Applications of ROS: Exploring Autonomous Navigation

Author: Josephine O.E. Odusanya

Student number: 9200353007

School of Automation and Electrical Engineering

Major: Robotics Engineering 201

For the course, Robotic Operation System, Third year, 2nd semester, 2023.

Introduction

ROS or Robotic Operating System is on open source software that can be used for the programming and simulation of robots.
This experiment is inspired by the immense potential and benefits that autonomous navigation offers in various domains, including robotics, self-driving vehicles, and industrial automation. Integrating ROS and autonomous algorithms/packages provides a powerful framework for intelligent decision-making and path planning, enabling robots and autonomous systems to navigate complex environments efficiently.

There is an increasing demand for autonomous systems that can operate seamlessly in dynamic and unpredictable environments. Traditional methods of navigation often rely on predefined paths or human intervention, limiting the adaptability and efficiency of autonomous systems. By harnessing the capabilities of ROS, we can develop a robust and versatile navigation system that can handle real-time decision-making, obstacle avoidance, and path planning to name a few.

The significance of this work stems from the broader implications of autonomous navigation in enhancing safety, efficiency, and productivity in various fields. In industries such as logistics, warehousing, and manufacturing, autonomous robots can optimise operations by navigating complex environments, avoiding collisions, and delivering goods with precision. Similarly, in the realm of self-driving vehicles, the integration of ROS can enable intelligent route planning, reducing traffic congestion, and improving road safety.

I aim to emphasise the accessibility and educational value of these technologies. To do this, I have highlighted the various dependencies that are important in this experiment.

Dependencies

- Ubuntu 20.04

- ROS Noetic
- Gazebo
- Turtlebot3 Packages

Ubuntu 20.04

AKA "Focal Fossa," is a long-term support (LTS) release of the Ubuntu operating system. It was officially released in April 2020. Ubuntu 20.04 features the GNOME desktop environment and offers several improvements in terms of performance, security, and software package updates. As an LTS release, it provides five years of support and regular updates, making it a reliable choice for both desktop and server environments.

ROS Noetic

ROS Noetic Ninjemys is the thirteenth ROS distribution release. It was released on May 23rd, 2020. It offers improved compatibility, stability, and other powerful features. With seamless integration of sensors, actuators, and

software modules, developers can create sophisticated robotic applications. Noetic revolutionises perception, navigation, and interaction, empowering robots in diverse environments. I chose to use Noetic as it is still under LTS (Long Term Support) and allows me to explore ROS 1 with very few limitations yet enhanced capabilities.

Gazebo

Gazebo is a widely-used open-source simulation tool designed for robotics and autonomous systems. It provides a virtual environment where developers can simulate and test their robotic applications before deploying them in the real world. It offers realistic physics engines, sensor simulation, and a wide range of pre-built models and environments, allowing users to accurately replicate real-world scenarios. It supports various robot platforms and can simulate complex behaviours and interactions. With its extensive set of APIs and plugins, it enables developers to create and customise simulations for research, development, and testing purposes.

Turtlebot3

TurtleBot3 is a small, affordable, programmable, ROS-based mobile robot for use in education, research, hobby, and product prototyping. The goal of TurtleBot3 is to dramatically reduce the size of the platform and lower the

price without having to sacrifice its functionality and quality, while at the same time offering expandability. The TurtleBot3 can be customised in various ways depending on how you reconstruct the mechanical parts and use optional parts such as the computer and sensor. In addition, TurtleBot3 is evolved with cost-effective and small-sized SBC that is suitable for robust embedded systems, 360 degree distance sensors and 3D printing technology. We use Waffle Pi in this experiment.

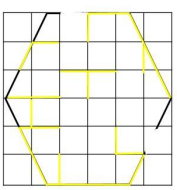
-https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
-https://github.com/ROBOTIS-GIT/turtlebot3.git
-https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
-https://github.com/ROBOTIS-GIT/turtlebot3_manipulation.git

SLAM

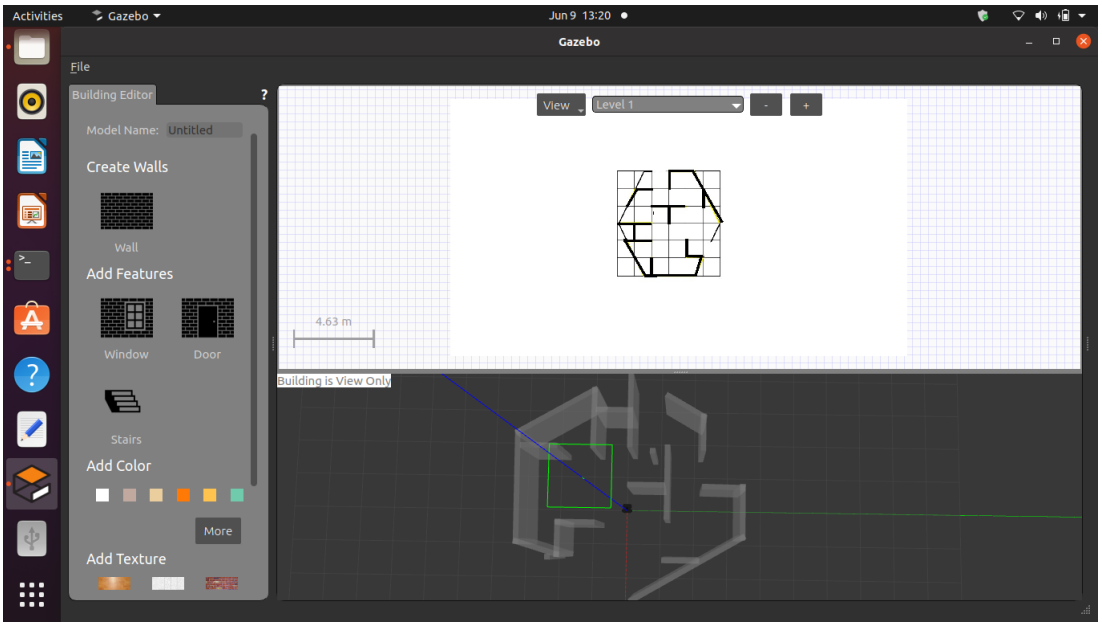
SLAM stands for Simultaneous Localization and Mapping, meaning creating and updating a map of the environment while also keeping track of where the robot is on this map.

Experiment (method):

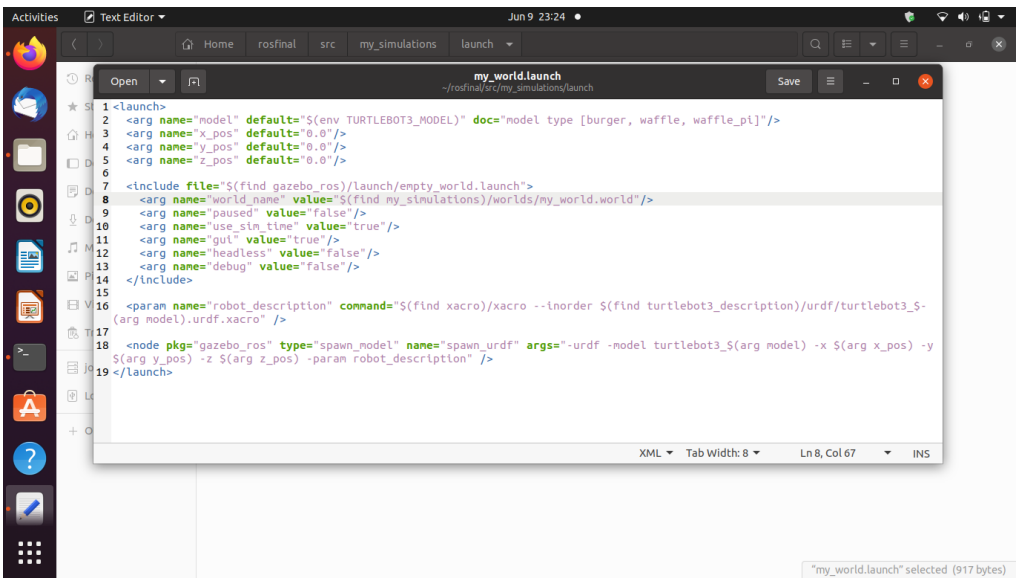
- Created a new workspace and labelled it ‘rosfinal’.
- Created a new package and labelled it ‘my_simulations’.
- Cloned the turtlebot3, turtlebot3_msgs and turtlebot3_simulations packages into my workspace.
- Add new folders ‘launch’ and ‘worlds’ to the my_simulations package. In the ‘launch’ folder, add a file and label it ‘empty_world.launch. Copy the contents of turtlebot3_empty_world.launch into the file. Then compile the packages.
- Launch the newly created launch file.
  $roslaunch my_simulations my_world.launch
- In the meanwhile, I used MS Powerpoint to create a custom maze. The consequent maze model was based on this image.



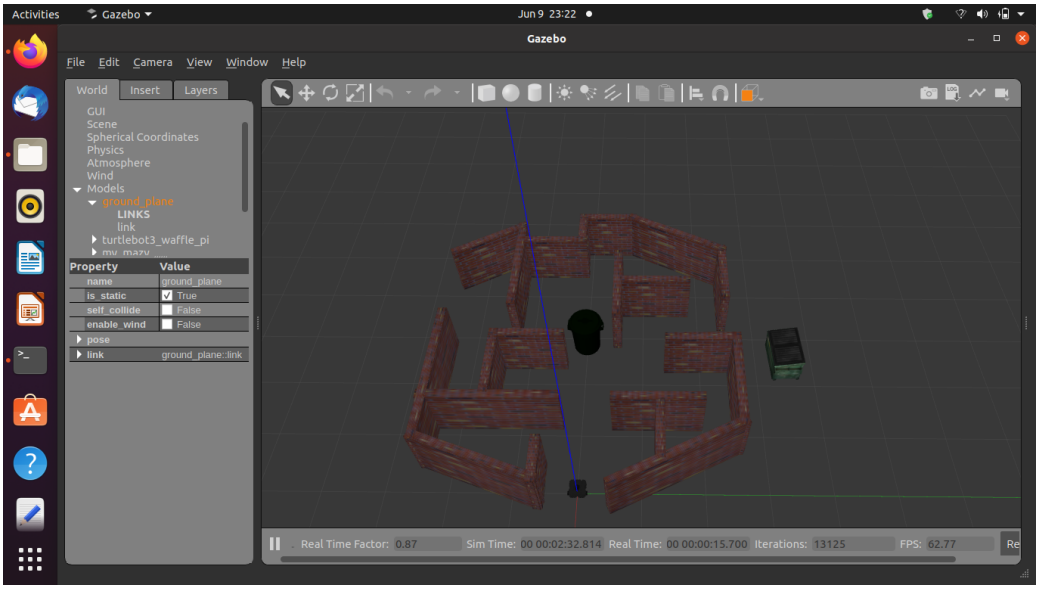
- To create the maze model, I used the Building Editor in Gazebo. After I save the model (in /home/jojo/.gazebo/models/my_maze) you will see two files inside my_maze folder (.config and .sdf). They are configuration and simulation description format files. Any object in Gazebo is described with these two files.

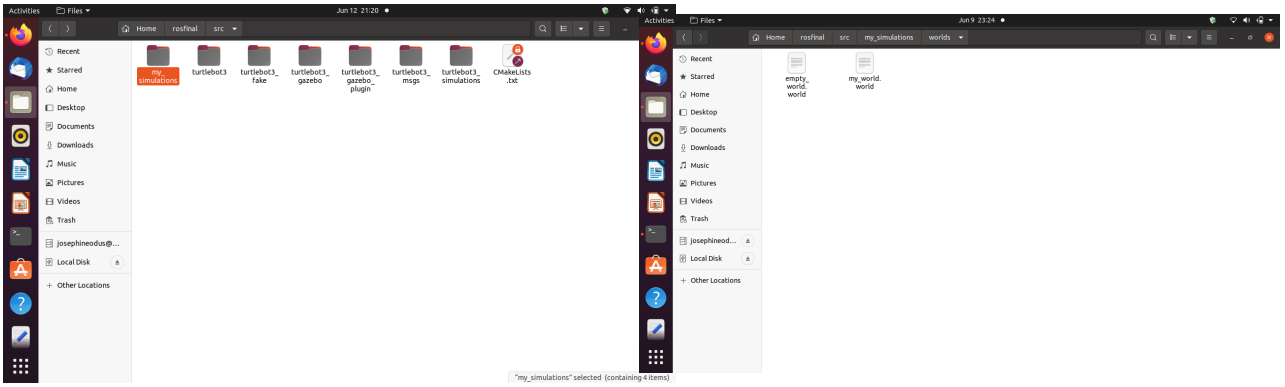


- I then saved the world in my ‘worlds’ folder and named it ‘my_world.world’.
- Make sure your .launch file has this new my_world.world written into it.



- I added a trash can and dumpster for extra touches.





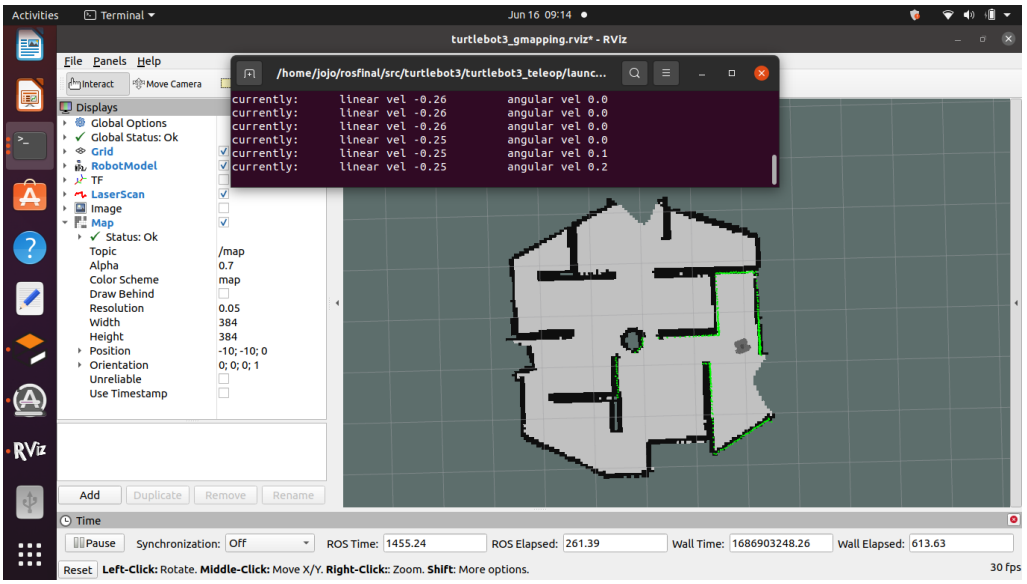
- Launched the SLAM mapping node.

  $roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping

- Launched the teleop node so we can drive around

  $roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

- I then drove around to collect data from the robot's sensors until I got an almost complete and accurate map.



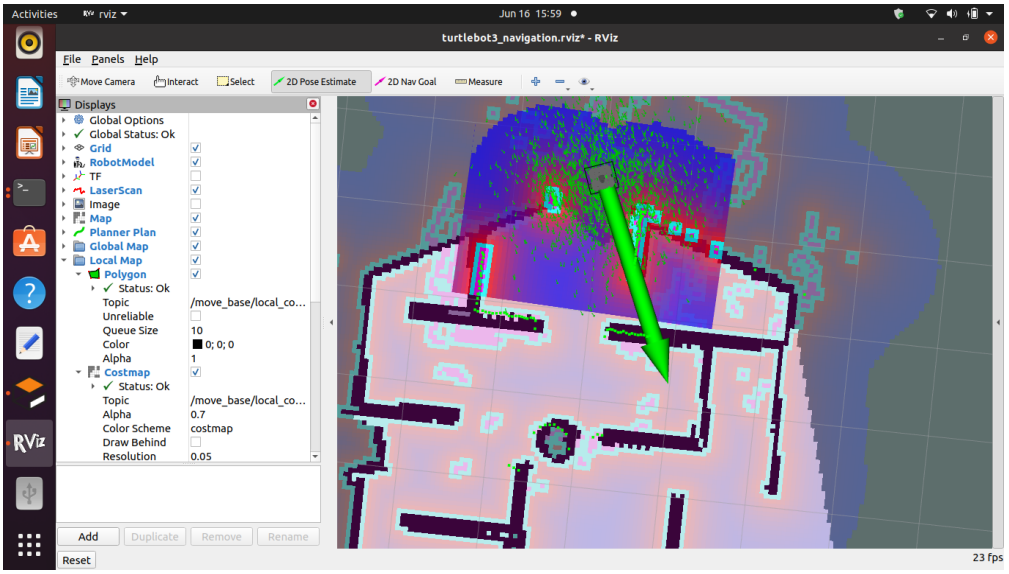
- I now used this code to save the maps to my home directory. Although the maps can be saved anywhere else.

  $rosrun map_server map_saver -f ~/map

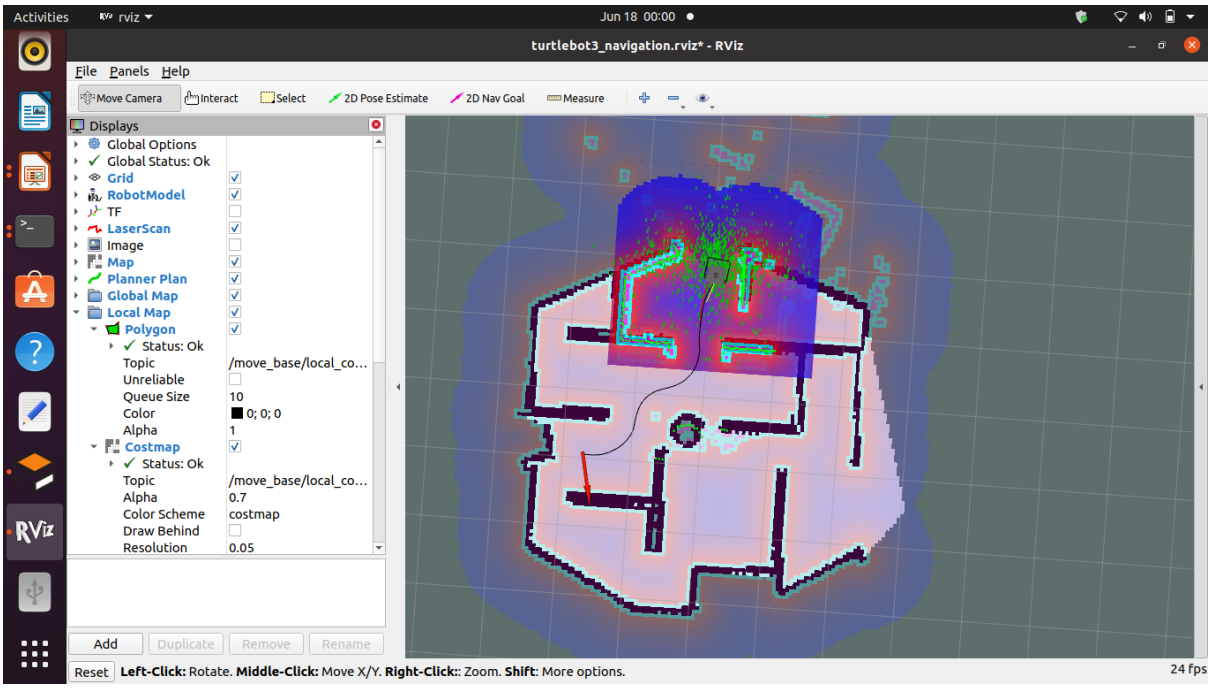
- Launched my ‘my_world.launch’ file.

  $roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml

- In RVIZ, I select the 2D pose selector. I drag an arrow that estimates where the robot is currently positioned relative to the map that I created.



- For the robot to navigate to the goal autonomously, I select 2D NAV goal.



Conclusion

This simple demonstration utilising the ROS Noetic and TurtleBot3 simulation packages has successfully demonstrated the autonomous navigation capabilities of the Waffle Pi robot. Through the integration of these powerful tools, I have witnessed the robot navigate its environment with precision and efficiency, showcasing the potential of robotic systems in various practical applications.

Throughout the experiment, I have observed the Waffle Pi robot intelligently plan its path, avoid obstacles, and reach its designated goals as long as it is a clear and defined path.

While the experiment yielded promising results, there are always opportunities for improvement. Future enhancements could include fine-tuning the robot's path planning algorithms to optimise for speed, energy efficiency, or specific task requirements. Additionally, incorporating advanced sensor technologies, such as LiDAR or stereo cameras, could enhance the robot's perception capabilities and further improve its navigation abilities in complex environments.

Upon concluding this experiment, I was filled with excitement for the future of navigation in the realm of robotics. The potential for autonomous robots to assist in various domains, including search and rescue, warehouse automation, and even space exploration, is both awe-inspiring and promising.

This experiment has not only showcased the capabilities of the Waffle Pi robot but has also fueled my passion and optimism for the future of navigation in robotics. By harnessing the power of ROS Noetic and TurtleBot3, we continue to take a significant step forward in unlocking the potential of autonomous systems, ultimately contributing to the advancement of science and benefiting society as a whole.

References

1. Wilkerson, Forsyth, Sperbeck, Jones, Lynn, “A Student Project using Robotic Operating System (ROS) for Undergraduate Research”. ~https://www.jasonforsyth.net/pdf/a-student-project-using-robotic-operating-system-ros-for-undergraduate-research.pdf
2. Wilkerson, Forsyth, Korpela, “Project Based Learning Using the Robotic Operating System (ROS) for Undergraduate Research Applications” ~https://www.researchgate.net/publication/325088637_Project_Based_Learning_Using_the_Robotic_Operating_System_ROS_for_Undergraduate_Research_Applications
3. https://github.com/Delmond/Navigation.git
4. https://navigation.ros.org/setup_guides/algorithm/select_algorithm.html
5. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0263841
6. https://github.com/sundevilrobotics/urc-code/wiki/4.-Learning-ROS-through-Simulation
7. ros gazebo maze simulation for turtlebot3 (beginner-level tutorial) ~https://youtu.be/J_vasKVhUQM