F1/10 Autonomous Racing

ROS F1/10 Autonomous Racing Simulator Installation

Lab Session CS4501

Madhur Behl madhur.behl@virginia.edu

Section [A]: Latest Simulator Full Installation (Recommended)

Section [B]: Docker Installation (Lightweight but old)

Section [C]: Keyboard TeleOp with F1/10 Simulator

Section [A] Latest Simulator Full Installation :

```
*For ROS Melodic, just replace the `kinetic` part of the instructions with `melodic`.*
```

Instructions to install the simulator on your local machine or VM.

Step 1: Installing Dependencies:

```
user@ros-computer: sudo apt-get install -y ros-kinetic-navigation
ros-kinetic-teb-local-planner*
ros-kinetic-ros-control
ros-kinetic-ros-controllers
ros-kinetic-gazebo-ros-control
ros-kinetic-ackermann-msgs
ros-kinetic-serial qt4-default
```

Step 2: Create a ROS workspace for the simulator

```
mkdir -p ~/simulator_ws/src
cd ~/simulator_ws/src
catkin_init_workspace
```

Step 3: Download source files for particle filter localization and mapping packages

In the `src` subdirectory of the `simulator_ws` run the following commands:

```
git clone https://github.com/f1tenth-dev/simulator
git clone https://github.com/mit-racecar/particle_filter
git clone https://github.com/kctess5/range_libc
git clone https://github.com/tu-darmstadt-ros-pkg/hector_slam
```

Step 4: `range_libc` configuration:

The source for the required packages, including the simulator, are now in the `simulator_ws` workspace. The simulator depends on the GPU particle filter and this package has to be configured before compiling the ROS packages. Navigate to the `range_libc` folder to compile the library necessary for the particle filter. Open a new terminal and enter the following commands:

```
user@ros-computer: sudo pip install cython
user@ros-computer: cd simulator_ws/src/range_libc/pywrapper
```

If you do not have `pip` installed, you can run the following first:

```
sudo apt install python-pip
```

There are two methods for installing the particle filter; with GPU support and without GPU support. We recommend using the package with GPU support, but leave the decision to you. For a VM it might be easier to install without GPU support.

Installing without GPU Support

In the same terminal [i.e. at the pywrapper direcotry], enter the following command and follow the instructions on the screen:

```
user@ros-computer: ./compile.sh
```

Installing with GPU Support

Keep this terminal open and pay special attention to the next step. The particle filter package depends heavily on the GPU and its architecture, so it becomes very important to match the architecture of the GPU in your local machine to the one listed in the configuration of `setup.py`. We are particularly interested in the `sm_xx` value associated with the GPU. Once the architecture type has been identified, go back to the terminal and open the setup file using an editor of your choice and navigate to line 96:

```
nvcc_flags = ['-arch=sm_20', '--ptxas-options=-v', '-c', '--compiler-options',
 "'-fPIC'", "-w","-std=c++11"]
```

Replace `-arch=sm_20` with the `-arch=sm_xx` value from the GPU. Once you have made the changes, save the file and exit back to the terminal and enter the following command to compile the library and follow the instruction on the screen:

```
user@ros-computer: ./compile_with_cuda.sh
```

Step 5: Build and source the simulator

With the final dependency installed, the installation process can now be continued using ROS `catkin_make`. This process is relatively simple; open a new terminal and enter the following command:

```
user@ros-computer: cd ~/simulator_ws
user@ros-computer: catkin_make install
```

The installation process in now complete, and the workspace needs to be sourced. Sourcing the workspace permanently helps launching the simulator easier. To do this, the `.bashrc` file in the home directory has to be modified; open a new terminal and enter the following command:

```
user@ros-computer: echo "source ~/simulator_ws/devel/setup.bash" >> ~/.bashrc
user@ros-computer: source ~/.bashrc
```

You have now completed the steps necessary to install the simulator.

Step 6: First Time Setup

The race track course may sometimes not be installed by `catkin_make` though the files are already present in the correct directory. To overcome this situation, the course will have to be manually transferred from the `simulator_ws` workspace to the `~/.gazebo` workspace. To do this open a new terminal and enter the following command:

```
user@ros-computer: cp -r ~/simulator_ws/src/simulator/world/race_track ~/.gazebo/models
```

The above step needs to be completed only once, and the Gazebo simulator will be able to read the course information everytime the F1/10 simulator is launched. The user can now enter the following command in the terminal to launch the simulator for the first time:

```
user@ros-computer: roslaunch f1tenth-sim simulator.launch run_gazebo:=true
```

Notice the information on the terminal; if there are any errors during installation, they would be shown in a red font. Not all errors on the terminal are fatal and not all fatal errors are necessarily displayed on the terminal. The Gazebo window should eventually come up on the screen and you should be able to see the simulated environment. Use the mouse to navigate and get yourself familiar with the simulator.

Press `ctrl + c` to exit the simulation session.

Section [B] Docker Installation (Old sim version):

Quick Start:

Run the f1tenth docker container and open port `6080`

```
docker run -it --rm -p 6080:80 madhurbehl/f1tenth
```

Then simply browse http://127.0.0.1:6080/

Using the `f1tenth` docker image:

The easist and fasted way to get started with the tutorials is to install docker and spin a container from the `f1tenth` docker image.

For this you will need to install Docker on your computer. If you have used Docker before this step should be straight forward, otherwise check https://www.docker.com/what-docker for more information about it.

- Installation's instructions for Ubuntu can be found https://docs.docker.com/engine/installation/linux/ubuntu/
- Installation's instructions for Windows can be found [https://docs.docker.com/docker-for-windows/install/
- Installation's instructions for MacOS can be found https://docs.docker.com/docker-for-mac/install/

After you have Docker installed, running the container should be as easy as typing:

```
docker run -it --rm -p 6080:80 madhurbehl/f1tenth
```

Then simply browse http://127.0.0.1:6080/

About the Docker sim (first time setup)

The F1/10 Docker Simulator has been setup to help the user get started with the simulator out-of-the box.

The `simulation` sub-package contains one-line commands that perform these tasks parallelly, but we recommend that first-time users understand the processes before using these commands. If you are already experienced in ROS, the launch files are present under `'/simulator/launch'` directory.

Starting the simulator in the Docker session

To bring up the F1/10 Gazebo simulator using the following command in a terminal in the web viewer of the Docker container.

```
user@computer:$ roslaunch racecar_gazebo racecar.launch
```

You should now see the default F1/10 world loaded into Gazebo and see the red F1/10 simulated car with the blue LIDAR rangefinder visualized.

Note: You may need to source the ROS environemnt in Docker

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Section [C] Keyboard TeleOp:

For the latest (non-Docker) simulator:

Keyboard teleoperation is the process of controlling the movement of the racecar using using computer's keyboard. The simulator release contains two teleoperation nodes meant to be ued with either a joystick game controller or the keyboard. This tutorial will focus only on the latter. For convenience, the keyboard teleoperation follows the conventional gaming control pattern `W-A-S-D` where W and S control forward and reverse velocities and A and D control steering position from left to right. **The teleoperation node latches onto the control state of the car during the last recorded key stroke and makes this control state persist until a new command is recieved.**

To bring up the simulator with the keyboard teleoperation node, kill any other simulation or Gazebo session that is active and enter the following command in a new terminal:

```
user@ros-computer: roslaunch f1tenth-sim simulator.launch run_gazebo:=true keyboard_cor
```

Wait for the simulator session to be launched and the Gazebo GUI to appear. The terminal from which the session was launch needs to be the active terminal and placed on top of the Gazebo GUI in order to accept the commands from the keyboard. Use the W-A-S-D keys to navigate the racecar around the racetrack. Use this opportunity to make yourself familiar with the handling of the racecar

For the Docker sim (old)

The F1/10 console package provides you with the option of using either keyboard control or joystick control. The package is built around the Logitech F710 game controller or the standard English(US) keyboard. IF you do not have the F710, you can use any other controller supported by the ROS joy-node and change the axis-mapping or just use the keyboard control by using the following command:

```
user@computer:$ roslaunch console keyboard_teleop.launch
```

Keyboard control in this package follows the WASD pattern: W: Move forward S: Move reverse A: Turn left D: Turn right

NOTE: When using *keyboard_teleop*, always keep the terminal from which the teleop node was initialized as the active terminal window over other screens, otherwise the keyboard data will not be sent to the car. You should now be able to see the car respond to manual control commands - steering, and accelration.