Wall-following Reactive Robot

António Ribeiro
Faculty of Engineering
University of Porto
Porto, Portugal
up201906761@edu.fe.up.pt

Filipe Campos
Faculty of Engineering
University of Porto
Porto, Portugal
up201905609@edu.fe.up.pt

Francisco Cerqueira
Faculty of Engineering
University of Porto
Porto, Portugal
up201905337@edu.fe.up.pt

***Abstract*—We present the development and testing of a reactive wall-following robot equipped with *LIDAR* sensors. The robot's functions include wall-following at a set distance, wandering, and stopping capabilities, structured under a subsumption architecture. Testing was conducted on a map featuring a question-mark-shaped structure with the objective of stopping at the bottom of the shape. The architecture was evaluated for a variety of different parameters and achieved an average distance error of approximately 0.18 meters when keeping a 1-meter distance from the wall. The achieved architecture is notable for its simplicity, speed, and effectiveness.**

***Index Terms*—robotics and automation, navigation, simulation**

I. Introduction

All mobile robots rely on algorithms that help them travel in their environment. One of the simplest and most studied algorithms is Wall-following (also known as boundary following). It is an intuitive movement technique where a robot contours an obstacle that it might find, aiming to prevent a collision, mimicking what we would do if we found ourselves walking indoors in a dark environment.

The algorithm's main information source is the distance to the wall, so in most cases, the robot has to be equipped with some type of ultrasonic, laser, or infrared sensor.

It can be applied as a main strategy or auxiliary behavior, in domains such as map building, obstacle avoidance, and improvement of the position estimate [1]. Its simplicity and practicality are useful in the context of reactive robots, that do not store any state memory, and consequently have to guide themselves on heuristics to reach their final objective and correct their trajectory along the way.

This project endeavors to conceive and assess the operational effectiveness of a reactive wall-following robot that ceases its movement upon encountering a specific undetailed region of the map.

II. Related Work

Wall-following robots have been explored exhaustively in literature but, typically, with a focus on complex behaviours such as [2], [3] and [4] which do not fit under the reactive robot category. Most relevant papers are either historical or not published, both of which are scarce.

Braunsting et. al [5] proposes a robot equipped with multiple ultrasonic sensors, which, due to their limited precision, create the need to develop algorithms that consider this. Namely, the detection range is divided into 4 discrete areas. This approach suffers from the unprecise sensors which is why multiple works shifted towards using *LIDAR* sensors. One such example is the work of [6] which proposes a reactive wall-following robot using the RANSAC [7] algorithm for error correction of *LIDAR* data and whose angular velocity is calculated using equation 1.

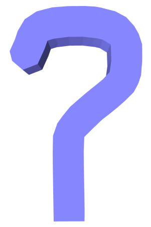
$$\omega = (-k(\sin(q_3) - (q_2 - T) + K)) \cdot v \tag{1}$$

where q_3 is the angle and q_2 the wall offset, k a fixed parameter, T the target wall distance, v the forward velocity and K the curvature estimate.

III. Environment

The environment where the robot was tested consists of a flat plane on which a question mark-shaped wall is placed. The robot spawns at a random position inside the imperfect circle defined by the topmost part of the wall, stopping when it finds the bottom-most edge of the shape, which is defined by a perfectly straight line with a width of approximately 3.5 m.

For this purpose, we use ROS 2 [8] and Gazebo Classic [9] which provides a 3D simulation environment (Figure 1).


(a) Aerial view of the 3D question mark placed on the simulated world

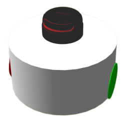

(b) Robot model

Fig. 1: Overview of the models present in the simulated environment

IV. Robot Architecture and Implementation

Compared to recent works, the objective of this study is not to improve upon the state of the art concerning precision or speed, but instead to simplify both the hardware and software to the bare essentials required to complete the task. This simultaneously improves the reaction speed of the algorithm, due

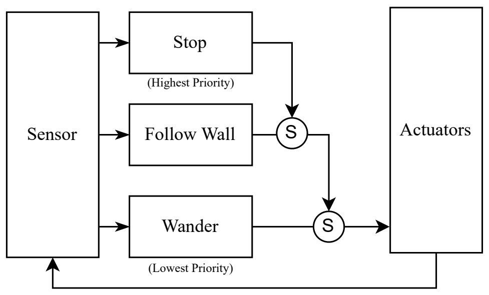


Fig. 2: Subsumption Architecture

to the reduced number of computations required, and makes the algorithm more interpretable and formally verifiable.

The main limitations imposed upon the hardware architecture are that the robot must be controlled via differential drive and it cannot have advanced localization sensors, such as GPS. On the software side, the most impactful limitation is that the robot must be reactive and, therefore, cannot have memory, ruling out mechanisms such as proportional–integral–derivative (PID) controllers.

A. Hardware Architecture

The robot chassis is cylindrical with radius $r = 10$ cm and height $h = 10$ cm with two lateral wheels used for differential drive. To stabilize it, we equipped two caster wheels at the front and the back.

It is equipped with a *LIDAR* sensor, located at the top, that detects obstacles within a configurable range, which, for our experiments was set between 0.1 m and 8.0m. This sensor detects 360 points around a $360°$ range, leading to an angular resolution of $1°$ with a 100Hz update rate. This configuration was used since it resembles a *Velodyne Puck LITE* sensor [10] with a reduced resolution and range.

B. Software Architecture

We propose a subsumption architecture, demonstrated in Figure 2 which has three distinct layers:

- **Stop** - having the highest priority, when the stop condition is met the robot should immediately stop its movement, making it impossible to resume travel assuming the world is static.
- **Follow wall** - this is the most common action, which occurs whenever at least one *LIDAR* point is within the detection range.
- **Wander** - when no wall is found, the robot enters into a wandering state in an attempt to re-discover a wall.

C. Angular velocity

We considered the angular velocity equation proposed by [6] as a starting point, further simplifying it to obtain a simple yet functional formulation, by removing the curvature estimate, K, and modifying how the robot angle is calculated to better fit our sensor. Simultaneously, we also added the ability to follow the wall from both sides of the robot, a major limitation in the original equation.

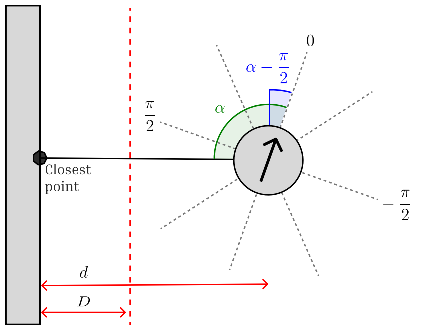


Fig. 3: Diagram demonstrating the angle correction required for a parallel trajectory. If the wall is located on the right side of the robot then the angle will be $\frac{\pi}{2} - \alpha$ instead.

The angular velocity takes into consideration two distinct objectives, straightening the robot's trajectory which should be parallel to the wall, and keeping the correct distance from the wall.

Consider a target distance that the robot should keep from the closest wall, D, and the closest point to the robot, P, defined by an angle α and its distance to the robot, d. In Equation 2 we define $\beta(\alpha)$ as being the rotation angle required to ensure that the robot's trajectory is parallel to the closest wall. This value is visually explained in Figure 3.

$$\beta(\alpha) = \begin{cases} |\alpha| - \frac{\pi}{2}, & \alpha \geq 0 \\ \frac{\pi}{2} - |\alpha|, & \alpha < 0 \end{cases} \tag{2}$$

In Equation 3 we define a value k which defines the distance to the closest wall, already considering which side the robot will turn towards.

$$k(\alpha, d) = \begin{cases} d, & \alpha \geq 0 \\ -d, & \alpha < 0 \end{cases} \tag{3}$$

Given this, we define the angular velocity ω, at each time step as demonstrated in Equation 4. The importance of each objective is adjustable by tweaking the parameters λ_1 and λ_2.

$$\omega(\alpha, d) = \lambda_1 \cdot k(\alpha, d) + \lambda_2 \cdot \beta(\alpha) \tag{4}$$

D. Linear velocity

The linear velocity is computed as follows:

$$v = v_c * \max\left\{\min\left\{\frac{1}{|\omega|}, \gamma_1\right\}, \gamma_2\right\} \tag{5}$$

Where v_c, γ_1, and γ_2 are constants that control the range of values. In our experiments, we set $v_c = 1.0$, $\gamma_1 = 1.0$, $\gamma_2 = 0.2$.

Our goal is to adjust the linear velocity according to the type of curve the robot is currently following. If it is heavily accentuated (requiring a high ω), then v decreases, allowing the robot to improve its control over the movement. On the contrary, if the curve is easily manageable, we can accelerate and traverse the distance faster.

E. Stop detection

To successfully stop the vehicle in the desired location (the bottom part of the question mark), we've grouped several heuristics that help us determine the robot's location and assert if it reached the final mark:

1) All laser points within the detection range must be adjacent.
2) The angles formed between the closest point and the two outermost must be roughly the same. This ensures that the robot is located in the middle point of the wall.
3) All points must form a straight line within a certain threshold. This is verified by fitting a straight line with the form $\bar{y}(x) = mx + b$ to the set of points detected by the *LIDAR* sensor, set P, using the least squares method [11]. Each point has coordinates $p = (p_x, p_y) = (\cos(\alpha) \cdot d, \sin(\alpha) \cdot d),\ \forall p \in P$ where α is the angle relative to the front of the robot and the d the distance to the sensor. Given the line equation and the set of real points the error is given by $\epsilon = \sum_{p \in P} (\bar{y}(p_x) - p_y)^2$. For our experiments, a line is considered straight if $\epsilon < 0.05$.
4) The Euclidean distance between the two outermost points must measure approximately 3.5m, which is the length of the bottom wall.

This procedure was developed taking into consideration the specifications of our test environment. Under different conditions, namely during the presence of more noise or reflective surfaces, some conditions may exhibit shortfalls. Condition 1 can fail if a single *LIDAR* pulse is undetected due to noise. To account for this we can loosen the restriction, allowing for gaps in the adjacent laser scans. Similarly, Condition 3 may be too strict, and an adjustment to the line error threshold might be required.

F. Wandering

When the robot does not detect a wall, i.e. all sensors are inactive, it performs the wandering behavior, rotating around in an imperfect circle until it finds a suitable wall according to equation 6.

$$v = v_c;\ \omega \sim \mathcal{N}(0.3, 0.1) \tag{6}$$

The robot may need more than one loop to find a wall that is on its right, waiting for the circle to randomly shift in the correct direction. One way to overcome this would be if the robot had basic memory, such as remembering the side of the previous closest wall, which could be used to go immediately in the right direction.

V. Experiments & Results

Each experiment was run on three distinct starting positions and poses randomly sampled from uniform distributions $x \sim U(-1.0, 1.0)$, $y \sim U(-3.0, 3.0)$, $\theta \sim U(-\pi, \pi)$. These starting values are the same across all experiments to allow for proper comparison. Ideally, each experiment would be run a large number of times to obtain statistical significance, but, due to the scope of the project and the compute-intensive simulation environment, we decided to forego this process.

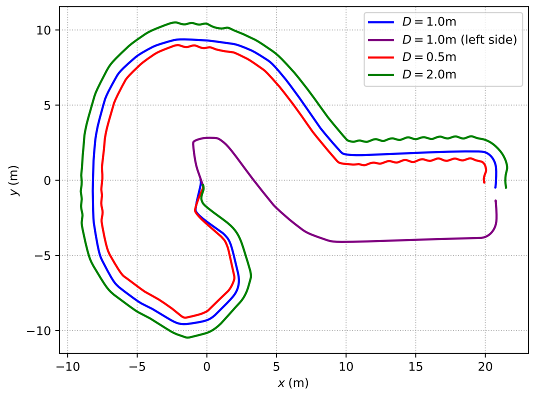


Fig. 4: Path trajectories recorded across Experiments 1,3,4 and 6 for the same starting position

Our experimentation scenarios targeted configurable factors that influence the robot's movement, such as weights in the angular velocity (equation 4), maximum values for both angular and linear velocity, movement direction, and target distance from the wall.

From the results table I, we gather the following conclusions:

- Inverting the λ_1 and λ_2 weights (with $\lambda_1 > \lambda_2$) results in a decreased relative distance error and an increase in lap time. This is due to prioritizing a constant distance to the wall and as such, the robot tends to veer from the tangent trajectory. This in turn brings more heading adjustments and lowers the linear velocity, slowing the movement. Figure 6 corroborates this reasoning, where we analyze the recorded relative distances in experiments 1 and 2.
- An increment in the target distance has the same outcomes as the aforementioned scenario, given that the robot travels farther away from the wall, with more distance to cover but fewer sharp turns that require heading corrections and a lower linear velocity, resulting in a less unstable wall distance.
- Diminishing the target distance by a factor of 2 did not yield a higher distance error compared to experiment 1, as initially expected.

Upon analyzing Figure 4, we found that our approach is suitable for the obstacle at hand, as demonstrated by the smooth curve trajectories described by the robot.

The stopping mechanism behaves as expected, considering the robot halts midway through the destination wall, in every recorded path.

Figure 5 illustrates the correlation between the relative distance error and the angular velocity, registered in experiment 1 (which places a larger emphasis on λ_1). From the graph peaks, we derive that abrupt heading adjustments are generally

TABLE I: Experiments table for different configurations. The experiments are divided according to which side the robot followed because they are not directly comparable.

Number	λ_1	λ_2	Max v ($\mathrm{m\,s^{-1}}$)	Max ω ($\mathrm{rad\,s^{-1}}$)	Target D	Direction	Lap Time (s)	Relative Distance Error (m)	Average v ($\mathrm{m\,s^{-1}}$)
1	1.0	5.0	10.0	10.0	1.0	Right	81.58 ± 4.50	0.31 ± 0.04	0.9547 ± 0.0125
2	5.0	1.0	10.0	10.0	1.0	Right	97.16 ± 6.36	0.18 ± 0.03	0.8896 ± 0.010
3	1.0	5.0	10.0	10.0	0.5	Right	81.96 ± 6.53	0.34 ± 0.04	0.9325 ± 0.0125
4	1.0	5.0	10.0	10.0	2.0	Right	100.57 ± 10.25	0.17 ± 0.01	0.9509 ± 0.0039
5	1.0	5.0	2.0	3.0	1.0	Right	89.70 ± 6.98	0.33 ± 0.05	0.9506 ± 0.0141
6	1.0	5.0	10.0	10.0	1.0	Left	44.54 ± 1.64	0.27 ± 0.01	0.9428 ± 0.0082

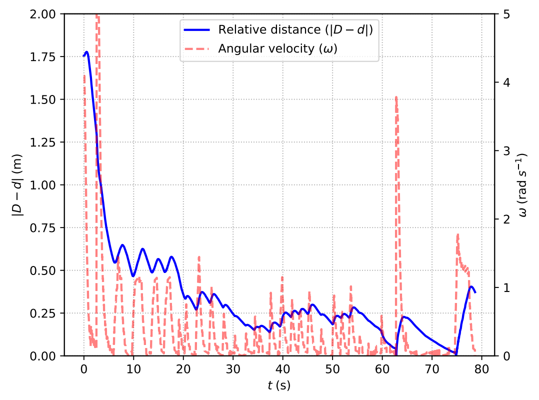


Fig. 5: Representation of correlation between angular velocity and relative distance error measured on Experiment 1

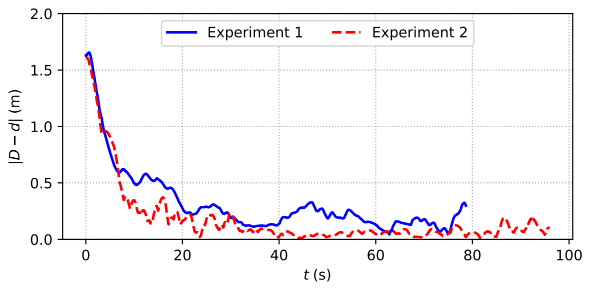


Fig. 6: Comparison between relative distance error measured on Experiment 1 and 2

followed by an increase in relative distance. This is explained by the fact that heading adjustments are a response to the detection of wall tangent variations, and thus prioritizing a parallel trajectory deviates the robot from the target distance.

When the robot approaches a wall head-on, its angular velocity sign fluctuates due to the discontinuity in its function (Subsection IV-C) causing it to collide and get stuck. We were unable to reproduce these conditions from a random starting position, instead, we were required to purposefully place the robot in specific locations. Although uncommon, this could be mitigated by introducing a slight bias in the calculation to ensure the robot never gets stuck unable to decide which direction to go towards.

VI. Conclusion

Our work proposes a minimalist approach to a wall-following robot, across both the hardware and software components. With that in mind, we achieved a solution that correctly traverses the necessary path (circumventing the question mark), and reliably reaches and stops on the proposed final position.

It has some limitations, namely, the simulation environment it was tested under is simplistic, therefore, under a real-world scenario, the robot could face issues it was not designed to deal with, such as sensor failures, reflections, or uneven terrain.

In future work, we could add more noise to both the sensors and actuators and adapt the algorithm to deal with those issues. Another facet that can be improved upon is the stability of the movement, which could be improved using a proportional integral derivative (PID) controller, for example.

References

[1] A. Bemporad, M. Di Marco, and A. Tesi, "Wall-following controllers for sonar-based mobile robots," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 3, pp. 3063–3068 vol.3, 1997.
[2] H. Suwoyo, Y. Tian, C. Deng, and A. Adriansyah, "Improving a wall-following robot performance with a pid-genetic algorithm controller," in *2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, pp. 314–318, 2018.
[3] I. Hammad, K. El-Sankary, and J. Gu, "A comparative study on machine learning algorithms for the control of a wall following robot," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2995–3000, 2019.
[4] I. Hammad, K. El-Sankary, and J. Gu, "A Comparative Study on Machine Learning Algorithms for the Control of a Wall Following Robot," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2995–3000, Dec. 2019.
[5] R. Braunstingl, "Fuzzy Logic Wall Following of a Mobile Robot Based on the Concept of General Perception," *ICAR '95, 7th INTERNATIONAL CONFERENCE ON ADVANCED ROBOTICS*, 1995.
[6] K. Bayer, "Wall Following for Autonomous Navigation," 2012.
[7] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, p. 381–395, jun 1981.
[8] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.
[9] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154 vol.3, 2004.
[10] "Puck LITE Lightweight Surround Lidar Sensor." Accessed on 22.10.2023.
[11] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, *The method of least squares*, pp. 329–340. London: Springer London, 2005.