ROB314 – Session 3 - Ex1

Exercise

The goal of this exercise is to close the control loop for the Husky robot. You will extract the position of a pillar from the laser scan and then control the robot such that it drives into the pillar.

1. Let's start from your `rob314_husky_controller` package of the session 2. Or the correction provided on the rob314 website: https://perso.ensta-paris.fr/~battesti/rob314

   In you file ~/.bashrc, at the end, you should have the line :
   ```
   export HUSKY_LMS1XX_ENABLED=1
   ```
   If you use Noetic, you should
   - use the `/front/scan` topic instead of the `/scan` topic in the *controller.launch* file
   - Use gazebo-11 instead of gazebo-9 in the *controller.launch* file
   - *empty_world.launch* instead of *husky_empty_world.launch* in the *controller.launch* file

2. Download the *world files* and put them in a new folder `worlds` in your package `rob314_husky_controller` :
   https://perso.ensta-paris.fr/~battesti/rob314_download/rob314_session3_worlds.zip

3. Adapt the launch file from the last exercise such that:
   - The keyboard twist node is removed. (One can also comment lines in launch file by framing the text with `<!--  -->`)
   - `singlePillar.world` should be loaded as the world, instead of the `robocup14_spl_field.world`. Be careful, the path is not the same : you can use the command `find` in the launch file.
     This file come from the Zip file `rob314_session3_worlds.zip`. It is a world with only a cylinder shape, a "pillar".

4. By modifying the `laserCallBack` function, extract the distance and angle of the pillar from the laser scan with respect to the robot. (check data in https://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/LaserScan.html)

5. In the idea of using the topic `/cmd_vel` to send a `geometry_msgs::Twist` message to Husky, you need to add `geometry_msgs` as a dependency of your package.
   Do it by modifying your `CMakeLists.txt` and `package.xml` (same structure as with `sensor_msgs`) (Session 2).

6. Create a publisher (called `m_commandVelocityPublisher`, for example) in the `MyController` class, on the topic `/cmd_vel` to be able to send a `geometry_msgs::Twist` message to Husky. For the moment, the node publishes nothing.

7. In the callback method of the laser scan topic, write some code that drives husky towards the pillar, by using the angle of the pillar from the laser scan with respect to the robot (see question 4). This can be a simple P (proportional) controller.

8. Add a new ROS parameter for your controller gain in your node and use it in the callback method (Session 2).

9. Adapt your launch file to add the controller gain parameter.
   Launch your launch file with `roslaunch`.
   Try different value of your gain to find a good value. What happens if the value is too high or too low.

10. Launch your launch file with `roslaunch`.
    - Add a “RobotModel” plugin to RViz to visualize the Husky robot, if necessary. (Session 2)
    - Make sure to set *odom* as the *Fixed Frame* (under Global Options) and adapt the size of the laser scan points, if necessary.
    - Show the laser scan in RViz with the plugins “LaserScan” (click *add* at the bottom left of the window if necessary).
    - Add a “TF display” plugin to Rviz,
    - Visualize all the TF on the robot. You maybe have to use the button “Focus Camera”.
    - Save the rviz configuration in your package in a *.rviz file.

11. In gazebo, you can translate the pillar. If you put the pillar behind the robot it cannot see the pillar. Find a (simple) algorithm and code it to solve this case.

12. We want to visualize the estimated position of the pillar in RViz. For that, we will use a special message : `visualization_msgs::Marker`.
    Have a look at this page : https://wiki.ros.org/rviz/DisplayTypes/Marker
    Publish a visualization marker for Rviz, a sphere for example, that shows the estimated position of the pillar. You can use, for example, a topic called `/pillar_marker`.
    We want to publish each time we detect the pillar, so inside the callback function `laserCallBack`.
    You can publish the marker at a position in the *frame* of the laser. So you will use a relative position according to the position of the laser. To make it work, you have to specify the correct `header.frame_id` for the message. It should be the same that the received `LaserScan`. RViz will automatically transform the marker into the odom frame.

13. Launch your launch file with `roslaunch`. Add a “Marker” plugin to Rviz to visualize your marker message. Make sure to set `/pillar_marker` as “Marker Topic” in Rviz.

14. You have more time ?

    In gazebo, you can use the `driveThrough.world`.
    Code a node for the robot to pass between the two pillars.