Reactive Robot with Simple Reactive Architecture

Joel Carneiro, *Student, FEUP,* and Pedro Moura, *Student, FEUP*

***Abstract*—In this study, we propose a simple reactive architecture of a robot that can follow a wall. This requires the acquisition of data from different kind of sources (lasers, sonars, etc) in order to make the robot capable of following the wall when the robot find it. In this proposal, we use Robot Operating System (ROS) for compiling and executing the reactive architecture of the robot and the Simple Two Dimensional Robot Simulator (STDR Simulator) to simulate the environment. The robot will try to find the wall that in our approach has the "D" character shape and then it will follow either the outside or the inside of the wall forever.**

***Keywords*—reactive, robot, paper, wall.**

I. Introduction

ROS is a meta-operating and open-source system for use in a robot implementation. It provides services like an operating system, including hardware abstraction, low-level device control, among others [1].

STDR Simulator implements a distributed based on server-client architecture. Each STDR Simulator node can run on a different machine and communicate through ROS interfaces. The most important characteristic of STDR Simulator to this study is that it provides a GUI developed in QT, that allows visualization of the environment. The GUI can be performed using command-line tools.

Reactive architectures have been used in many robots, with distinguished use. There are three types of robot architectures, Hierarchical, Reactive and Hybrid. In this proposal will be used the reactive architecture for the robot that follows a wall. This architecture style has some main ideas: Modules should be grouped into layers and modules in a higher level can subsume or replace behaviors at a lower level [2]. In other words, the subsumption architecture relies on the presuppose of splitting complex behaviors in simple ones. Thus, this architecture is composed by a state machine that has simple behaviors to translate the complex behavior of the robot, without any central control. However, in many cases, the system can have multiple goals in parallel. Thus, some goals of one layer could conflict with other goals from another layer. In these cases, higher-priority goals should override the lower-priority goals. In short, subsumption architecture relies on the subsumption of the lower-layer simple tasks by the higher-layer ones to have complex behaviors. [3].

With this architecture, it's possible to build a reactive robot that follows a wall with simple lines of code.

The next section presents the architecture of the robot used in this study, *Section III* show the results obtained after testing the implementation of the reactive robot. Moreover, in *Section IV* are presented the found limitations in this implementation and the last section presents the conclusions of this study.

II. Architecture

In this section will be described the reactive architecture implemented. Reactive architectures have been used with distinguished use in many robots. In this proposal was used the reactive architecture for the robot implementation. Thus, the priority of the layer levels in the architecture was chosen in account with the goals of the study.

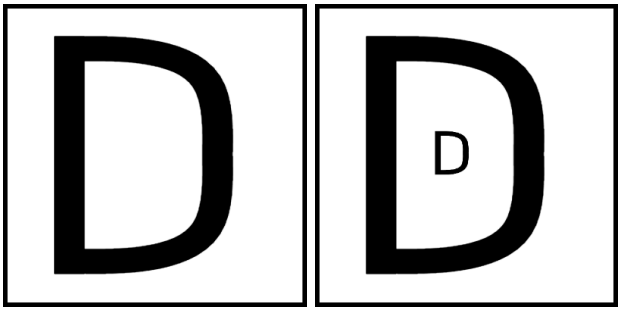

Fig. 1. "D" shaped wall's map.

When we run the code we can specify a limit to X and Y (maximum and minimum) to the position where the robot will start. Hence, the robot will start in a random place between the X and Y specified. In another hand, we can run the code without specifying the limits to the start place of the robot. In this case, the robot will start from the position that it has in the STDR simulator. Furthermore, the robot also starts with a random rotation angle.

The robot has two main goals: a) searching for a wall; b) following a wall. About the first mentioned goal, we have some important questions here. When the robot is searching for a wall, it evaluates the sonar sensors in order to know if they hit something. When a sonar sensor hit something, the robot will only follow that sonar sensor and ignore the other one. This allows having a simple algorithm to deal with the tracking of the wall. With this approach, we don't have to deal with different readings from the two sonars, hence, it's more easier to build a mathematical expression for the robot behavior. Still related with the searching a wall goal, the robot evaluates if the sonar sensor reading has something and we have two situations: a) The wall is out of the sonar sensor range and the robot increases it linear velocity to faster find a wall. b) The wall is at the sonar sensor range, the robot set its linear velocity to fast linear velocity and it knows that the wall is found.

Now, the robot knows where there is the wall. Some previously specified range limits are used to allow the robot to follow the wall: a) minLimit, that is the minimum distance admitted between the wall and the robot; b) minRange; c) maxRange, these last ones (b) and c)) are the limits of the

optimal interval of distances between the robot and the wall that allows to the robot has a good tracking of the wall. To be more clear, when the robot follows the wall it needs to be in a good distance to the wall. The minRange and the MaxRange are the limits of this interval; d) maxLimit, that is the maximum distance admitted between the robot and the wall: e) outOfRange, that means that the sonar sensor cannot reach the wall.

Hence, the robot will have different behaviors taking into account some different sonar sensors readings: a) if the range of sonar is less than the minLimit specified, the robot will stop its linear velocity, which means that the robot will stop; b) if the range of the sonar sensor is less than the minRange specified, the robot will turn to the opposite direction, which means it increases it angular velocity and its linear speed is set to low linear velocity; c) if the range of the sonar sensor is less than the maxRange and greater than the minRange, the robot will stop turning, which means that its angular velocity is zero and its linear velocity is set to fast linear velocity; d)if the range of the sonar sensor is less than the maxLimit and greater than the maxRange, the robot will turn into the wall direction. Thus, the robot increases its angular velocity and set its linear velocity to low linear velocity; e)if the range of the sonar sensor is less than the outOfRange and greater than the maxLimit, the robot will put its linear velocity at zero and it will turn to the wall direction by changing its angular velocity; f)if the range of the sonar sensor is greater than the outOfRange, the robot will turn to the wall direction and it will set its linear velocity to fast linear velocity.

Figure 1 shows the architecture of the reactive robot.

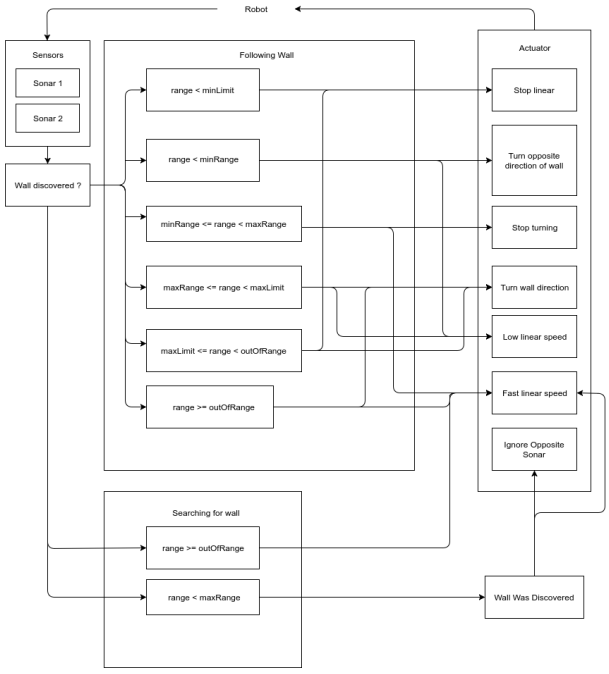

Fig. 2. Reactive Architecture of the Robot.

In this study, it was analyzed the ideal values to the minLimit(1), minRange(2), maxRange(3) and maxLimit(4) values to have a better tracking of the wall. All of this values depend on the maximum range of the sonar sensor, taking into account the ”D” shaped maps that we used. In minLimit value case, we divide the maximum range of the sonar sensor in eight equals parts. In minRange value case, we multiply two times the minLimit value, thus, the minRange value equals to the maximum range of the sonar sensor divided by four. The maxRange value is minRange multiplied by two, hence, the maxRange value equals to the maximum range of the sonar sensor divided by two. The maxLimit value is the minLimit multiplied by seven, so, this value is seven-eighths of the maximum range of the sonar sensor. These values allow the robot to have a good tracking of the wall, without being too far or too close to it.

In order to an easier reading, the limits specification can be seen below.

```
1 #define minLimit(max_range) max_range / 8.0f
2 #define minRange(max_range) 2.0f * max_range / 8.0f
3 #define maxRange(max_range) 4.0f * max_range / 8.0f
4 #define maxLimit(max_range) 7.0f * max_range / 8.0f
```

However, when we the robot are following the inner ”D” shaped wall it could lose the tracking when it arrives at one of the two abrupt wall’s turnings. Here, the sonar sensor cannot reach anything. Thus, the robot will try to find a wall again. In this case, the robot will have into account only the sonar sensor that has used to track the wall. Moreover, the robot could find a different wall. To overcome this we added a behavior that if the range of the reading is greater or equals to outOfRange, the robot will turn in the wall direction and its velocity is set to fast linear velocity. With this kind of hack, the robot can overcome the abrupt wall’s turnings and keep tracking the wall.

III. Results

After writing the code and test the reactive robot architecture some results have been achieved. As described in the previous sections the reactive robot relies on the subsumption architecture, thus, the robot complex behavior was decomposed in simple behaviors, without a central control. This allows the robot to follow the wall successfully.

The ”D” shaped wall has two abrupt turnings in the transition of the curve line to the straight line. Due to this, the robot could bump into the wall or simply lost the wall’s tracking. In this robot implementation, the most difficult part was the capability of do not lose the wall when we have an abrupt wall’s turning. However, our approach showed that it is capable of dealing with this kind of problems, taking into account the two specifics maps used. In the curve part of the ”D” shaped wall, the robot is always adjusting its position (using linear and angular velocity) in order to follow the wall and don’t bump into it. In another hand, in the straight part of the wall, the robot just continue with the same linear velocity without any angular velocity added, keeping the same distance between it and the wall.

Our implementation allows that someone specify the X and Y limits to the start place of the robot. After that, the system

will randomly set the start place of the robot somewhere inside of the limits. If, after this random calculation, the robot start place collides with any wall, the system will try another random. The system will do it until fifty times, after that, if always get into a wall, the start place of the robot is the position at the STDR Simulator. Thus, the robot always starts in a place that isn't the wall.

After trying several times, running our code, the robot always was capable of following the wall forever, without ever bumping into it.

IV. Limitations

Analyzing the results we can say that there are some limitations in this reactive architecture. One of the most explicit limitations is the abrupt wall's turning issue. Our approach effectively solves this problem in the proposed maps, however, if the shape of the inner wall is too small the robot possibly could lose the wall's tracking. If we are looking into the outer wall it will be different. We don't have the same kind of abrupt wall's turning because the sonar always hit the wall.

In our implementation, the robot follows the sonar sensor that hit first the wall. Hence, if the sonar sensor has some kind of malfunction, or if it simply doesn't work after the robot finds the wall, the robot will lose the wall's tracking and it will try to find a wall again. However, in the new tries, just the other sonar sensor is working correctly. Due to this, wall's tracking loses will occurs frequently.

Another important limitation is that our approach doesn't deal with any kind of obstacles. Thus, if some obstacles are added to the maps the robot cannot follow the wall and it will be stuck at some obstacle, following it forever. In order to overcome this limitation, in a future work, the reactive architecture should include a simple behavior that allows the robot to avoid the obstacles.

Another limitation, however, a smaller one, is the fact that the robot will follow the first wall that it finds. Due to this, if for some reason you want that the robot follows the outer wall and it finds first the inner one, it is necessary that you change the position of the robot so that it finds the wall again and vice-versa.

V. Conclusion

In this paper, we propose a simple reactive architecture of a robot that can follow a wall. With the base on simple behaviors in order to have a complex one, the reactive robot developed pass in the main goals of this study. With this study, it was proved that it's possible to build a robot that follows a wall forever, without bumping into it, with a reactive architecture based on subsumption of the smaller-priority behaviors.

Some difficulties were detected and then overcome. One of the most difficulties was the wall's continue tracking when the robot faces an abrupt wall's turning. Without an extra effort in this matter, the robot could go against the wall or could lose the tracking of the wall.

However, this implementation may not be the best one in order to have the shortest path until the robot finds the wall and when the robot follows it. The path just depends on our algorithm that uses the sonar sensors results. Thus, some adjustments to the algorithm can bring different paths and wall's tracking problems.

In a future work, it can be added other simple behaviors to deal with obstacles. In the present study, the maps used just have the wall, or two walls (inner and outer wall), without any obstacle. Hence, the robot can complete its task. In another hand, if we add some obstacles to this maps the robot will be stuck in one of them and cannot accomplish its mission. It will be a good advance if it is added a new behavior that is capable of dealing with the obstacles issues.

In our approach, the robot begins its search for a wall with a random rotation and then simply walks forward in a straight way. This means that we cannot predict the wall that the robot finds first. Because of that, it could be very difficult to put the robot following a specific wall that we want. Some consideration should be done and this behavior should be studied to have a better approach on that. However the main goal of the study is to develop a reactive robot that follows a wall, and it does it.

References

[1] D. Thomas. Ros/introdution, 2014.

[2] H. Yanco. Lecture notes on robot architectures, 2014.

[3] Eyal Amir and Pedrito Maynard-Zhang. Logic-based subsumption architecture. *Artificial Intelligence*, 153(1):167 – 237, 2004. Logical Formalizations and Commonsense Reasoning.