Introduction to TurtleBot

EECS 275: Fundamentals of Robotics

INTRODUCTION

The robot hardware used for this course is version of the Turtlebot 2 that is produced by Autonomous. The sensors on this platform include a microphone, bumper sensors, and a first generation Microsoft ® Kinnect ® 3D depth camera. This particular TurtleBot uses an NVIDIA ® Jetson TK1 quad-core ARM ® board with 192 GPU CUDA ® core.

TurtleBot makes extensive use of ROS. This laboratory will provides an introduction to using the platform and demonstrates the methods to control the robot from a secondary computer over the network. *Please note:* This type of operation is subject to difficulty. Debugging and understanding the difficulties is an express purpose of the exercises below.

EXERCISES

There are six robots in the Glennan 210 Laboratory. Groups should use the same platform for subsequent laboratories. The robots have the names: deeplearning0<1-6>w and have static IPs. These are the names associated with the Wi-Fi interfaces, removing the "w" are the wired interfaces were the robot to have a wired connection. Labels identifying the robots are located on the top of the robots.

Turn on the Robot

Set the master power switch for the robot to the "on" position. The NVIDIA ® Jetson TK1 board must be powered up manually after the master power switch.

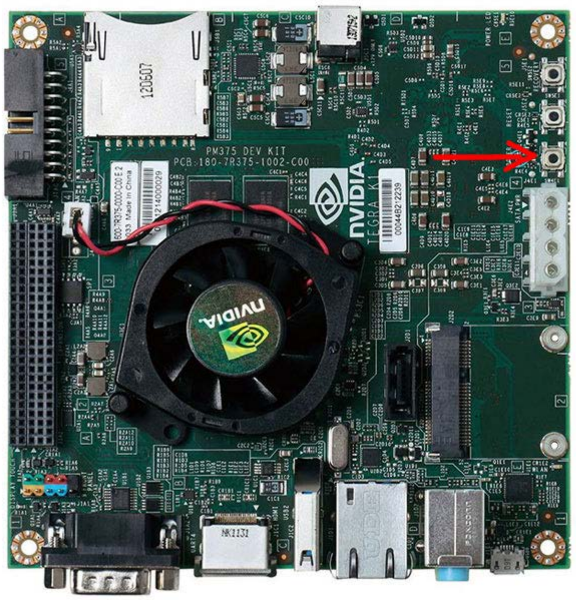

Figure 1: The power button for the NVIDIA ® Jetson TK1 board must be pressed after the main power has been switched on. A green LED near the switch will illuminate when the board is turned on.

Connecting to the Robot

The next few steps are done manually to understand the process, but it is possible to put these commands into the script that runs during the login process, though the lab computers may not retain those settings.

ROS is designed to exist on the network, though it defaults to local operation. To leverage the network distributed capabilities of ROS, a single ROS Master (in this case, the robot) must be configured to use the external network interface, not the "localhost" interface. This is part of the bootup scripts on the robots and is already complete. Any other computers that will connect to the ROS Master must be configured to use the robot as the ROS Master instead of starting a local one. This configuration is applied after the ROS `setup.bash` script has been run, but before running anything else in ROS. (Note: If you have taken to manually running `roscore` instead of allowing `roslaunch` to manage this automatically, you must not run `roscore` manually, the ROS Master is located on the robot now and this second ROS Master will conflict.)

```
# Setup ROS
source /opt/ros/indigo/setup.bash
# Where is the ROS Master
export ROS_MASTER_URI=http://deeplearning0{1-6}w.eecs.cwru.edu:11311
```

To verify that you are connected to the robots, list the available ROS topics.

```
rostopic list
```

The list of topics will be fairly long and include `/mobile_base` if everything is working. If there is no `/mobile_base` entry, it is possible that the list of ROS topics is from the local machine and not the robot. (There should be not ROS Master [`roscore`] running locally.)

```
# The following command will filter all the processes on the local computer looking
for roscore.
ps aux | grep roscore
```

Note: Logging directly into the robots (e.g., via `ssh`) is heavily discouraged. No part of this laboratory should require any actions be performed directly on the robots. It is possible this may be more necessary later this semester, though.

Drive the Robot

ROS contains many useful command line and GUI tools to inspect the system. Executables such as `rosnode`, `rostopic` and `rosparam` allow users to retrieve information about ROS nodes currently connected to the system, view the messages that are being either published or subscribed to by nodes, and inspect the parameter server. The executable `rqt_gui` is a GUI interface to the ROS system that has a number of plugins that extend from low level listing and reading of messages, and writing specific messages from and to the ROS system, to higher level tools that show the hierarchy of nodes and messages, allow robots to be driven and to view the data from sensors. To start the `rqt_gui` executable, use the following command:

```
rosrun rqt_gui rqt_gui &
```

This interface contains a tool that can drive the robot around. To use it, add the Steer Robot tool from the Plugins->Robot Tools menu. The default address for driving robots is `/cmd_vel`. That must be changed to `/mobile_base/commands/velocity`. Use the interface to drive the robot.

This exact same functionality can be used with the simulator by setting an environment variable to simulation.

Initiate a 3D TurtleBot Simulation

One of the most powerful benefits of ROS is the ability to swap between simulation and hardware operation with ease. Launch a simulation of the TurtleBot and launch the ROS nodes to run the robot.

```
# Start the Gazebo simulation with a TurtleBot
roslaunch turtlebot_gazebo turtlebot_world.launch &
# Launch the ROS nodes to operate the robot in simulation mode
roslaunch turtlebot_bringup minimal.launch simulation:=true
```

With the simulation running, it is possible to connect to the simulated robot in exactly the same was as above.

```
rosrun rqt_gui rqt_gui &
```

It is possible to drive the simulated robot exactly as the real robot. Do not kill your simulation, it will be used in the next section.

Visualize the Robot Sensor Information

The `rqt_gui` is a more of a debugging, level interface to the ROS system (though a sizeable step up from the command line utilities and with some higher level snap-ins). To see what the robot sees, the `rviz` tool is quite useful. For this exercise, create a new directory in the `src` directory of a catkin workspace. Create a new package named turtle-test. Remember to run the setup script in this workspace to make sure everything that should be there is.

```
# Create the basic catkin workspace directory structure
mkdir –p catkin_ws/src
cd catkin_ws/src
# Create a catkin package for now
catkin_create_pkg turtle-test turtlebot
cd ../
# Make the packages
catkin_make
# Run the setup script to let ROS know about the packages in the catkin workspace.
source devel/setup.bash
roscd turtle-test
# Start from scratch
rm –rf ~/.rviz
rviz
```

There should be a new RVIZ window showing nothing. To see a representation of the robot, add a robot model by selecting Add->RobotModel. A new element should now be visible in the Displays window. Expand the “RobotModel” element, there will be an error. Inspect the error. To fix the error, change the “FixedFrame” under the Global Options element to be `base_link`. A robot should now be visible in the main part of the window.

Add a “PointCloud2” element to the Displays and change the Topic to `/camera/depth/points`.

Add an Image and change the Image Topic to `/camera/rgb/image_raw`. Add a second image and set the Image Topic to `/camera/depth/raw_image`.

Drive the Robot

Open the `rqt_gui`. Add the Plugins->Robot Tools->Robot Steering to the panel. Change the `/cmd_vel` to `/mobile_base/commands/velocity`. Try driving the robot. Try it with the actual robot hardware, carefully!

QUESTIONS

Take a screenshot of someone in your group as viewed through the RVIZ interface. The screenshot utility can be found in the Program Listings of the computer desktop.

Please provide edits and comments for improvements of this laboratory. Please use the “Review” functionality of MSWORD to edit and alter the text of this laboratory. Enable Review->Track Changes and edit the document as you wish. Add directed comments using the Review-New Comment functionality. General comments and recommendations for improvements in the next section.

COMMENTS AND RECOMMENDATIONS