# ROBO: Wall following Reactive Robot

1st António Martins
*Masters in Informatics and Computer Engineering*
*Faculty of Engeneering of the University of Porto*
Porto, Portugal
up201404422@fe.up.pt

2nd Arthur Matta
*Master in Informatics and Computer Engineering*
*Faculty of Engeneering of the University of Porto*
Porto, Portugal
up201609953@fe.up.pt

***Abstract*—This document describes the approach taken towards the implementation of a Reactive Robot capable of following a wall through the usage of laser sensors. The robot was designed and tested using a simulator provided by the *Robot Operating System (ROS)* ,version *Melodic Morenia*.x**

***Index Terms*—reactive, robot, walls, follow, smart**

## I. Introduction

Reactive robots are the most basic type of artificial intelligence systems. This type of robot uses its sensors to perceive the world directly and act according to it. They have the ability neither to form memories nor to use past experiences to inform current decisions - meaning they can't function beyond the specific tasks they're assigned and are easily fooled. The fact these robots will behave exactly the same way every time they encounter the same situation is what make them trustworthy [1].

In this scenario, our goal is to developed a reactive robot tasked to follow an angled wall in a simulated environment using *Robot Operating System (ROS)* and a *Simple Two Dimension multi-Robot (STDR)* simulator. In the State of the Art chapter we mention some modern reactive robots. The Implementation chapter describes the robot architecture. Following we have the Experiments and the Results chapters which describes the tests we performed and their results, respectively. We then conclude our work and suggest some future work.

## II. State of the Art

Robots capable of wall and obstacle tracking have existed for many years and range from simple reactive robots capable of following a curved wall with nothing but laser sensors, to more complex systems carrying an array of sonar, laser and often times cameras. An example of these kinds of robots are the robotic vacuum cleaners "Roomba". These robots rely on a combination of touch sensors, infrared sensors and cameras to path inside a house and clean floors and wall skirtings.

## III. Implementation

### A. Simulator

The STDR simulator is a simple Server-Client simulator capable of simulating two dimensional environments and a virtual robot with an array of sensor. It's modules and a brief description of each one of them can be found in the table bellow.

TABLE I
Simulator modules description

| Module | Description |
|---|---|
| stdr_gui | A gui in Qt for visualization purposes in STDR Simulator. |
| stdr_launchers | Launch files, to easily bringup server, robots, guis |
| stdr_msgs | Provides msgs, services and actions for STDR Simulator. |
| stdr_parser | Provides a library to STDR Simulator, to parse yaml and xml description files. |
| stdr_resources | Provides robot and sensor descripiton files for STDR Simulator. |
| stdr_robot | Provides robot, sensor implementation, using nodelets for stdr˙server to load them. |
| stdr_server | Implements synchronization and coordination functionalities of STDR Simulator. |

The project's implementation follows the simulator modular implementation with each module executing independently. A total of three modules were used, as follow:

- **Server**: this module is responsible for providing the simulation services required by the other modules. For this purpose, the standard server module that comes with the simulator was used.
- **Launcher**: this module is responsible for loading the interactive GUI, the map on which the simulation will run, and the robot. For experiment purposes, 4 launchers were created each with a different map.
- **Behaviour**: this module provides the robot with instructions on how to behave and it is the focus of this project.

### B. Robot's Architecture

Sensors are the main source of information for a reactive robot about the environment that surrounds it. The STDR package provides an example of a simple robot and many different types of sensors. For the purpose of this project, the use of laser sensors is enough as we only need information about distances and angles.

The virtual robot is equipped with 200 laser sensors capable of detecting objects from a distance of 0.1m to 2m and gather this information at a refresh rate of 10Hz. The sensors are distributed in a 200 degree range in front of it, allowing it to detect obstacles both ahead and to the sides.

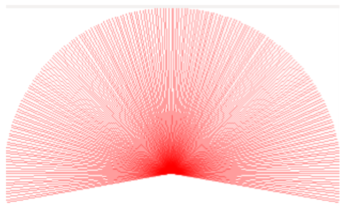
Fig. 1. Distribution of the laser sensors

Its movement decisions are based on how he perceives the environment (See Figure 2.). The robot starts to wander in the map until its sensors detects a wall, after which it starts to follow it - keeping a distance of 1.0m from the wall. The direction of the movement is determined by the range of sensors that detected the obstacle, as follow:

- Left, sensors ranging from 15 to 120 degrees;
- Right, sensors ranging from -120 to -15 degrees;
- Front, sensors ranging from -15 to 15 degrees.

The robot moves with a linear speed of 0.5 meters/s and turns with an angular speed calculated by the expression bellow

$$-15 * (sin(alpha) - (distance - ideal\_distance)) * Speed$$

where *alpha* is the angle the robot is currently facing; *distance* is the distance from the robot to the wall; *ideal˙distance* the ideal distance from the robot to the wall (1.0m); and *speed* the linear speed of the robot (0.5 m/s).

In an event where the robot is cornered, it will turn around with an angular speed calculated by the expression described above with linear speed of 0.1 meters/s.

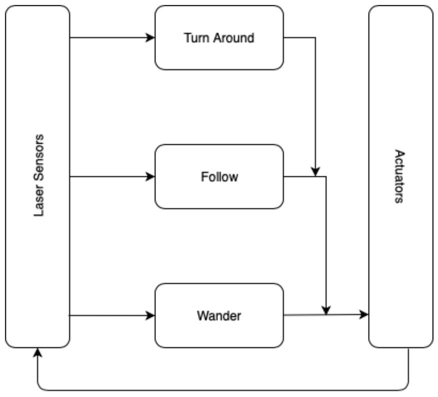

Fig. 2. Robot behaviour diagram

## IV. Experiments

Our experiments were performed in the simulated environment created by the STDR simulator. Four different maps were used to test the system: a filled V, a filled W, an unfilled W, and a filled C (Figures 3., 4., 5., and 6. respectively). For simplicity purposes, some assumptions were made:

- Walls are approximately 1 meter long
- Angle between two walls are between 0 to 180 degrees (except for C map)
- Wall thickness is between 0.2 and 0.6 meters (except for C map)

We started the system with the V-shaped map to test the robot capabilities when facing a angled wall.

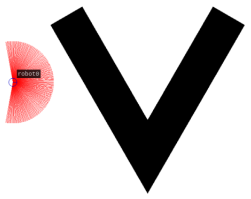
Fig. 3. Robot in V map

After the robot presented a good performance against the V wall, we switched to a W-shaped wall in order to increase the difficulty.

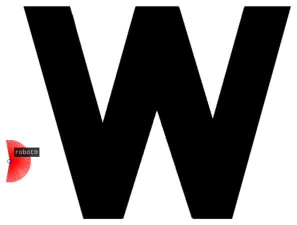
Fig. 4. Robot in W map

Furthermore, the robot was tested with a unfilled W shaped map. However, now the robot was placed inside the map, testing his tracking capabilities when presented with walls on both sides.

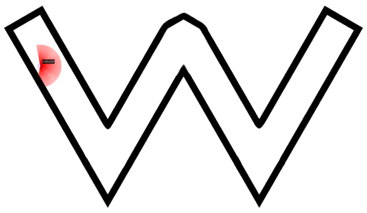
Fig. 5. Robot in unfilled W map

The last test was a C-shaped map to test the robot capabilities with curved walls.

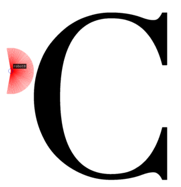

Fig. 6. Robot in C map

## V. Results

The robot has a number of variables that can be tweaked to attempt to improve its speed and efficiency. The metrics chosen for evaluating performance were whether the loop was completed or the robot wandered off the map and the time spent on the loop. The table bellow show the results obtained using the V-shaped map for multiple values of the robot linear speed and ideal distance from the wall, where *Loop* indicates that the robot performed a successful loop, and *Lost* indicates that the robot wandered of the map or was trapped spinning in the same point forever.

TABLE II
Results of the experiments on the V map

| Linear Speed | Ideal Distance | Action | Time(s) |
|---|---|---|---|
| 0.5 | | Loop | 85 |
| 0.75 | | Loop | 60 |
| 1 | 1 | Loop | 45 |
| 1.25 | | Lost | N/A |
| 1.5 | | Lost | N/A |
| 0.5 | | Loop | 90 |
| 0.75 | | Loop | 62 |
| 1 | 1.25 | Loop | 47 |
| 1.25 | | Lost | N/A |
| 1.5 | | Lost | N/A |
| 0.5 | | Loop | 95 |
| 0.75 | | Loop | 65 |
| 1 | 1.5 | Loop | 48 |
| 1.25 | | Lost | N/A |
| 1.5 | | Lost | N/A |

As can be seen from the table of results, maximising the speed results in more cases where the robot gets lost, but at 1 meter per second optimal results are found, with the optimal case being 1m/s with ideal distance equal to 1 meter.

## VI. Limitations and Future Work

Although the robot is capable of tracking and following walls very accurately and efficiently, there are some shortcomings that can be fixed through different algorithmic implementation and/or the usage of different sensor types.

The robot starts his routine by looking for a wall. This process has him continuously moving in a straight line until a wall is found. If the robot does not find a wall, he will leave the simulation area, resulting in a simulator crash. To fix this, a more robust wandering algorithm could be implemented to help the robot find walls more efficiently and avoid going out of bounds.

Speed is another limitation of the robot. The sensor refresh rate and the algorithm implemented does not allow for high speed turns and in most cases results on the robot losing track of the wall. Laser sensors are good for the purpose of this project but perhaps sonar sensors, or even a mix of both these types of sensors, could provide a better alternative. This could allow for faster robot movement due to the better obstacle detection.

## VII. Conclusion

Through testing in different environments we validated the robot's performance and the results were satisfactory. The robot was able to traverse the 4 different maps developed (V, W, unfilled W and C), following the walls and performing turns in both concave and convex parts, with an adjustable speed and ideal distance.

The implemented algorithm work as we intended and even surpass some of our expectations for this project but there are some key features already mentioned in the previous chapter that could be implemented to improve the results even more.

## VIII. Acknowledgements

We would like to thank both our teachers Armando Jorge Miranda de Sousa and Luís Paulo Gonçalves dos Reis for their guidance throughout the first half of the semester and all our colleagues that have helped us improve our skills in the area of informatics.

## References

[1] A. Hintze, "Understanding the four types of ai, from reactive robots to self-aware beings," November 2016. [Online]. Available: https://theconversation.com/understanding-the-four-types-of-ai-from-reactive-robots-to-self-aware-beings-67616
[2] J. M. O'Kane, *A Gentle Introduction to ROS*. Independently published, Oct. 2013. [Online]. Available: http://www.cse.sc.edu/ jokane/agitr/
[3] M. Tsardoulias, C. Zalidis, and A. Thallas, "Stdr˙simulator." [Online]. Available: http://wiki.ros.org/stdr_simulator
[4] J. Costa and M. Ferreira, "Reactive robot," 2018, supplied as additional material.
[5] M. Abreu, "Simple but efficient wall-following reactive robot," 2018, supplied as additional material.
[6] D. Duque and S. Fernandes, "Wall following robot," 2018, supplied as additional material.
[7] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," *ICRA workshop on open source software*, 2009.