ROBO 2017/2018 Assignment 2

João Gabriel Marques Costa
Faculty of Engineering
University of Porto
up201304197@fe.up.pt

Gustavo de Castro Nogueira Pinto
Faculty of Engineering
University of Porto
up201302828@fe.up.pt

***Abstract*— In this document we will explore working with a robotic simulation environment (ROS and STDR) as well as the concept of reactive robot behavior, more specifically, we'll explore the Subsumption Architecture.**

Keywords – simulation; robotics; reactive behavior; subsumption architecture; artificial intelligence;

I. INTRODUCTION

The project was developed with the goal of both exploring Robot Operating System (ROS) and Simple Two Dimension Robot Simulator (STDR) as well as get a better understanding of reactive robot behaviors and implementations.

The simulation is set up a with a STDR robot equipped with two sonars. Each sonar has a range of 2 meters and a 50 degree arc. They are 90 degrees apart: one of them is effectively sensing the "front left" and the other the "front right" of the robot, angled at 45 degrees and -45 degrees respectively.

The simulation is run on a map (Figure 1) which consists of an inner *D* shaped obstacle which is, itself, inside a *D* shaped wall.

NOTE: Sonar measurement readings in the code are treated as a percentage of the distance read based on the maximum range of the sonar.

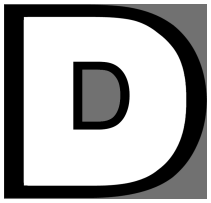

Fig. 1: Main simulation map

II. ARCHITECTURE

Subsumption architecture is a control architecture that, instead of guiding behavior by symbolic mental representations of the world, couples sensory information to action selection in an intimate and bottom-up fashion.

It does this by decomposing the complete behavior into various sub-behaviors. These sub-behaviors are organized into a hierarchy of layers where the lower layers have precedence over the higher ones. The layers, which all receive sensor-information, work in parallel and generate outputs. These outputs can be commands to actuators, or signals that suppress or inhibit other layers.

In this assignment's specific implementation, which is depicted in Figure 2, since the sensory information is relatively small, the layers aren't operating in parallel but are simply a sequence of *if's* and *else's* in which the lower (higher precedence) layers are placed at the top of the chain.

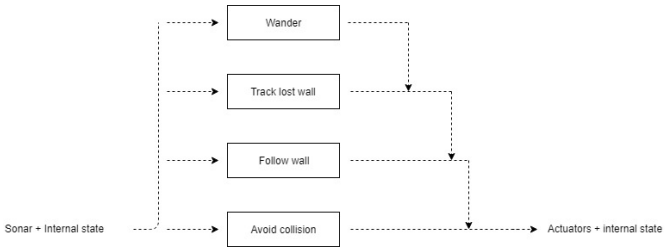

Fig. 2: Subsumption Architecture diagram

A. Avoid Collision

Avoiding any collision takes precedence over any other behavior. When any of the sonars reads a distance that is below the defined threshold (in this case 10% of the sonar's range), the robot must immediately react.

Table I shows the defined action for each case, with both sonars.

TABLE I: Behavior of "Avoid Collision" state

	Left Sonar	
Right Sonar	**Above Threshold**	**Below Threshold**
Above Threshold	—	Stop and rotate to the right
Below Threshold	Stop and rotate to the left	Stop and rotate right

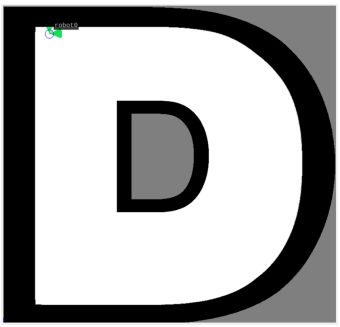

Fig. 3: Situation where both sonars are below the threshold (stop and turn right)

B. Follow Wall

When any of the sonars registers a distance that is not below the previously defined threshold, the Follow Wall layer will be active. In this state, the robot will attempt to follow a wall, in parallel, at

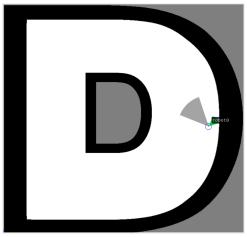

Fig. 4: Robot following outer wall

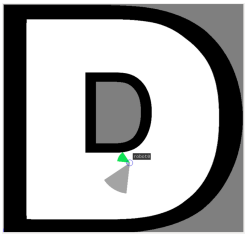

Fig. 5: Robot following inner wall

a fixed distance (25% of the sonar's range in the implementation). To achieve this, the robot will maintain a given linear velocity (of 0.6 units) and adjust its angular velocity by the formulas described in equations 1 and 2 if following a wall to the left or to the right, respectively:

$$angular.z = -4 * (0.25 - LeftDistance) \quad (1)$$

$$angular.z = 4 * (0.25 - RightDistance) \quad (2)$$

TABLE II: Behavior of "Follow Wall" state

	Following		
Sonar	**None**	**Left**	**Right**
Left	Follow Left	Follow Left	—
Right	Follow Right	—	Follow Right

C. Track Lost Wall

If the robot is following a wall and suddenly looses signal from both sonars, it will, for 1 second, try to recover the signal by stopping in place and rotating towards the direction of the wall that was previously being followed.

This effectively covers the edge case of sharp corners, where by the robot following the wall parallel to it leads to losing signal abruptly once reaching the corner, effectively not having enough time to compensate. With this behavior in place, once the robot suddenly loses signal, it will stop, turn in place and, very quickly, regain signal back (if it really is a corner).

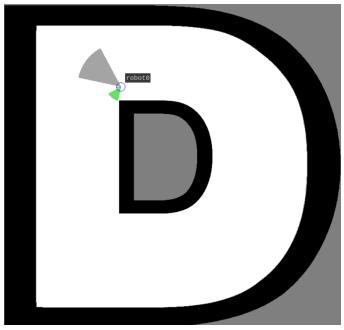

Fig. 6: Situation where the robot is about to suddenly lose signal from the left sonar upon reaching the corner

D. Wander

The last and default layer is where the robot doesn't have any sensory information from the sonars and didn't regain track of the lost wall after 1 second. When in this state, the robot effectively wanders around randomly forever until it gets any signal back from the sonars.

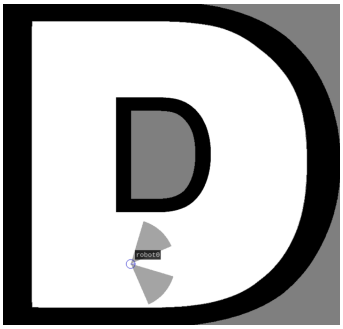

Fig. 7: Robot wandering around in search of a wall to start following

III. RESULTS

The emergent complex behavior based on the effective cooperation of the previously described layers is that of a robot that can seamlessly wander around a map and follow any wall that it finds, both inner and outer, without bumping or crashing into them. It is also able to "intelligently" recover from any lost wall, which enables it to follow around corners.

IV. LIMITATIONS

Through the extended testing of the simulation in different conditions, some undesired situations were found:

1) If the robot gets too close to an outer corner facing it's direction, it can get stuck by repeatedly alternating between the two avoid collision actions by having each sonar report below threshold sequentially
2) Due to the nature of *if* precedence in the subsumption behavior implementation, some cases can happen where the robot is following a wall and, upon reaching a corner, it will loop back around because, while rotating away from the corner, the opposite sensor gets triggered

Also, as it is implemented at the moment, it is not possible to have multiple robots in a single simulation.

V. CONCLUSIONS

Subsumption architecture provides a low computational effort and a relatively quick way to implement a working solution for a simple robot with a basic mission and no need for direct knowledge of the world it is acting on. However, as we found out during development, it isn't without it's downsides. Due to its nature, it isn't easily scalable, as it is hard to come up with the different layers in a way that together the desired complex behavior emerges.

Coming up with a different abstraction for the different layers and how they cooperate with each other would have certainly mitigated the problem of relying on simple control flow logic. Even so, we believe that the main challenge would still apply and, in order to develop more complex reactive behavior, a different architecture would need to be used.

REFERENCES

[1] E. Ruiz, R. Acua, P. Vlez and G. Fernndez-Lpez, "Hybrid Deliberative Reactive Navigation System for Mobile Robots Using ROS and Fuzzy Logic Control," 2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR), Uberlandia, 2015, pp. 67-72.

[2] R. A. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation, vol. 2, no. 1, pp. 1423, 1986.

[3] M. Kseolu, O. M. elik and . Pekta, "Design of an autonomous mobile robot based on ROS," 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 2017, pp. 1-5.