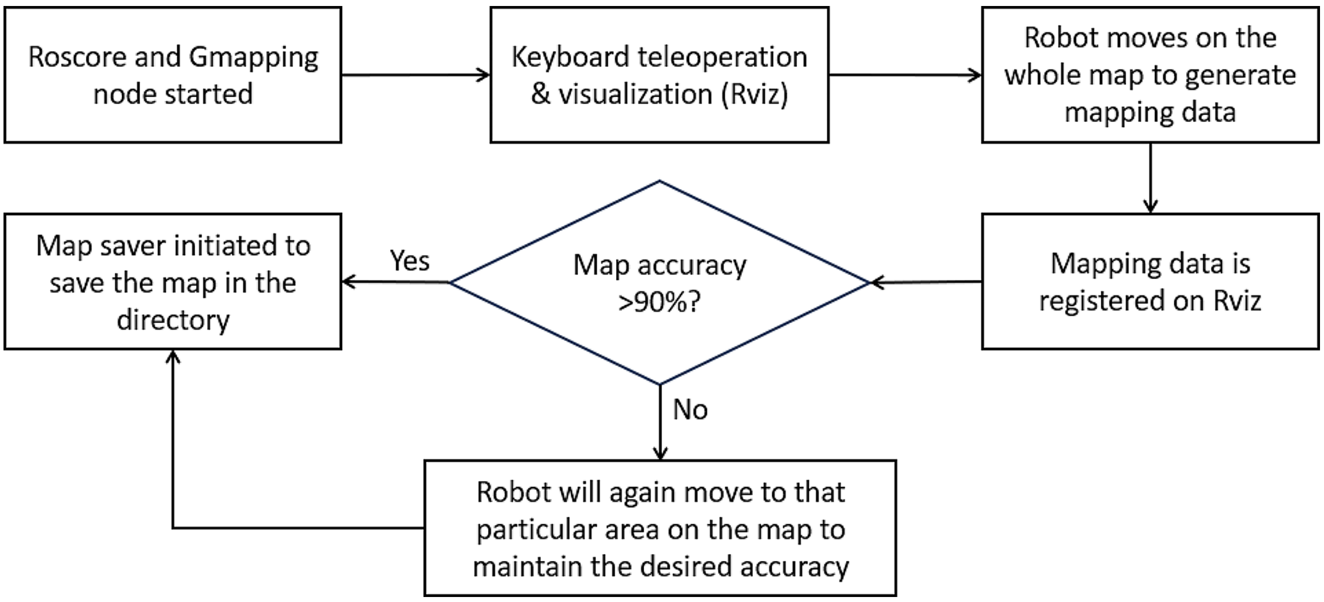

Roscore and Gmapping node started
Keyboard teleoperation & visualization (Rviz)
Robot moves on the whole map to generate mapping data
Mapping data is registered on Rviz
Map accuracy >90%?
Yes
Map saver initiated to save the map in the directory
No
Robot will again move to that particular area on the map to maintain the desired accuracy

Fig. 4 Flowchart of SLAM implementation

The computing device processes the sensor data and runs the SLAM algorithm. Additionally, a keyboard is needed to control the Turtle Bot using teleoperation.

The software components required for implementing SLAM on Turtle Bot include ROS, Gazebo, the teleoperation node, and the SLAM algorithm. ROS is middleware that provides tools and libraries for robot applications, making implementing SLAM for the Turtle Bot easy. Gazebo is a popular robot simulation software that can be used to test and evaluate the SLAM algorithm before deploying it on a real robot. The teleoperation node is a software module that allows us to control the Turtle Bot using a keyboard. The SLAM algorithm uses sensor data to create a map of the environment and estimate the robot's position and orientation within that environment.

To implement SLAM for the Turtle Bot with teleoperation, we first set up the hardware and software components as discussed above. We then start the Gazebo simulation and launch the SLAM algorithm. We use the teleoperation node to control the Turtle Bot and move it around the environment. As the Turtle Bot moves, the laser range finder generates a map of the environment, which the SLAM algorithm uses to estimate the robot's position and orientation.

As we move the Turtle Bot around the environment using teleoperation, we can observe the map generated by the SLAM algorithm in real-time. The map shows the obstacles and features in the environment, as well as the location of the Turtle Bot. The SLAM algorithm updates the map continuously as the Turtle Bot moves around the environment, allowing us to create a complete and accurate map of the area.

In addition to teleoperation, we can also use autonomous navigation to control the Turtle Bot and collect data for SLAM. Autonomous navigation involves programming the Turtle Bot to move around the environment on its own using a predefined path or algorithm. It can be useful for collecting data in difficult or unsafe areas to navigate using teleoperation.

Once we have collected enough data using teleoperation or autonomous navigation, we can use the SLAM algorithm to generate a complete and accurate map of the environment. This map can be used for a variety of applications, such as navigation, object detection, and path planning.

4 Results and discussion

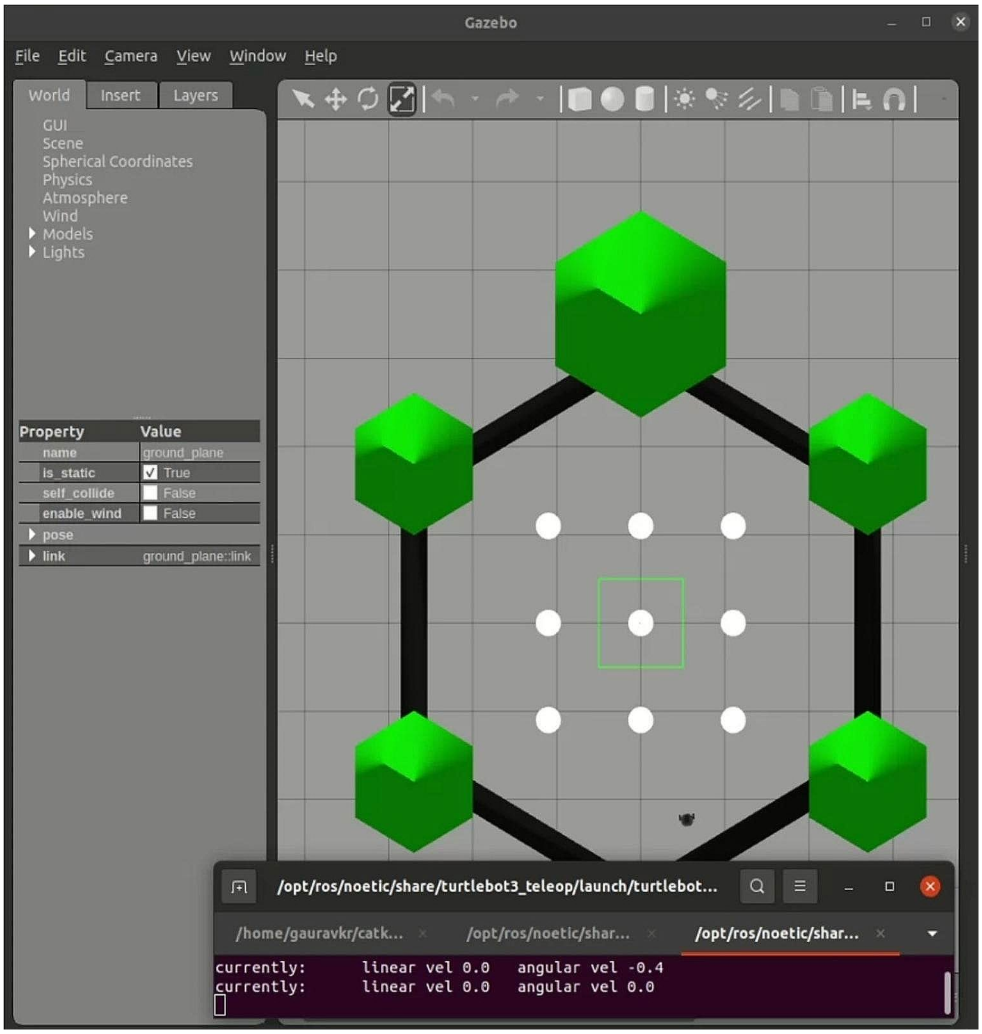
The implementation of SLAM for Turtle Bot using a teleoperation node has been successfully demonstrated in this research paper. The hardware and software components required for the implementation were discussed in detail, and the steps involved in implementing SLAM with teleoperation were explained thoroughly. The resulting map generated by the SLAM algorithm was observed in real-time as the Turtle Bot moved around the environment using teleoperation. The map shows the obstacles and features in the environment. The implementation of the SLAM algorithm updated the map continuously as the Turtle Bot moved around the environment, creating a complete and accurate map of the area. The accuracy of the map generated by the SLAM algorithm will be further improved by using the camera data in addition to the laser range finder data. The resulting map generated by the algorithm can be used for a variety of applications, such as navigation, object detection, and path planning.

Overall, the implementation of SLAM for the Turtle Bot with the help of a teleoperation node was successful, and the results can provide valuable insights for researchers in the field of robotics. Figure 5 shows the starting position of the Turtle Bot in the Simulation Environment.

From Fig. 6, it can be observed that less than half of the whole map is generated by the SLAM algorithm. Figure 7 shows the mid-position of the Turtle Bot in the Simulation Environment. Table 1 simulation position indicates different stages of the Turtle Bot's movement in the simulation environment, including the starting position, mid-position, ending position and the map coverage provides the percentage of the map covered by the SLAM algorithm. Table 2 provides key performance metrics of the SLAM algorithm, such as mapping accuracy, localization error and computational time per frame. Table 3 evaluates the quality of the generated SLAM map based on metrics such as feature sharpness, consistency of features, and noise level. Table 4 summarizes the elapsed time for the SLAM implementation at different positions. Similarly, Table 5 calculates the time difference between different positions for SLAM Implementation. Figure 8 reveals that more than half of the whole map is generated by the SLAM algorithm. Figure 9 shows the ending position of the Turtle Bot in the Simulation Environment. Figure 10 shows that the whole map is generated by the SLAM algorithm.

.

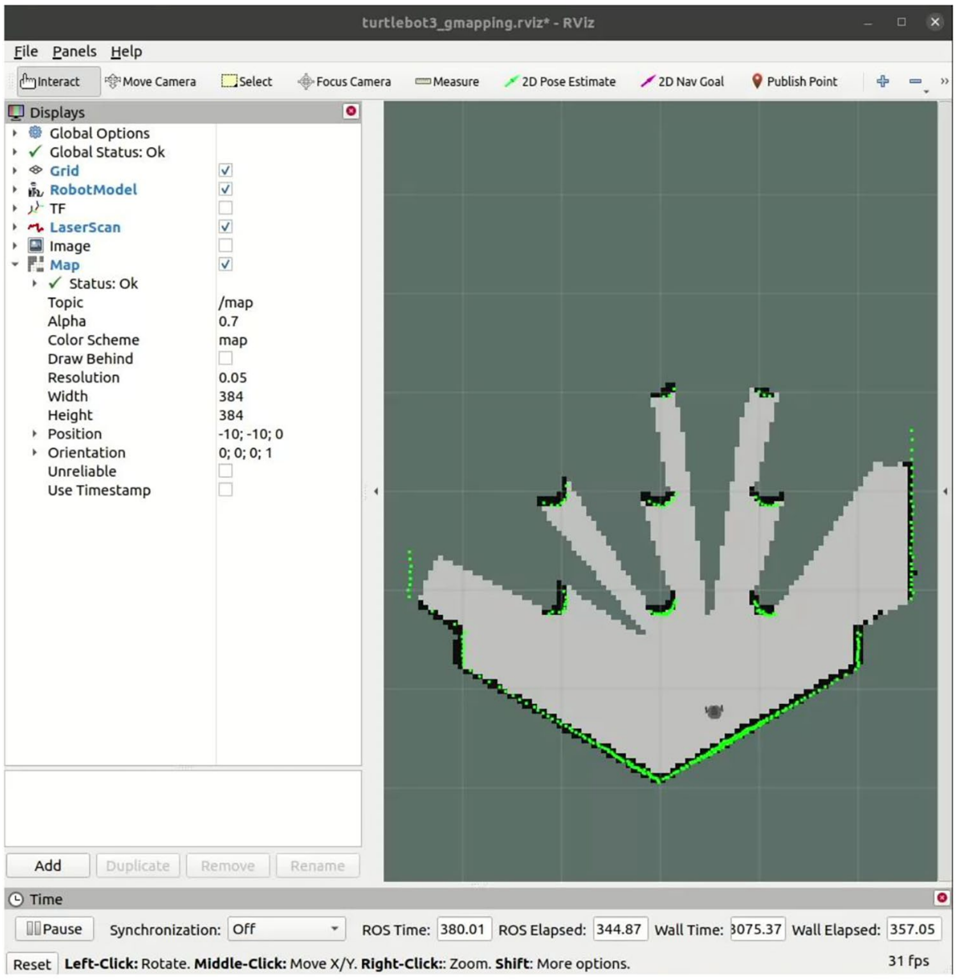
In the presented simulation results (Figs. 5, 6, 7, 8, 9 and 10), the SLAM algorithm was executed a total of five times to generate the robot's path. The iterations include the following stages: initial position allocation, moving to the mid-position, holding the mid-position allocation, moving to the final position, and holding the final position allocation. Each run contributed to the continuous updating of the SLAM-generated map, resulting in a comprehensive representation of the environment. This iterative approach aimed to enhance the accuracy and reliability of the SLAM implementation, capturing variations in the robot's movement and environment mapping at different stages of the experiment.



Property	Value
name	ground_plane
is_static	✓ True
self_collide	False
enable_wind	False
pose	
link	ground_plane::link

```
currently:	linear vel 0.0	 angular vel -0.4
currently:	linear vel 0.0	 angular vel 0.0
```

Fig. 5 Starting point of Turtle Bot



```
Global Options
Global Status: Ok
Grid                 ☑
RobotModel           ☑
TF                   ☐
LaserScan            ☑
Image                ☐
Map                  ☑
  Status: Ok
  Topic              /map
  Alpha              0.7
  Color Scheme       map
  Draw Behind        ☐
  Resolution         0.05
  Width              384
  Height             384
  Position           -10; -10; 0
  Orientation        0; 0; 0; 1
  Unreliable         ☐
  Use Timestamp      ☐

Synchronization: Off   ROS Time: 380.01   ROS Elapsed: 344.87   Wall Time: 3075.37   Wall Elapsed: 357.05

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.   31 fps
```

Fig. 6 Starting point map visualization

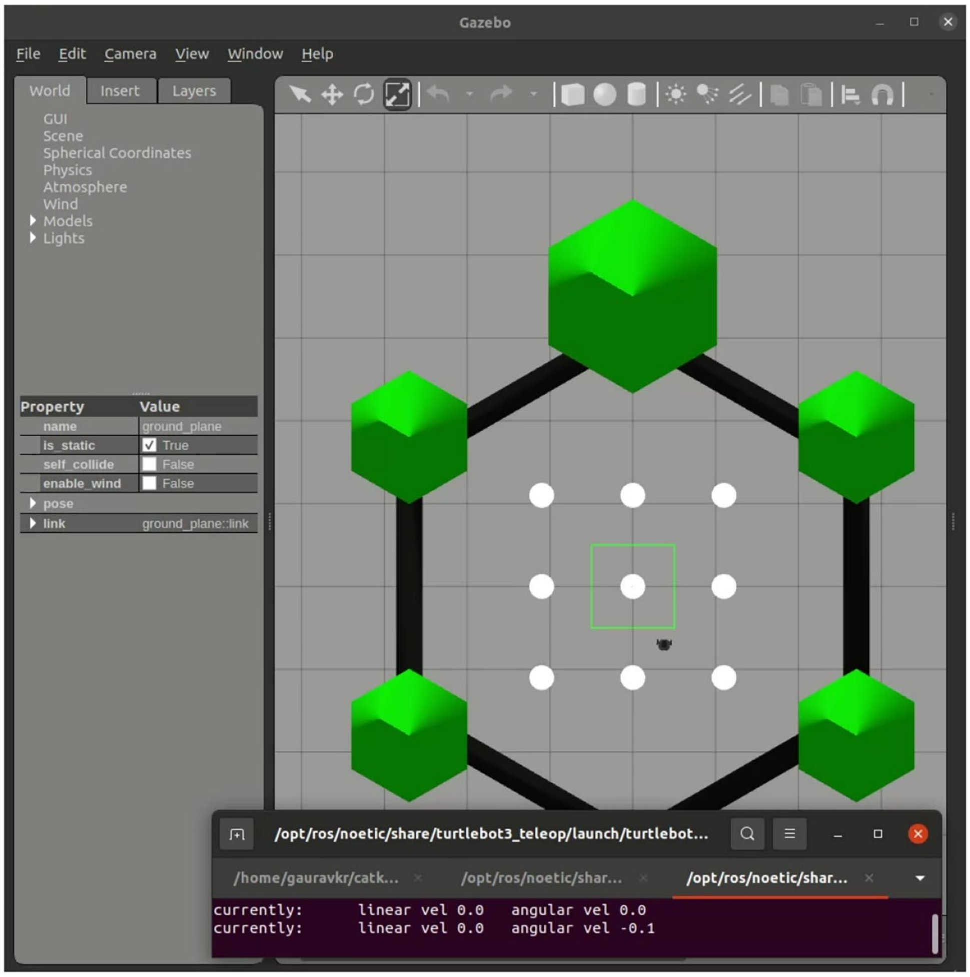
5 Conclusion

In this paper, we used ROS to analyze and successfully implement SLAM. TurtleBot3 was used to test and implement autonomous navigation in a simulated environment built with the aid of Gazebo. Turtle Bot successfully generated the map and made it to the designated location, as demonstrated above. The existing work can increase navigational precision through better software development and proper analysis. The entire hardware implementation and calibration for improved precision will be the main topics of future efforts.

This research addresses a critical need for advancing robotic capabilities through the implementation of Simultaneous Localization and Mapping (SLAM) on the TurtleBot robot within the Robot Operating System (ROS) framework. Motivated by the increasing demand for versatile and cost-effective robotic solutions, the study focuses on augmenting the capabilities of the TurtleBot. The integration involves a meticulous combination of hardware components, including laser and odometry sensors, and software tools such as the Gmapping algorithm for mapping. The research is propelled by the overarching goal of enabling the TurtleBot to autonomously navigate and explore unknown environments, crucial for applications ranging from search and rescue missions to autonomous transportation. Through this work, the authors seek to bridge existing research gaps, specifically in the realm of SLAM implementation with teleoperation on the TurtleBot. The study not only explores challenges but also looks ahead to potential future enhancements, emphasizing the motivation to contribute valuable insights to the field of robotics. Simulation results, detailing successful SLAM execution through teleoperation, underscore the practical significance of this research.

In future work, enhancements to the SLAM implementation for the TurtleBot could include the integration of additional sensors, such as cameras or depth sensors, to improve mapping accuracy and environmental perception. Furthermore, exploring advanced SLAM algorithms and machine learning techniques could contribute to more robust navigation in complex and dynamic environments. Additionally, investigating methods to optimize computational efficiency

Fig. 7 Mid-position of Turtle Bot



Property	Value
name	ground_plane
is_static	✓ True
self_collide	False
enable_wind	False
pose	
link	ground_plane::link

```
currently:      linear vel 0.0    angular vel 0.0
currently:      linear vel 0.0    angular vel -0.1
```

Table 1 Summary of SLAM implementation

Simulation Position	Map Coverage (%)
Starting Position	< 50%
Mid-Position	> 50%
Ending Position	100%

Table 2 SLAM algorithm performance metrics

Metric	Value or Description
Mapping Accuracy	High Accuracy
Localization Error	Low Error
Computational Time per Frame	31 FPS (Frames Per Second)

Table 3 SLAM map quality metrics

Quality Metric	Value or Description
Feature Sharpness	High
Consistency of Features	Good
Noise Level	Low

Table 4 Elapsed time for SLAM implementation at different positions

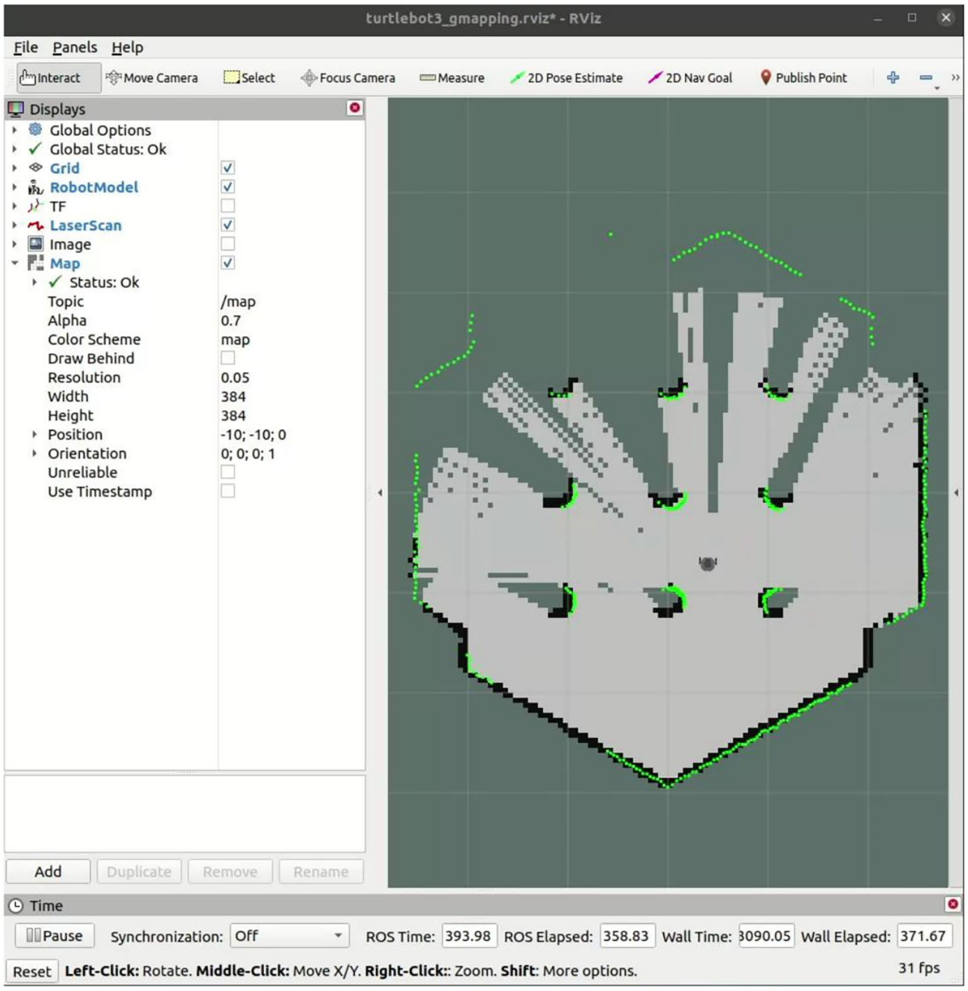
Simulation Position	Elapsed Time (seconds)
Starting Position	344.87 s
Mid Position	353.83 s
End Position	401.00 s

Table 5 Time difference between positions for SLAM implementation

Position Pair	Time Difference (seconds)
Starting to Mid	08.96 s
Mid to End	47.17 s
Starting to End	56.13 s

and real-time processing can enhance the overall performance of the robot. These potential enhancements aim to push the boundaries of the TurtleBot's capabilities, making it a more versatile and adaptable tool for a wide range of applications in the field of robotics.

Overall, the implementation of SLAM on the TurtleBot with the help of a teleoperation node has demonstrated the potential of robotics to solve complex problems and automate tasks in a variety of industries. As technology advances, we can expect to see even more sophisticated robotics systems capable of performing increasingly complex tasks with greater accuracy and efficiency.



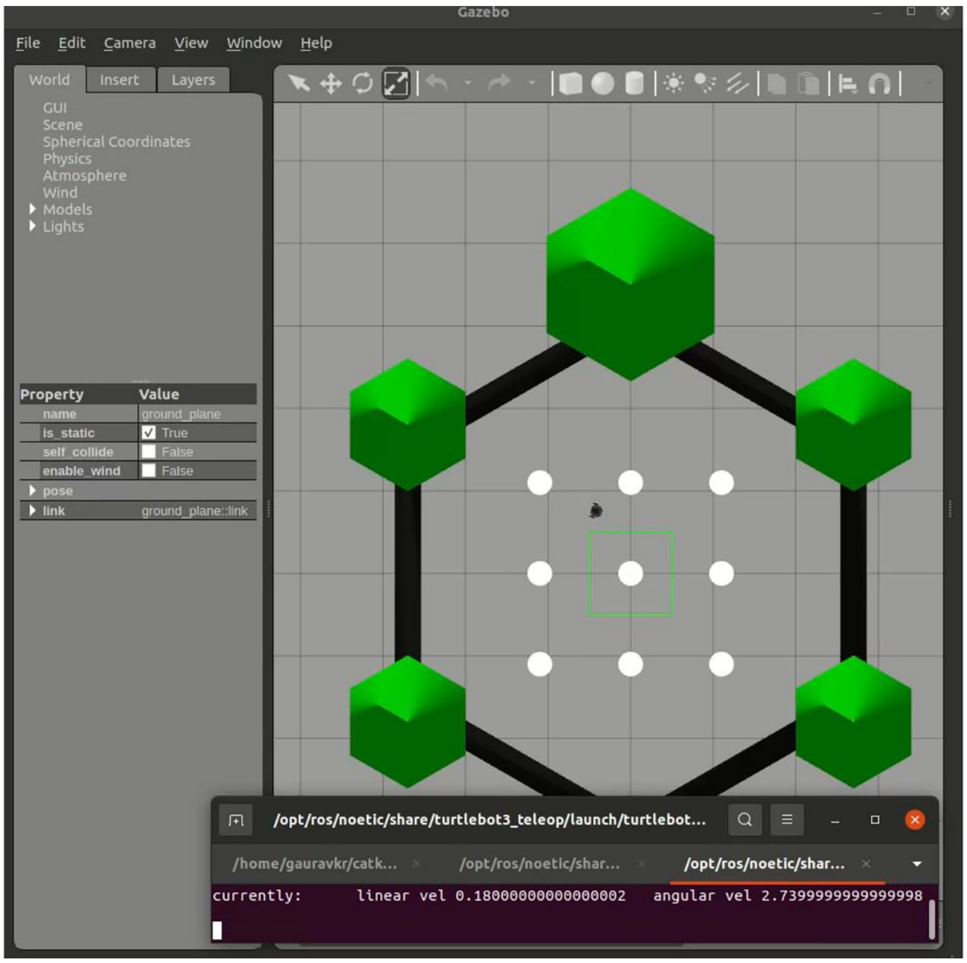
Global Options	
✓ Global Status: Ok	
Grid	✓
RobotModel	✓
TF	☐
LaserScan	✓
Image	☐
Map	✓
✓ Status: Ok	
Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.05
Width	384
Height	384
Position	-10; -10; 0
Orientation	0; 0; 0; 1
Unreliable	☐
Use Timestamp	☐

Synchronization: Off ROS Time: 393.98 ROS Elapsed: 358.83 Wall Time: 3090.05 Wall Elapsed: 371.67

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click::** Zoom. **Shift**: More options. 31 fps

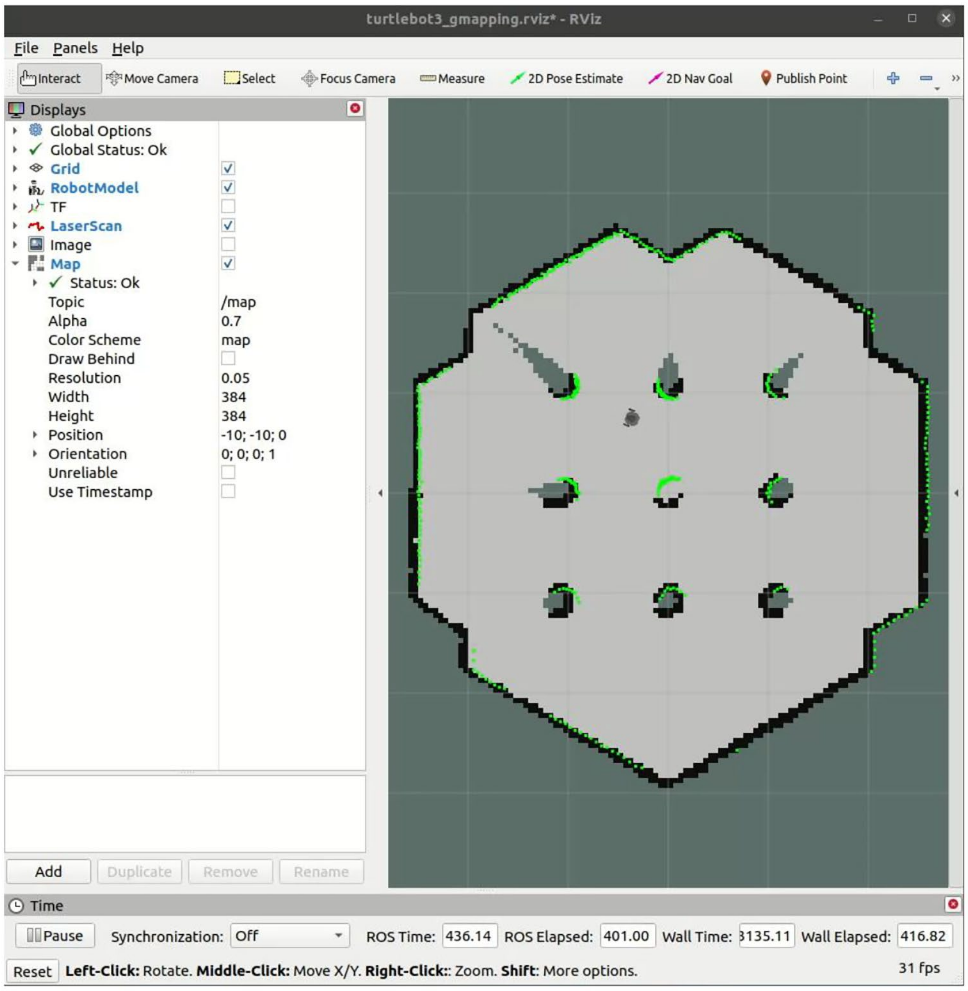
Fig. 8 Mid-position map visualization

Fig. 9 Ending point of Turtle Bot



Property	Value
name	ground_plane
is_static	✓ True
self_collide	False
enable_wind	False
pose	
link	ground_plane::link

```
currently:	linear vel 0.18000000000000002	 angular vel 2.7399999999999998
```



```
Global Options
Global Status: Ok
Grid                ✓
RobotModel          ✓
TF
LaserScan           ✓
Image
Map                 ✓
  Status: Ok
  Topic             /map
  Alpha             0.7
  Color Scheme      map
  Draw Behind
  Resolution        0.05
  Width             384
  Height            384
  Position          -10; -10; 0
  Orientation       0; 0; 0; 1
  Unreliable
  Use Timestamp
```

```
Synchronization: Off   ROS Time: 436.14   ROS Elapsed: 401.00   Wall Time: 3135.11   Wall Elapsed: 416.82
Reset  Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.   31 fps
```

Fig. 10 Ending point map visualization

Abbreviations

SLAM	Simultaneous localization and mapping
ROS	Robot operating system
Rviz	ROS visualization
EKF	Extended Kalman Filter
DFT	Dynamic field theory
LiDAR	Light detection and ranging

Acknowledgements None

Funding This research is not funded by any sponsoring agency.

Code availability For image processing, the code was written in the Python language and implemented in the ROS. This code is used to produce the results in this article and can be obtained upon request from the corresponding authors.

Data availability All data and materials used to produce the results in this article can be obtained upon request from the corresponding authors.

Declarations

Consent to publish The authors declare that all authors agree to sign the transfer of copyright for the publisher to publish this article upon acceptance.

Competing interests The authors declare that there are no conflicts of interest.

References

1. Housein, A.A., Xingyu, G.: Simultaneous Localization and Mapping using differential drive mobile robot under ROS. In Journal of physics: conference series (Vol. 1820, No. 1, p. 012015). IOP Publishing. (2021), March
2. Kamarudin, K., Mamduh, S.M., Shakaff, A.Y.M., Zakaria, A.: Performance analysis of the microsoft kinect sensor for 2D simultaneous localization and mapping (SLAM) techniques. sensors, 14(12), 23365–23387. (2014)
3. Huang, P., Zeng, L., Chen, X., Luo, K., Zhou, Z., Yu, S.: Edge robotics: Edge-computing-accelerated multirobot simultaneous localization and mapping. IEEE Internet Things J. **9**(15), 14087–14102 (2022)
4. Thale, S.P., Prabhu, M.M., Thakur, P.V., Kadam, P.: ROS based SLAM implementation for Autonomous navigation using

Turtlebot. In ITM Web of conferences (Vol. 32, p. 01011). EDP Sciences. (2020)

5. Omara, H.I.M.A., Sahari, K.S.M.: Indoor mapping using kinect and ROS. In 2015 International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR) (pp. 110–116). IEEE. (2015), August
6. Macias, L.R., Orozco-Rosas, U., Picos, K.: Simultaneous localization and mapping using an RGB-D camera for autonomous mobile robot navigation. In: Optics and Photonics for Information Processing XV, vol. 11841, pp. 119–129. SPIE (2021, August)
7. Gao, L.F., Gai, Y.X., Fu, S.: Simultaneous localization and mapping for autonomous mobile robots using binocular stereo vision system. In 2007 International Conference on Mechatronics and Automation (pp. 326–330). IEEE. (2007), August
8. Ibáñez, A.L., Qiu, R., Li, D.: An implementation of SLAM using ROS and Arduino. In 2017 IEEE International Conference on Manipulation, Manufacturing and Measurement on the Nanoscale (3 M-NANO) (pp. 1–6). IEEE. (2017), August
9. Chen, H., Huang, H., Qin, Y., Li, Y., Liu, Y.: Vision and laser fused SLAM in indoor environments with multi-robot system. Assembly Autom. **39**(2), 297–307 (2019)
10. Ibáñez, A.L., Qiu, R., Li, D.: A simple, cost-effective and practical implementation of SLAM Using ROS and Arduino. In 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) (pp. 835–840). IEEE. (2017), June
11. Hasan, H.M., Mohammed, T.H.: Implementation of Mobile Robot's Navigation using SLAM based on Cloud Computing. Eng. Technol. J., **35** (2017). (6 Part A).
12. Reynolds, S., Fan, D., Taha, T.M., DeMange, A., Jenkins, T.: An Implementation of Simultaneous Localization and Mapping Using Dynamic Field Theory. In NAECON 2021-IEEE National Aerospace and Electronics Conference (pp. 80–83). IEEE. (2021), August
13. Putra, I.A., Prajitno, P.: December). Parameter tuning of g-mapping slam (simultaneous localization and mapping) on mobile robot with laser-range finder 360 sensor. In: 2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), pp. 148–153. IEEE (2019)
14. Gao, X., Zhang, T.: Robust RGB-D simultaneous localization and mapping using planar point features. Robot. Auton. Syst. **72**, 1–14 (2015)
15. Durdu, A., Korkmaz, M.: Autonomously simultaneous localization and mapping based on line tracking in a factory-like environment. Adv. Electr. Electron. Eng. **17**(1), 45–53 (2019)
16. Ahmed, H.A., Jang, J.W.: Design of cloud based indoor autonomous navigation with turtlebot3. In International Conference on Future Information & Communication Engineering (Vol. 10, No. 1, pp. 118–122). (2018), June
17. Achour, A., Al-Assaad, H., Dupuis, Y., Zaher, E., M: Collaborative Mobile Robotics for Semantic Mapping: A Survey. Appl. Sci. **12**(20), 10316 (2022)
18. Hercik, R., Byrtus, R., Jaros, R., Koziorek, J.: Implementation of autonomous mobile robot in smartfactory. Appl. Sci. **12**(17), 8912 (2022)