F1tenth Simulator QuickStart

January 15, 2021

1 Creating your workspace

First you will need to create a workspace for your ros project, for this you can either follow the tutorial here or the instructions bellow.

Before anything else it is important to verify tha ros is installed in your machine. You can use the command below to check your ros version, for this activity, version **melodic** is recommended.

```
$ echo $ROS_DISTRO
```

For the next commands, you should replace melodic for the ros version that you are using.

```
$ source /opt/ros/melodic/setup.bash
$ mkdir -p ~/f110_ws/src
$ cd ~/f110_ws/
$ catkin_make
```

Then, you should run the next command inside your workspace directory each time you open it in a new terminal.

```
$ source devel/setup.bash
```

2 F1tenth environment

Your workspace should be all set and now you can clone the f110 environment inside your source directory. The f110 environment is set in an OpenAI gym structure defined on the repository f1tenth_gym_ros.

The dynamic equations and the physical structure of the f110 car are defined on this environment along with its sensors. In addition, it subscribes to control commands, that are published in the form of a ros topic, and apply these inputs to the car, in order to actualize the system's state and its sensor's readings.

The simulated sensor's readings are periodically published by the environment on its respective topics and you should use this information to plan the next control command to be applied to the system. For this we should create another ros node, in order to facilitate the task we will start with the model proposed here.

Before following the instructions bellow, if you do not have docker installed yet, you should follow the tutorial here.

```
$ cd ~/f110_ws/src
$ git clone https://github.com/f1tenth/f1tenth_gym_ros
$ git clone https://github.com/cosynus-lix/f1tenth_quickstart
$ cd f1tenth_gym_ros
$ ./build_docker.sh
$ ./docker.sh
```

If you meet the problem on downloading docker image (if it takes too much time) or it could not correctly compile, you could try to launch the f110 environment **alternatively**:

```
$ cd ~/f110_ws/src/f1tenth_gym_ros
$ ./start.sh
```

The f110 environment is now running and you can check the topics being published by the environment, in another terminal (do not forget to source your new terminal ;)), with the command bellow.

```
$ rostopic list
```

For checking what is being published by one of the topics you can do,

```
$ rostopic echo name_topic
```

You can also visualize the relation between nodes/topics by using following command.

```
$ rqt_graph
```

For launching the ros node that will control the robot you can follow the instructions bellow in a new terminal,

```
$ cd ~/f110_ws
$ source devel/setup.bash
$ catkin_make
$ roslaunch f1tenth_controller_example wall_following_agent_node.launch
```

Now you have a robot that follows the right wall of the race track. This is a simple solution to the control problem and now you should propose your own solution by modifying the proposed code at ~/f110_ws/src/f1tenth_quickstart/src/wall_following_agent_node.py. Note that you are not allowed to change the environment and both nodes must be running at the same time, so you will need at least two terminals.

For testing head-to-head competition mode, you should firstly switch the source code version of f110 enviroment by

```
$ cd ~/f110_ws/src/f1tenth_gym_ros
$ git checkout origin/multi_node
```

and then compile and re-launch the simulator.

```
$ sudo ./build_docker.sh (it takes several minutes)
$ sudo ./docker.sh
```

At the side of controller, you should edit the class Agent() in file wall_following_agent_node.py: comment the part about "single vehicle racing" and uncomment the part about "head-to-head racing". Then you could launch the modified controller.

```
$ cd ~/f110_ws/
$ catkin_make
$ roslaunch f1tenth_controller_example wall_following_agent_node.launch
```

3 The drive topic

The topic */drive* publishes the control command that should be applied to the system in the form of an Ackermann message. You can control the robot using :

- Steering angle
- Longitudinal Speed

For single vehicle racing mode, the controller receive the environment information and send drive command as shown in Fig. 1. For head-to-head racing mode, the communication between different nodes via different topics is shown in Fig. 2 (use the same controller for both vehicles) and Fig. 3 (use different controllers).

4 Write the controller code by yourself!

We refer to the instructions at the end of the file README.md.

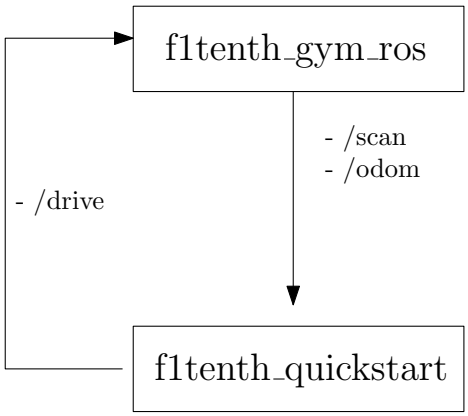

Figure 1: single vehicle racing mode

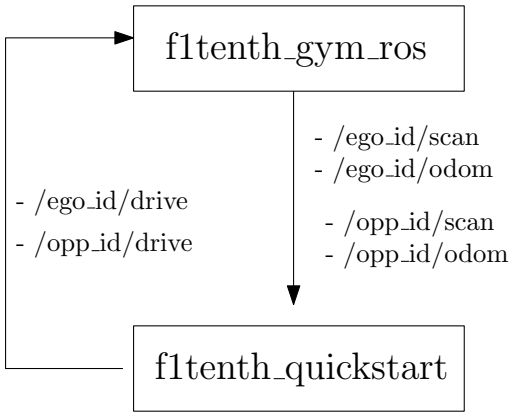

Figure 2: head-to-head racing mode

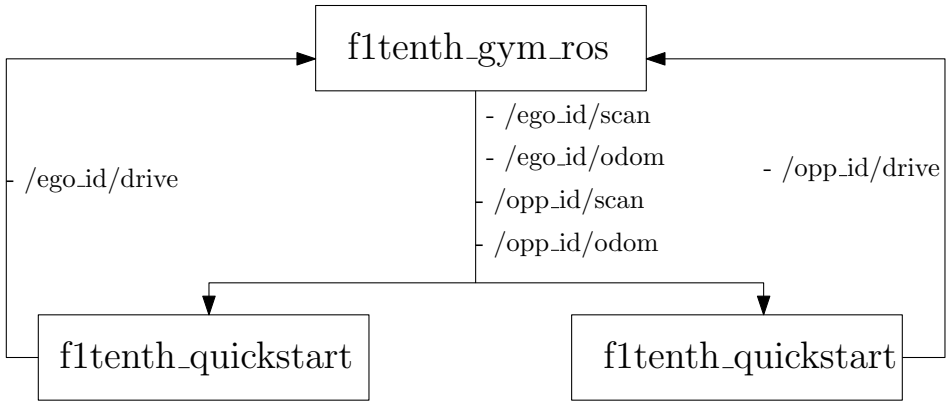

Figure 3: head-to-head racing mode