SOFAR Lab evaluation

Read the whole text (annex excepted) to get a general idea of the tasks before doing anything else.

Acknowledgements

This exercise uses the « Star Wars » simulator developed by Olivier Kermorgant for one of his former evaluation exercises. Thanks to Olivier for sharing his code with me.

Conditions of the exercise

During this exercise, you must remain connected to my Zoom session and **your screen must be shared at all times**. I must be able to connect to your room at any moment and see your screen.

Aim of the exercise

The aim of the exercise is to develop a mobile robot control node designed to make a robot follow another one at a certain distance.

Setting up the exercise

- I recommend you use an initially empty catkin workspace. Thus, if you decide at some point to force complete re-compilation, the compilation time will be minimized. Backup your existing files from the src folder of your workspace to a safe location, then empty the src folder and remove the build, devel and log folders of the catkin workspace. On the VM the folders would be /home/ecn/ros/build, /home/ecn/ros/devel and the source folder would be /home/ecn/ros/src.
- Copy the provided zip file to the src folder of your catkin workspace (/home/ros/src with the VM) and unzip it.
- Open a terminal and go to the root of your catkin workspace. Compile, then source setup files (source devel/setup.bash). Then launch file star_wars.launch. You should see in rviz thre Star Wars robots, one moving and two motionless R2D2 robots. Stop execution.
- If you agree to participate in the data collection operation to analyze the way students progress along a ROS exercise, open the file “mail_status.py” and enter your ECN email address in the required field line 24. The password will be asked at execution and will not be saved anywhere. You have the code, you can check that it’s not the case.
- Run the script in a terminal: go to the folder where the script is present and type “python mail_status.py”. Enter your password when requested and check that there are no errors: if the script cannot connect to the ECN mail with the provided address and password, you get an error message.

Task 1

This task must be solved using subscribers and publishers. **No use of the tf package**.

Run “star_wars.launch”.

For you to perform the tasks of this lab, it is important to understand that, contrary to what you have seen in lab 3, there is not one node for each robot. The only node you have to interact with is the “simulator” node, which simulates the behavior of all three robots of the application. In lab 3, the “master” node was the simulator of the master, and the “dog” node was used in as many instances as there were dogs, each instance simulating one dog.

You must submit answers to the questions with your package. I suggest you use LibreOffice Writer and save the file in the src folder of you package. Thus, when you submit the zipped package, you will automatically submit the document.

1.1. Identify the topics which inform you of the position of each robot. List them and determine their type.

1.2. Identify the topics you can use to move the robots. Which robots can you move? List the topics you must use and identify their type.

1.3. Read the annex to the subject. It describes the control law you need to implement. **Then** draw the graph showing the simulator node, your own node and how they connect through topics. You have to do this prior to starting any programming.

1.4. Program the node. The file “controller_1_node.cpp” already exists as a skeleton node and is taken into account in the CMakeLists.txt file. You do not need, and indeed must not modify it.

1.5. Create a launch file to make robot 2 follow robot 1. The code will be designed so that all connections between nodes are automatic in this case.

1.6. Complete the launch file so that robot 3 follows robot 2. Use remappings.

1.7. Submit to Hippocampus the zipped package star_wars

Task 2

Task 2 consists in solving the same problem, but using the tf package. Under rviz, you can visualize that each robot has a “ground” frame at its origin point, /robot1/ground, /robot2/ground…

2.1. Using these frames and a tf_listener, write a new version of the controller in controller_2_node.cpp that makes robot 2 follow robot 1 using the same control algorithm.

2.2. Modify it to use parameters to define who follows who and run the application where the three robots follow each other.

2.3. Submit to Hippocampus.

Annex – Equations of the control algorithm.

The control is based on

- The distance between the follower and the leader, along axis x of the follower, with a desired value of d0.
- The angle between vector FL and the x axis of the follower, where F (resp. L) is the reference point of the follower (resp. leader).

Figure 1. shows the geometric variables taken into account in the control.

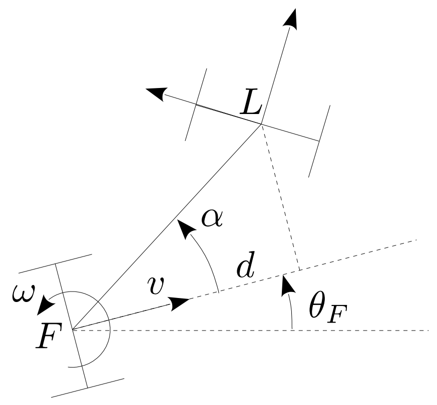

Figure 1. Variables of the control.

The speed and rotation speed of the follower are calculated as:

$v = k_\rho \rho = k_\rho (d - d_0)$

$\omega = k_\alpha \alpha$

The gains will be set to $k_d = 2.7$ and $k_\alpha = 2.9$, and the desired distance to $d_0 = 1$.

When the positions are known in the absolute reference frame, the expressions of the variables of the control are:

$$\begin{cases} d_x = {}^0x_L - {}^0x_F \\ d_y = {}^0y_L - {}^0y_F \\ \alpha = atan2(d_y, d_x) - \theta_F \\ \rho = d_x \cos\theta_F + d_y \sin\theta_F - d_0 \end{cases}$$

When the position of the leader is known in the frame of the follower, the equations become:

$$\begin{cases} \alpha = atan2({}^Fy_L, {}^Fx_L) \\ \rho = {}^Fx_L - d_0 \end{cases}$$