# Robotics Lab: Homework 4

Control a mobile robot to follow a trajectory

**Mario Selvaggio**

This document contains homework 4 of the Robotics Lab class.

## Control a mobile robot to follow a trajectory

The goal of this homework is to implement an autonomous navigation software framework to control a mobile robot. The `rl_fra2mo_description` package must be used as a starting point for the simulation. The student is requested to address the following points and provide a detailed report of the methods employed. In addition, a personal GitHub repo with all the developed code must be shared with the instructor. The report is due in one week from the homework release.

1. Construct a gazebo world and spawn the mobile robot in a given pose
   (a) Launch the Gazebo simulation `/launch/gazebo_fra2mo.launch.py` and spawn the mobile robot in the world `leonardo_race_field` in the pose
   $$x = -3 \text{ m}, \quad y = 5 \text{ m}, \quad Y = -90\,\text{deg},$$
   with respect to the map frame. The argument for the yaw in the call of `spawn_model` is `Y`.
   (b) Modify the world file of `leonardo_race_field.sdf` moving the obstacle 9 in position:
   $$x = -3 \text{ m}, \quad y = -3.3 \text{ m}, \quad z = 0.1 \text{ m}, \quad Y = 90\,\text{deg}\,.$$
   (c) Place the ArUco marker number 115[1] on obstacle 9 in an appropriate position, such that it is visible by the mobile robot's camera (you have to add it to the robot) when it comes in the proximity of the object.
2. Using the Nav2 Simple Commander API enable an autonomous navigation task
   (a) Define 4 goals in a dedicated `.yaml` file. They must have the following poses with respect to the map frame:
   - Goal_1: $x = 0$ m, $y = 3$ m, $Y = 0\,$deg;
   - Goal_2: $x = 6$ m, $y = 4$ m, $Y = 30\,$deg;
   - Goal_3: $x = 7.0$ m, $y = -1.4$ m, $Y = 180\,$deg;
   - Goal_4: $x = -1.6$ m, $y = -2.5$ m, $Y = 75\,$deg.

   (b) Modify `follow_waypoint.py` or `reach_goal.py` to send the defined goals to the mobile platform in a given order. Go to the next one once the robot has arrived at the current goal. The order of the explored goals must be Goal_3 $\rightarrow$ Goal_4 $\rightarrow$ Goal_2 $\rightarrow$ Goal_1.
   (c) Record a bagfile of the executed robot trajectory and plot it in the XY plane.
3. Map the environment tuning the navigation stack's parameters
   (a) Modify, add, remove, or change pose, the previous goals to get a complete map of the environment, and save it (put in the report the .png of the map).
   (b) Change the parameters of the navigation config (try at least 4 different configurations). The suggested parameters that you can change are:
   - In file `slam.yaml`: tune parameters `minimum_travel_distance`, `minimum_travel_heading`[2], `resolution` and `transform_publish_period`.

[1] Generate it here.
[2] For these first two parameters observe what happens launching `fra2mo_slam.launch.py`

- In file `explore.yaml`: change the `inflation_radius` and `cost_scaling_factor` for global and local costmaps.

(c) Comment on the results you get in terms of robot trajectories, execution timings, map accuracy, etc.

4. Vision-based navigation of the mobile platform

(a) Create a launch file running both the navigation and the `aruco_ros` node using the robot camera you previously added to the robot model.

(b) Implement a 2D navigation task following this logic

- Send the robot in the proximity of obstacle 9.
- Make the robot look for the ArUco marker. Once detected, retrieve its pose with respect to the map frame.
- Return the robot to the initial position.

(c) Publish the Aruco pose as TF following the example at this link.

**NOTE:** to make correction easier, modify, when necessary, the launchfiles such that Rviz automatically opens with the proper configuration (all configuration files are in the `rviz_conf`).