ECSE 683 Assignment - 1 Report

Adharsh Mahesh Kumaar

McGill ID: 260905451

1 Examples

Turtlebot mobile robot is used for obstacle avoidance in this assignment as shown in Fig 1 [5]. The controller designed to solve obstacle avoidance is kinematic as we control the robot using linear velocity v and angular velocity ω_z.

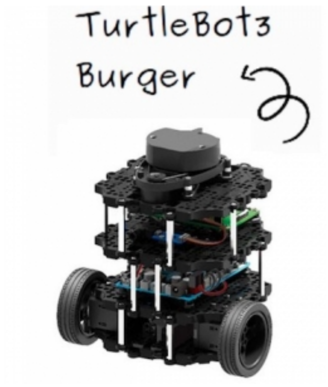

Figure 1: Turtlebot3 - Burger

The configuration space of the robot is $R^2XS^{'}$ where R^2 is the set of positions in 2D x-y plane and $S^{'}$ is the rotation in 2D [1]. $R^2XS^{'}$ is called SE(2) which is the group of homogeneous transformation in 2D.

The task space is all the positions the turtlebot can occupy which is the 2D (x-y) plane so the degrees of freedom (DOF) is 2.

Similar to any ground robot, the Degrees of Freedom (DOF) is 3 (x - position, y - position, and orientation of the robot θ) which is the dimension of C-space. The robot is controlled using command velocity messages in ROS. So the input to the robot is linear and angular velocities.

The state of the robot(q) is given by the position and orientation.

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

The action of the robot is defined by the linear velocity v and angular velocity ω_z The algorithm for obstacle avoidance is tested in 2 environments in ROS - world (Fig. 2), and house (Fig. 3). These two environments form the test cases.

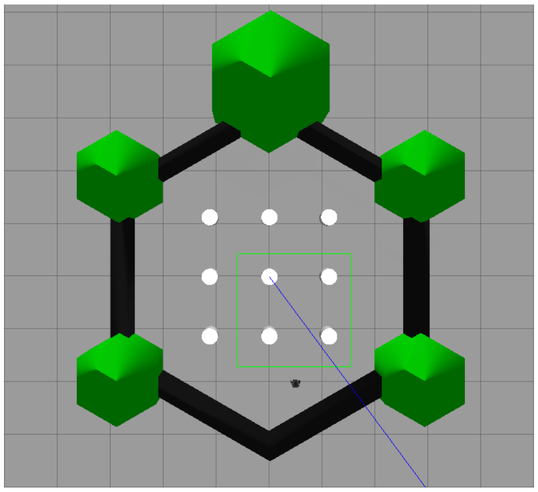

Figure 2: World Environment in ROS

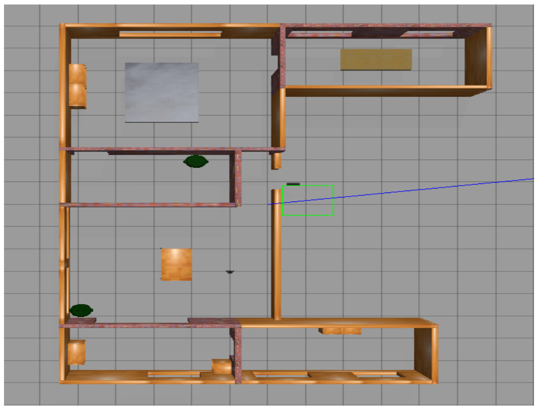

Figure 3: House Environment in ROS

2 Method

The method to avoid obstacles is by using a laser scanner of the turtlebot. By default, the robot is made to move forward. If the laser scanner detects any obstacles, then the turtlebot is given angular velocity, and the linear velocity is made to be 0. The robot turns at an angle and checks for any other obstacle. If obstacles are not present then the robot moves forward. The robot keeps turning until it does not detect any obstacles. Based on the laser scanner measurements, the angular or linear velocity of the robot is given.

3 Code

The code is implemented using Subscriber and Publisher messages in ROS [4]. The Laser scanner measurements are subscribed and are passed to the call back function. Inside the call back function, we provide the appropriate velocities (linear or angular) as described above. These velocities are published to the robot in the form of command velocity messages. The program keeps running until it is manually stopped. This is done with the help of the spin() function in ROS.

Algorithm for Obstacle Avoidance:

- Move forward if there are no obstacles within -10 °to +10 °in the direction of the robot. This range is taken so that other parts of the robot like the wheels do not get into a collision.
- Rotate counterclockwise if there are any obstacles within -10 °to +10 °in the direction of the robot.
- Keep turning till there are no obstacles in its direction of motion.

4 Justification of approach

This algorithm is free from deadlocks because it uses sensor measurements while avoiding obstacles. The video of the simulation has been attached to this assignment. The algorithm is tested in two environments in ROS - world, and house. The distance threshold for the laser scanner is given to be 0.7. So, the maximum distance at which it detects any obstacle is 0.7. From the video, we can observe that the robot moves forward until it detects any obstacles. Once it detects any obstacle, it starts turning in the counter-clockwise direction. There is no problem with deadlocks because our algorithm only detects the presence of obstacles and plans its motion accordingly to another point in case it detects any obstacles.

5 Limitations

- The algorithm does not work all the time. Since I am using a 2D laser scanner [2], the robot detects only the obstacles which are present in the line of sight of the Laser scanner. If the obstacles are smaller than the robot then the laser scanner cannot detect any obstacles and the obstacle avoidance algorithm will not work.
- In this algorithm, obstacles are checked within -10 °to +10 °. But this range is found using trial and error. Although this constant works in simulation, it may not be accurate in real time.
- Since we are using only a 2D laser scanner, other parts of the robot may collide. For example in Fig 4, the wheels of the robot may collide with walls and might get stuck.

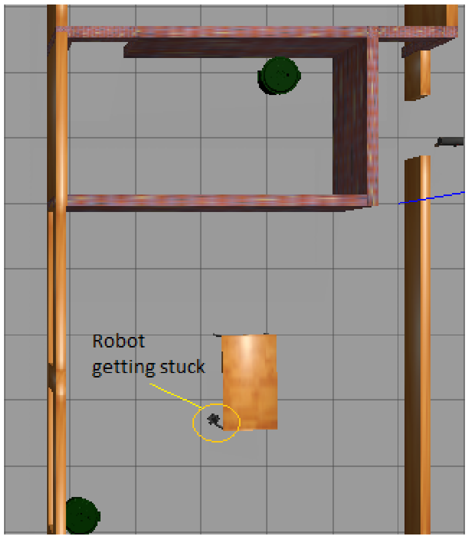

Figure 4: The wheels of the robot getting stuck

6 Overcoming Limitations

- The first and the last limitation can be overcome by using a 3D laser scanner and incorporating those measurements as well.
- Accurate threshold and constant values can be calculated using the size of the robot as given in [3].

References

[1] C-space of turtlebot

[2] Turtlebot specifications

[3] T.R. Madhavan. and M. Adharsh., "Obstacle Detection and Obstacle Avoidance Algorithm based on 2-D RPLiDAR," 2019 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, Tamil Nadu, India, 2019, pp. 1-4

[4] Publisher and Subscriber ROS

[5] Turtlebot3 Simulator