Introduction to Robotics
Assignment 2

Course lead: Dr. Amanda Prorok

Michaelmas Term, 2021-2022

1 Assignment 2 (100 points)

For this assignment, you will hand in a zip file named `A2_CRSID.zip` that contains your report (PDF file) as well as your code. For this set of exercises, you will review concepts from navigation and path planning.

1.1 Potential Field Method

Exercise 1 (33 points)
In this exercise, you will implement a potential field navigation method in Python. We will use a first-order method (velocity only) to control the Turtlebot robot. Open the file `~/catkin_ws/src/exercises/part2/python/potential_field.py`.

(a) **[5 points]** Implement a velocity field that reaches the goal in `get_velocity_to_reach_goal`. Run the program with `python potential_field.py --mode=goal`. Elaborate your method and include the resulting figure in your report.

(b) **[5 points]** Implement a velocity field that avoids the obstacle. Run the program with `python potential_field.py --mode=obstacle`. Elaborate your method and include the resulting figure in your report.

(c) **[5 points]** Test the combination of both fields with `python potential_field.py --mode=all`. Include the resulting figure in your report and explain this result.

(d) **[5 points]** Try placing the obstacle at $[0, 0]$. What do you expect to happen? What problem do you expect the robot to encounter? Describe a potential solution to mitigate this problem (for the given scenario).

(e) **[5 points]** Implement the mitigating solution proposed in (d). Place your obstacle at $[0, 0]$. Explain your method and include the resulting figure in your report.

(f) **[8 points]** What happens if you add a second obstacle, so that we have an obstacle at $[0.5, 0]$ and $[0, 0.5]$. Does your solution still work? If not, implement another solution, explain the method used, and include the resulting figure in your report (the resulting code should be present in `potential_field.py`).

Hand in your solution `potential_field.py` as well as your answers to (a)-(f). Do not exceed 250 words per answer.

Exercise 2 (20 points)
This exercise ports the previous exercise to Gazebo. Open the file `~/catkin_ws/src/exercises/part2/ros/potential_field_navigation.py`.

(a) **[3 points]** Write the set of equations that provide feedback linearization for a differential drive robot, by controlling a holonomic feedback point at a distance ϵ from the robot.
(b) **[4 points]** Explain the utility of feedback linearization.
(c) **[8 points]** Implement this system of equations in the `feedback_linearized` function. Start Gazebo with `roslaunch exercises gazebo_simple.launch` and then run your controller with `roslaunch exercises potential_field_navigation.launch`. When the robot reaches the opposite corner of the arena, pause Gazebo and plot your robot's trajectory with the `plot_trajectory.py` script in the same folder. Explain the resulting trajectory and include that plot in your report.
(d) **[5 points]** The implementation uses the absolute pose of our robot. Why is this not really needed? Implement `get_relative_position`, set `USE_RELATIVE_POSITIONS` to `True` and test your controller again. Mention the benefit of this method.

Hand in your solution `potential_field_navigation.py` as well as your answers to (a)-(d). Do not exceed 250 words per answer.

1.2 Rapidly-Exploring Random Trees

Exercise 3 (25 points)
This exercise implements a Rapidly-exploring Random Tree (RRT) algorithm for our differential-drive robot.

(a) **[5 points]** Open the file `~/catkin_ws/src/exercises/part2/python/rrt.py`. Implement the `sample_random_position` function. Explain your method.
(b) **[7 points]** Implement `adjust_pose` and run the program with `python rrt.py`. Explain your method and add the resulting plot to your report.
(c) **[5 points]** Run the program multiple times. What seems to be the main drawback of RRTs in the current implementation? Propose a solution (and justify).
(d) **[8 points]** Implement the proposed solution (copy the `rrt.py` file to `rrt_improved.py` beforehand). Explain your method and the resulting performance.

Hand in your solution `rrt.py`, `rrt_improved.py`, as well as your answers to (a)-(d). Do not exceed 250 words per answer.

Exercise 4 (22 points)
This exercise combines the Rapidly-exploring Random Tree (RRT) algorithm developed in the previous exercise with navigation.

(a) **[5 points]** Open the file `~/catkin_ws/src/exercises/part2/ros/rrt_navigation.py`. Implement the `feedback_linearized` function. You are free to copy-paste from the code you developed in Exercise 2. Include figures to show your working solution.
(b) **[5 points]** Explain the advantage/disadvantage of using motion primitives for path generation.
(c) **[7 points]** Implement the `get_velocity` function and explain your method. You can run your program in Gazebo by issuing the following commands in different terminals:

```
roslaunch exercises gazebo_simple.launch
roslaunch exercises slam.launch
```

```
roslaunch exercises rrt_navigation.launch
```

To send a target goal to your robot, click the "2D Nav Goal" button and click anywhere in the free space. To visualize the path generated by your program, add the corresponding topic in RViz by clicking "Add" and following the `/path` topic. Take a screenshot of your RViz window. It should clearly show your robot, the path and the map. Include this image in your report.

(d) **[5 points]** Explain what `roslaunch exercises slam.launch` does and how it is used by the RRT algorithm.

(e) **[Optional; please email ql295@cam.ac.uk to loan a robot]** We can finally run our program on a real robot. See [1] for a tutorial.
First, connect your laptop to "robotlab-wifi" (the password is **r0b0t**), and run the following command in a terminal:

```
hostname -I # Make a note of the output (e.g., 192.168.2.123).
ROS_MASTER_URI=http://192.168.2.123:11311 # Terminal 1.
ROS_HOSTNAME=192.168.2.123 # Terminal 1.
roscore # Terminal 1.
```

In another terminal, run the script that will synchronize the clocks between your laptop and the robot:

```
ROS_MASTER_URI=http://192.168.2.123:11311 # Terminal 2.
ROS_HOSTNAME=192.168.2.123 # Terminal 2.
roslaunch exercises send_clock.launch # Terminal 2.
```

Turn on your robot and make a note of its identifier (e.g. turtlebot-05). Once the robot is turned on, connect to it by running the following commands in a new terminal:

```
ROBOT_IP=192.168.2.155 # Use 150 + identifier.
scp receive_clock.py robotlab@192.168.2.155:~ # Password is r0b0t.
ssh robotlab@192.168.2.155
ROS_MASTER_URI=http://192.168.2.123:11311 # Robot terminal.
python receive_clock.py # Robot terminal.
roslaunch turtlebot3_bringup turtlebot3_robot.launch # Robot terminal.
```

Finally, on the laptop again, run the following in two additional terminals:

```
ROS_MASTER_URI=http://192.168.2.123:11311 # Terminal 3 & 4.
ROS_HOSTNAME=192.168.2.123 # Terminal 3 & 4.
roslaunch exercises slam.launch # Terminal 3.
roslaunch exercises rrt_navigation.launch # Terminal 4.
```

Explain the underlying procedure (i.e., . Take a screenshot of your RViz window and include it in your report. Take a photo of the environment that the robot navigated in and include that, too (it should correspond to the RViz image). Your report must clearly demonstrate the successful deployment onto the real Turtlebot.

Hand in your solution `rrt_navigation.py` as well as your answers to (a)-(d). Do not exceed 250 words per answer.

[1] https://drive.google.com/file/d/1gyb4cUd3GujW68roM904JAY-7kY1wOPO/view