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` and `fra2mo_2dnav` packages 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 and spawn the mobile robot in the world `rl_racefield` in the pose

   $$x = -3 \quad y = 5 \quad yaw = -90\,\text{deg}$$

   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 `rl_racefield` moving the obstacle 9 in position:

   $$x = -17 \quad y = 9 \quad z = 0.1 \quad yaw = 3.14$$

   (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 when it comes in the proximity of the object.

2. Place static tf acting as goals and get their pose to enable an autonomous navigation task

   (a) Insert 4 static tf acting as goals in the following poses with respect to the map frame:

   - Goal_1: $x = -10 \quad y = 3 \quad yaw = 0\,\text{deg}$
   - Goal_2: $x = -15 \quad y = 7 \quad yaw = 30\,\text{deg}$
   - Goal_3: $x = -6 \quad y = 8 \quad yaw = 180\,\text{deg}$
   - Goal_4: $x = -17.5 \quad y = 3 \quad yaw = 75\,\text{deg}$

   Follow the example provided in the launch file `rl_fra2mo_description/launch/spawn_fra2mo_gazebo.launch` of the simulation.

   (b) Following the example code in `fra2mo_2dnav/src/tf_nav.cpp`, implement tf listeners to get target poses and print them to the terminal as debug.

   (c) Using `move_base`, send 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* → *Goal_4* → *Goal_2* → *Goal_1*. Use the *Action Client* communication protocol to get the feedback from `move_base`. Record a bagfile of the executed robot trajectory and plot it as a result.

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.

   (b) Change the parameters of the planner and `move_base` (try at least 4 different configurations) and comment on the results you get in terms of robot trajectories. The parameters that need to be changed are:

   - In file `teb_locl_planner_params.yaml`: tune parameters related to the section about trajectory, robot, and obstacles.

[1] Generate it here.

- In file `local_costmap_params.yaml` and `global_costmap_params.yaml`: change dimensions' values and update costmaps' frequency.
- In file `costmap_common_params.yaml`: tune parameters related to the obstacle and raytrace ranges and footprint coherently as done in planner parameters.

4. Vision-based navigation of the mobile platform

   (a) Run ArUco ROS node using the robot camera: bring up the camera model and uncomment it in that `fra2mo.xacro` file of the mobile robot description `rl_fra2mo_description`. Remember to install the camera description pkg: `sudo apt-get install ros-<DISTRO>-realsense2-description`

   (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.
   - Set the following pose (relative to the ArUco marker pose) as next goal for the robot

   $$x = x_m + 1, \quad y = y_m,$$

   where x_m, y_m are the marker coordinates.

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