**Lab 1. ROS Tutorial and Racecar Simulator**

Project Lead: submit codes and peer evaluation form.

Subscriber: submit lab report and peer evaluation form

Due date: Wednesday Feb. 26, 2020 on Course Site

**Major Tasks:**

1. Configure and test the Lidar sensor;
2. Learn the basics of ROS Melodic Morenia;
3. Run racecar simulator;
4. Write python code for publisher and subscriber nodes.

**Preparation before the lab:**

0.1 Have Ubuntu 18.04, NVIDIA SDK Manager, and ROS Melodic Morenia installed on the host computer and Jetson device;

0.2 Check that your F1/10 car hardware has been built/assembled properly, including the Lidar.

0.3 If you are not familiar with Linux command lines, please go through the tutorial here: <https://ubuntu.com/tutorials/command-line-for-beginners>. It takes ~50 minutes for a beginner to work through it. I suggest you also practice effortful retrieval (<http://sites.gsu.edu/scholarlyteaching/effortful-retrieval/> ) with your teammate. Your time is well spent on these tutorials since we will be using the basic commands for the rest of the semester. Also refer to the basic Linux/Unix command list here <https://files.fosswire.com/2007/08/fwunixref.pdf> .

**Part 0. Lidar Installation:**

Procedures for Lidar installation and testing:

* Follow the instructions on the lecture slides to install your lidar (UST-10LX) on your car.
* Set up Ethernet connection between the lidar and the Jetson with a static IP address.
* Install Lidar package in ROS by $sudo apt-get install ros-melodic-urg-node
* (optional) change the factory default IP address to 192.168.1.xxx by the IP Changing Tool from <https://www.hokuyo-aut.jp> (need to setup an account to download the tool)
* Run the lidar scan in ROS by

Terminal 1: $roscore

Terminal 2: $rosrun urg\_node urg\_node \_ip\_address:="192.168.0.10"

Terminal 3: $rostopic echo /scan

Terminal 4: $rosrun rviz rviz

Refer to Rviz tutorial: <http://wiki.ros.org/rviz/Tutorials>. If you changed the IP address of the lidar, then use the new IP address in the command of Terminal 2. Also run rosnode list, rosmsg, rqt\_graph, etc to gather more information about the Lidar node.

Your report shall include your experience working with the scan messages and answer the following questions:

Q0.8. What type of node (publisher or subscriber) is the urg\_node? What channel does it communicate on? What type of message does it have?

Q0.9. What other nodes does the Lidar communicate with? How do they show on the rqt\_graph?

Q0.10. If you tried to change the IP address of the Lidar, report your experience for a max of 10 bonus points.

Your code submission shall include: the log file of the “rosrun echo /scan” messages.

**Part 1. ROS Tutorials:**

Follow the videos and tutorials 1&2 provided on the F1/10 website:

ROS Tutorial Part 1: [http://f1tenth.org/code.html#s1t1](http://f1tenth.org/code.html)

ROS Tutorial Part 2 here: [http://f1tenth.org/code.html#s1t12](http://f1tenth.org/code.html)

Work through the tutorials on both host computer and Jetson device. You may also refer to the original ROS tutorials for Beginners here: <http://wiki.ros.org/ROS/Tutorials>. Specifically, Tutorial Part 2 is covered in Creating a ROS Workspace <http://wiki.ros.org/catkin/Tutorials/create_a_workspace> and Creating a Package <http://wiki.ros.org/ROS/Tutorials/CreatingPackage>.

The objectives of Part 1 of Lab 1 are getting familiar with ROS framework, understanding how publisher and subscriber are implemented, and working with graphical tools such as rqt\_graph and Rviz.

Your report shall demonstrate/state that both team members work through the tutorials. Make sure to also answer the following questions.

Q1.1. What are the three layers of ROS hierarchy? What are the contents in each of the layers in the turtlesim example?

Q1.2. What is a topic, a message, a publisher, or a subscriber in the turtlesim example?

Q1.3. How do you open a new terminal using a key shortcut? How do you save the rqt graphs?

Q1.4. How many terminals does the turtlesim example use? What are they? How to exit a ROS process without closing the terminal?

Q1.5 In which directory would you run catkin\_make? What are the files and folders generated by catkin\_make in the beginner\_tutorial example? What is a CMakeLists.txt? What is the difference between the CMakeLists.txt in ROS and a make file used for compiling C++ objects?

Q1.6. What are the $source commands in the tutorials? What is the significance of sourcing the setup files? (or What does source do?)

Q1.7. What problems did you encounter working through the two tutorials? Did you watch the tutorial video and/or work through the original ROS tutorials? If so, report your experience: how are they helpful to your learning?

Your code submission for Part 1 of Lab 1 shall include: none, or any video or film you made that may help others with the procedures.

**Part 2. Implementing a publisher and subscriber:**

* 1. Work through the ROS tutorial “Writing publishers and subscribers in python” <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>
  2. Work through the Racecar Simulator: Download the Lab1Readme.MD file from Course Site Lab1 assignment. Follow the instructions in the readme to install the racecar simulator package and its dependencies. When creating a workspace, name your workspace <team#\_ws> instead of <catkin\_ws>. When creating a package, name your package <team#\_roslab1> instead of <beginner\_tutorials>. If you run into an error after running catkin\_make saying: “CMake Error at /opt/ros/melodic/share/cv\_bridge/cmake/cv\_bridgeConfig.cmake:113 (message):

Project 'cv\_bridge' specifies '/usr/include/opencv' as an include dir, which is not found.”

then use this solution: edit the file to change the opencv installation directory:

$sudo vim /opt/ros/melodic/share/cv\_bridge/cmake/cv\_bridgeConfig.cmake

On line 96: set(\_include\_dirs "include;/usr/include;/usr/include/opencv"); add the number “4” after opencv, then save and exit. Catkin\_make again to continue the installation.

Run the racecar\_simulator and record your findings. Find additional information about the simulator and lidar messages by running the following commands in different terminals:

$rostopic list

$rostopic info /scan

$rosmsg show sensor\_msgs/LaserScan

$rostopic echo /scan

The “ranges” data contain 1080 distance measurements per sequence at a fixed angle increment. Refer to data type documentation for details on the sensor\_msgs type.

<http://docs.ros.org/melodic/api/sensor_msgs/html/msg/LaserScan.html>

2.3. Write a python program to create a new node which subscribes to the /scan topic and publishes processed lidar data to two separate new topics: /closestpoint and /farthestpoint. The subscriber node shall receive the lidar data, then process the “ranges” data. The data processing function shall keep the values in “ranges” that are between “range\_min” and “range\_max”, then find the minimum and maximum values in the remaining data for each sequence. The publisher node shall publish nearest\_point = min(ranges) and farthest\_point = max(ranges) on the corresponding topics. Demonstrate your work to the TAs when you complete this task.

Requirements: Make sure your code include the author names, course and lab numbers, date and other related info in the comments. Your codes shall follow the Python-ROS coding style guide: <http://wiki.ros.org/PyStyleGuide>

Tips:

1. You may create your own python codes or modify the sample codes talker.py and listener.py. To edit a file in a terminal, use rosed which uses a default editor vim or nano. Alternatively, you may also use other text editors such as Atom, Subline Text, emacs or gedit.
2. To test or debug your python code, use the pdb (python debugger). Refer to this tutorial for details: <https://realpython.com/python-debugging-pdb/>.
3. Import the proper msg types Float64 and LaserScan.

Your report for Part 2 of Lab 1 shall include answers to the following questions:

Q1.8 What does rospy.init node() do in the program? What happens if you do not include the option “anonymous=True” in rospy.init node()?

Q1.9 How do you control callbacks for the subscriber?

Q1.10 Rate, time or duration are important in robotics. How do you use rospy.rate() in this example to make best use of the lidar data?

Code submission for Part 2 of Lab 1: the python code implementing the subscriber and publisher nodes.

**Part 3. Implementing custom messages and working with bag files:**

* 1. Work through the ROS tutorial “Creating s ROS msg and srv” <http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv>.
  2. Create a new custom message “scan\_range.msg” which includes the maximum and minimum values of the Lidar ranges. Publish the message over a new topic called /scan\_range. Make sure to include the ROS Header in your message.
  3. Work through the rosbag tutorial <http://wiki.ros.org/rosbag/Tutorials> Part 1 “Recording and playing back data” <http://wiki.ros.org/ROS/Tutorials/Recording%20and%20playing%20back%20data>
  4. Run the racecar\_simulator and record a few seconds of the simulation with “$rosbag record –a”. Note what have been recorded. Demonstrate this to the TAs with “$rosbag play –r 2 ”.

Your report for Part 3 of Lab 1 shall include answers to the following questions:

Q1.11 What is the ROS Header msg? How do you include the header msg in your custom messages?

Optional: Incorporate the custom message in the python code you have developed in Part 2 of Lab 1. Run the node and show your message with “$rosmsg show” in a new terminal.

Q1.12 What did you observe when you run rosbag play in the tutrtlesim example and the racecar\_simulator example? Where are the bag files saved when you run rosbag? How do you change the location of the bag files when saving them?

Q1.13 (Optional) If you are to rosbag record only the Lidar data and extract a specified number of sequences, how do you do this? Show your implementation to the TAs.

Code submission for Part 3 of Lab 1 shall include: the scan\_range.msg file and the related files that you have modified for Part 3.2. Please zip them in a .zip file. Optional: the new py file that makes use of the new message for Q1.11 and any programs and data files for Q1.13.

**Part 4. Using launch files:**

* 1. Work through the ROS tutorials “Rviz” <http://wiki.ros.org/rviz/Tutorials> and roslaunch <http://wiki.ros.org/roslaunch>
  2. Create a new custom launch file “team#\_lab1lidar.launch” to start the Lidar (as in Part 0) with the code you developed in Part 2, along with rosbag and Rviz. You may modify the simulator.launch in the racecar\_simulator to create your team#\_lab1lidar.launch. Your rosbag shall have a specified location for rosbag record, and the Rviz shall have a customized Rviz configuration. When launching Rviz, the lidar topic has to be manually added to the Rviz GUI. Once added, save the configuration file as team#\_lab1lidar.rviz. Your launch file shall then use the new Rviz configuration file and automatically start the lidar in Rviz GUI when launching the package. Demonstrate your implementation to the TAs.

Tips: Refer to <https://answers.ros.org/question/11845/rviz-configuration-file-format/> for saving the Rviz configuration file. To include the Rviz configuration in your launch file, refer to:

<https://answers.ros.org/question/287670/run-rviz-with-configuration-file-from-launch-file/>

Your report for Part 4 of Lab 1 shall include answers to the following questions:

Q1.14 What are the launch files used in the original racecar\_simulator? How do the multiple launch files work together?

Q1.15 Describe the procedure you take in accomplishing Part 4 of Lab 1. What difficulties do you encounter with Part 4 of Lab 1? What did you learn in this experience?

Code submission for Part 4 of Lab 1 shall include: the new launch file “team#\_roslab1.launch” and its related configuration files.