Drive the Husky

Assignment 8

Dec 18

In this assignment, you will navigate the husky robot to its destination. You can use either a lidar or a stereo camera to detect obstacles. You are allowed to use python or Matlab in this assignment.

1 Prepataion

- Download the starting code here. Unzip the package, and place `base_controller` in your repository `robotic-coursework-f2022`. This package contains the launch files for your assignment.
- Download additional world files and add it to your catkin workspace
  `> git clone https://github.com/clearpathrobotics/cpr_gazebo`

  Build the packages by
  `> catkin build cpr_office_gazebo`
  `> catkin build cpr_agriculture_gazebo`

- If ROS-perception is not installed on your machine, install ros perception package
- Install the external packages for the sensors if needed

2 Base Controller

2.1 Launch and yaml

There are five launch files in the `base_controller/launch` folder. Each of them will launch gazebo with a different world file. (Note: It may take some time to load the first time.)

`empty.launch` will launch with an empty world, so you can use it for your initial testing. Here is the list of launch files with interesting environments (from easy to hard):

```
mud.launch
cafe.launch
agriculture.launch
office.launch
```

There are also 5 `.yaml` files (for each `.launch`) for storing the default values and parameters. You cannot change the initial and the target positions in the yaml files. You can modify the other parameters or add more if needed.

2.2 Enable/Disable the sensors

In this assignment, you can choose any sensors for perceiving the environment. To enable/disable the sensors, you can export the environment variables

- Enable 2D Lidar: set `HUSKY_LMS1XX_ENABLED` to 1
- Enable Realsense: set `HUSKY_REALSENSE_ENABLED` to 1
- Enable 3D lidar : set `HUSKY_LASER_3D_ENABLED` to 1

Once you start, you should see the extra topics available

- Laser Scanner: `/front/scan`
- Realsense Camera: `/realsense/depth/color/points`

You should also see extra sensor in gazebo

2.3 Rviz

You can open RViz to visualize the sensor data
```
> roslaunch husky_vis view_robot.launch
```

You might need to adjust some values in RViz

- If you do not see the robot, change the value in `Fixed Frame` from `odom` to `base_link`
- For 2D lidar, change the value in `/scan` to `/front/scan`
- For Realsense, change the value in `PointCloud2/Topic` to `/realsense/depth/color/points`

3 Tasks

Your job is to implement a node `base_controller` that publishes the twist for the robot.

3.1 Planner

Plan the horizontal velocity from the robot's current position to the target and avoid collisions. You can use any planning methods (e.g., potential fields, via points, bugs algorithms, RRT, etc). Your planner should output the desired base velocity in 2D plane $\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$

3.2 Controller

Take the planned velocity and converted it into linear speed and angular speed.

```
twist_msg_.linear.x  = linear_speed ;
twist_msg_.angular.z = angular_speed ;
```

Publish it to `/husky_velocity_controller/cmd_vel`

3.3 README

In your readme file, briefly explain your work; e.g.,

- What sensors are required to run your planner?
- What is your planning algorithm?
- If your work only runs with different instructions, please specify your instructions in README.md in your Gitlab repository. (You will lose marks for running different instructions)

4 Evaluation

4.1 How to submit

Add `base_controller` to your repository `robotic-coursework-f2022` and push your changes to the same repository.

4.2 Test

We will test your implementations as follow:

- Build your work by running `catkin build base_controller`
- Change the environment variables depending on your choice of sensors.
- Open one terminal and run `> roslaunch base_controller xxx.launch`
- Open another terminal and run `> rossercall base_controller/start`

The robot should start moving toward the target.

4.3 Marking Scheme (15 points total)

- (2 points) program runs without compilation or run-time error with the instructions in 4.2.
- (1 points) reach the target in mud.launch
- (2 points) reach the target in cafe.launch
- (2 points) reach the target in agriculture.launch
- (2 points) reach the target in office.launch
- (6 points) reach the target for random initial and target positions