ROS2 Wall-Following Differential Reactive Robot

Bruno Mendes
M.EIC
FEUP
Porto, Portugal
up201906166@edu.fe.up.pt

Fernando Rego
M.EIC
FEUP
Porto, Portugal
up201905951@edu.fe.up.pt

José Costa
M.EIC
FEUP
Porto, Portugal
up201907216@edu.fe.up.pt

Marcelo Couto
M.EIC
FEUP
Porto, Portugal
up201906086@edu.fe.up.pt

***Abstract*—This paper exposes the development of the software for a wall-following differential drive robot with differential locomotion and reactive behaviour, with the ability to stop upon detection of sharp, small, straight sections through sensors. ROS2 was used for the implementation and Flatland2 simulator was used for validation and evaluation. The methodology developed uses a 2D LiDAR sensor and simple proportional controllers to control the robot's motion. The experimentation carried out validated the system and gave insight on the relevance of parameter tuning.**

***Index Terms*—reactive robot, differential drive, wall following robot, ROS2, ROS2 Flatland, LiDAR**

I. Introduction

It is common for robots to build an internal world representation to support intricate actions through complex decision-making. However, not all tasks require such complicated, computationally and hardware-expensive systems to be completed. Simple reactive behaviours can mimic intelligence. A common example is the first 6 generations of Roomba vacuum cleaners (up to 2015), which lack complex world representation and make use of bumper contact and infrared sensors to detect and avoid obstacles. State of the art of wall-following robots will be discussed in Section II.

This article describes the architecture and implementation of a robot that searches for a wall and follows it until it detects a sharp small straight segment, stopping afterwards. The robot has no internal state representation or world map, as it follows a purely subsumption model, relying on its sensors to decide on the next action to execute. Section IV will expand on the behaviour model and Section III will define the test environment, assumptions, and constraints.

Functionality is built with ROS2 "Humble Hawksbill". A modified version of Flatland to support ROS2 was used to simulate the environment. Visualization is provided by RViz. The robot simulated by Flatland2 is based on FEUP's SERP2 robot, a differential drive robot built around an Arduino + RPi platform. It includes a LiDAR sensor to detect obstacles. Relevant details on the supporting technologies and the implemented algorithm will be analyzed in Section V.

The achieved results will be presented and discussed in Section VI. Section VII concludes the article and suggests possible improvements.

II. State of the art

Wall-following is a common task in the robotics field; as such, the literature on the subject is extensive and several approaches for solving the issue emerged.

A. Navigation algorithms

Popular approaches include mapping, especially simultaneous localisation and mapping (SLAM), potential-field approaches and reactive approaches. [1]

1) Mapping approaches: Mapping approaches keep the robot's state and location, estimated or inferred by some environment marker, as well as a detailed environment map, either uploaded or built and updated on the fly (SLAM), and navigate the environment by using this information. This method proves itself computationally expensive as the robot's view of the environment needs to be constantly updated.

2) Potential-field approaches: Making use of gradients, robots are expected to move from locations with high potential to locations with lower potential. In this paradigm, obstacles represent locations of very high potential, which the algorithms avoid. This methodology requires prior environmental knowledge, which isn't always available.

3) Reactive approaches: Navigation decisions are made solely on currently available data provided by the robot's sensors. This means that the robot has limited knowledge of the environment around it. However, this renders the method less computationally intensive, yielding faster response times.

The aforementioned navigation methods are unable to solve dynamic obstacle avoidance. By using current data, a side-effect of reactive algorithms is being capable of reacting to changing environments, making these methods desirable for applications that require simple navigation of dynamic heterogeneous layouts, like cleaning and disinfection [2].

B. Perception

Recent publications focus on the use of fuzzy logic to process sensor data and control the robot. In this context, distances to the wall translate to intervals that represent relative distances to the wall used as fuzzy sets (near, medium, far). These intervals are then used as input to an inference engine with fine-tuned thresholds that decide on what direction to turn, deciding between left, right or straight. The decision is then converted to crisp logic through defuzzification, outputting a turning angle.

The work of Li and Wang [3] uses notions from human experience to define the relative input thresholds and the fuzzy decision engine criteria. Genetic algorithms have been studied to tune these parameters [4].

C. Control

Most work being developed regarding the control of autonomous agents is in the field of autonomous driving. On this front, model-based approaches are popular, and much of the work developed explores different methodologies of employing improving the fidelity of the models and performance of the algorithms, such as in [5]. In recent years, with the development of computational power and the rise of Machine Learning techniques, a shift in the focus on mobile robotics can also be noticed towards the study of Reinforcement Learning as a tool for motion control. The usage of fuzzy logic has also been trending. Cheng-Hung and Jeng [6] have employed both these techniques to come up with a unique controller for wall-following robots. However, these are not all the approaches available for the control of a wall-following robot. Simpler methodologies have proven their success in earlier ages such as in [7]. It is also worth noting the versatility of PID controllers, making these a valid and commonly applied option in mobile robotics, notably in [8], [9].

III. Problem statement

The goal of the proposed project was the development of a 2D Reactive robot designed to autonomously track and follow a wall, with the specific goal of navigating until the detection of a sharp rectangular edged zone of the wall within the world it operates in.

A. The Map

The map is shaped like a question mark "?", with an imperfect circle above and a circular termination at the upper end. The lower part forms a straight and edgy contour.

Fig. 1. Simulation map with start area highlighted in green

Figure 1 represents the map and highlights the robot's initial position, in an undetermined place inside the green area.

B. The Robot

A robot with a round structure and employing a differential locomotion system was used. This differential locomotion system implies that the movement is based on two independently controlled wheels on either side of the body. However, the simulation environment used only allowed for the control of the motion of the robot as a whole, not the wheels independently. Furthermore, the robot is to be equipped with sensors that enable it to perceive the world state. Our exteroceptive sensor of choice is a 2D LiDAR (Light Detection and Ranging) sensor, which calculates the distance to a surface or object by measuring the time for the reflected light to return to the receiver. Particularly, this sensor performs measurements in a complete 360º range around the robot, at 4º intervals, featuring 90 laser beams. This configuration offers, to some extent, comprehension of the state of the world.

C. Representation of the world

The state of the world can be represented by three major aspects. Firstly, the angle relative to the perpendicular to the wall, θ, which, having in mind that the robot should maintain a parallel course alongside the wall, this angle would ideally be 90º or -90º, depending on the direction of circulation.

Secondly, it was considered the distance to the wall, d. It's necessary to possess knowledge regarding the distance between the robot and the wall. Additionally, an ideal distance to the wall can be defined to perform adjustments along the robot's course.

The third aspect is the current velocity of the robot, v.

The motion of the robot is controlled via linear acceleration a and angular velocity ω commands. In pursuit of a more realistic simulation and for experimental reasons, configurable limits were imposed on the maximum and minimum variables of these control variables.

IV. Behavioural model

Evoking the subsumption model proposed by Brooks [10], it is possible to build higher-order behaviours, materialised in the robot's actuators' outcomes, by using sensor data and outputs of lower competence layers, subsuming their roles, either by agreeing with their outputs or changing them to achieve the greater goal.

In the following subsections, the desired behaviours will be staged from lower to higher order.

A. Wander until wall

Move around the environment until a wall is found.

For the sake of simplicity, we assume that this behaviour is implemented through another mechanism, as all the initial robot positions for testing guarantee that a wall is always found.

B. Keep distance from the wall

If a wall is found, face the robot side and move parallel to it, optionally with an ideal distance to the wall.

C. Stop if a small straight section is found

Upon detection of a small straight section bring the robot to a halt.

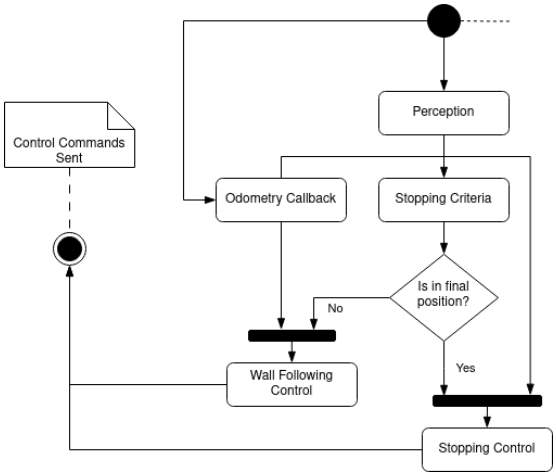

Fig. 2. Activity Diagram

V. Architecture and Implementation

A. Simulation Environment

As previously noted, the simulation environment used was an altered version of the Flatland2 simulator [11], made to work with ROS2. This version of the simulator was further altered to contain a second implementation of a differential drive robot, which would take acceleration commands instead of velocity commands [12]. Such implementation was carried out to provide closer control of the motion of the robot and match a real scenario.

Flatland2 is a two-dimensional simulator that uses RVIZ for visualisation purposes. It is particularly useful for simple simulations that can benefit from time acceleration, such as Reinforcement Learning experiments or evaluation of a system with multiple parameter combinations. The simulator also allows for the usage of a laser scanner, which is our sensor of choice.

B. Architecture

We used ROS2 to develop our system and, as such, our program maps to a ROS2 node. Being a simple robot, the robot was programmed as a single object with multiple functions. The robot system developed follows a reactive architecture, only divided into 4 different models:

- Perception
- Wall Following Control
- Stopping Control
- Stop Criteria

Additionally, there is an odometry callback function only used to draw the necessary information regarding the velocity of the robot at a certain point in time. Figure 2 depicts an activity diagram that illustrates the workflow of the program. The components diagram in Figure 7 can help visualise the environment in which the system is tested.

C. Perception

The perception model is the one that transforms the data sent by the simulator regarding the laser scanner into the image of the current world state. It does so following a simple algorithm:

1) select the laser with the smallest distance measurement
2) calculate the angle to between the direction the robot is facing and the perpendicular to the wall taking into account the angle increments between lasers and the index of the laser selected

$$\theta = i \times a + m \tag{1}$$

The formula used to calculate the angle is described in 1, where θ is the angle intended to be calculated, i is the index of the laser in the array, a is the angle between lasers and m is starting laser's angle value.

The choice behind this controller can be justified by the incremental spirit of the development of this project: following investigation, the simplest solution available is implemented; if it does not suit the problem, it is upgraded; otherwise, it is left as is.

D. Stop Criteria

To detect the final sharp small straight section in order to stop the robot, two distinct approaches were developed. It is important to note that both approaches share in common the main following steps:

1) The robot iterates through the array of distances, given by LiDAR, until the wall is detected
2) Upon wall detection, each approach validates the distance to the wall of each laser beam until the first laser beam that does not detect any object. If any validation fails it means that does not correspond to the final wall
3) Finally, the robot confirms that no other object is present

To introduce some flexibility to each validation method, the robot can not only identify the final straight wall but also accommodate situations where the robot detects more distant walls from the map. In these scenarios, the validation methods accept detected distances that deviate significantly from the calculated distances, d_i in the first criteria and d_{ij} in the second criteria.

The two distinct criteria utilised to detect the final straight are described below.

1) Straight line based criteria: The first criteria developed is based on straight-line calculations. It assumes that the laser beam with the shortest detected distance represents the normal relative to the wall (straight line), thereby allowing the calculation of the distance to the wall for the other laser beams.

$$\alpha_i = (|i - i_{min}|) \times a \tag{2}$$

$$d_i = d_{min} / \cos \alpha_i \tag{3}$$

The formula to calculate the angle and the associated distance are 2 and 3, respectively, where:

- i_{min} is the index of the laser in the array that detect the smallest distance

- i is the index of the laser in the array to validate
- a is the angle between lasers
- α_i is the angle of the laser beam in index i with the normal relative to the wall
- d_{min} is the smallest distance measurement
- d_i is the calculated distance for the sensor in index i

The validation in this approach is straightforward. It involves a direct comparison between the distance measure by the laser beam to the calculated distance. Naturally, considering sensor noise and potential inaccuracies, a margin of error of 0.3 units was allowed after fine-tuning the validation method.

2) Distance based criteria: The second criteria, which relies on the distances between consecutive wall points detected by each laser, was chosen due to specific relevant aspects according to the map. The distances between consecutive wall points can be calculated using the Law of cosines, where:

- a is the angle between lasers
- d_i is the distance measurement of the laser beam in index i
- d_j is the distance measurement of the laser beam in index j
- d_{ij} is the calculated distance between detected points by laser beam in index i and j
- i and j are consecutive array indexes

$$d_{ij} = \sqrt{d_i^2 + d_j^2 - 2 * d_i * d_j * \cos a} \tag{4}$$

Given the short length of the final wall and the narrow angle between the laser beams, and in line with the first criteria, accounting for sensor noise and potential inaccuracies, a margin of 0.15 units was allowed as the maximum distance between consecutively detected points after refining the validation method.

E. Wall Following Control

The wall following control aims to maintain a certain distance from the wall and a certain velocity at the same time. Despite the many advanced techniques used for motion control nowadays, this project did not require such complex methodologies to yield good results. As such, the controllers implemented for the linear acceleration and angular velocity commands were simple Proportional controllers.

For angular velocity control, two controllers were developed:

- The first controller only took the angle error into account (i.e. the difference between the θ and its ideal value)
- The second controller took both the angle error and the distance error

$$\omega = \theta_e * k\_a \tag{5}$$

$$\omega = \theta_e * k_a + d_e * k_a * dir \tag{6}$$

The formulas of both of these controllers are 5 and 6 respectively, where:

- ω is the angular velocity
- θ_e is the angle error
- d_e is the distance error
- dir is a binary variable that denotes the signal of the ideal angle (1 or -1, depending on on which side of the robot was the wall)
- k_a is a multiplying linear factor that was tuned for the best performance

For linear acceleration control, the controller developed can be viewed as a chain of two controllers:

- the first controller outputs goal velocity given the fixed target velocity and the current angle error. The idea is to slow down when making trajectory adjustments to not get too far from the trajectory e.g. get far away from the wall or collide with it
- the second controller takes the velocity error (i.e. the difference between the current velocity and the aforementioned goal velocity) and outputs a proportional linear acceleration value

$$a = k_l * gv_e \tag{7}$$

$$gv = \frac{v}{(1 + v * \mod \theta_e)} \tag{8}$$

Once again, the first controller in the chain is given by 8 while the second is given by 7, where:

- gv is the goal velocity
- θ_e is the angle error
- v is the target velocity
- a is the linear acceleration
- gv_e is the velocity error
- k_l is a multiplying linear factor that was tuned for the best performance

The formula in 8 was idealised such that the goal velocity would be the same as the target velocity if the angle error was 0, but it would also be smaller the greater the angle error and the target velocity values were. This last measure was taken following experimental trials in which the actual velocity of the robot was too great while the robot was facing forward against the wall for higher values of target velocity, leading to collisions.

F. Stopping Control

The stopping control robot aims to control the acceleration command in order to bring the robot to a stop. It uses the formula described in 7, where the velocity error is just the symmetrical of the current velocity. The robot is considered to have come to a complete stop when the indicated absolute velocity is smaller than 0.01 for more than 2 following measures. The latter measure is taken to reduce the influence of odometry noise. These values were obtained through experimental tuning.

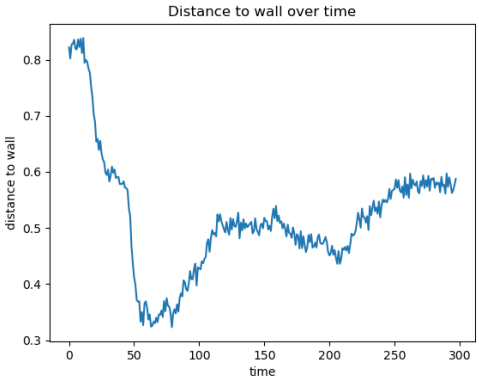

Fig. 3. Distance to the wall with target_velocity=0.4 and ideal_distance=0.5.

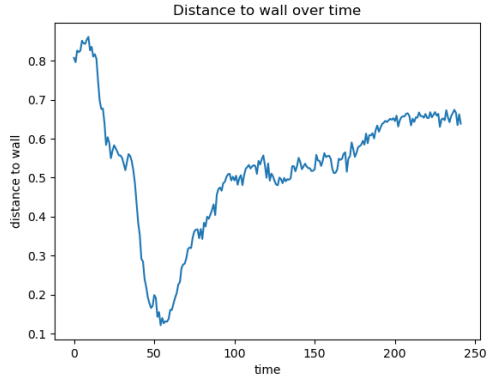

Fig. 4. Distance to the wall with target_velocity=0.6 and ideal_distance=0.5.

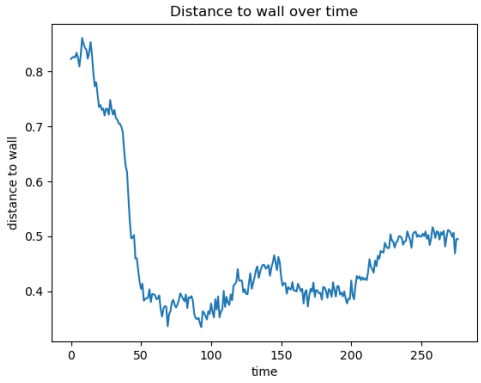

Fig. 5. Distance to the wall with target_velocity=0.6 and ideal_distance=0.5 using old controller.

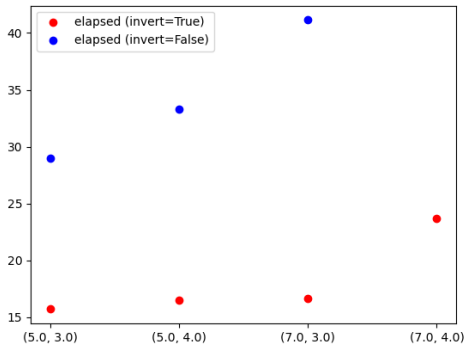

Fig. 6. Elapsed time according to start position.

VI. Results and Discussion

To evaluate the effectiveness of different hyper-parameters, such as k_ang or k_lin, and to compare the two different controllers implemented, an external custom script was developed to orchestrate the launch several instances of the robot node, taking advantage of *ROS2 Parameters*.

Increasing the target velocity of the robot makes it reach its final destination sooner, but it leads to slightly greater instability in its movement, as depicted in VI and 4. Results also indicate that the new controller was not that great of an improvement, as the old one was already very stable, which can be seen in 5.

Although not shown here, due to the lack of results, increasing the target velocity past the unit and the target distance past the unit renders the stopping criteria blind to the final destination. That is expected due to the way it is implemented (refer to section IV).

The robot is able to wander close to the wall and reach its final position, independently of the starting position as per the requirements. Inverting its direction (i.e. maintaining a positive perpendicular angle to the wall) haves the time needed to travel the path, as it is shorter.

k_ang and k_lin play a significant role in the outcome of the robot's journey. Modifying the angle of the robot and its linear acceleration in greater steps means that the robot tries to fix its trajectory more aggressively: it is then closer to its target distance to the wall but it may also lead to it hitting the wall if it is too close to it.

VII. Conclusions and Future Work

The developed reactive robot, although not as accurate as recent SLAM robots, is able to wander close to the wall under different circumstances, and does not show a significant amount of false positives in the stopping criteria.

Future improvements could include the switch to a PID controller, to improve the overall quality of the travelled path. The stopping criteria also deserves some attention: a switch to a more complex SLAM-based agent enabling recognition of the target region would most likely improve results.

REFERENCES

[1] A. De and D. E. Koditschek, "Toward dynamical sensor management for reactive wall-following," in *2013 IEEE International Conference on Robotics and Automation*, pp. 2400–2406, 2013.

[2] K.-J. Joo, S.-H. Bae, A. Ghosh, H.-J. Park, and T.-Y. Kuc, "Wall following navigation algorithm for a disinfecting robot," in *2022 19th International Conference on Ubiquitous Robots (UR)*, pp. 343–346, 2022.

[3] X. Li and D. Wang, "Behavior-based mamdani fuzzy controller for mobile robot wall-following," in *2015 International Conference on Control, Automation and Robotics*, pp. 78–81, 2015.

[4] S. F. Desouky and H. M. Schwartz, "Genetic based fuzzy logic controller for a wall-following mobile robot," in *2009 American Control Conference*, pp. 3555–3560, 2009.

[5] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094–1099, 2015.

[6] C.-H. Chen, S.-Y. Jeng, and C.-J. Lin, "Mobile robot wall-following control using fuzzy logic controller with improved differential search and reinforcement learning," *Mathematics*, vol. 8, no. 8, 2020.

[7] T. Yata, L. Kleeman, and S. Yuta, "Wall following using angle information measured by a single ultrasonic transducer," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 2, pp. 1590–1596 vol.2, 1998.

[8] K. Lee, D.-Y. Im, B. Kwak, Y.-J. Ryoo, *et al.*, "Design of fuzzy-pid controller for path tracking of mobile robot with differential drive," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 18, no. 3, pp. 220–228, 2018.

[9] J. E. Normey-Rico, I. Alcalá, J. Gómez-Ortega, and E. F. Camacho, "Mobile robot path tracking using a robust pid controller," *Control Engineering Practice*, vol. 9, no. 11, pp. 1209–1214, 2001.

[10] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[11] J. Costa, "Flatland2 - flatland ros2 adapatation," 2022.

[12] M. Couto, "Flatland2 fork with acceleration based reactive robot," 2022.

APPENDIX

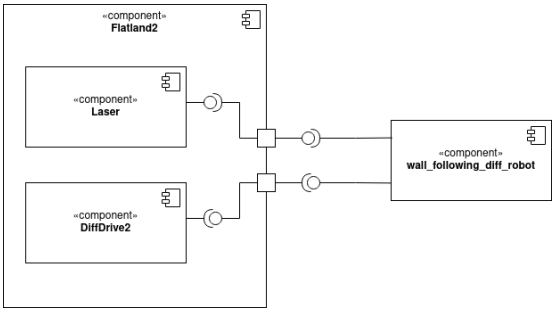

Fig. 7. Components Diagram

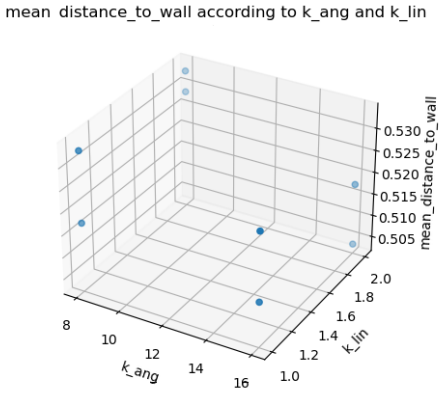

Fig. 8. Mean distance to the wall according to hyper-parameters (ideal_distance=0.5).