Initial setup for the UR5 communications with ROS

Angadh Nanjangud

started on: November 7, 2017
edited on: November 17, 2017

1 Introduction

This is a document that will inform the user on using ROS to communicate with the UR5 on their personal computer. It will also discuss the usage of MoveIt! for motion planning of the UR5 and execute said plans on the physical robot.

The pre-requisites are that the user have ROS Kinetic. The operating system that is being used for this tutorial is Ubuntu 16.04. YMMV depending on what you have. It is recommended that the user go through the ROS tutorials available at `wiki.ros.org/tutorials`.

2 Setting up UR5 to communicate with an Ubuntu machine via ROS

The UR5 is decently documented (but not exceptionally well). The ROS page on 'Getting Started with a Universal Robot and ROS-Industrial' is a good starting point. However, I will document the entire process here for sake of completeness.

- **Step 1: Setup a catkin workspace** Open up a terminal and create a catkin workspace following the instructions here. I chose to call the catkin workspace driver_ws.

```
$ cd ~/driver_ws
$ catkin_make
$ source devel/setup.bash
```

- **Step 2: Obtain UR5 driver** First, acquire the necessary universal robot ROS packages by running the following command in your terminal:

```
$ sudo apt-get install ros-kinetic-ur-driver
    ros-kinetic-universal-robot
```

Since the UR5 robot is running Polyscope 3.x, an additional installation of the `ur_modern_driver` is necessary which can be obtained via git. Ensure that this installation is being done in the previously built catkin workspace

from step 1; in our case, this is the `/driver_ws/src` directory. We then run `catkin_make` again and also source the `setup.bash`file again:

```
$ cd ~/driver_ws/src
$ git clone https://github.com/Zagitta/ur_modern_driver.git
$ cd ~/driver_ws/
$ catkin_make
$ source devel/setup.bash
```

- **Step 3: Connect UR5 and computer** Follow instructions on step 3.2 on 'Getting Started with a Universal Robot and ROS-Industrial' to configure the IP of the robot. This is used in Step 4 to connect the robot to the dektop computer. In my setup, the desktop and robot's CB3 unit are wired to the router. The robot's IP has been set to 192.168.0.50.
- **Step 4: Testing the connection** In the same terminal where the `source devel/setup.bash` was executed, run the following:

```
$ roslaunch ur_modern_driver ur5_bringup.launch
    robot_ip:=192.168.0.50
```

MAKE SURE NOBODY AND NOTHING IS IN THE VICINTY OF THE ROBOT BEFORE RUNNING THIS NEXT STEP. Open a new terminal and run:

```
$ rosrun ur_driver test_move.py
```

You should see the robot come to life at this point. `test_move.py` is a script within the `ur_modern_driver` that we installed from git.

3 MoveIt!, RViz, and the real UR5

With the driver still running in one of the terminals from step 4 (i.e. `roslaunch` command), open a new terminal and run:

```
$ roslaunch ur5_moveit_config ur5_moveit_planning_execution.launch
    limited:=true
```

which basically loads what is needed for motion planning. In the above snippet, the flag `limited:=true` loads limits the joint motions to the range $[-\pi, \pi]$. Then, we launch athe RViz gui with MoveIt! motion planning plugin by running:

```
$ roslaunch ur5_moveit_config moveit_rviz.launch config:=true
```

4 MoveIt!, RViz, and the UR5 on Gazebo

Instead of connecting to the real robot as explained in step 4, open a new terminal and run:

```
$ roslaunch ur_gazebo ur5.launch
```

and then do the same as mentioned in the previous section.

```
$ roslaunch ur5_moveit_config ur5_moveit_planning_execution.launch
    limited:=true
```

which basically loads what is needed for motion planning. In the above snippet, the flag `limited:=true` loads limits the joint motions to the range $[-\pi, \pi]$. Then, we launch athe RViz gui with MoveIt! motion planning plugin by running:

```
$ roslaunch ur5_moveit_config moveit_rviz.launch config:=true
```