Python Applications for Robotics

Spring 2021

Instructor:	Zeid Kootbally	**RWA#2:**	Robot Motions
Email:	zeidk@umd.edu	**Due date:**	04/09/2021

1 Introduction

For this assignment you have to write a ROS package to move a TurtleBot3 in Gazebo. You reuse the package shown in class and modify it. This assignment must be done individually and not as a group.

- You will work with the following world:

```
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

- You can eventually use RViz for debugging:

```
roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch
```

2 Reaching Goals

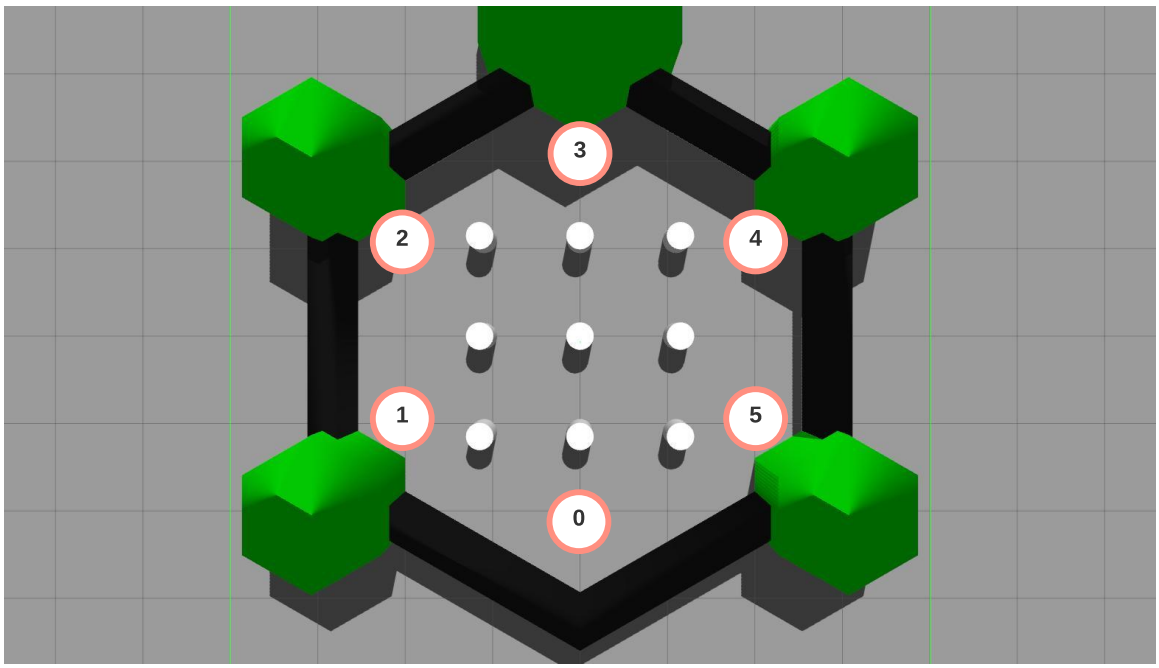

Figure 1: Your robot should be able to drive from any location to any location.

1. Write your program.
2. Spawn the TurtleBot3 at position $(-2, 0)$ (shown by location 0 in Figure 1).
3. Run your node with a location argument passed to the command line (we saw how to read arguments passed to the command line during class). Based on the argument passed, move your robot from its current location to the goal location. For instance, the commands below should move the robot in this order: (0)→(1) and then (1)→(5). **Note**: For this assignment we only need 2 arguments: the name of the executable (Python file) and the goal.

   ```
   rosrun bot_controller my_bot_controller 1
   ```

   ```
   rosrun bot_controller my_bot_controller 5
   ```

   The coordinates of each goal are defined below:

   - (0) : $(-2, 0)$ (this is the start location of your robot).
   - (1) : $(-1, 2)$
   - (2) : $(1, 2)$
   - (3) : $(2, 0)$
   - (4) : $(1, -2)$
   - (5) : $(-1, -2)$
4. **Note**: The example above runs your node only twice but we will grade your assignment by asking the robot to move to 5 different goals ((where `x` is a value in [0,5])):

   ```
   rosrun bot_controller my_bot_controller x
   ```

   ```
   rosrun bot_controller my_bot_controller x
   ```

   ```
   rosrun bot_controller my_bot_controller x
   ```

   ```
   rosrun bot_controller my_bot_controller x
   ```

   ```
   rosrun bot_controller my_bot_controller x
   ```
5. The second command below asks the robot to move to an already reached goal. You have to take care of this situation by displaying a message in the terminal ("Goal reached"). Since the robot will not precisely reach the goal, we can use a threshold of 0.5 m to consider a goal reached.

   ```
   rosrun bot_controller my_bot_controller x
   ```

   ```
   rosrun bot_controller my_bot_controller y
   ```

   - Where `x` = `y`

3 Obstacle Avoidance

To move from one location to the other, your robot has to move in a straight line while avoiding the obstacles. In other words, you need to combine both go_to_goal and read_scan functions (seen in class) to do obstacle avoidance (see Figure 2).

- Hardcoding waypoints will result in an automatic 0 for the assignment.

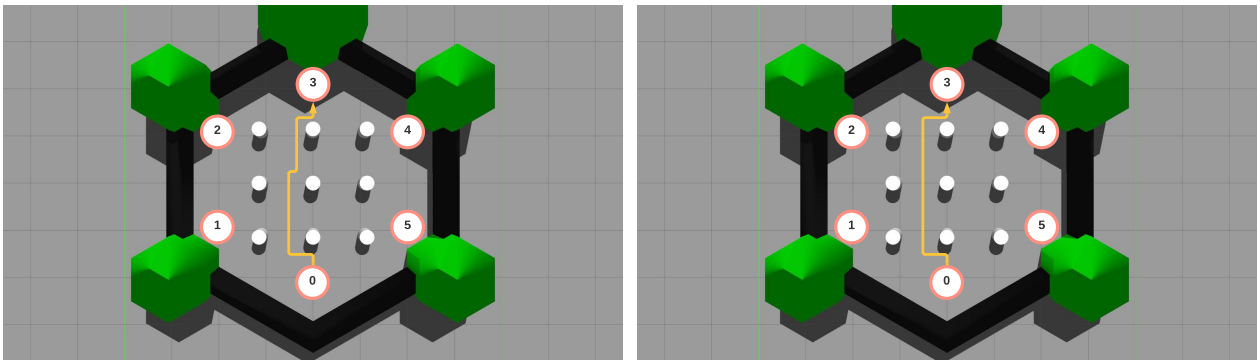

Figure 2: Correct way (left figure) and wrong way (right figure) to do obstacle avoidance.

- Hardcoding the position of the poles in your program will result in an automatic 0 for the assignment.
- You HAVE to use LIDAR sensor data for obstacle avoidance.
  - In class we only inspected 3 beams (front, left, and right). You will need to inspect other beams to properly do obstacle avoidance.

4 Package and Submission

- Rename your catkin package as follows: rwa2_lastname.
  - For instance, we should run your node as follows:
    `rosrun rwa2_lastname my_bot_controller x` (where `x` is a value in [0,5])
- Currently, `computation.py` is part of the velocity_publisher. You need to move this Python package to the package rwa2_lastname (see Lecture07 for instructions).
- Create an *instruction* directory inside your package and place a file `instructions.txt`, which explains how to run your node.
- Zip your catkin package and submit in on Canvas. Zip only your package and not your whole workspace.
- Submit your package before the deadline. Late submissions will incur a penalty.
- If you need help on the assignment, set up a meeting for at least 3 days before the due date (last day to meet with me is 04/06). I will not be able to meet with you or answer assignment related questions after this date.

5 Grading Rubric

+20 pts – **Reaching goals**: We will ask the robot to move to 5 different goals. These goals are the same for each student and we will run your node exactly 5 times. One of the following situations will be applied to grading this section.

+4 pts – Reaching a goal without colliding with poles.

+2 pts – Reaching a goal with at least one collision.

+0.5 pts – Not reaching a goal.

+5 pts – **Documentation**: Code properly documented. There is no need to generate any documentation from now on. You only need to document your code properly.

+5 pts – **Python guidelines**: PEP8 is followed.

-10 pts – **Hardcoded information**: Points will be removed if you hardcode waypoints, pole locations, or other things you were not supposed to hardcode. In case of doubts of what to hardcode and not to hardcode, post your questions in the Discussions section on Canvas (do not send us emails).

-15 pts – **Plagiarism**: This an individual assignment and each one of you has to write a program. It is very easy to detect plagiarized programs. You can talk with each other to share ideas but definitely not programs. Students with similar programs will have 15 pts taken off from the final grade for this assignment.