Autonomous Navigation of a Mobile Robot on Robot Operating System Using Graphical User Interface

Henok Tegegn Warku
Department of Electronic Engineering, Interdisciplinary Program in IT-Bio Convergence Systems,
Chosun University
Gwangju 61452, South Korea
heni1032.tegegn@gmail.com

Nak Yong Ko*
Department of Electronic Engineering, Interdisciplinary Program in IT-Bio Convergence Systems,
Chosun University
Gwangju 61452, South Korea
nyko@chosun.ac.kr

Gyeongsub Song
Department of Control and Instrumentation Engineering,
Chosun University
Gwangju 61452, South Korea
sgs0369@naver.com

Da Bin Jeong
Department of Electronic Engineering, Interdisciplinary Program in IT-Bio Convergence Systems,
Chosun University
Gwangju 61452, South Korea
jjdabin@naver.com

Boeun Lee
Department of Electronic Engineering, Interdisciplinary Program in IT-Bio Convergence Systems,
Chosun University
Gwangju 61452, South Korea
dlqhdms9503@naver.com

Tegen Eyasu Derbew
Department of Electronic Engineering, Interdisciplinary Program in IT-Bio Convergence Systems,
Chosun University
Gwangju 61452, South Korea
eyasuderbew2008@gmail.com

Suk-Seung Hwang
Department of Electronic Engineering, Interdisciplinary Program in IT-Bio Convergence Systems,
Chosun University
Gwangju 61452, South Korea
hwangss@chosun.ac.kr

2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS&ISIS) | 978-1-6654-9924-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SCISISIS55246.2022.10001940

***Abstract*—Many applications of mobile robotics necessitate the safe execution of a collision-free motion to a defined place. Real-time obstacle avoidance strategies enable reactive motion in dynamic and unpredictable situations, whereas planning approaches are best suited for achieving a goal position in known static environments. A ROS-based approach is presented to address the challenge of robot SLAM, which has been used in real-time applications to map the environment, localize the robot within the environment, plan paths, and avoid obstacles. It is demonstrated that all navigation modules can coexist and work together to reach the destination without colliding with static and dynamic obstacles. This paper provides a platform for guiding a mobile robot in a real-world environment autonomously for different waypoints while avoiding static and dynamic obstacles using the Graphical User Interface (GUI). The developed GUI-based navigation simplifies the complex ROS navigation system control to an easy user-friendly interface so that anyone can control the mobile robot platform. Our results were validated at Chosun University in South Korea, through the experimental test in an indoor environment.**

Keywords—navigation, Localization, Mobile Robot, Robot Operating System (ROS), Graphical User Interface.

I. Introduction

Mobile robotics has recently become more popular due to technological advancement, the availability of many robot platforms, and robotic system architectures. Numerous autonomous robots have been developed and used for different application areas. Some of them are space and ocean exploration, underground mining, underwater exploration, manufacturing industries, and an autonomous self-driving car that can operate alongside pedestrians and cars driven by humans.

Simultaneous Localization and Mapping (SLAM) is commonly used to create metric maps since it can build a map of the environment whilst localizing the robot [1]. SLAM can be categorized according to different sensors used for the collection of information into vision-based and laser-based SLAM. SLAM based on the laser is mostly used to generate occupied grid maps. Representative algorithms include Hector SLAM [2], Cartographer [3], GMapping [4], and Karto SLAM [5].

With the rapid development of the use of mobile robots, the ability of mapping and localization is crucial for autonomous navigation through the environment. Without the knowledge of the current pose and the map of the environment, the robot cannot make its own decisions and actions. To achieve autonomous navigation in indoor environments, mobile robots must be able to obtain information from the environment using range sensors (e.g., laser sensor, 3D sensor, ultrasonic sensor) to construct a map of their environment and determine their location.

This paper addresses the experimental implementation of map-based autonomous navigation for a mobile robot in an indoor environment. We also integrated a Graphical User Interface (GUI) with the mobile robot to implement the waypoint's autonomous navigation of a mobile robot in an indoor environment. The GUI can be used with any ROS-based robot and allows anyone to handle the navigation system without having to write complex commands.

II. System Architecture

A. Robot Operating System

It is an open-source platform for programming robots. It's a set of tools, libraries, and conventions that make the work of

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1I1A3A01057691).

building flexible and reliable robot behavior across a wide range of robotics platforms easier. It is used in both research and commercial applications and provides robot programming capabilities such as high-level programming language support and tools, message passing interface between processes, availability of third-party libraries, community support, extensive tools and simulators, and so forth [6,7].

In order to represent the three-dimensional model of the robot in ROS, the Unified Robot Description Format (URDF) is used. It is XML file that describes the model of the robot and it contains the robot-specific information that nodes require to do their tasks. URDF file is constructed in such a way that each link is connected to joints, each robot link is a child of a parent link, and joints are specified by their offset from the parent link's reference frame and rotation axis [8].

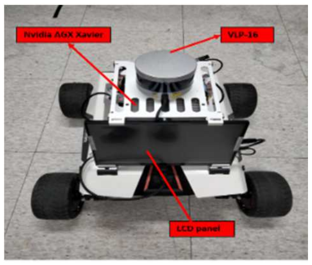
B. *ROS Simulation Environment*

1) Gazebo: It is a 3D dynamic simulator used to simulate a robot accurately and efficiently by developing complex indoor and outdoor environments. It provides a more accurate physical simulation of the system, as well as a wider range of sensors with user and program interfaces. Multiple physics engines, a rich library of robot models and environments, and convenient programmatic and graphical interfaces are some of the basic features of Gazebo [9].

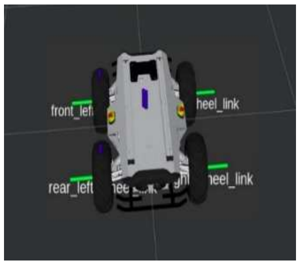
2) Rviz: ROS Visualization (Rviz) is an impressive 3D visualization tool for ROS that enables users to view or visualize the simulated robot model; that is what is the robot doing, seeing, and heading. It also lets users see 3D sensor data from lasers, stereo cameras, Kinect, and other 3D devices as point clouds or depth images [10].

C. *Robot Platform*

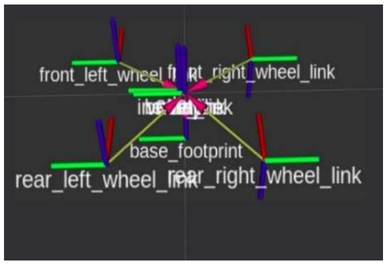
The mobile robot used for the experiment is a four-wheeled differential robot called "Scout Mini". The robot platform is shown in Fig. 1 and the hardware specification is listed in TABLE I. Fig. 1 (a) depicts the Scout Mini mobile robot with the USB-HUB expansion interface, USB to CAN module, Intel RealSense D435, LCD screen, and Velodyne VLP-16 installed at the top layer. Fig. 2 shows the external connection topology of the Scout Mini. The external topology connection of Nvidia AGX Xavier as the core computing unit of Xavier's network port is connected to the router's network port, which is convenient for remote desktop connection, access, and debugging and easy to expand to other network devices.



(a)



(b)



(c)

Fig. 1. The hardware platform of the mobile robot is used for experimentation. (a) Robot Hardware Setup, (b) Mobile robot on Rviz, and (c) Frame transformation on Rviz.

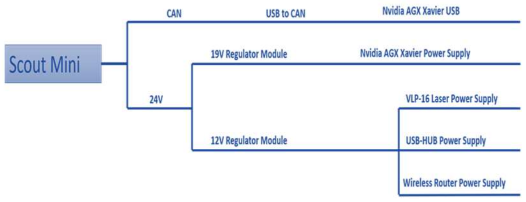


Fig. 2. External topology connection of Scout Mini

TABLE I. HARDWARE SPECIFICATION

Dimension	627 X 550 X 252 mm
mode of operation	4-wheel Differential-Drive Model
Wheelbase	452mm
Minimum ground clearance	107mm
Battery	24V / 15Ah
Maximum speed	10km/h
Communication environment	Standard CAN / RS232

D. *Velodyne VLP-16 Laser Sensor*

It is the most advanced and smallest lidar sensor. The VLP-16 is less expensive than comparable sensors while retaining some of the basic features such as real-time, 360^0 field of view, 3D obstacle detection, and reflectance measurements with calibration. The VLP-16 has a measuring range of up to 100 meters, a low power consumption (8W), is lightweight (830g), has a small size (103mm x 72mm), and has dual return capability, making it ideal for man-machine mounts and other mobile devices. It is used to make a map of the environment and for navigating the robot from one point to the goal point.

III. STRUCTURE OF ROS NAVIGATION

ROS is an open-source platform that could be considered a middleware that provides high-level abstraction between low-level hardware and drivers and high-level software APIs. The problem of navigation is fundamental in robotics and other important technologies. For the sake of making the mobile robot navigate autonomously, it must first understand where it is, where it is going, and how it will get there.

ROS navigation stack is used to accomplish mobile robot navigation from one point to a target point. The ROS navigation stack is a collection of ROS nodes and algorithms that are used to autonomously move a robot from one location to another while avoiding any obstacles in its path. The followings are the input for the navigation stack shown in Fig. 3 to send commands to our robot to move to its desired goal:

- Current pose: The position and orientation of the robot.
- Goal pose: The robot's desired location for achieving its goal.
- Odometry data: Using the motion sensor's data for estimating the change in pose over time. These sensors include wheel encoders, IMU, and GPS.
- Laser sensor: Data from the laser (lidar) sensor is needed to recognize objects in the environment.

Taking the above as input, the navigation stack in exchange will output the velocity commands that are needed and send them to the mobile base for moving the robot to the designated target position.

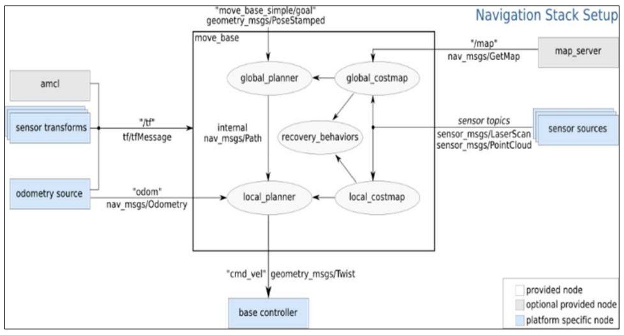


Fig. 3. ROS Navigation stack structure.

A. *Specific Objectives*

- Experimental indoor mapping of an environment.
- Localizing the mobile robot inside the constructed map.
- Path planning of the robot after the robot is localized in the environment. Path planning includes local and global path planning.
- Obstacle avoidance. This includes static and dynamic obstacle avoidance.
- Navigation of a mobile robot from one point to another point.
- The development of a Graphical User Interface (GUI) to control the robot's autonomous navigation in the environment.

B. *SLAM-gmapping*

SLAM is the process of creating the environment's map while keeping track of the position of the robot on the map. This problem is basically what mapping is solving. For reducing the common depletion problem that is associated with the Rao-Blackwellized particle filter, an adaptive resampling technique is employed by the gmapping package [11, 12, 13]. To create a map, a two-dimensional occupancy grid method is used by the gmapping package. An obstacle is inserted into a cell or the cell is cleared using sensor stream data. Clearing a cell for each successful laser-scan sample includes ray-tracing through a grid.

The gmapping ROS package [14] is used to create a 2D map from the robot's laser and pose data as it moves around the environment. It includes a slam gmapping node, which reads data from the laser and transforms it from the laser source to the base link, broadcasting the transform from the "map" to "Odom" frames, resulting in an occupancy grid map (OGM). To obtain the data needed to build a map, the slam_gmapping node subscribes to the laser topic (*/scan* topic for hardware implementation) and an extensive transform topic (*/tf*). The map_server package, which is part of the ROS navigation stack, saves the OGM map output and provides the *map_saver* node that allows access to mapping data from the ROS service, and saves it into a file.

C. *Adaptive Monte Carlo Localization (AMCL)*

Determining the robot's pose inside a mapped environment is known as localization. Robot localization occurs when a robot moves around a map and needs to know its position and orientation within the map using sensor readings. Monte Carlo Localization (MCL) is the most popular algorithm in robotics, which is known as Particle Filter Localization (PFL) because it uses particles for locating the robot. After the deployment of MCL, the robot would be navigating through its known map and collect sensory information by the use of a laser sensor. Then, the MCL will use these sensor data for measuring and keeping track of the pose of the robot [15].

The ROS package AMCL [16] includes the amcl node, which employs the MCL algorithm to track the location of the movement of the robot in a two-dimensional space. This node subscribes to laser data, the laser-based map, and the robot's transformation, and then publishes the estimated pose on the map as shown in Fig. 4.

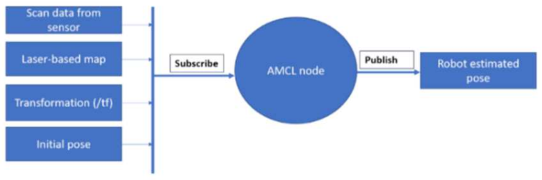


Fig. 4. AMCL node data subscription and publication.

D. *Move-Base Package*

Move base is the key component of the ROS navigation stack that connects all planner and controller behaviors. The move base package includes the *move_base* node, which moves the robot from its current position to the target point. It is a *SimpleActionServer* implementation that accepts a target pose with *geometry_msgs/PoseStamped* message type so that *SimpleActionClient* can send a target point to this node. One of the topics provided by the move base action server is *move_base/goal*, which is the navigation stack's input that will be used to provide the goal pose [17]. Sending a goal to the move base node as shown in Fig. 3 activated some other processes such as local and global planner that involve some other nodes that result in moving the robot to the target pose.

IV. Experimental Results

The experiments were carried out in our laboratory room, and we used the Rviz platform to analyze the internal state of the robot and compare it to the actual robot status. The indoor environment used for the experiment has an area of approximately 34m^2 as shown in Fig. 5.

A. Mapping of the Environment

The gmapping algorithm's experimental performance was evaluated in the experimental scenario of Fig. 5. Fig. 6 depicts the mapping result obtained from the experiments while the Scout Mini mobile robot is teleoperated around the environment, and the final mapping outputs, as well as the real-time data visualization from the real sense camera, which is used to compare the obstacles detected by the grid map with the robot's actual position and attitude. The experimental results showed the creation of a compatible map of the actual environment with the same dimensions and features.

During the experiment, the gmapping algorithm uses a static transformation between the laser's base link frame (with link name "Velodyne") and the robot base link frame (with link name of base_link) to generate three-dimensional measurement data with a topic of */velodyne_points*, which is changed to a 2D laser scan data using the ROS package called *pointcloud_to_laserscan* to publish a */scan* topic with

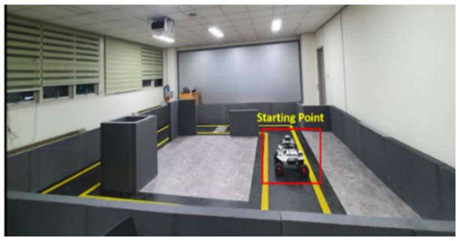


Fig. 5. Experimental indoor environment for navigation of a mobile robot.

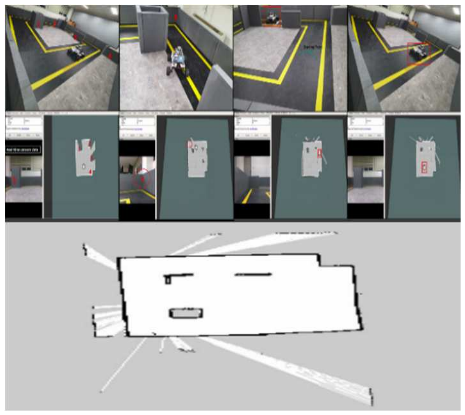


Fig. 6. Mapping of the actual environment.

a message of *sensor_msgs/msg/LaserScan* from the point cloud.

The data from the wheel encoder is used to provide odometry data to the *slam_gmapping* node with the topic */Odom*, which is then linked with the lidar topic to publish the */map* topic to generate the map of the environment, as shown in Fig. 7. Additionally, several parameters of the slam_gmapping node have been optimized, and some of the major parameters and their values for our experimental case. The created map is used by localization and path-planning algorithms to complete real-time autonomous robot navigation.

B. Localization of a Mobile Robot in the Map of the Environment

The map generated in the previous section from actual environments is used to localize the robot using the AMCL method as shown in Fig. 8. The yellow color arrows show the particles used to estimate the robot's pose.

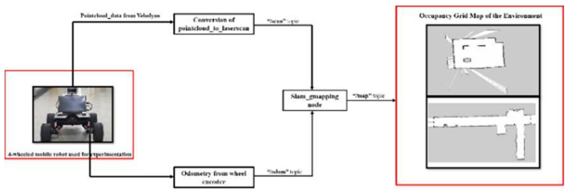
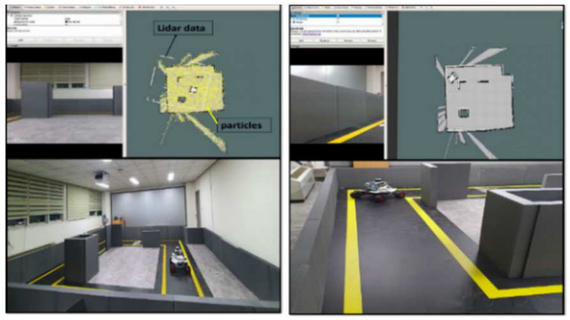

Pointcloud_data from Velodyne
Conversion of pointcloud_to_laserscan
"/scan" topic
Slam_gmapping node
"/map" topic
Occupancy Grid Map of the Environment
4-wheeled mobile robot used for experimentation
Odometry from wheel encoder
"/odom" topic

Fig. 7. Block diagram representation of the mapping algorithm of the experiment.



(a) (b)

Fig. 8. Global localization of a mobile robot after calling AMCL's *global localization service* (a), and the convergence of particles after the robot translates 4.15 meters.

The operation of particle filters includes:

- Particle initializations: Each particle is a belief of where the robot can be located, and each particle has a position (x,y) and orientation to keep the detail of the particle's position guess. Each particle also has an associated weight.
- Prediction step: When the robot moves, all the particles move with the same motion. The prediction step adds multivariate Gaussian noise to each particle motion, causing particle dispersion across the map.
- Correction step: Sensor measurements are processed in this step. It corrects the state for the next filter iteration based on sensor measurements. It processes incoming measurements, compares them to the measurements of the particles on the map, and then prioritizes particles with the lowest error between measurements and the map.
- Resampling step: After the correct step, the weights of the particles have changed. Resampling is the process of replacing the particles with small weights by others with high probability poses.

During the experiment, the particle filters took roughly 10 to 15 seconds to converge to the actual pose.

C. Development of a Graphical User Interface (GUI)

The Graphical User Interface (GUI) has been designed to control the mobile robot's navigation in both simulation and hardware implementation. The PyQt toolkit is used to create the robot's graphical user interface. PyQt is a Python binding for the Qt cross-platform widget toolkit and application framework [18].

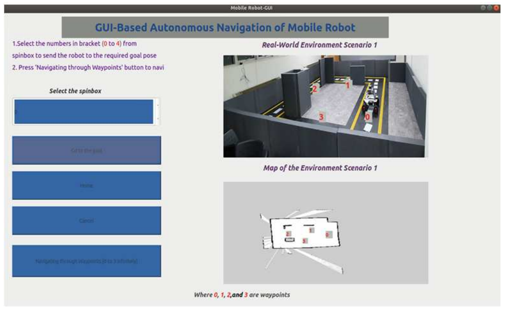
The Qt designer is used to create and insert controls into the Qt GUI. The Qt graphical user interface (GUI) is an XML file that has information about its controls and components. To control the robot with Qt GUI, we must first create the platform in Qt designer. The Qt designer provides a variety of options and tools to make the user interface simple and convenient.

To use the tool, we must first create an empty widget by selecting the *Widget* option from the *New Forum* window lists. The fundamental building blocks of the Qt graphical user interface are Qt widgets. Using this tool, we created an application for a mobile robot's autonomous navigation in a mapped environment via a different waypoint based on the pose that we want the robot to take.

D. Autonomous Navigation of the robot using GUI

Once the map of the environment is generated and the robot can know its pose in the environment relative to the global map frame, path-planning, and obstacle avoidance is carried out. In the experiment section, we have tested the navigation performance for the static and dynamic obstacles in the environment.

The main goal of developing a GUI is easy to control a mobile robot to send it to the desired location and cancel the operation at any time without having to know complex commands to start and stop the robot. PyQt, ROS, and the Python interface are used to create the GUI. Fig. 9 depicts the developed graphical user interface platform for controlling the robot navigation system.

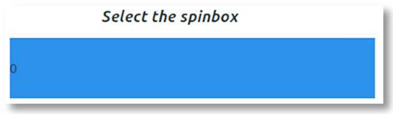

Mobile Robot-GUI
GUI-Based Autonomous Navigation of Mobile Robot
Real-World Environment Scenario 1
Select the spinbox
Map of the Environment Scenario 1
2
1
3
0
Where 0, 1, 2,and 3 are waypoints

1.Select the numbers in bracket (0 to 4) from spinbox to send the robot to the required goal pose
2. Press 'Navigating through Waypoints' button to navi

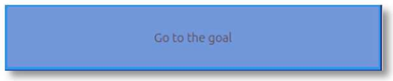
Fig. 9. The designed GUI for autonomously navigating a mobile robot through different waypoints.

The GUI buttons in Fig. 10 have the following features:

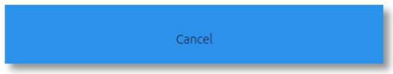
- SpinBox: This widget is used to insert the waypoint position numbers as shown in the environment map, ranging from 0 to 3. To navigate the robot to the target goal, we considered four pose values. We can select or insert any waypoint from the list of poses to send the robot to the goal position and attitude.



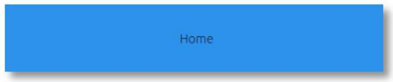
- Go to the goal: This button has a function called "***GO***" and it is used to send a command to the robot to the goal pose of the specified position based on the input from the spin box. By pressing the ***Go to the goal*** button, the position is sent to the navigation stack, where the robot plans its path and arrives at the desired destination.



- Cancel: This button has a" ***Cancel***" function that allows canceling of the robot's current operation. When clicking the *Cancel* button, the robot will stop moving to any point on the map.



- Home: This button has a function called "***Home***" to return the robot from any location in the environment to the initial position.



- Navigate through the waypoint: This button has a function called "***navigate through waypoints***" that allows the mobile robot to autonomously navigate through the poses shown in TABLE II in an ordered manner from 0 → 1 → 2 → 3 → 0 infinitely.

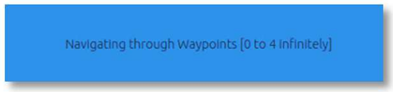
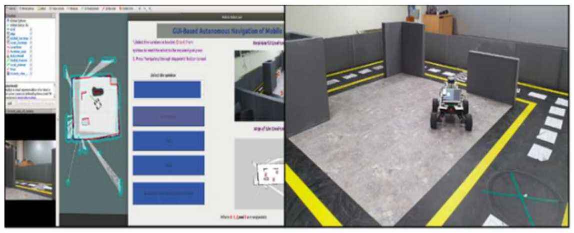


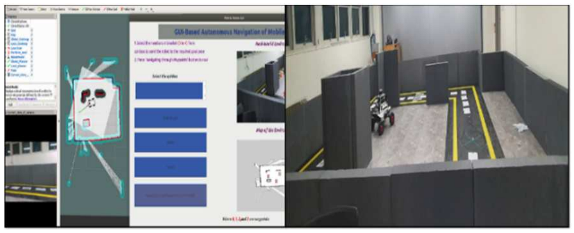
TABLE II. POSITION AND ATTITUDE QUANTITATIVE VALUES OF THE WAYPOINTS WHILE NAVIGATING AROUND THE ENVIRONMENT.

Assigned numbers for waypoint poses	**Absolute value of X-Position (meters)**	**Absolute value of Y-position (meters)**	**Attitude (rad)**
0	0.012	0.205	3.112
1	1.348	0.819	2.542
2	2.926	0.588	0.485
3	1.125	1.332	2.086

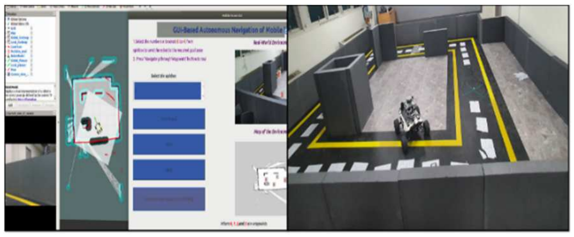
Fig. 10 shows the real-time autonomous navigation of the Scout Mini mobile robot through all waypoints while controlling via the GUI platform.



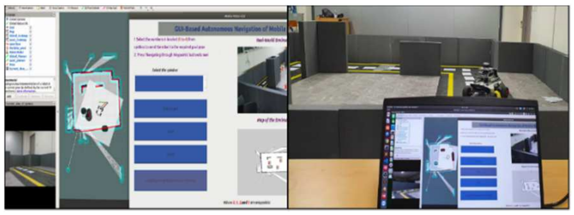
(a)



(b)



(c)



(d)

Fig. 10. GUI-based autonomous navigation of a mobile robot in the actual environment.

(a): Shows the mobile robot's real-time autonomous navigation to the first waypoint (1).

(b): Depicts the mobile robot's real-time autonomous navigation to the second waypoint (2).

(c): Illustrates the mobile robot's real-time autonomous navigation to the third waypoint (3).

(d): Shows the mobile robot's real-time autonomous navigation to the map's initial pose (0) using the GUI platform.

V. Conclusion

In this paper, we have done practical experiments to test and validate the autonomous navigation algorithm of a mobile robot using the ROS platform. The 16-channel Velodyne lidar data is used for mapping, and localization of the robot inside the environment using particle filters, path planning (global and local), and navigating through the environment from one location to the goal pose while avoiding static and dynamic obstacles in a two-dimensional environment. We developed a Graphical User Interface (GUI) based autonomous navigation of a mobile robot through different waypoints of the actual environment that controls the robot using various buttons, and infinite navigation of the mobile robot through different waypoints of the map, as well as an emergency button, to stop the robot at any time. From the experimental result, we observed that the mobile robot was able to reach its target by manually pressing the necessary buttons on the GUI. The performance of the navigation of the robot can be improved with careful optimization of localization and path planning parameters.

References

[1] C. Landsiedel, V. Rieser, M. Walter, and D. Wollherr, "A review of spatial reasoning and interaction for real-world robotics," Advanced Robotics, vol. 31, no. 5, pp. 222-242, 2017.

[2] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *2011 IEEE international symposium on safety, security, and rescue robotics*, 2011: IEEE, pp. 155-160.

[3] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE international conference on robotics and automation (ICRA)*, 2016: IEEE, pp. 1271-1278.

[4] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics,* vol. 23, no. 1, pp. 34-46, 2007.

[5] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2D mapping," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010: IEEE, pp. 22-29.

[6] L. Joseph, *Robot Operating System (ROS) for Absolute Beginners*. Springer, 2018.

[7] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O'Reilly Media, Inc.", 2015

[8] Playfish. "URDF." ROS wiki. http://wiki.ros.org/urdf.

[9] O. S. R. Foundation. "What is Gazebo?" Gazebo. https://gazebosim.org/tutorials?cat=guided_b&tut=guided_b1.

[10] H. Do Quang *et al.*, "Mapping and navigation with four-wheeled omnidirectional mobile robot based on robot operating system," in *2019 International Conference on Mechatronics, Robotics and Systems Engineering (MoRSE)*, 2019: IEEE, pp. 54-59.

[11] W. B. Sebastian Thrun, Dieter Fox, *Probabilistic Robotics*. ©Massachusetts Institute of Technology, 2006.

[12] R. Tang, X. Q. Chen, M. Hayes, and I. Palmer, "Development of a navigation system for semi-autonomous operation of wheelchairs," in *Proceedings of 2012 IEEE/ASME 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, 2012: IEEE, pp. 257-262.

[13] N. Kwak, I.-K. Kim, H.-C. Lee, and B.-H. Lee, "Analysis of resampling process for the particle depletion problem in FastSLAM," in *RO-MAN 2007-The 16th IEEE International Symposium on Robot and Human Interactive Communication*, 2007: IEEE, pp. 200-205.

[14] B. Gerkey. "gmapping." ROS Wiki. http://wiki.ros.org/gmapping.

[15] S. Thrun, "Particle Filters in Robotics," in *UAI*, 2002, vol. 2: Citeseer, pp. 511-518.

[16] B. P. Gerkey. "amcl." ROS Wiki. http://wiki.ros.org/amcl.

[17] K. Zheng, "Ros navigation tuning guide," in *Robot Operating System (ROS)*: Springer, 2021, pp. 197-226.

[18] J. M. Willman, "Getting Started with PyQt," in *Beginning PyQt*: Springer, 2022, pp. 1-11.