Reactive robot path-finding in a tilde-shaped map

1st João Alves
MEIC
FEUP
Portugal
up202007614@g.uporto.pt

2nd Marco Rocha
MEIC
FEUP
Portugal
up202004891@g.uporto.pt

3rd Rúben Monteiro
MEIC
FEUP
Portugal
up202006478@g.uporto.pt

***Abstract*—This paper presents the development of control laws for two differential-drive, reactive robots with wall-following and robot chasing behaviors, respectively. The implementation was carried out in *ROS2*, with the *Flatland2* simulator used for validation and evaluation. The methodology employs a *LiDAR* (Light Detection and Ranging) sensor alongside simple controllers to manage the robot's motion. The team's experimental results confirmed the system's functionality and highlighted the importance of parameter tuning for optimal outcomes.**

***Index Terms*—reactive robot, wall-following robot, robot chasing**

I. Introduction

In the field of robotics, real-time navigation tasks frequently require control architectures that are both efficient and straightforward, especially in scenarios where particular mapping or sophisticated computational resources are either impractical or unnecessary. Reactive robots serve as a good example of this methodology, as they operate based only on immediate sensor inputs without relying on memory. These robots utilize basic control algorithms and sensors, such as *LiDAR* (Light Detection and Ranging), to engage with their environment without the need for an internal representation of the world. This reactive paradigm facilitates rapid decision-making and effective navigation in dynamic surroundings.

This paper details the design and execution of a reactive, differential-drive robot that is capable of performing wall-following and inter-robot chasing behaviors. The main aim of this project, explained in Section III, is to establish a reliable wall-following strategy for a blue robot, which is designed to maintain a consistent distance from walls, while a second, faster red robot tracks the wall until it can follow the blue robot. The robots, design, implementation and architecture explained in Sections IV and V respectively, were simulated in the *Flatland2* environment within *ROS2*'s "Humble Hawksbill" version, allowing for the validation of their functionalities and highlighting the significance of precise parameter tuning for achieving behavioral accuracy and consistency. Through experimental analysis, presented in Section VI, this study investigates the application of two simple control laws to realize effective path-following and chasing behaviors within a reactive robotic framework. Finally, proper results, discussion and conclusions are displayed in Sections VII and VIII.

II. State of the Art

Reactive robotics, also referred to as behavior-based robots, perform tasks through simple primitive behaviors, instead of complex and though out actions [1]. This approach serves as a simple solution to two main problems: the close world issue, and the frame problem.

These two problems describe the limitations of creative a robot that interacts with its environment. In short, close world problem exposes that the robot cannot know all of its environment in real time, as it is too much information to process [2]. The frame problem, however, explains the difficulty/impossibility of representing all the knowledge a robot would gather from its environment [3].

This approach, of purely reactive robots, comes with inherent limitations. One such is the lack of a memory, meaning that the robot cannot learn, and therefore, it is limited to the environment its programmed behaviors can withstand. A solution for this problem, and eventual surpassing paradigm, would be a hybrid deliberative/reactive proposal [4].

For the purposes of this study, the purely reactive paradigm is studied in lieu of other approaches.

III. Problem Definition

The goal of this project was the development of two 2D reactive robots designed to track and follow a tilde-shaped wall within a specific and stable distance. The first robot, the one in blue, should follow the wall while the red robot, which is substantially faster, should either follow the wall or tail the first robot, whichever is closest. The simulation map at the start is as presented in Figure 1.

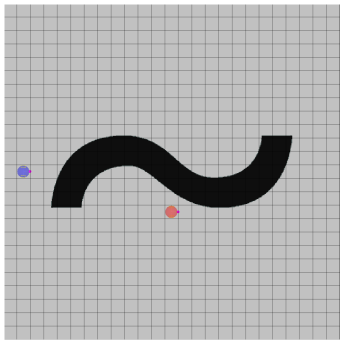


Fig. 1. Simulation Map with both robots in the starting positions

IV. Robot Design

The robots share a similar physical design, with minor differences to support their respective control behaviors. Each robot consists of a main circular base, two wheels, and a set of sensors for navigation and collision detection:

- **Body Components**
  - Circular base with a radius of 0.45 meters;
  - Two wheels welded on both sides of the base for stability and motion control via differential locomotion;
  - Small colored front component serving only as indicator and not impacting dynamics;
- **Plugins and Sensors**
  - DiffDrive plugin to control movement, subscribing to unique command velocity topics and publishing odometry data;
  - Bumper plugin for collision detection, publishing to distinct collision topics;
  - 360-degree laser scanner that detects walls within a 10-meter range, divided into 90 evenly spaced rays (4 degrees intervals), ensuring, to some extent, a comprehension of the world state;

Additionally, R2 has a second laser scanner specifically configured to detect only R1, taking advantage of the concept of layers in *Flatland2*, enabling it to switch between wall-following and trailing R1 based on the environment. The values of this laser are parsed before being used in the reactive component to make sure the robot can't use any laser that should be occluded by the wall, which is done by comparing each ray with its analogous pair wall laser.

A. Parameterization

The behavior and control of both robots are defined and finely tuned through a set of parameters defined in a Python file as global variables. These values allow for a flexible configuration for both general settings and robot-specific behavior. For instance, there are general settings related to logging and statistical analysis collection via performance tracking. Concerning robot-specific parameters, there are speed predefined maximum limits (*LIN_VEL_MAX* and *ANG_VEL_MAX*) and also wall distance related constraints (*IDEAL_DISTANCE* and *IDEAL_DISTANCE_TOLERANCE*). Additionally, for the following robot, there are two exclusive parameters related to its speed boost in relation to the slower robot and the distance at which it should start following it (*SPEED_DIFF2* and *FOLLOW_DISTANCE*).

This structured parameterization contributes to a swifter debugging and dynamic control adaptation of the robot's behavior.

V. Robot Architecture

This section gives an overview of the working of two reactive robots' control systems, Robot 1 and Robot 2, from here on out referenced only as *R1* and *R2*, respectively. By discussing topic data handling and overall decision-making, we give a abridged look on how the robots can navigate independently based exclusively on sensory data and without any memory or communication.

A. Robot 1 Architecture

R1 is designed to continuously monitoring its environment through a laser scan and navigate through the world with a wall-following behavior, keeping a consistent distance from said wall on its right side, thus circling the wall in a clockwise fashion. The condensed architecture is as follows:

1) **Data Acquisition:** R1 subscribes to laser scans (*/robot1/static_laser*), odometry (*/odom1*), and bumper collision data (*/collisions1*). This data provides real-time information on the robot's surroundings.
2) **Decision Process:** The robot parses the received laser scan data, allowing it to calculate distances to the front and right obstacles. Based on these measurements, R1 determines appropriate linear and angular speeds through a *follow_wall* function.
3) **Control Execution:** The computed speed commands are published to the */cmd_vel1* topic, dictating the robot's movements. This wall-following control law enables R1 to maintain an ideal distance from the wall, adjusting its heading as needed to stay aligned with the wall surface.

B. Robot 2 Architecture

R2, besides having a higher linear and angular speeds than R1, operates in a more complex manner, choosing between following R1 or a nearby wall depending on its sensory data. Its architecture is as follows:

1) **Data Acquisition:** R2 subscribes to two separate laser scan topics: one for walls (*/robot2/static_laser1*) and another for tracking R1 (*/robot2/static_laser2*). Additional odometry (*/odom2*) and collision data (*/collisions2*) are also received.
2) **Control Logic**
   a) **Following Robot 1:** If R1 is detected within a specific range, R2 follows it in the shortest path possible.
   b) **Wall-Following:** When R1 is not in range, R2 shifts to wall-following behavior.
3) **Control Execution:** The selected speeds (based on the presence or absence of R1) are published to */cmd_vel2*, guiding R2 to either tail R1 or follow a wall independently. This dual-mode controller allows R2 to adjust its navigation dynamically based on environmental factors.

This robot operates using a dual-mode control, making a binary decision of choosing which reactive behavior to adopt. R2 subscribes to two laser scan topics to distinguish between R1 and the walls in its environment. This information is processed to selectively "ignore" distances that should be blocked by walls while identifying the closest visible distance to R1.

When R1 is not within a designated range (*FOLLOW_DISTANCE*), the wall-following behavior kicks in and mirrors the wall-following control law of R1. Otherwise, the robot calculates the direct angle and distance and moves towards the other robot with a linear speed proportional to the distance. This guarantees that R2 can rush and catch up to R1 and makes it impossible to ever collide with it as R2 gets progressively slower the closer it gets and can even go back if the distance becomes less than 1 meter.

VI. Robot Control Law Implementation

For the decision-making of both robots two different control laws were developed: one rule-based approach and a PD (proportional–derivative) controller. These two options will be further explored in the following subsections.

A. Rule-based Approach

The team started by devising a simple decision-making process that revealed itself to be quite effective for the problem at hand, requiring in practice only three lasers (as shown in Figure 2) to implement and demanding only a few cases as control laws, presented in descending order of priority:

1) **Obstacle in Front:** If an obstacle is directly in front, the robot stops and rotates left in-place to avoid a collision.
2) **No Wall on Right:** If the right-side ray is infinite, there is no wall there. Hence the robot rotates in-place until it finds the wall on its right to grab onto.
3) **Right Curve Detection:** When the right-front distance is shorter than the right-side distance, the robot detects a curve and adjusts its heading to the left to follow the wall's curve.
4) **Distance Adjustment:** Focusing on the right-side distance, it should ideally be perpendicular to the wall and kept relatively consistent. Thus, if that distance is too large, the robot turns right, otherwise left.
5) **Ideal Wall Distance:** When the ideal distance is achieved, with a small distance tolerance delta to avoid continuous adjustments, the robot makes only small angular corrections to stay on course.

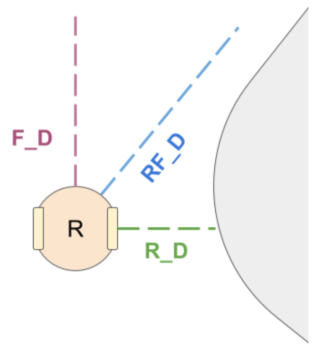


Fig. 2. Visualization of the three rays used

The above described conditions produce linear and angular velocities that are then published, directing the robot's movements in real-time. It is worth noting that, for the sake of simplicity, the team always assumes that the robot will be placed within 10 meters of any wall to remove the need for aimless wandering to find it. This facilitates the initialization of any simulation as the *No Wall on Right* condition will only happen once at the beginning, if it even happens, to first find the wall and it never leaves its right side again.

The angular velocity is always proportional to the angle error, within the set limits, thus leading to swifter adjustments. The linear velocity is kept at 50-70% of the max for the wall-following conditions for finer adjustments. The two high priority cases of *Obstacle in Front* and *No Wall on Right* both exhibit an angular rotation at max speed and null linear speed to quickly deal with the problem without risking moving out of place and colliding. Finally, the max linear speed is only achieved when the robot finds itself in the narrow range of ideal distance plus or minus the tolerance interval.

B. PD Controller Approach

The second method employs a PD controller as a control law for the wall-following behavior. Instead of using only three rays, the team used the entire 180 degree field of the right of the robot as depicted in Figure 3. By averaging all rays that hit a target, the robot can develop a sense of the average distance of anything to its right, independently of the wall shape or curvature.

There are a few additions to make the behavior more reliable: first, at the start, the robot safely spins in-place until finding something to its right to start linearly moving; second, there is a fail-save override at the end of the control function that checks if the front of the robot is unblocked and is safe to move there (if it is not, the PD decision is ignored and the robot starts to slow down and rotate left to dodge it).

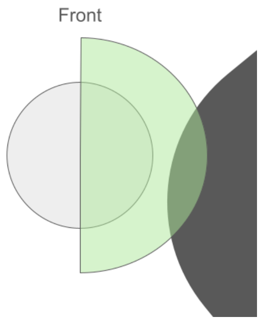


Fig. 3. PD Controller LiDAR use

VII. Experiments

The team explored other approaches until settling with these two as the others were discarded for being even worse or not promising. For example, a common solution for these kinds of problems would be the implementation of a PID (Proportional, Integral, Derivative) controller which was briefly explored by the team but the results were far from optimal, with very inconsistent path taking. The situation improved when the 'I' part, responsible for accumulated error over time, was removed as it was actually hurting the system which requires quick response and fine-tuning to minimize overshoot.

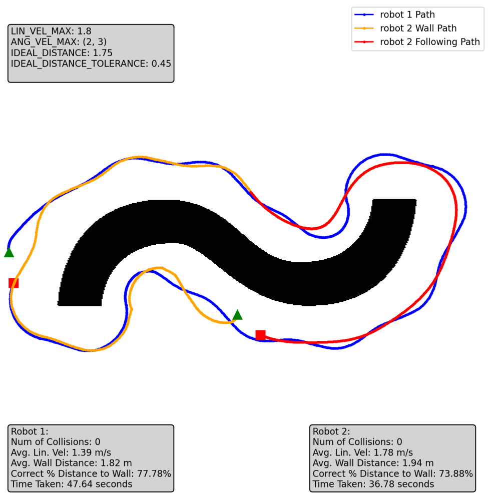
VIII. Results and Discussion

To make it easier to test the developed control laws, a script for batch running episodes was developed. This script starts a new world simulation and runs an episode (a single lap) with specific defined parameters in the configuration file. In essence, the script does a grid search with all possible configuration combinations and stores all result data into a CSV file. The following subsections will dive into further conclusions of the collected results for the final settled decision-making system.

A. Loop Path

A single full episode loop path for both robots can be seen in Figures 4 and 5, containing also the parameters used and the overall of-interest collected statistics.

In the figures, it can be clearly seen that the robots can do a relatively good job at keeping their distance from the wall with either approach. However, there are two things worth noting for the rule-based approach on Figure 4: first, the wall-following control law is not ideal in the two concave parts of the tilde, getting excessively close and having to reduce speed to correct their course; second, the path taken by the robots is not perfectly linear, having some waves. These waves happen as a result of the robot being at the ideal wall distance, correcting itself more strongly whenever the distance tolerance is crossed (which is corroborated by higher amplitude waves whenever the tolerance is increased).



```
LIN_VEL_MAX: 1.8
ANG_VEL_MAX: (2, 3)
IDEAL_DISTANCE: 1.75
IDEAL_DISTANCE_TOLERANCE: 0.45
```

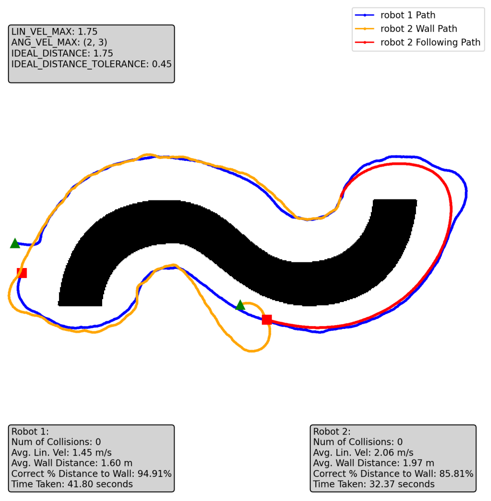
```
Robot 1:
Num of Collisions: 0
Avg. Lin. Vel: 1.39 m/s
Avg. Wall Distance: 1.82 m
Correct % Distance to Wall: 77.78%
Time Taken: 47.64 seconds
```

```
Robot 2:
Num of Collisions: 0
Avg. Lin. Vel: 1.78 m/s
Avg. Wall Distance: 1.94 m
Correct % Distance to Wall: 73.88%
Time Taken: 36.78 seconds
```

Fig. 4. Rule-based Control Law Episode

The emergence of these waves is not ideal and served as the catalyst for the team to pursuit a new solution: a PD controller. In comparison with the previous method, the path is much smoother (see Figure 5). There are still some minor disturbances in the path, mainly due to the sharp corners, that briefly confuse the robot but they are not very pronounced (and they are still a big improvement from the previous path drawing).

The choosing of the sensitive values for *Kd* and *Kp* was a very finicky process due to drastically different outcomes.



```
LIN_VEL_MAX: 1.75
ANG_VEL_MAX: (2, 3)
IDEAL_DISTANCE: 1.75
IDEAL_DISTANCE_TOLERANCE: 0.45
```

```
Robot 1:
Num of Collisions: 0
Avg. Lin. Vel: 1.45 m/s
Avg. Wall Distance: 1.60 m
Correct % Distance to Wall: 94.91%
Time Taken: 41.80 seconds
```

```
Robot 2:
Num of Collisions: 0
Avg. Lin. Vel: 2.06 m/s
Avg. Wall Distance: 1.97 m
Correct % Distance to Wall: 85.81%
Time Taken: 32.37 seconds
```

Fig. 5. PD Control Law Episode

A lot of combinations were tested, with some not working at all and leading to robots losing themselves or going straight against the wall, until the most optimal configuration was reached.

Besides traveling time and drawn path, the main metric of interest being measured was the correct distance to wall percentage time. This was calculated by checking at each received odometry message, the position of the robot and the closest wall ray hit. By doing so, we could collect how much of the traveling time was done within the ideal distance +/- margin range. Therefore, ideally, a robot would be able to keep itself circling the wall at a consistent ideal distance and achieve 100%. In light of that, the episode heat-map maps for both robots regarding correct wall distance percentage are presented in Figures 6, 7.

The missing cells represent instances where either R1, R2 or both took more than two minutes to finish the episode and were thus terminated as failures. Moreover, there are some clear outlier results that don't fit the trend gradient that the heat-map describes.

This could happen for many reasons, like collision or robot leaving the map entirely however, the team testified multiple times that due to the absurd number of consecutive executions, the script would sometimes be delayed or straight out fail and either the robots would not receive published instructions or the *Rviz* map would not load the walls correctly.

Due to time constraints, these episodes were not retried but we think the present values already give a very good picture, showing the best possible achievable correct wall distance values to be 97 and 94 percent for R1 and R2, respectively.

Our data analysis further concluded that while there are value combinations that indeed maximize a specific metric, it normally comes at the cost of others. For example, a bigger *Kd* tends to indicate a lower correct percentage of wall distance, which happens because the path becomes more circular, disregarding the complexities of the tilde-shaped wall and in-turn causes traveling time to be slightly reduced. Moreover, again

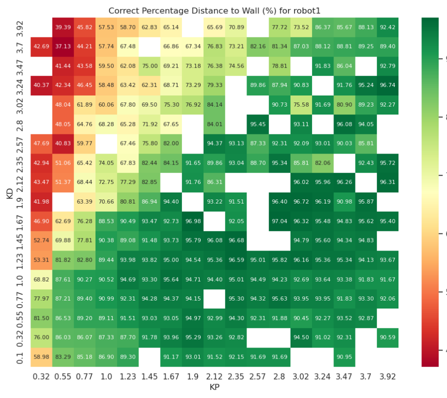


Fig. 6. R1: Correct Wall Distance Time Percentage

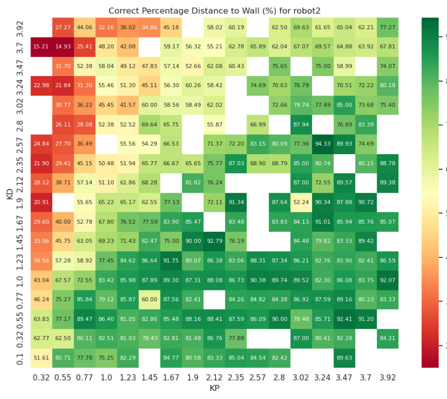


Fig. 7. R2: Correct Wall Distance Time Percentage

because of time constraints, the robots shared the coefficients in a single episode and so, what could be great for one, could be awful for the other as they behaved drastically differently due to their speed differences.

The path time remains fairly consistent between runs with modified parameters. The rule-based approach is a few seconds slower (as it can be seen on the statistics on the bottom left on Figures 5 and 4) mainly because of the artificial capping to lower than max speed during maneuvering for more movement stability.

The linear and angular speeds were notoriously influential in the behavior of the robots, as bigger values tended to make the robots harder to control effectively using either of the two control laws which couldn't respond fast enough to the changes taking place. Hence, the goal was to keep the speed as high as possible for faster laps, while keeping a good stable path-taking behavior.

In practice, the results are quite telling and positive. The first method yields consistently between 70-80% correct distance for both robots with the best parametrization tested, while the PD controller approach was consistently able to keep the robots within the correct wall distance for longer, at 80-95% of the circling time, improving on the base method.

B. *Stuck Robot and Collisions*

During all iterations of testing, there were seldom instances where the robot got stuck. However, there are some parameter combinations that inevitably cause collisions which are detected using the Bumper plugin for later statistical analysis. After inspection, these combinations were primarily the ones where the linear speed was great and the angular velocity was small, making it so the robot couldn't react in time to the wall. Moreover, during the entire testing, there was a single occurrence of an edge case where R2 blindly followed R1 and briefly collided with a sharp corner.

C. *Randomness*

To further verify the robustness of our solutions, randomness was added in the form of uncertainty in the *LiDAR* readings (and therefore more realistically simulating this kind of sensor) and by randomizing the initial robot position within a parametrized distance. After multiple episodes with different values, we can confidently say that both control laws were able to deal with these challenges, reinforcing the confidence in the solutions.

IX. Conclusions

In conclusion, the team was able to develop two very simple reactive robots that effectively circles a tilde-shaped wall keeping a stable distance from it without collisions. We acknowledge that the developed rule-base reactive control flow, while a solution for the problem at hand, is a somewhat simplistic representation of what can be achieved through reactive means. Thus, the team pursued a PD controller as a mean to improve the pathing of the robot and it was successful. Despite those results, some other improvements could have been made, like a subsumption architecture, which could further prevent collisions and make the decision-making more robust and efficient. Finally, while the PD solution was indeed better performing, the team still feels more confident about the base control law when considering the robot ability to do the full lap without issue as it was very challenging to get good PD coefficient values and, together with the randomness, the robot sensitiveness results in unstable outcomes each time it is executed like corroborated by the heat-maps.

References

[1] R. C. Arkin, "Reactive Robotic Systems," *The Handbook of Brain Theory and Neural Networks*, pp. 793-796, 1995.
[2] R. R. Murphy, *Introduction to AI Robotics*. MIT Press, 2019.
[3] R. C. Arkin, *Behavior-Based Robotics*. MIT Press, 1998.
[4] L. De Silva and H. Ekanayake, "Behavior-Based Robotics and the Reactive Paradigm: A Survey," in *2008 11th International Conference on Computer and Information Technology*, 2008, pp. 36-43. doi: 10.1109/ICCITECHN.2008.4803107.