Constructing a 2D Occupancy Grid Map Using TurtleBot3

Babu Pallam (P2849288)
MSc. Artificial Intelligence
Submitted to: Dr. Aboozar Taherkhani
(Intelligent Mobile Robotics, De Montfort University)

January 21, 2025

Introduction

This report details the process of constructing a 2D occupancy grid map of an unknown environment using the TurtleBot3 and ROS (Robot Operating System). The implemented system utilizes sensor data from the TurtleBot3's LiDAR and odometry to accurately map the surrounding environment. This report describes breifly about some of the experiments carried out with my research to improve the accuracy of the map by filtering the noise in the data transmission from the robot, dynamic map expansion as the robot moves, applying multiple probabilities of grid update rather than binary update which is done in the basic implementation.

Methodology

The project involved creating a setup to map an unknown environment using a TurtleBot3 robot and ROS. First, the necessary ROS packages for TurtleBot3, mapping, and visualization were installed and configured. A simulation environment was launched, representing a 3D world with obstacles. The RViz visualization tool was used to display the map in real-time. A custom mapping package was created, containing two main components: the Map Class, which stores the occupancy grid as a 2D numpy array, and the Mapper Class, which processes sensor data. The robot's LiDAR provided laser scan data via the '/scan' topic, and its position and orientation were fetched from the '/odom' topic. These inputs were processed to identify obstacles and update the map grid.

In addition to mapping, a wandering behavior was implemented to autonomously explore the environment. The wandering package guided the robot to move forward in open spaces and turn when obstacles were detected. This ensured that the robot covered as much of the environment as possible, collecting detailed sensor data. All packages, including the mapping and wandering algorithms, were run simultaneously along with RViz to visualize the real-time map updates. By combining the wandering behavior with dynamic map updates, the robot was able to navigate and map the environment efficiently while generating visualizations in RViz.

Implementation and Experiments

Began with basic implementation, several tests have been done to improve the mapping process. In the initial version (**mapperv1 - Basic Mapping**), `/scan` and `/odom` were utilized to detect obstacles and update the map, with basic coordinate transformations and grid updates implemented. In the next version (**mapperv2 - Noise Filtering & Index Mapping**), noise filtering was added to eliminate abrupt changes in laser scan data, and the `get_index` method was introduced to convert global coordinates to grid indices for accurate updates. The, (**mapperv3 - Dynamic Map Expansion**) implemented to test with high resolution and dynamic resizing of the occupancy grid to handle larger exploration areas while preserving existing map data. Finally, rather than using binary update of the

grid (0 or 1) while obstacle detection, (**mapperv4 - Probabilistic Mapping**) introduced probabilistic grid updates to reflect sensor uncertainty, using clamped probabilities to represent obstacles and free spaces in the range of 0 to 1.

Results and Analysis

The experiments successfully generated 2D occupancy grid maps that improved in accuracy and robustness through different implementations. The produced maps of each tests can be seen in Figure 1. In the first version (**mapperv1**), basic mapping worked well and provided a graph with less noise, but resolution and noise were a problem. The second version (**mapperv2**) improved data quality by adding noise filtering, significantly reducing false positives. The third version (**mapperv3**) enabled dynamic map expansion, allowing the robot to explore larger areas without losing previously mapped data. Finally, the fourth version (**mapperv4**) introduced probabilistic updates, improving the accuracy of the map and better representing uncertain areas. All versions successfully visualized the maps in RViz, with real-time updates published to the `/map` topic.

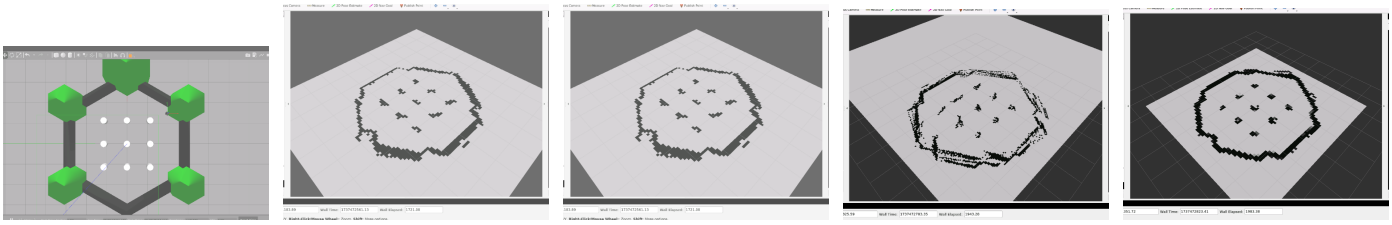

Figure 1: Maps generated by each version: (Left to Right) Original Map, mapperv1 - Basic Mapping, mapperv2 - Noise Filtering & Index Mapping, mapperv3 - Dynamic Map Expansion, mapperv4 - Probabilistic Mapping.

Conclusion and Future Directions

This project helps me to understand how the ROS works and how a robot behaves from independent units, such as sensors, movements, and others. This work demonstrates the development of a robust 2D mapping system for unknown environments using TurtleBot3. Through iterative improvements and extensions, the mapping system achieved high accuracy, scalability, and noise resilience. Future work could explore the integration of SLAM (Simultaneous Localization and Mapping) for improved localization and map consistency.

References

1. RViz User Guide.
2. Joseph, Lentin. *Learning Robotics using Python - Second Edition: Design, simulate, program, and prototype an autonomous mobile robot using ROS, OpenCV, PCL, and Python.* Packt Publishing, 2018.
3. ROS Navigation Documentation.
4. Zhang, A. "ROS Tutorial for Robotics: Introduction to ROS and Navigation,"
5. How to Use RViz and Other ROS Graphical Tools - YouTube.