SOFAR Continuous Evaluation 2 – Programming a node

The test consists in programming a node that controls a differential drive robot to follow another robot. The idea is to form elementary platoons by using several such nodes to control several robots. Those who work on platoons in the project will of course note that the provided solution is not suitable for a platoon in a street or road, as followers do not track the same path as the leader.

Provided files/nodes

- The `platoon` package, which contains three types of nodes :
  - `control_node`: the node to be developped.
    - Subscriptions and publications : you must define and implement them in such a way that the application runs using the provided launch file.
    - The control node uses two local parameters, « `dist` » and « `Kp` », both with default value of 1.0, which you need to implement.
  - `leader_node`: it defines a virtual leader moving along a circular path, which the first vehicle will follow. The leader node captures keystrokes '+' and '-' and increases/decreases the circle radius accordingly. **There is no need for the `capture_key` package**. The node publishes:
    - A visualization marker for `rviz` (red sphere).
    - Its posture (used by the control node of the following vehicle).
    - Its twist (speed and rotation speed, also used by the control of the next vehicle).
  - `vehicle_node`: it simulates the kinematics of a vehicle.
    - It subscribes to the control twist (speed and rotation speed).
    - It publishes:
      - A visualization marker for `rviz` (a blue arrow).
      - The posture of the vehicle (useful for the control of the follower).
      - The twist of the vehicle (useful for the control of the follower).
- The « `two_vehicles.launch` » file.

Step 1.

Configure your account for use of ROS using the file « account configuration for ROS.zip » from the shared folder. Since we will not be using the Baxter, **ignore the steps involving the file « baxter.sh »**. Do not execute « baxter.sh » when working on this evaluation. The « config.sh » file includes a command to copy the « baxter.sh » file, which will fail. You can ignore the error.

Step 2.

- Retrieve the file « ce2.zip » using the link given on the whiteboard.
- Copy it to the « catkin_ws/src » folder of your account.
- Decompress it there. You should a new folder named « `platoon` ».

Your tasks

1. Use the « control_node.cpp » file as a skeleton. Program the control node to control the robot at 20 Hz to follow the vehicle that precedes it. In order to do that, you need to determine what the input and output topics of the node should be (name and type).
   As far as the control is concerned, it derives from the static feedback control and is calculated in the following way for a controller which controls the rear vehicle to follow the

front vehicle:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x_{rear} + dist * \cos(\theta_{rear}) \\ y_{rear} + dist * \sin(\theta_{rear}) \end{bmatrix}$$

$$\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = \begin{bmatrix} v_{front} * \cos(\theta_{front}) + K_p * (x_{front} - x_p) \\ v_{front} * \sin(\theta_{front}) + K_p * (y_{front} - y_p) \end{bmatrix}$$

$$\begin{bmatrix} v_{control} \\ w_{control} \end{bmatrix} = \begin{bmatrix} \omega_1 * \cos(\theta_{rear}) + \omega_2 * \sin(\theta_{rear}) \\ \frac{1}{dist}[(-\omega_1 * \sin(\theta_{rear}) + \omega_2 * \cos(\theta_{rear})] \end{bmatrix}$$

Where $\begin{bmatrix} x_{rear} \\ y_{rear} \\ \theta_{rear} \end{bmatrix}$ and $\begin{bmatrix} x_{front} \\ y_{front} \\ \theta_{front} \end{bmatrix}$ are the poses of the front and rear vehicles respectively, and v_{front} is the speed of the front vehicle.

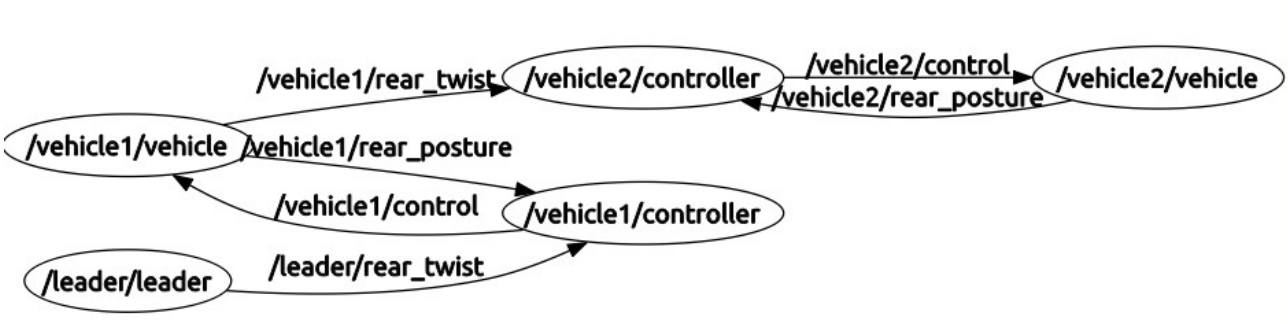
2. K_p and «$dist$» will be local parameters of the control node. To begin with, you can hard-code them and handle the parameters once your node works.

3. Modify the launch file so the first vehicle uses the default Kp and dist, while the second uses dist=2.0 and Kp=0.5 (this requires to have handled the parameters in the code).

Advices/information

- To get a pass grade, you need to have a functional control node (the robot tracks the path), even if it uses hard-coded constants instead of parameters.
- You should try to complete the pass grade task before attempting any improvement. Once you get to that level, call the teacher who will record your status.
- Code quality will be taken into account in the grading.

To help you understand the role of the nodes, here is the rqt_graph of the running application (without visualization topics).



Submission

Zip you whole platoon folder into a zip file named LastName_FirstName.zip and send as attached file to **gaetangarcia_8a6a**@sendtodropbox.com

Your submission will be date-stamped and automatically added to my dropbox account.