Reactive robot under ROS

Philius Moraless

Faculdade de Engenharia da Universidade do Porto
University of Porto
Porto, Portugal
Email: moraless.philius@universite-paris-saclay.fr

***Abstract*— This document presents a comprehensive overview of a reactive robot project utilizing ROS. The robot's primary objective is to autonomously follow walls in various environments. The document explores the robot's performance, characteristics, limitations, and ongoing efforts for enhancements.**

I. INTRODUCTION

Reactive robots, which can quickly react and adapt to their surroundings, are taking center stage in the world of autonomous robotics. And one of the key tools making this possible is the Robot Operating System, or ROS. In this article, we're going to dive into the world of ROS and use it to create a robot that's fantastic at following walls.Our main goal is to show how the latest technology can be used in a very practical way. We'll dig deep into the robot's design, its brain (the control laws), and how it uses sensors to understand its environment. We'll also test it extensively to see how well it performs, where it struggles, and how we can make it even better for real-world use.This project isn't just about experiments; it's also a stepping stone for our future work. We want to learn more about using ROS for reactive robots and get ready for more advanced projects

II. STATE OF ART

A. *Learning ROS and Training with Tutorials*

When we embarked on this project, our foremost objective was to master the Robot Operating System. ROS, an open-source platform widely used in the field of robotics, was integral to our project. We dedicated time to comprehend its core principles, such as the creation of ROS nodes and communication between them. This understanding was pivotal in designing, developing, and effectively controlling our reactive robot.

To gain this understanding, we utilized a multitude of online tutorials, which served as a wealth of knowledge and guidance. These tutorials covered a wide array of topics, ranging from the creation of ROS nodes to the programming of sensors and actuators. They provided a practical grasp of the fundamental concepts of ROS.

B. *Embracing Gazebo and Continuous Exploration*

Another critical facet of our project was the utilization of Gazebo, a 3D robotics simulator seamlessly integrated with ROS. We delved into Gazebo's capabilities, mastering its functions for modeling and simulating a robotic environment. Simulation within Gazebo enabled us to virtually test and refine our robot before real-world deployment.

Our pursuit of knowledge extended beyond tutorials. We conducted comprehensive research, consulting books, research papers, and ROS community forums to enhance our understanding and address specific challenges. This continuous exploration proved invaluable for overcoming technical obstacles and improving our reactive robot.

III. ROBOT'S IMPLEMENTATION AND ARCHITECTURE

A. *3D Model with URDF*

First and foremost, our initial goal was to construct the 3D model of the robot. To achieve this, I created a 3D representation of our robot, enabling it to move within the xOy plane and perform rotations around the z-axis. The URDF (Unified Robot Description Format) played a crucial role in this process. It is an XML-based file format widely used in the field of robotics to describe a robot's structure and kinematics. URDF provides a standardized approach for defining a robot's physical components, including links, joints, sensors, and their relative positions. This file format is commonly employed in conjunction with the Robot Operating System to facilitate robot modeling, simulation, and control.

Our robot is composed of a circular-shaped chassis, featuring two controllable wheels and an additional pair of wheels that function as caster wheels. The two controllable wheels are responsible for propelling and steering the robot, while the caster wheels provide stability and help maintain balance. This design allows the robot to move efficiently and navigate its environment with ease, making it suitable for a variety of tasks and applications. The combination of these components ensures that the robot can achieve the desired

mobility and control to carry out its intended functions effectively.

Component	Height(m)	Radius(m)	Mass(kg)
Chassis	0.1	0.25	5
Wheel	0.04	0.1	2.5
Caster Wheel	None	0.05	O.05

Table 1. the robot's characteristics

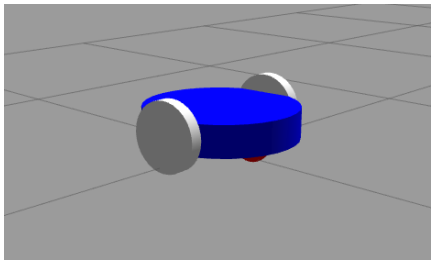

Fig1. Robot in Gazebo

B. Efficient Robot Control

For robot control, the implementation required the utilization of the gazebo_ros_control plugin in conjunction with joint_state_controllers. These components enabled the robot to receive velocity commands, translating them into motion. To facilitate robot control, the robot_state_publisher package was employed, streamlining the management of the robot's kinematic information. To further simplify the robot's movement, we opted to control both wheels simultaneously.

For this purpose, the gazebo library's differential_drive_controller plugin was utilized. This approach allowed us to efficiently command the robot's motion, making it easier to navigate and interact with its environment.

C. Lidar integration : Hokuyo

The third part of the implementation focused on integrating a Lidar, a laser range sensor, to enable the robot to perceive its surroundings by performing distance measurements in a field of view ranging from -π to π radians (equivalent to -180 degrees to 180 degrees). For this application, we selected the Hokuyo sensor, well-known for its reliability and ability to provide accurate data on the spatial configuration of the surrounding objects. This Lidar allows the robot to create a virtual representation of its environment, which is crucial for tasks such as mapping, localization, and autonomous navigation.

By combining the Lidar data with kinematic information from the robot's URDF, we were able to develop a comprehensive representation of the state of the environment, enabling the robot to make decisions and autonomously respond to obstacles and space configurations in real-time.

D. Communication Flow and Data Handling Between Nodes and Toics

In this section, I will provide an overview of how the nodes in the system communicate with each other, ensuring the system's proper functioning using topics. To achieve this, I employed the "teleop_twist_keyboard" package, enabling me to send velocity commands to my robot.

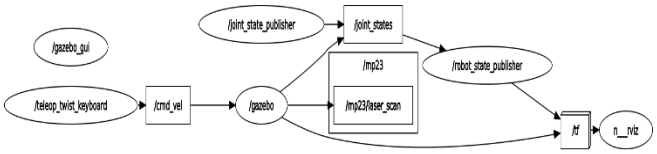

Fig2. Robot's rqt_graph

To begin, the generated data is transmitted to the /cmd_vel topic. Subsequently, this topic relays the data to the /gazebo system, which further distributes it to my robot's simulation system, identified as /mp23, through the /Joint_states topic. This data transmission chain enables the robot to receive motion commands.

Furthermore, in this demonstration, I utilized the Rviz software tool, which is why you can observe various communications towards it. This adds a visualization aspect to the entire system.

IV. EXPERIMENTS

Our wall-following algorithm experiment is conducted on a custom-designed map featuring a point-shaped landmark that we modelled ourselves. The primary objective is to navigate the path leading to the bottom rectangular side of this landmark and come to a stop there. To facilitate this, we have meticulously organized our project by creating a dedicated package to keep our robot model and wall-following algorithm files separate. Within this package, we've developed a node responsible for processing data from our Lidar sensor and using this information to guide the robot's movements.

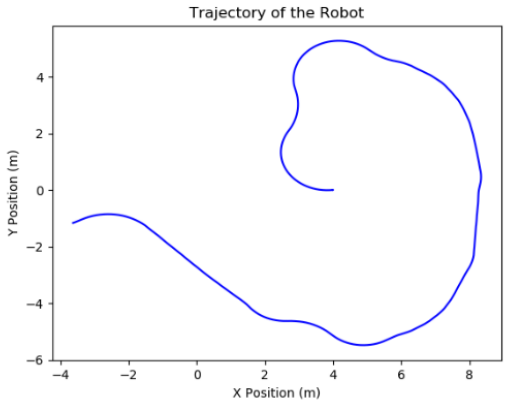

Fig. 3 Travelled Path

Here is an overview of the map with the robot coming to a stop at the destination point.

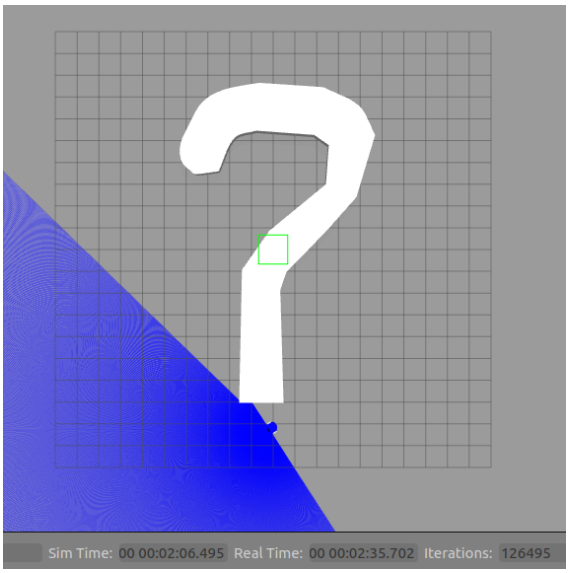

Fig.4 Overview of the map

V. RESULTS AND DISCUSSION

A. *Perfomance analysis*

Our robot successfully navigated the predefined path, circumnavigating the interrogation point and coming to a halt as intended. However, it is crucial to highlight that the algorithm's performance is sensitive to the initial parameter values. When these values were altered, the algorithm's effectiveness was compromised, causing the robot to deviate from the desired trajectory. This sensitivity to initial conditions underscores the need for more robust and adaptive algorithms to enhance the robot's performance.

Initial			Real Time(min)
x = 3	y = 0	theta = -3	1.45
x = 4	y = 0	theta = -3	1.36
x = 4	y = 1	theta = -3	1.47

Table.2 Travel time

B. *Limitations*

One of the primary limitations is the algorithm's dependence on predefined parameters, making it less adaptable to dynamic environments and varying wall shapes. Additionally, the algorithm's response to unforeseen obstacles or irregularities in the wall-following path is currently limited, necessitating further development to enhance its adaptability.

To address these limitations, future work should focus on improving the algorithm's ability to adapt to varying environmental conditions and wall shapes. This could involve implementing more sophisticated algorithms for obstacle avoidance, refining parameter tuning procedures, and incorporating advanced sensor fusion techniques. Moreover, real-world deployment would require the development of a robust localization and mapping system to enable autonomous navigation in complex and unstructured environments.

CONCLUSION

This project has been an exciting journey into the realm of reactive robotics using ROS. We have successfully created a robot capable of consistently following a wall, yet there remain challenges to transform it into a truly adaptable and robust tool. The lessons learned throughout this process are invaluable, emphasizing the importance of understanding initial parameters and the need to enhance the algorithm's agility in the face of changing environments. This project marks the beginning of our path towards creating autonomous robots capable of navigating complex environments reliably.

ACKNOWLEDGMENT

My sincere thanks go to our instructors and mentors for their guidance and expertise, which proved invaluable throughout the development process. I also extend my appreciation to the open-source robotics community for providing the tools and resources that enabled me to work with ROS and Gazebo

REFERENCES

I began the project by watching some videos from – [1]- this channel, which I found to be quite valuable. Simultaneously, I explored this channel - [2] -, which aided me in gaining a more comprehensive understanding of how ROS operates, such as the concepts of nodes, topics, and node creation. Ultimately, this resource – [3] - was instrumental in resolving certain errors I encountered during the project.

[1] Articulated Robotics: https://www.youtube.com/@ArticulatedRobotics
[2] Robotic Systems Lab: https://www.youtube.com/@leggedrobotics
[3] Wiki ROS: https://wiki.ros.org