CSE 468/568 Assignment 2
Robot Control and Using `tf` in ROS

The objective of this assignment is to use ROS `tf` to relate the coordinate system of one robot with the other.

Before you get started, please make sure you have done the following:

- Complete ROS tutorials 1 through 12 (in Section 1.1 Beginner) and be able to write a simple publisher and subscriber in ROS

Simulation Setup

First install required packages:
`$ sudo apt-get install ros-noetic-turtlebot3-gazebo ros-noetic-turtlebot3-fake ros-noetic-turtlebot3-simulations`

Alternatively, you can install all of these packages (along with some more), using:
`$ sudo apt-get install ros-noetic-turtlebot3*`
(Note the asterisk at the end)

Download the provided package template lab2.zip and extract the contents into the `~/catkin_ws/src` directory. You should have a `~/catkin_ws/src/lab2` folder, and a `~/catkin_ws/src/turtlebot3_simulations` folder which contains related packages.

- The package lab2 is where you will do all your programming.
  - Do NOT change the names of the launch and rviz files.
  - You may choose names for your python scripts but they should always be in the `src` directory.
- The package `turtlebot_simulations` contains the turtlebot simulator you need for the second half of this assignment.
- Read the README file in `lab2` carefully, and run the commands to verify installation.

Please read the roslaunch tutorial to understand the `roslaunch` file.

Evader Controller

In this section, you will write your own controller for a robot that avoids crashing into obstacles. We'll use `husky_gazebo` package for this part. Type (don't copy from PDF):

```
$ export HUSKY_UST10_ENABLED='1'
$ roslaunch husky_gazebo husky_playpen.launch
```

Familiarize yourself with the given robot by checking the topics it publishes and subscribes to, and the available information. Move the robot by publishing a `geometry_msgs/Twist` message using `rostopic pub /cmd_vel`.
The robot also has a laser range finder attached to it. Monitor the output of the laser by using `rostopic echo /front/scan` in the command line. You should be able to understand the output of the sensor from gazebo and its output you observed using echo.

Write a controller node that drives the robot straight at a constant speed of 2m/s. When the robot is close to an obstacle, the robot should stop, turn randomly to left or right until there is no obstacle in front of it, and then drive forward at the same speed. Edit the existing `lab2_evader.launch` file so that your controller is launched. You may launch only your controller and can assume the gazebo launch file will be used when grading.

Pursuer-Evader

In this section, you will write controllers for two robots. One Evader with similar behavior as previous section, and one Pursuer that follows the Evader. Read through the `tf` tutorial.
We'll use the turtlebot3 simulator for this part (launch file provided) to spawn 2 robots. Type (don't copy from PDF):

```
$ export TURTLEBOT3_MODEL=waffle
$ roslaunch turtlebot3_gazebo dual_turtlebot3.launch
```

First, write a controller for the evader. You may re-use the code from evader controller script you created in previous section and make it work with tutlebot3. Once the Evader works as expected, update it so it publishes the coordinate frame of the robot with regards to the global frame. Then write a controller node for the pursuer that subscribes to the `tf` messages from the evader, and follows the evader by going to the spot it was at, two second ago (NOT its current location).

Edit the existing `lab2_pursuer.launch` file so that both your controllers are launched. You may launch only your controllers and can assume the gazebo launch file will be used when grading.

Submission Instructions

You will submit `lab2.zip`, a compressed archive file containing the lab2 folder. The folder should be in the same format as provided and should compile when droped into `catkin_ws/src`. Please take care to follow the instructions carefully so we can script our tests. Problems in running will result in loss of points.

Please use UB Learns for submission.
The assignment is due Friday, October 7 before midnight.