AUE-893 Autonomy: Science and Systems

Repository: https://github.com/rmerco-clemson/AuE893Autonomy.git

Commands for launching the applications

The assignment is totally contained in the package "assignment_04" with its own launch scripts.
The package "assignment_04" has the following folder tree:

- Launch:
  - race1_world.launch. It launches the race world raceTrack1 in Gazebo by typing the command:
    - `roslaunch assignment_04 race1_world.launch`
  - race1_obstacles_world.launch. It launches the race world raceTrack1_obstacles in Gazebo by typing the command:
    - `roslaunch assignment_04 race1_obstacles_world.launch`
  - race1_obstacles2_world.launch. It launches the race world raceTrack1_obstacles2 in Gazebo by typing the command:
    - `roslaunch assignment_04 race1_obstacles2_world.launch`
  - race2_obstacles_world.launch. It launches the race world raceTrack2_obstacles in Gazebo by typing the command:
    - `roslaunch assignment_04 race2_obstacles_world.launch`
  - turtlebot_app.launch. It launches the python script able to run the mybot autonomously in order to avoid the obstacle.
    - `roslaunch assignment_04 turtlebot_app.launch`
- worlds:
  - raceTrack1.world: Gazebo world with a rectangular track without obstacles. In figure 1.
  - raceTrack1_obstacles.world: Gazebo world with a rectangular track with some obstacles. In figure 2.
  - raceTrack1_obstacles2.world: Gazebo world with a rectangular track with more obstacles. In figure 3.
  - raceTrack2_obstacles.world: Gazebo world with uneven walls and obstacles. In figure 4.
- Scripts:
  - turtlebot_app.py. It is the python script which runs the nodes able to create the logic that the mybot uses to race in the track avoiding the obstacles.

Designed Worlds

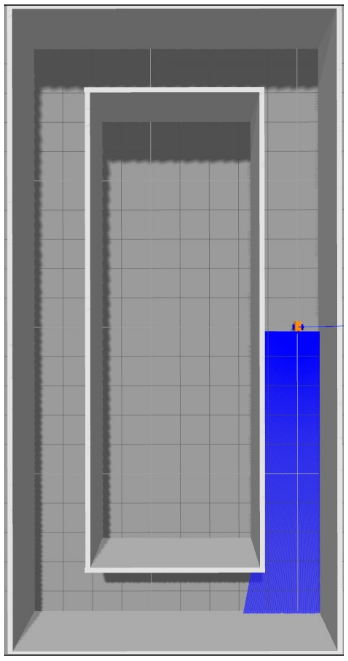
Illustration 1: raceTrack1

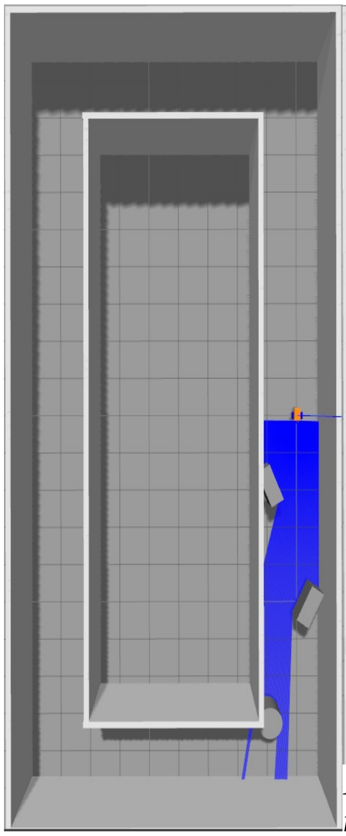
Illustration 2: raceTrack1_obstacles

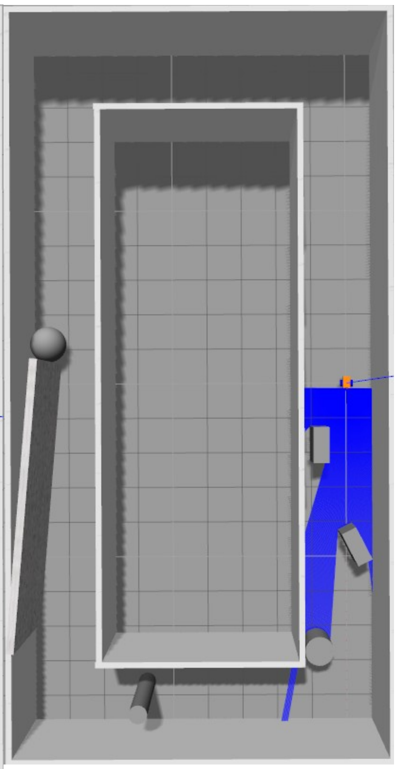
Illustration 3: raceTrack1_obstacle2

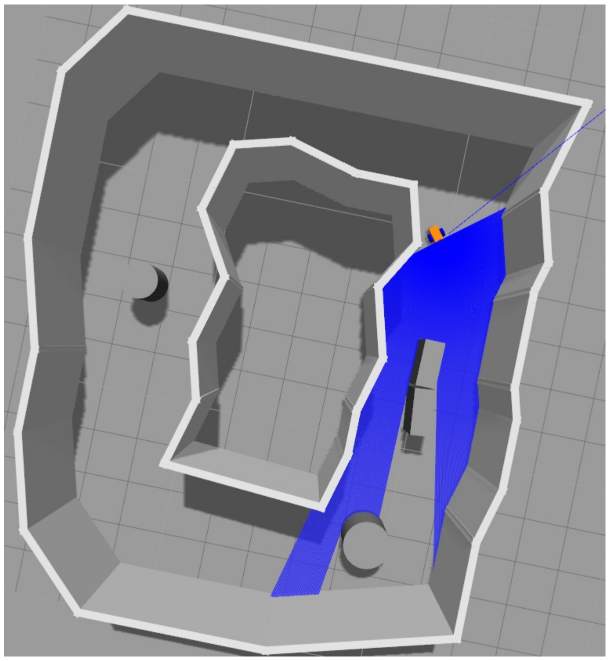
Illustration 4: raceTrack2_obstacles

Logic of track following and obstacles avoidance

The robot is provided of a front laser sensor able to identify the presence of obstacles in front of itself in a range of (-90,90) degree. Similarly to the previous assignment, this range is divided by interval in order to provide a matrix where each element represents a portion of the cone of visibility.
The full range of spectrum in front of the robot is divided by 5 bigger sectors. The first 2/5 of the sectors are used to create a sector which identifies the objects on the right, the middle (third) sector is used to identify the obstacles in the center and the last two sectors are merged to create a bigger sector which is used to identify the obstacles on the left. For each of these 3 macro sectors is calculated the minimum distance detected to the object.

An error based on the difference between the left and right minimum distance is used in order to feed a PD controller which keeps the robot in the center of the track. This logic is also used to avoid the obstacles which are not directly in front of the robot. A deadband of 0.1 is added to the error.

The center minimum distance is used in order to implement a proportional controller for the forward speed.

When the obstacle are closer than 0.35, the controller is switched to the wanderer program which is integrated in my code.