Final Project

Due May 5 by 11:59pm   **Points** 100   **Submitting** a text entry box or a file upload

Available Apr 3 at 10:30am - May 5 at 11:59pm

This assignment was locked May 5 at 11:59pm.

First: make sure your update your local copy of the MRTP GitHub repository to get the final version of the navigation package. Read the documentation and see the provided examples (testnavigator.cpp and testpackage.cpp) to understand how to use it. In this final project you are encouraged to use the navigation package as much as possible to simplify your solution.

The setup is the same as for lab7, i.e., you start as follows:

```
export TURTLEBOT3_MODEL=waffle
export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:/opt/ros/foxy/share/turtlebot3_gazebo/models
ros2 launch gazeboenvs tb3_simulation.launch.py
```

When you start the simulation the robot and the navigation stack are provided with a map of the environment including the location of the the nine cylinders and of the surrounding walls. However, when your code will be tested (see below), there will be an extra cylinder positioned at an undisclosed location. Your task is to design a program that explores the environment, finds the extra post, and returns its position in the map frame.

Notes:

- how do I find the extra post? Use the onboard range finder (providing data on /scan) to find mismatches between the information you get from the map and what you receive from the sensor;
- the map of the environment is found **here (https://catcourses.ucmerced.edu/courses/26775/files /5887062?wrap=1)** ⤓ **(https://catcourses.ucmerced.edu/courses/26775/files/5887062 /download?download_frd=1)** . The map is aligned with the x axis to the right and the y axis up. The center of the map has coordinates (0,0). Each pixel represents a square with side 0.05m. Values in the map are in the range 0-1. Values above 0.65 are occupied space and values below 0.196 are free space. Values in between are unknown. Note that if you want to process the map you do not have to read it from the file, but you can retrieve it from the topic `/map` (the file is only made available to you to simplify your design stage).
- the robot always starts at position (-2,-0.5,0) with 0 yaw (same as testnavigator.cpp)
- to know the estimated pose of the robot, you can subscribe to the topic `/amcl_pose` that provides the estimated pose of the robot
- to get the transformation of the laser with respect to `base_link`, subscribe to `/tf_static`

- if you want to test your solution in an environment with an extra post, in a separate shell, run the following:

```
export TURTLEBOT3_MODEL=waffle
export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:~/MRTP/MRTP/src/gazeboenvs/models
ros2 launch gazeboenvs tb3_simulation.launch.py
```

  This assumes that the MRTP repository is installed in your home folder. If this is not the case, adjust the path accordingly, i.e., replace `~` with the path where MRTP is installed. If you want to experiment with different poses for the extra post, edit the file `model.sdf` that is found in the `turtlebot3_world` folder inside the `gazeboenvs` package (look for the two tags with name `extra` and edit the pose field as you wish).
- we will discuss this project in class and answer any question you may have.
- the task you have to implement is not dissimilar to what a surveillance robot would to -- patrol a known environment and find out anomalies.

Important: this assignment can be solved in groups with up to three students, but if you want to, you can work alone or in a team of two. The decision is up to you. However, past experience shows that students working alone often struggle to come out with a working solution. So you are encouraged to work in groups.

At the end you submit:

- your code, as a standalone package (file upload).
- instructions on how to run your code (in the textbox).
- a one pager describing how your solution works (to be entered in the textboox).
- you will have to demonstrate your code in the lab, during one of the lab sessions associated with your team members.
- if you work in groups, just one submission and one demonstration per group is needed. All members in the group get the same grade.
- your overall grade depends both on the quality of your solution and of your description.

The following rubric will be used to evaluate your submission:

Item	Points	Remarks
Compile / Run	10	*Your solution compiles to an executable that can be run under your project's ROS2 overlay.*
Moves	10	*Your solution moves the turtlebot.*
Safely Moves	20	*Your solution moves the turtlebot safely, without running*

			into any obstacles.
Explores	20		*Your solution causes the turtlebot to explore the environment.*
Returns Coordinate	20		*Your solution returns the location of the extra cylinder, with a reasonable margin of error. If the solution returns multiple results, only the last result will be considered.*
Writeup	20		