Lab 1: Introduction to ROS

CSCI 545

Spring 2024

1 Intro

In this lab, you will be setting up your system and getting acquainted with ROS. This lab is very important since we will build upon this for subsequent labs. You will need to submit a lab report (in pdf) and two python files (for the publisher/subscriber exercise).

Submission format: Your submission should be based on your group's GitHub repository. Create a folder called `Lab1`. Files should be placed in that folder like this:

```
Lab1/report.pdf
Lab1/code/<files>
```

See Section 9 for more details about the report.

2 Working with your project repository

1. First, clone your repository for the lab.
   ```
   git clone https://github.com/csci-545-spring-2024-classroom/<your-repo>.git
   cd <your-repo>
   ```
2. Make your changes for the lab as necessary.
3. If you haven't already, commit all your changes.
   ```
   git add <files to add>
   git commit -m <Your commit message>
   ```
4. Push your changes to the repository (we will only look at the main branch for grading).
   ```
   git push origin main
   ```

3 Pre-requisites for this tutorial

- Your VM or native install should have ROS-kinetic (for Ubuntu 16.04) or ROS-noetic (for Ubuntu 20.04) installed.

4 Verify ROS Install

In the following text for this (and subsequent) sections, we assume that your catkin workspace is in `~/catkin_ws`, but if you are using our ROS-noetic VM, you might be using `~/ros_ws` instead. If that is the case, just replace `catkin_ws` with `ros_ws`.

1. Create a catkin workspace

   ```
   mkdir -p ~/catkin_ws/src
   cd ~/catkin_ws
   catkin build
   source ~/catkin_ws/devel/setup.bash
   ```

   Permanently add to path

   ```
   echo 'source ~/catkin_ws/devel/setup.bash' >> ~/.bashrc
   ```

2. Check environment variables

   ```
    printenv | grep ROS
   ```

   Check if `ROS_ROOT, ROS_PACKAGE_PATH`, etc. are setup. If not, run
   `source /opt/ros/kinetic/setup.bash`
   (for ROS-kinetic) or run
   `source /opt/ros/noetic/setup.bash`
   (for ROS-noetic) and then
   `source ~/catkin_ws/devel/setup.bash`

   For a permanent solution, you should add these lines to your `~/.bashrc`

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

   (for ROS-kinetic) or

   ```
   echo ‘source /opt/ros/noetic/setup.bash‘ >>~/.bashrc
   ```

   (for ROS-noetic)

   ```
   echo ‘source ~/catkin_ws/devel/setup.bash‘ >>~/.bashrc
   ```

3. You should be able to see which ROS packages are installed. `rospack list` You can see where a package is installed with `rospack find <package>`

5 Create your own package

The simplest possible package might have a structure which looks like this:

```
my_package/
    CMakeLists.txt
    package.xml
```

Make a new package as follows:

1. `cd ~/catkin_ws/src`
2. `catkin_create_pkg cs545_lab1 std_msgs rospy roscpp`
   This creates a package named cs545_lab1 that lists std_msgs, rospy, and roscpp as dependencies
3. Build your catkin workspace `cd ~/catkin_ws/`
   `catkin build`
4. After build succeeds `source ~/catkin_ws/devel/setup.bash`
5. Verify that your package has installed
   `rospack list | grep cs545`
   Should return the name of your package

   If you ever need to change dependencies for you package, you would have to edit the package.xml and CMakeLists.txt. See this tutorial page for more info.

6 ROS exercises

Note: You will need multiple terminals for this section. Consider using programs to handle this use-case, such as tmux or Terminator.

1. Run turtlesim
   (a) Terminal1: `roscore`
   (b) Terminal2: `rosrun turtlesim turtlesim_node`
   (c) Terminal3: `rosrun turtlesim turtle_teleop_key`
   (d) Select Terminal3 and then use the arrow keys to move the turtlebot
2. Find topics using rostopics
   (a) Get list of topics with `rostopic -v`
   (b) Run `rqt_graph` to see a graphical representation of the nodes. Explain what you see in 1-3 sentences.
   (c) Get some info from a rostopic.
3. Record data using `rosbag record`
   (a) Terminal4: `mkdir /tmp/bagfiles # gets cleared on reboot`
   (b) `cd /tmp/bagfiles`
   (c) `rosbag record -O baggy -a`
   (d) Move turtlebot using arrow keys in Terminal3
   (e) After recording for a few seconds, kill (Ctrl-C) the rosbag record in Terminal4
4. Play it back using `rosbag play`
   (a) Kill (Ctrl-C) the turtle teleop key in Terminal3

(b) `cd /tmp/bagfiles`

(c) `rosbag info baggy.bag`

(d) `rosbag play baggy.bag`

5. Explain in 3-5 sentences the difference between rosservice, rostopic, rosparams, and rosbag.

7 Publisher and Subscriber Topics

This section requires you to write code to create a publisher and subscriber for a topic. Refer to the ROS tutorial slides presented in class or the official ROS tutorials for more information on this. You may need to use multiple terminals.

1. Write a publisher, using the skeleton code provided in `publisher_exercise.py`
2. Move the file to `~/catkin_ws/src/cs545_lab1/scripts/`
3. `catkin build cs545_lab1`
4. Make the publisher executable
   `chmod +x ~/catkin_ws/src/cs545_lab1/scripts/publisher_exercise.py`
5. Run `roscore` to start ROS
6. Run your publisher: `rosrun cs545_lab1 publisher_exercise.py`
7. Verify that the topic is being published using `rostopic list` and `rostopic echo`
8. Write a subscriber, using the skeleton code provided in `subscriber_exercise.py`
9. Do the necessary steps to build and run the subscriber. Make sure you are able to receive the message and are printing it out.

8 Additional Questions

As a very rough guideline, we anticipate that for most teams, the answers to each question below will be approximately one paragraph long (or about 4-5 bullet points). However, your answers may be shorter or longer if you believe it is necessary.

8.1 Resources Consulted

Question: Please describe which resources you used while working on the assignment. You do not need to cite anything directly part of the class (e.g., a lecture, the CSCI 545 course staff, or the readings from a particular lecture). Some examples of things that could be applicable to cite here are: (1) did you get help from a classmate *not* part of your lab team; (2) did you use resources like Wikipedia, StackExchange, or Google Bard in any capacity; (3) did you use someone's code (again, for someone *not* part of your lab team)? When you write your answers, explain not only the resources you used but HOW you used them. If you believe your team did not use anything worth citing, *you must still state that in your answer* to get full credit.

8.2 Team Contributions

Question: Please describe below the contributions for each team member to the overall lab. *Furthermore, state a (rough) percentage contribution for each member.* For example, in a team of 4, did each team member contribute roughly 25% to the overall effort for the project?

9 Report

Write a report comprising of all the textual answers required in this lab. The report should be named `report.pdf`). *Your report must also answer BOTH questions in Section 8. Include both in the report and make it **very easy** for us to find.*