CSCI-473/573 Human-Centered Robotics
Project 2: Reinforcement Learning for Robot Wall Following

Project Assigned: Feb 15, 2021
Multiple due dates (all deliverables must be submitted to Canvas)
Deliverable 1 (Gazebo Simulation) due: Feb 22, 23:59:59
Deliverable 2 (Problem Formulation) due: March 8, 23:59:59
Spring Break: March 18–26
Deliverable 3 (Reinforcement Learning) due: April 3, 23:59:59

In this project, students will design and implement reinforcement learning algorithms to teach an autonomous mobile robot to follow a wall and avoid running into obstacles. Students will be using the Gazebo simulation in ROS Melodic to simulate an omni-directional mobile robot named Triton, and using an environment map that is provided to you. Students will be using a laser range scanner on the robot to perform sensing and learning, where the robot is controlled using steering and velocity commands. Students are required to program this project using C++ or Python in ROS Melodic running on Ubuntu 18.04 LTS (i.e., the same development environment used in Project 1). Also, students are required to write a report following the format of standard IEEE robotics conferences using LaTeX. Please **START EARLY!**

A note on collaboration: Students are highly encouraged to discuss this project amongst yourselves for the purposes of helping each other understand the simulation, how to control the robot, how to read sensor values, etc. This collaboration is solely for the purpose of helping each other get the simulation up and running. You may also discuss high-level concepts in helping each other understand the reinforcement learning algorithms. However, each person must individually make their own decisions specific to the learning aspects of this project. In particular, this means that individuals must make their own decisions over the representations of states, actions, and rewards to be designed in this project, and must implement their own learning algorithms and prepare all the written materials to be turned in on their own.

I. The Wall Following Problem

Wall following is a common strategy used for navigating an environment. This capability allows an autonomous robot to navigate through open areas until it encounters a wall, and then to navigate along the wall with a constant distance. The successful wall follower should traverse all circumferences of its environment at least one time without getting either too close, or too far from walls, or running into obstacles. Many methods were applied to this problem, such as control-rule based methods and genetic programming. In this project, students are required to solve the problem using reinforcement learning.

If you have any questions or comments, please contact the instructors Mark Higger at mhigger@mines.edu, Terran Mott at terranmott@mines.edu or TA Shane Copenhagen at scopenhagen@mines.edu.
This write-up is prepared using LaTeX.

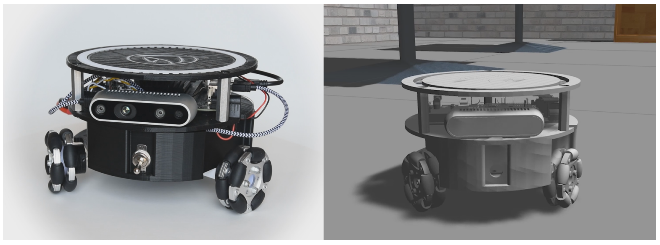
Fig. 1: The Triton robot and its simulation in Gazebo.

II. Deliverable 1: Gazebo and SLAM

Before working directly on the wall following robot, students will first familiarize themselves with using Gazebo in ROS. Gazebo is a powerful tool used in robotics for robot simulation. It allows you to simulate a robot and its environment for testing or data-driven methods. In order to simulate the wall-following robot for reinforcement learning we will use a gazebo. As a tutorial for utilizing gazebo you are expected to launch a robot in the gazebo environment, map the environment using SLAM, and understand what ROS topics are being used. To do this you will follow the steps below:

1) Install ROS packages for simulating Turtlebot3 in Gazebo, mapping, and planner:
```
sudo apt-get install ros-melodic-turtlebot3*
sudo apt-get install ros-melodic-slam-gmapping
sudo apt-get install ros-melodic-dwa-local-planner
```
2) Go through Gazebo beginner tutorial 1-3 “Overview and Installation, Understanding the GUI, and Model Editor” to get familiar with Gazebo: http://gazebosim.org/tutorials?cat=guided_b&tut=guided_b2

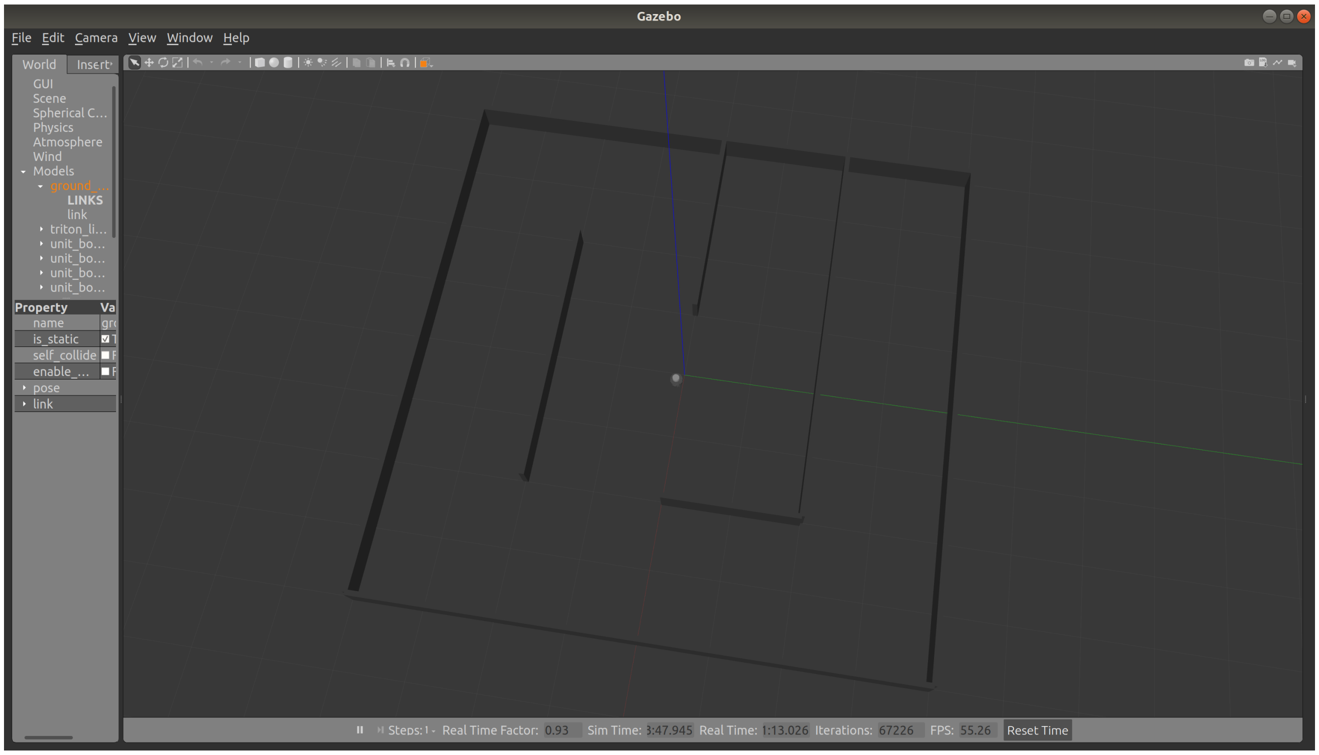

Fig. 2: Gazebo simulation of the Triton robot and the experiment environment for wall following.

3) Go through TurtleBot3 Simulation tutorial from 6.1.1 “Install Simulation Package” to 6.3.4 “Set Navigation Goal.” `https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/`
   Notes: Choose waffle_pi as your TURTLEBOT3_MODEL since it is also equipped with a camera.
   Make sure you select Melodic at the top of the page.
   You will need to (1) load a pre-defined simulated world, (2) execute RViz to visualize the robot sensing, (3) build a map of your simulated world, and (4) use RViz to navigate through the world.
4) Find what ROS topic gazebo is subscribing to control the robot and what topics you would need to subscribe to for getting lidar data. Students should use rqt_graph to find the available ros topics: `http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics`

III. Deliverable 2: Problem Formulation

As part of the project, students are responsible for formulating the wall following problem. Specifically, in this part of the project, students will need to determine how to represent the input state (based on the input Lidar data) and the motor actions, and the best level of discretization of these values. Please note that you are NOT allowed to use the grid-world representation of states. States should be a function of sensor values (and other measurements, as appropriate), for example, as implemented in [1]. As always, there is not just one single way to define states and actions. Explore various design possibilities and find the one that you believe works best.

Based upon the discretized states and actions, students are required to implement a Q-table and manually set the values of the Q-table to determine the policy. The manually selected Q-values will be applied for two purposes:

- Allowing the robot to follow a straight wall using these manually defined Q-table as the policy.
- Using these manually defined Q-values to encode expert knowledge to facilitate Q-learning in Deliverable 2.

In Deliverable 2, students are required to demonstrate that the manually defined Q-values as the policy enable the robot to follow a straight wall. Accordingly, the ROS package you submit in Deliverable 2 should allow a user (e.g., the TA or instructor) to test the policy manually defined in the Q-table for a robot to follow a straight wall in Gazebo. This means that (1) your submitted ROS package in Deliverable 2 should include your manually defined Q-table, and (2) a launch file designed to start the whole simulation to show the capability of robot following a straight wall in Gazebo.

Additionally, you are required to use LaTex to generate a 1-2 page report in IEEE 2-column conference format for deliverable 2. You can find the IEEE LaTex template at `https://www.overleaf.com/gallery/tagged/ieee-official`. To write this report in LaTex it is recommended to use Overleaf, which provides a web platform for writing and compiling documents in LaTex. A tutorial for how to use LaTex and overleaf can be found here: `https://www.overleaf.com/learn/latex/Tutorials`.

A. ***CSCI-473***: *What to Submit for Deliverable 2*

CSCI-473 students are required to submit a single tarball, named *D1_firstname_lastname.tar* (or .tar.gz) to the Canvas portal named P2-D1, which must contain the following items:

- A 1-2 page **Report** written in LaTex which contains:
  - An introduction to the problem
  - Definitions of your states and actions as well as why those states and actions were chosen
  - A table including your manually defined Q-values for each state-action combination (HINT: this table should n_{states} x $n_{actions}$ in size and populated by mostly zeros)
  - A screenshot of your robot in the simulation environment
  - Some notes on the performance of the robot (i.e. Does your solution wobble? Is it generalizable to corners and turns?)
  - Bibliography containing any references that you used to complete this assignment (including [1]).
- Your **ROS package** (that must include the source files, launch files, package.xml, CMakeLists.txt, world files, README, etc.) to demonstrate the robot capability of following a straight wall. The launch file must automatically start your demonstration in Gazebo (as shown in Fig. 2). The README file must provide sufficient information of your package, and clearly describe how to use the launch file to run the demonstration.
- A short **demo video** showing that the robot successfully follows a straight wall (no need to implement or show the capability of turning around wall corners; following a *straight* wall is sufficient for this deliverable). You can either submit a video or provide a link to the video on YouTube. If you are submitting a video, make sure the video size is less than 5M. You can use *ffmpeg* to speed up the video in Ubuntu. If you choose to provide a link, you are responsible to ensure that the video link allows public access.

B. ***CSCI-573***: *What to Submit for Deliverable 2*

CSCI-573 students are required to submit a single tarball, named *D1_firstname_lastname.tar* (or .tar.gz) to the Canvas portal named P2-D1, which must contain the following items:

- A 1-2 page **Report** written in LaTex which contains:
  - An introduction to the problem
  - Definitions of your states and actions as well as why those states and actions were chosen
  - A table including your manually defined Q-values for each state-action combination (HINT: this table should n_{states} x $n_{actions}$ in size and populated by mostly zeros)
  - A screenshot of your robot in the simulation environment
  - Some notes on the performance of the robot (i.e. Does your solution wobble? Is it generalizable to corners and turns?)
  - Bibliography containing any references that you used to complete this assignment (including [1]).
- Your **ROS package** (that must include the source files, launch files, package.xml, CMakeLists.txt, world files, README, etc.) to demonstrate the robot capability of following a straight wall. The launch file must automatically start your demonstration in Gazebo (as shown in Fig. 2). The README file must provide sufficient information of your package, and clearly describe how to use the launch file to run the demonstration.
- A short **demo video** showing that the robot successfully follows a straight wall (no need to implement or show the capability of turning around wall corners; following a *straight* wall is sufficient for this deliverable). You can either submit a video or provide a link to the video on YouTube. If you are submitting a video, make sure the video size is less than 5M. You can use *ffmpeg* to speed up the video in Ubuntu. If you choose to provide a link, you are responsible to ensure that the video link allows public access.

REFERENCES

[1] D. L. Moreno, C. V. Regueiro, R. Iglesias, and S. Barro, “Using prior knowledge to improve reinforcement learning in mobile robotics,” *Proc. Towards Autonomous Robotics Systems*, 2004.