Assignment 1

DEADLINE: 23rd December at 23:59 (CET)

Implement a routine that lets Tiago navigate inside the environment, see Figure 1. The environment comprises two rooms with movable obstacles (i.e. you can move them in Gazebo) and a narrow space. Tiago has to navigate from point *Starting Pose* to *Pose_B* (Figure 1).

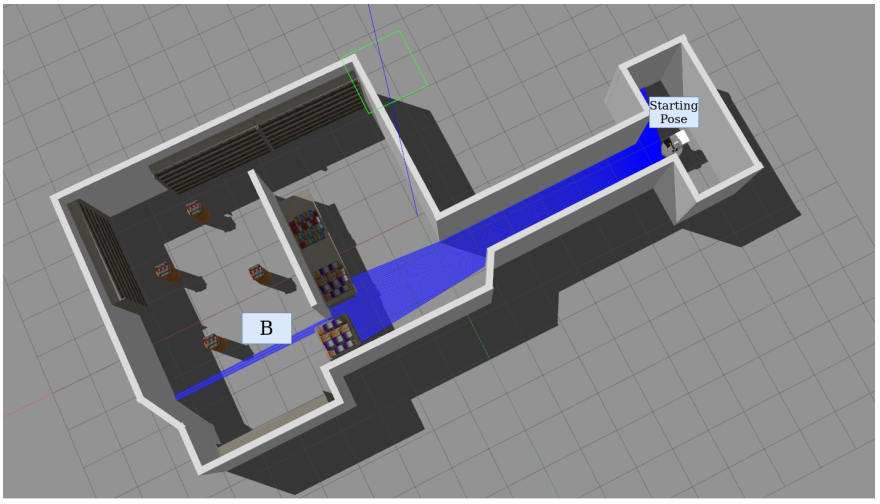

Figure 1

The *Pose_B* must be inserted by the user via the command line, i.e., it is an input from the command line for your code. Once the robot reaches *Pose_B*, it has to detect the obstacles, and print on the screen, the position of the movable obstacles, e.g. the cylindric tables, visible from that pose. (Figure 2). The static obstacles that are part of the map, e.g. the walls, or the shelves, must not be detected as obstacles.

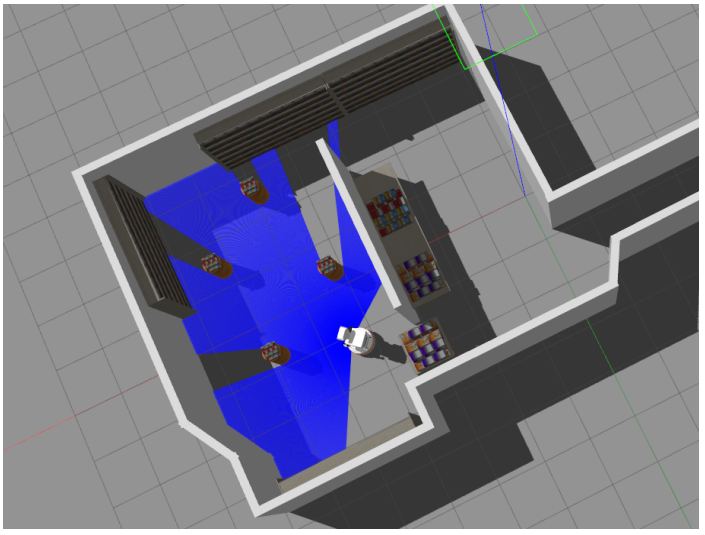

Figure 2

Suggestion: use the laser range finder mounted on the robot (topic /scan).

For the assignment, inside your Bitbucket repository, create a package which contains the code to implement.

Your code must be implemented using the following structure:

- The user can input the pose of *Pose_B* by command line;
- Your code **must** be implemented with an Action Client/Server structure:
    - The action client receives the input from the user;
    - The action client calls the action server that executes all the tasks;
    - The action server sends the final list of positions of the obstacles as result to the action client;
    - The action client also implements callbacks to the feedback of the action server, and prints the task's current status in the terminal. The feedback **must** reflect the current status of the robot, e.g.: *the robot is stopped, the robot is moving, the robot started the detection of the obstacles, the detection is finished, etc.*
    -

Extra points

Typically, the ROS Navigation Stack is inefficient in narrow passages. Usually, the programmer has to implement an ad hoc routine to directly process the sensor data (e.g., laser and/or sonar data) and generate velocity commands for the robot, i.e. the programmer creates a **"motion control law"** for the robot. In this assignment, to get extra points, we ask you to implement your control law to navigate through the narrow corridor using the laser data. The routine has to use the laser and send the velocity commands to the topic */cmd_vel* to move the robot without calling the *move_base stack*.

N.B.: Please, specify in the submission email and in the report that you implemented the extra point part of the assignment.

Instruction to start the homework

Follow these instructions to install the environment and start the homework:

- Clone your repository into your workspace, it will initially contain all the environment and launch files necessary to start the development. You **must** create your package into your group repository and deliver this one as a whole:
  ```
  $> git clone
  https://username@bitbucket.org/iaslab-unipd/ir2223_group_XX.git
  ```

- Start the simulation (cmd console 1):
  ```
  $> roslaunch tiago_iaslab_simulation start_simulation.launch
  world_name:=ias_lab_room_full
  ```

- Navigation stack (cmd console 2):
  ```
  $> roslaunch tiago_iaslab_simulation navigation.launch
  ```