Technical Report on Wall Following Mobile Robot

Batuhan Ozgur Basal

November 8, 2021

Abstract

This technical report examines the Pioneer 3dx named robot's movement, it's attitude towards the wall, and tracking along the wall using the robot simulation platform named CoppeliaSim(VREP) with the help of Python language. It is aimed to measure the distance of the robot to the wall and maintain its proximity to the wall by using almost every sensors. It is aimed to measure and regulate the attitude of the robot against the wall by increasing and decreasing the power coming to the motors with the activation of the sensors. Although the robot's wall tracking has been successful, it is clear that there are some problems with the robot's acceleration and deceleration after detecting the wall. However, there is no barrier for it to be turned into a machine that moves more smoothly by working on it.

1 Introduction

One of the most important features of a mobile robot is that it can find its way without colliding with objects or walls. Robots used in homes and workplaces definitely have this feature. Otherwise, the physical impact from the collision may results as a hardware problem, it can stop the robot's motion indefinitely. To prevent this, the robot must determine It's distance from the object and position itself by it. Robot vacuums can be considered as an example for this situation. The function of robot vacuum cleaners is to clean the floors by traveling around the environment. For this, The robot must enter every environment he can and circulate in such a way that there is no space left. Here, the wall tracking algorithm comes into play. Before going into more detail, it is necessary to learn about the simulation platform, the robot, and its sensors.

1.1 CoppeliaSim

Figure 1: CoppeliaSim

CoppeliaSim is a simulation platform for mobile and non-mobile robots. This platform provides users with a realistic environment to use robots as they wish. This platform offers a near-real experience with its different features. On this platform, you can create the environment you want, use whichever robot you want to use, measure, and analyze the reactions of the robot to the external environment by using different sensors.

1.2 Python

CoppeliaSim platform supports a lot of programming languages for its controller. These are: Lua(main one), C/C++, Python, MATLAB, Octave, Java and Urbi. In this study, python language was used for controls. First of all, one of the most valid reasons to use Python is that the programming language is open source and always free. In addition, since it is very simple compared to other programs due to its simplicity, it can be learned quickly without losing any time.

Figure 2: The Symbol of Python

1.3 Robot

This study uses the Pioneer 3dx robot in the CoppeliaSim simulation platform. This robot has 2 wheels and 2 motors on the right and left. It is a very useful robot due to its light weight and small structure. It needs only one battery for its energy and There are 16 sensors placed around the robot. Thanks to the wheel encoders that the robot has, it can read the notification from the sensors and position the wheels accordingly.

Figure 3: The Figure of Pioneer 3DX Robot

1.4 Sensors

There are 16 sensors positioned in front and rear of the robot. These sensors are ultrasonic sensors. It sends out a sound wave signal, which reaches the object and returns to the sensor. Thus, it can measure how far is the distance to the object. Since it is located on many places, it can approach objects from different angles and position itself accordingly..

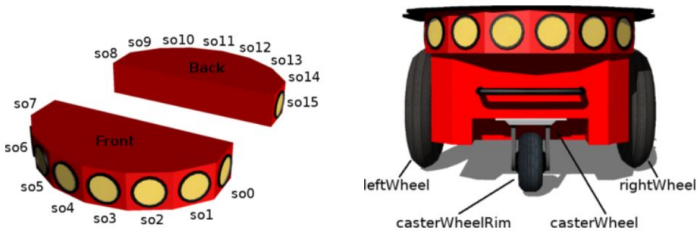

Figure 4: Sensors of the Pioneer 3DX Robot

2 Algorithms

For mobile robots, the algorithm of finding and following the wall has become a widely used task. Although the robots or sensors used are different from each other, the task is the same. There are two algorithms in this study. The first is to find the wall and the second is to follow the wall.

2.1 Wall finder

The purpose of this study is to create and develop the ability to find and follow the wall by starting from the same position and following different paths. The robot's primary task is to find the wall so that it can follow. Thanks to the sonar sensors located on the front of the robot, the detection of the wall can be easily achieved. The threshold value is very important because as the value decreases, the robot starts to approach the wall, the most appropriate threshold value must be entered for the sensors, otherwise the robot may detect the wall too late and can hit the wall while turning left or right, or it may detect the wall too early and start maneuvering to turn and it could be too far from the wall.

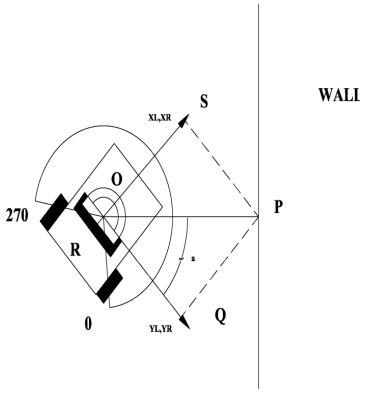

Figure 5: Robot position with respect to wall

2.2 Wall Follower

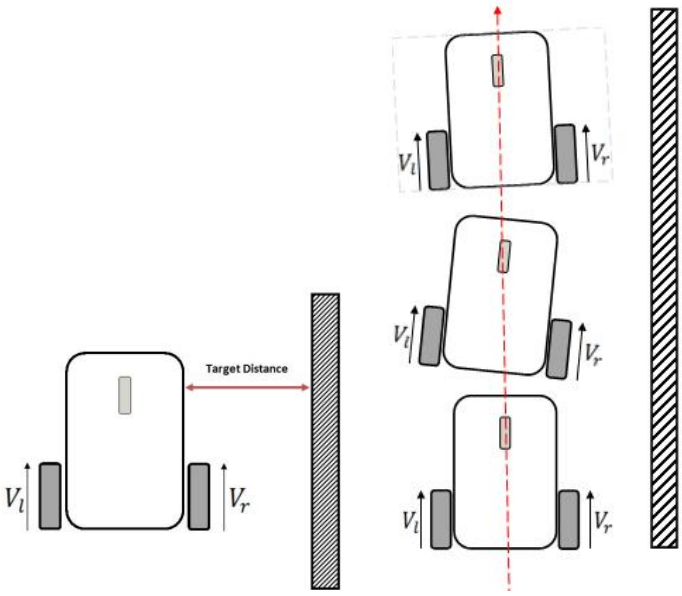

Figure 6: Maintaining the distance between robot and wall

After reaching the wall, the robot turns right or left and takes a different path until a new object appears in front of it. Thanks to this algorithm, the robot starts to follow the wall by keeping its distance from the wall. In short, wherever the wall goes, the robot goes.

This algorithm is very important especially for vacuum robots. Starting from the outer surface, they traverse the interior and achieve this by following the wall or nearby objects. In this case, the sensors on the right and left come into play. If the robot is to the left of the wall, the right side sensors

will be active and keep its distance from the wall, ensuring that it does not overflow. The same is true for the right side. This time, the sensor on the left side becomes active and maintains its distance from the wall, ensuring that it does not overflow. When the sensor is activated, the motors will be notified and both the left and right motors will accelerate to different speeds to maintain and keep the distance to the wall.

3 Simulation and Results

3.1 Random Wander

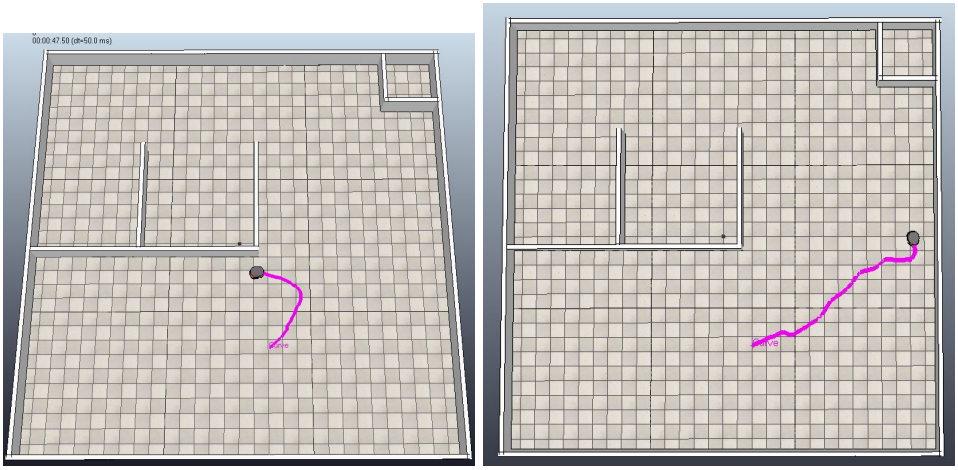

Figure 7: The Robot finds the wall in different places

Before the robot interacts with the wall, the robot has to travel around randomly. For this, random numbers must be given to the motors. In Python, we can get random numbers using many functions through the random library. The computer gives us these numbers automatically and gives different numbers each time it runs. Let's say there are two variables, if we connect the random.randint function to them, both variables will have different numbers and the robot will be able to navigate to different places each time it is started. But the problem here is either the numbers given are far from each other, so one engine freezes the wheel fast because it works stronger than the other. In this case, the robot keeps spinning around itself and cannot detect the wall, or the motors are rated close to each other and move straight ahead. whichever number is high, a movement opposite to the number belonging to that engine is observed. In this case, the robot draws an arc to the left and crashes into the wall.

To prevent this, it is necessary to determine a single number and give the motors a value close to that number. Thanks to the def (Wander) function, it gives the motors a different number between 0 and 10 each time they are started. This allows the robot to reach everywhere and points to different coordinates each time it is activated.

3.2 Wall Follow

The robot meets the wall after its wander. It needs to follow the wall without leaving the wall and keeping the distance. In this case, it is certain to get help from the sensors. 15 sensors were used for this study. it was necessary to take a different approach for each conditions using the if... else statement in python. if the threshold value is set for each sensor, both motors of the robot can operate at different levels, determining how the robot will act in the current situation. When only the front sensor is activated, the robot was able to move forward without hitting anything. But it's not enough for wall tracking. Also, after a while, it stays in a loop and keeps spinning around. But it is enough for the robot to navigate without crashing into anything.

It was mentioned that when the robot reaches the threshold value, motors will take different values.The robot only works with the 5th sensor. This sensor starts to change direction when the distance between the sensor and the wall is less than 0.8 meters. The values of the motor are given

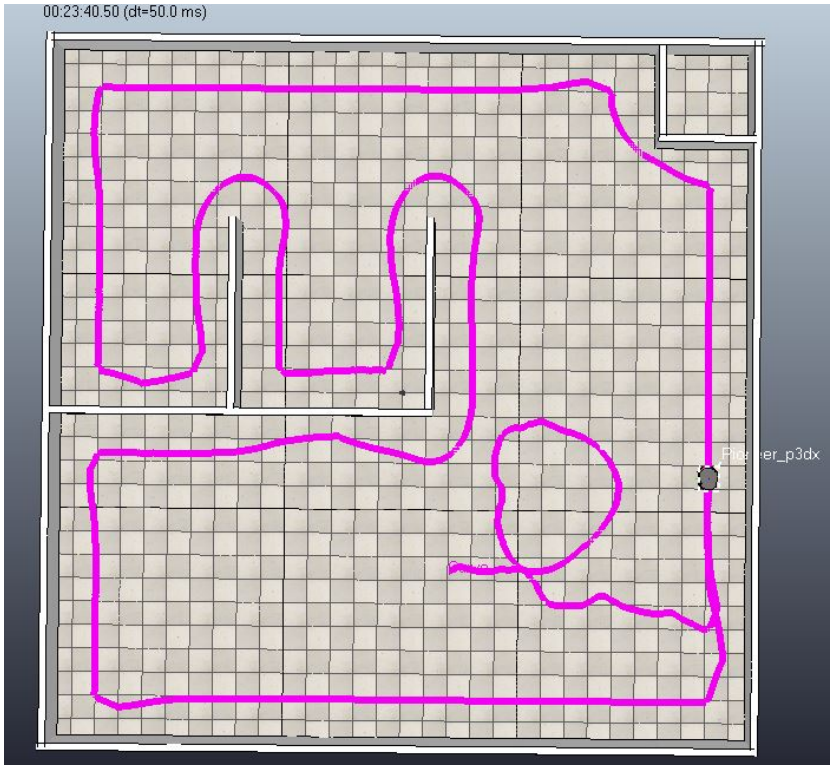

Figure 8: Simulation of Wall follower robot

manually. These values are 2.1 for the right motor and -2.1 for the left motor. When the values are equal and opposite to each other, the wheel turn to the same direction. If the signs were reversed, the robot would turn right instead of left. In order for the robot to move straight without making too much left and right, the sensors on the front usually take values at similar levels. The task of the sensors on the right and the left side of the robot is to keep the distance between the wall and the robot. sensor 8(right sensor) and sensor 15(left sensor) have been used twice because one task was to keep it close to the wall and the other was to keep it away from the wall. Therefore, different threshold and motor values were used in these sensors.

4 Conclusion

This study, observed the movements of the robot named Pioneer 3dx, examined its approach to the wall, and developed the wall tracking algorithm. The robot had 2 tasks to perform. The first one was to find the wall by following different directions from where it started, and the second was following the wall that the robot found. In both tasks, the robot's motors and sensors played a major role. While the motors took the wheels in different directions with the force they applied, the sensors ensured that the robot did not hit the wall and followed it, just like an eye.

Many problems were faced in this study and all of them were successfully solved. The biggest problem was following the wall without leaving it. Some of the turning points were very challenging. It was especially challenging to follow the walls consisting of 2 blocks. In that part, the effect of the sensors on the left and right back was very big.

Another positive reason why this study was successful was the setting of the appropriate threshold values for the sensors. A small error in these values could bring the robot against the wall. That is why almost all sensors were active for the wandering of the whole wall.

Finally, the initial speeds of the motors made the simulation run faster, but on the other hand, there were problems in the simulation for following the wall. As the speed increased, the errors also increased. for this reason, appropriate engine start level was determined as 1.

Sensor Number	Right Motor Value	Left Motor Value	threshold Value
1	2.1	-2.1	0.8
2	2	-2	0.8
3	1.9	-1.9	0.8
4	2	-2	0.8
5	2.1	-2.1	0.8
6	0.1	0.2	0.8
7	0.2	0.1	0.8
8	0.3	0.1	0.4
8	0.1	0.4	0.8
9	0.1	1.0	1.8
10	0.2	1.1	0.8
11	-	-	-
12	0.3	1.2	0.8
13	0.4	1.3	0.8
14	1	0.1	1.8
15	0.1	0.3	0.4
15	0.4	0.1	1.8
16	0.5	0.2	0.8

References

[1] M.FINKELSTEIN, B.HUNTE. Wall Following with Collision Avoidance and Mapping Using a Laser Range Finder (2008)

[2] GALLOWAY, KEVIN. Basic Operating Manual for ISL Systems

[3] Cyberbotics Ltd. https://cyberbotics.com/doc/guide/pioneer-3dx. Pioneer p3dx.

[4] Karoline de M. Farias; R. Wilson Leal; Ranulfo P. Bezerra Neto; Ricardo A. L. Rabelo; Andre M. Santana. An approach for environment mapping and control of wall follower cellbot through monocular vision and fuzzy system. (4-8 Sept. 2017)

[5] Coppelia Robotics, Ltd. https://www.coppeliarobotics.com/.

5 APPENDIX

5.1 Random Wander

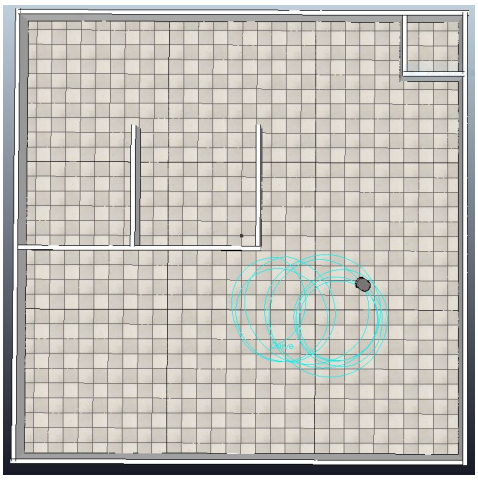

Figure 9: Robot spinning around

Some problems have been mentioned before.If the values of the motors were far from each other, spinning around itself was observed.

5.2 Wall Follow

These values were measured as the most appropriate values. But before that, the simulation was tested by giving different values for distance. Threshold values of sensors 9 and 14 were 0.8 like all the first values. At first, the robot had no trouble following the wall. But as you can see in the figure above, it broke its connection with the wall when it had to make a maneuver to the right. This was because the threshold value was low for those sensors.

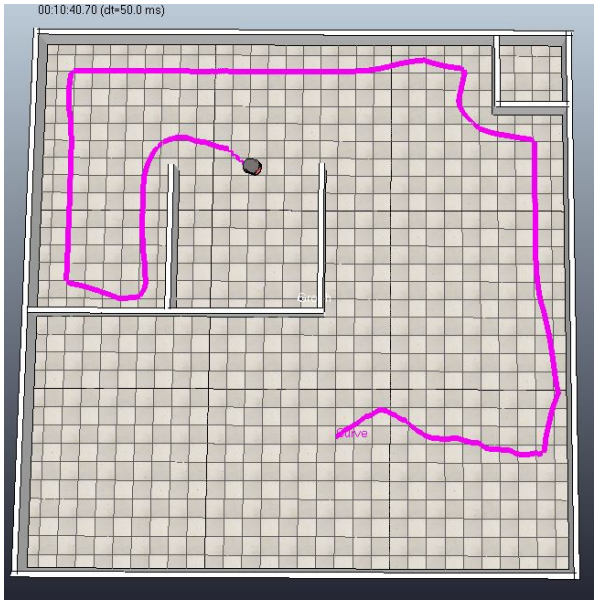

Figure 10: (When Sensor 9 and Sensor 14 are lower than 0.8 threshold value)

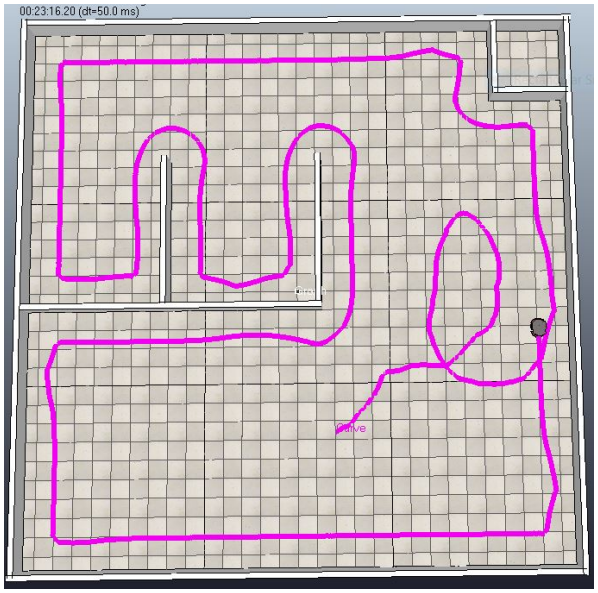

Figure 11: When (Sensor 9 and Sensor 14 are lower than 1.8 threshold value)

The same is correct for the side sensors. A threshold value of 0.8 means that the robot loses its connection with the wall. When the value is reduced to 0.4, Wall tracking will be performed successfully.

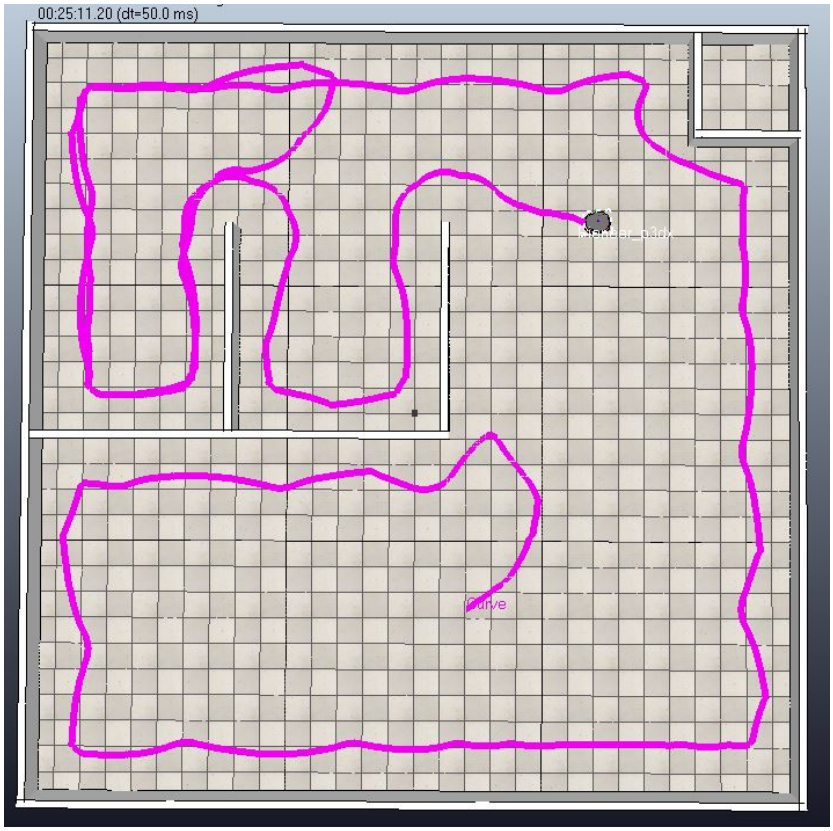

Figure 12: (When the threshold value is lower than 0.8 threshold value)

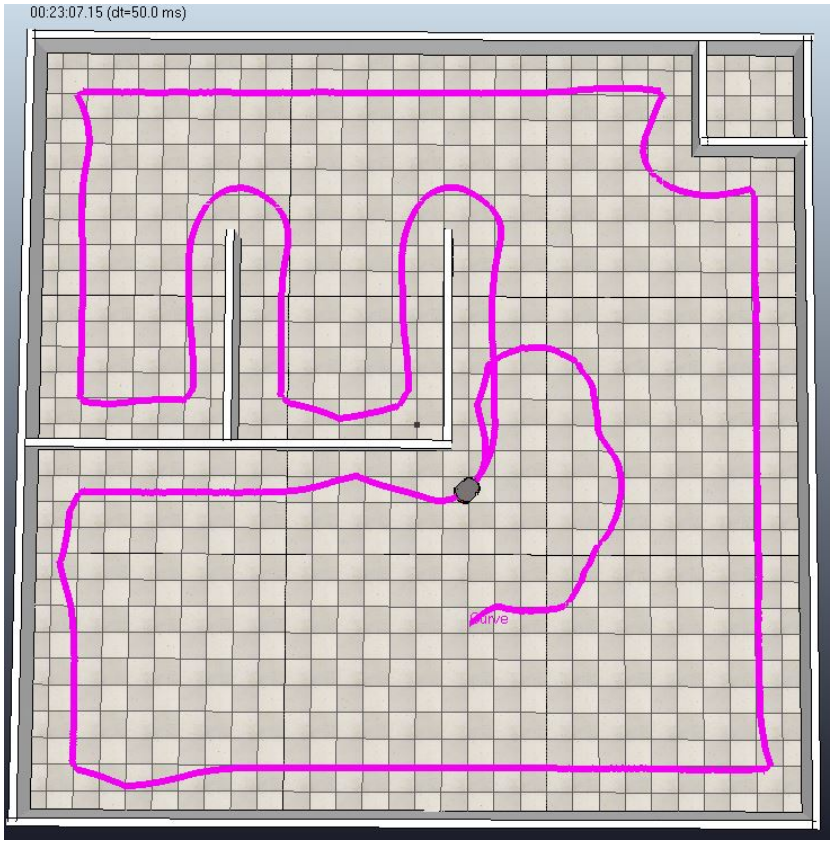

Figure 13: (When the threshold value is lower than 0.4 threshold value)

the angular velocity of the robot before it finds the wall and after it finds it:

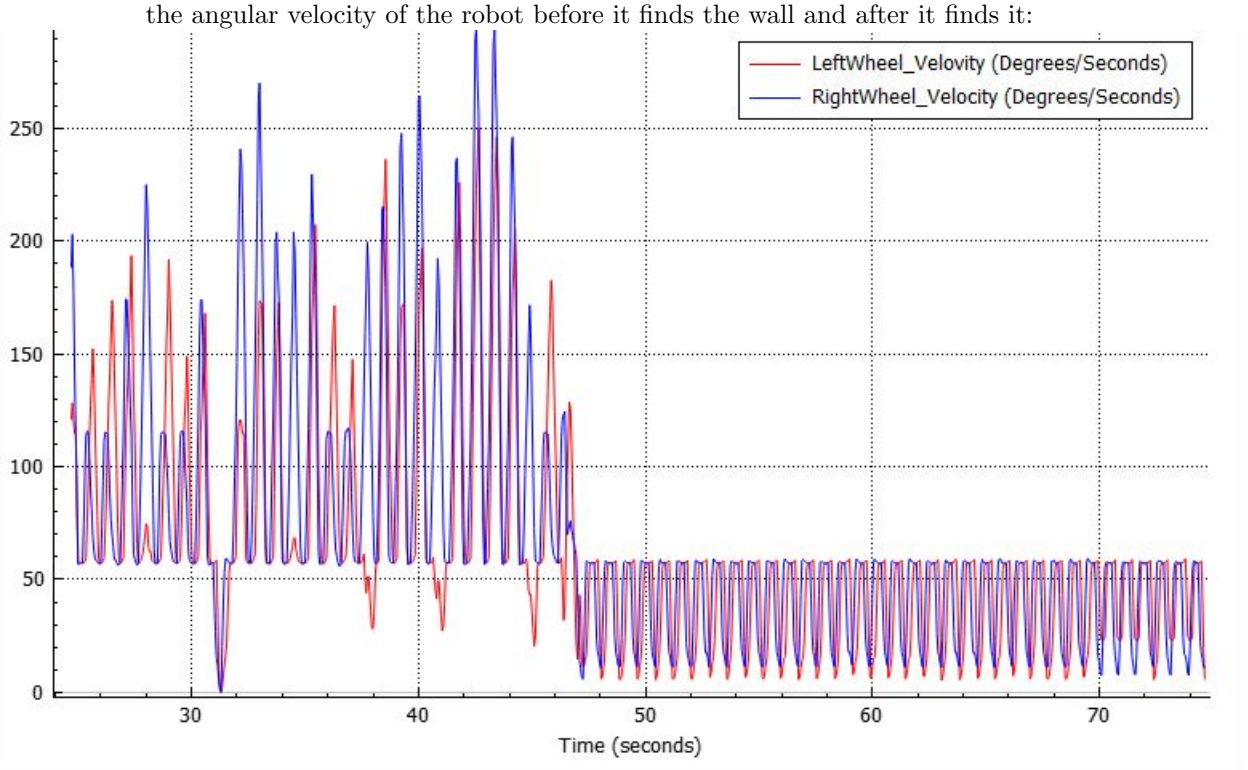

Figure 14: (Angular velocity of the robot changes when it finds the wall)