Wall Following STDR

Ângela Cardoso and Inês Caldas

Abstract—The abstract goes here.

✦

1 INTRODUCTION

The most basic types of robotics systems are purely reactive. These systems don't have neither the ability to form memories nor to use past experiences to inform current decisions. Since they are fast and rely only on the current sensor readings instead of an accurate map, the use of which requires very accurate localization capabilities, reactive navigation approaches are often used for robot navigation. However, reactive navigation does not plan ahead and is therefore susceptible to local minima.

In this project we developed a stable wall follower behavior for a simple reactive robot. The design, implementation and testing of our solution, were made with the support of the *ROS framework* and the *STDR Simulator* package. The robot is equipped with two sensors that allow him to sense the world around and act according to the information he receives from them.

In the following sections we will discuss the architecture of our design, the results and limitations of our approach and some last remarks.

[1]

2 ARCHITECTURE

3 RESULTS

4 LIMITATIONS

4.1 Map delimiting walls

In order to keep the robot from trying to leave the map, we had to create squared walls around the map. These walls are not the ones the robot is supposed to find and follow, but if it does find these walls it will follow them forever.

Of course this only happens for the small D, whose walls the robot is supposed to follow from the outside. In the cases where the robot is inside the large D or between the two D's, it cannot move through those walls, therefore it will never find and follow the square map delimiting walls.

For the small D, we defined an initial position for the robot that is closer to the D than to the outside wall, although it sees neither of these walls. Thus, even though the initial movement of the robot will be random, it is more likely to find the small D wall and follow it. In case it finds the outside wall, the program should be stopped, but if it is

not, the robot will still follow the outside wall without any problem.

4.2 Initial random movement

Until it finds the wall, the robot moves randomly, that is, at each step it chooses random linear and angular velocities. Depending on the initial distance of the robot to the nearest wall and its orientation, this random search may take a long time.

We believe this wall finding time could be improved by making it more likely for the robot to move forward than for it to turn. This is clear if the robot is inside the wall it is suppose to find, like when it is inside the large D or between both D's. But in the case where the robot is outside the small D, going forward with higher probability might increase the likelihood of the robot finding the square walls delimiting the map, before it found the D walls.

Since the map delimiting wall is only there to stop the robot from trying to leave the map and not to be followed, we kept the robot moving randomly. In any case, the only downside is longer waiting time.

4.3 Desired wall distance

The distance the robot tries to keep from the wall can be configured by the user. If the chosen distance is too small, the robot may not be able to turn on sharp corners. This happens because while turning the robot gets too close to the wall and perceives that it has run into it, which makes it stop altogether.

Perhaps there is some way to make the robot resume its movements if it runs into a wall, but we did not pursue that. Especially because the correct way to handle this would have been to force the robot to avoid all obstacles, which we also did not try.

As an example, for the large D that the robot follows from the inside, if the defined distance to the wall is 0.6 or less, the robot runs into the wall at the left corners of the D.

Although we did not test it, we are convinced that for smaller angled corners, the desired distance that the robot should keep from the wall must increase in order for it not to crash.

4.4 Initial robot position

If the initial position of the robot is on top or too close to a wall, it will not move. This is related to the previous

limitation, as we did not try to get it to move after running into a wall. So the only way to avoid this error is to make sure that a safe initial position is configured for the robot, according to each map.

4.5 Following two walls

When the robot is between two walls, like when it is between the two D's, it does not follow both, that is it does not try to keep itself in the middle of both walls. We are not sure if that was the intention of the two D's assignment, but such behavior does make sense to us. The problem with trying to stay in the middle of two walls is that such behavior is different from that of trying to follow a wall at a given distance.

There are two reasons why the robot may receive data from both lasers: either each laser detects a different wall, that is, the robot is between two walls; or both lasers detect the same wall, which may happen because the robot is facing the wall or because it is at a corner. Our way of dealing with data from both lasers is to use the laser that is closest to a wall. Unless the robot has already chosen a laser, in which case it should stick to it. This way, we avoid running into a wall because we where paying attention to the wrong laser. We also avoid turning around in an inner corner, because suddenly the other laser is closer to the wall. Plus, when the robot is facing a wall, it will eventually chose one laser, which means it will chose a direction in which to follow the wall. Unfortunately, it also means that when it is between two walls, the robot will pick the closest wall and follow it. To change this behavior, we need to be able to distinguish between the two scenarios above, and honestly we did not think further about this.

5 CONCLUSION

The conclusion goes here.

REFERENCES

[1] D. Duan, L. Yang, Y. Cao, J. Wei, and X. Cheng, "Self-organizing networks: From bio-inspired to social-driven," *Intelligent Systems, IEEE*, vol. 29, no. 2, pp. 86–90, Mar 2014.