

Project 1

Learning Robot Operating System (ROS)

Assigned: January 29, 2019

Deliverable 1 (Task 1) Due Date: February 5, 23:59:59

~~Deliverable 2 (Tasks 2 & 3) Due Date: February 17, 23:59:59~~

Deliverable 2 (Tasks 2 & 3) Due Date: February 19, 23:59:59

Note: Project 1 has a four-day late submission period for each deliverable.

Late submissions will lose 25% points per day of that specific deliverable.

Introduction

In this project, the objective is to setup the ROS development environment, write a “Hello World” program using ROS, get a robot simulation working, and test the mapping capability provided by ROS. This project provides an opportunity for you to understand open-source software for robotics applications and become ready for future course projects.

Project 1 must be done **individually**, although group discussion is encouraged.

Task 1: Setup Development Environment

You are required to finish the following steps:

1. Install **Ubuntu 18.04 LTS**:

<https://ubuntu.com/download/desktop/thank-you?country=US&version=18.04.3&architecture=amd64>

Download the ISO file named: ubuntu-18.04.3-desktop-amd64.iso

IMPORTANT NOTE: Please install the 18.04 version, you may have issues if other versions of Ubuntu are installed.

Note: A physical installation is preferred, either Ubuntu only or a dual boot system. If you don't want a physical installation, you can use Amazon Web Services (AWS) to launch a virtual machine. The connection with AWS can be slow depending on your Internet speed. Instructions of using AWS are also provided on the course website: <http://inside.mines.edu/~hzhang/Courses/CSCI473-573/assignment.html>

2. Install **ROS Melodic**, and Make sure to perform “**Desktop-Full Install**”:

<http://wiki.ros.org/melodic/Installation/Ubuntu>

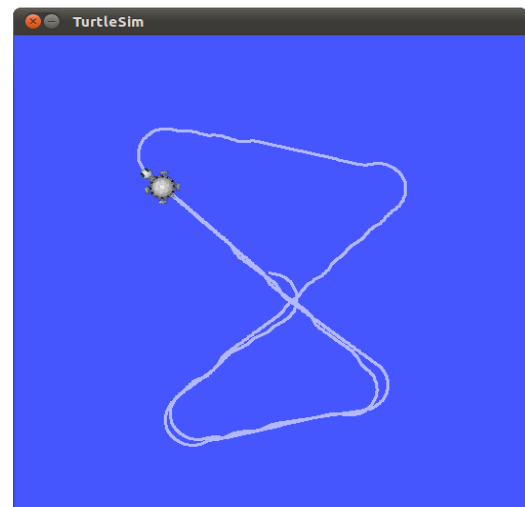


Figure 1. Snapshot of a working turtlesim in ROS.

IMPORTANT NOTE: Please install the exact ROS version and follow the exact steps!

3. Go through ROS tutorial 1.1.1-1.1.9, i.e., “Beginner Level” from the beginning (1.1.1) to “Using roscd to edit files in ROS” (1.1.9). The tutorial is available at: <http://wiki.ros.org/ROS/Tutorials>. Follow the tutorial: <http://wiki.ros.org/turtlesim> to play with the Turtlesim tool and understand how ROS works (to some extent). For Turtlesim, make sure you follow the tutorial under ROS Melodic.

What to submit:

You are required to submit a screenshot of a working turtlesim window for grading, similar to Figure 1. Name the figure T1_*firstname_lastname*.png (or jpg) and submit to the Canvas portal named P1-T1.

Task 2: Understand the Gazebo Simulator

Install ROS packages for simulating Turtlebot3 in Gazebo, mapping, and planner:

```
sudo apt-get install ros-melodic-turtlebot3*
```

```
sudo apt-get install ros-melodic-slam-gmapping
```

```
sudo apt-get install ros-melodic-dwa-local-planner
```

Go through Gazebo beginner tutorial 1-3:

http://gazebo.org/tutorials?cat=guided_b&tut=guided_b2

Go through TurtleBot3 Simulation tutorial from 11.2.1.1 “Simulate in Various World” to 11.2.1.5 “Virtual Navigation with TurtleBot3”:

<http://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#simulation>

Note: Choose waffle_pi as your TURTLEBOT3_MODEL since it is equipped with a camera.

You will need to (1) load a pre-defined simulated world and customize it, (2) execute RViz to visualize the robot sensing, and (3) build a map of your simulated world, as shown in Figure 2-4 respectively.

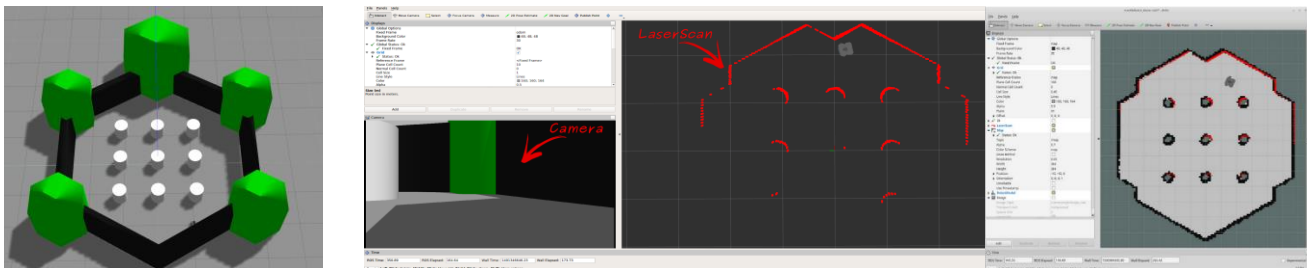


Figure 2. Simulation world

Figure 3. Simulated 3D scene

Figure 4. Mapping simulation

What to submit:

You are required to submit snapshots of (1) your customized simulation world, (2) visualization of the simulated robot sensing, and (3) simulated mapping result, as shown in Figures 2-4, respectively. Include all files in a tar or zip file named `T2_firstname_lastname.tar` (or zip) and submit to the Canvas portal named P1-T2.

Task 3: Write a ROS “Hello World”

You need to go through ROS Tutorial 1.1.11 (if you use C++) or 1.1.12 (if you use Python), as well as complete 1.1.13.

Write a ROS node that uses the turtlesim simulator to draw the inner-boundary of the Mines’ logo shown by the **RED CURVE** in Figure 5:

- Create a new package called `mines_lastname_firstname` (e.g., `mines_zhang_hao`). Your package should contain an executable with the same name as the package.
- Your program should assume that `roscore` and `turtlesim_node` have been started independently.
- The drawing should not take longer than two minutes to complete.
- You should NOT use the teleport services offered by turtlesim.

For CSCI 473:

- An open-loop control can be implemented (i.e., only tell the turtle driving directions without pose feedback).

For CSCI 573:

- A close-loop control is required (i.e., designing a subscriber to receive turtle’s pose data and use it to improve navigation accuracy).



Figure 5. Logo of Mines. The *red curve* is the trajectory the turtle needs to draw.

What to submit

You are required to submit (1) your code, (2) a README to detail how to compile and run your code (and other information), and (3) a snapshot of the result. Include all files in a tar or zip file named `T3_firstname_lastname.tar` (or zip) and submit to the Canvas portal named P1-T3.

Submission and Grading

Project 1 has two due dates, with a four-day late submission period for each of the two deadlines. The project deliverables must be submitted to the **Canvas**, but you are encouraged to use version control systems (GitHub or GitLab) to manage your code.

Grading: Your report will be graded as follows.

- 2/10: Task 1.
- 3/10: Task 2.
- 5/10: Task 3.