Laboratory exercise 6

Autonomous mobile robot navigation in Stage

Name: **JMBAG:**

Preparation and helpful instructions

- **Do not: consult, read, or examine ANY materials or solutions for assignments from the previous years' editions of this course, or from your colleagues. The assignment solution must *fully* be your work. In case signs of plagiarism are detected, you will get a zero score and may be subject to be reported to the Faculty's Ethics Committee.**
- **Do: consult the teaching staff (for this assignment, teaching assistant Vlaho-Josip Štironja) via Teams DM, if you have any problems, if anything is unclear, or if you need any help with any part of the assignment.**
- Review the lecture slides about mobile robot localization and navigation.
- Unpack the prepared archive, which contains folders related to each task.

Assignments

Task 1: Implement Bug0 Algorithm for Robot Navigation

a) In this task, you will implement the Bug0 algorithm's logic in a Python script and test it using the Stage simulator.

The Bug0 algorithm is an approach to reactive navigation for mobile robots. It enables a robot to navigate from a starting position to a defined goal while avoiding obstacles in its path. The algorithm operates based on the following principles:

(a) **Goal-Oriented Motion**: The robot initially moves in a straight line toward the target position, maintaining a direct trajectory whenever no obstacles are encountered.

(b) **Obstacle Detection and Avoidance**: Upon detecting an obstacle within a predefined distance, the robot stops its forward motion and begins navigating around the obstacle by following its boundary while keeping the obstacle to one side.

(c) **Path Resumption**: Once the robot clears the obstacle, it resumes its direct motion toward the goal.

The Bug0 algorithm does not require prior knowledge of the environment, making it suitable for online path planning.

Your task is to complete the `move_to_goal` function in `bug0.py` and run this python script after launching the Stage simulation with the `single_obstacle.world` file.

Your function should address the following tasks:

- Stop when the robot reaches the goal
- Detect obstacles and rotate to avoid them
- Move toward the goal when no obstacles are detected

Hint: Do not worry if your path does not closely resemble the image (Figure 1) below. The shape of the path depends on factors such as the rotational velocity applied and the translational velocity (if any) during rotation.

Hint: One possible approach is to treat the angular velocity for moving toward the goal as the difference between the desired angle to the goal and the robot's current orientation, represented by the `self.yaw` angle.

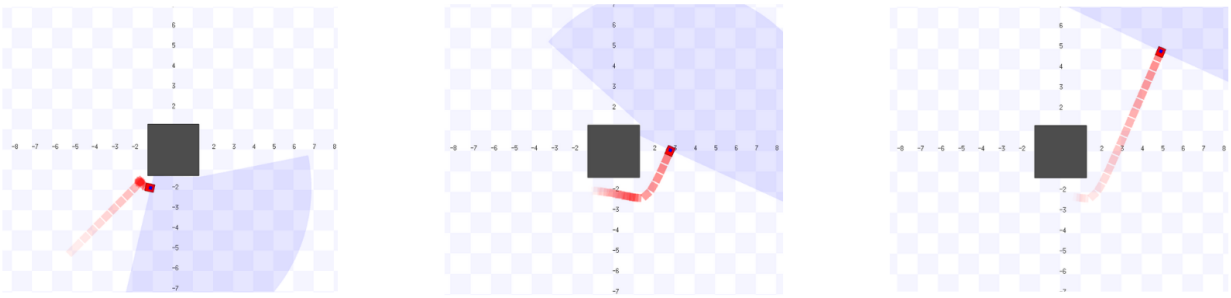

Figure 1: Example of the desired Bug0 path while running the Stage simulation.

Submit a screenshot showing the robot reaching the goal and the completed `bug0.py` script and answer tthe following questions.

The Bug1 algorithm is an improvement over the Bug0 algorithm. Unlike Bug0, which resumes moving toward the goal immediately after clearing an obstacle, Bug1 incorporates additional logic to enhance path efficiency. Explain how the Bug1 algorithm modifies the behavior of the Bug0 algorithm. What are the key differences in their approach to navigating around obstacles?

Assume you are tasked with upgrading a Bug0 implementation to Bug1. What changes would you need to make in the code? (**Bug1 should not be implemented and it is okay to answer general**).

Examine and explain how the detection of obstacles is implemented in the code.

Task 2: **Mapping the environment with gmapping package**

a) Following the steps presented in the lecture "Mobile Robot Localization and Navigation", run the Stage simulator with the `simple_rps.world` environment. Run the robot keyboard teleoperation and RViz. Try moving the robot around the environment via keyboard to make sure everything is working. Check the `gmapping` package documentation and run `gmapping` with correct arguments. In RViz, add `LaserScan`, `Map` and `TF` for visualization. Use your keyboard to manually move the robot around until you completely map the environment. Save the resulting map as `firstname_lastname` (look here for help).

Send us a screenshot of RViz with visualized `LaserScan`, `Map` and `TF` where the resulting map you obtained can be seen. Write a launch file named `firstname_lastname.launch` which will run `gmapping` package, `Stage`, `Rviz`, and keyboard teleoperation at once.

Task 3: **Navigation with Move Base package**

a) Following the steps presented in the lecture "Mobile Robot Localization and Navigation", run the Stage simulator with the `simple_rps.world` environment. Run the map server with the map you obtained in the Task 2 (`firstname_lastname.yaml`). Run the robot keyboard teleoperation, RViz and AMCL similarly to the lecture. Ensure that everything is running correctly and that the robot is properly localized. You will need to use the Move Base package to make the robot autonomously navigate in the environment. Your task is to fill in the required parameters in place of the question marks (???) in the provided parameter files. Launch the prepared `move_base_rps.launch` file which runs the required node and loads all of the necessary parameter files which you edited. In RViZ select the 2D Nav Goal and give the robot a destination. If everything is correct the robot should autonomously navigate in the environment using a map that you obtained in Task 2.

In RViz show the following and submit it as screenshots:

1) Costmap generated by the Move Base package.
2) Global and local planned path (both need to be clearly visible).

Exercise submission

Create a zip archive containing **this PDF with the completed answers** and the following:

- A folder named `task1` containing a screenshot and the `bug0.py` script.
- A folder named `task2` containing a screenshot of the map in RViz and the launch file.
- A folder named `task3` containing a screenshot of the global and local planned paths, along with the parameter files with completed values.

Upload the zip archive to Moodle.