Lab 0: Run and Characterize the Lynx in Gazebo (Python)

MEAM 520, University of Pennsylvania

September 2, 2020

This exercise is due on **Wednesday, September 9, by midnight (11:59 p.m.)** Submit your answers to the questions at the end of the document as a pdf on Gradescope. Late submissions will be accepted until midnight on **Saturday, September 12**, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation such as illness. This assignment is worth 5 points.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. When you get stuck, post a question on Piazza or go to office hours!

1 Set Up Simulation Environment

The purpose of this mini-lab is to get you familiar with the Lynxmotion robot manipulator ('Lynx') in ROS+Gazebo simulation environment. The Lynx is a small robot arm with a parallel-jaw gripper. Specifically, we are modelling a Lynxmotion AL5D with the heavy-duty wrist rotate upgrade and SSC-32U Servo Controller.

1.1 Setup an Ubuntu virtual machine with ROS+Gazebo[1]

Because the simulator is built with ROS and Gazebo, it must run on Ubuntu. Since most students do not have access to a machine running Ubuntu, we will run the simulation on a Virtual Machine. **Note:** These steps require you to have a computer powerful enough to run the virtual machine (recommended allocation to VM is 10 GB hard-disk space, 8 GB of RAM, 2 CPU cores). If your computer is not able to run the virtual machine, you can also set up the VM image through Penn's Virtual PC Lab, which already has Oracle VM VirtualBox installed. See `https://cets.seas.upenn.edu/answers/virtuallab.html` for details on accessing Virtual PC Lab.

1. **Install VirtualBox:** Install the `Oracle VM VirtualBox 6.1.6` for your OS from `https://www.virtualbox.org/`. Versions are available for Windows, Mac and Linux.
2. **Download Virtual Image:** Download the `MEAM520F20.ova` virtual image from `https://drive.google.com/file/d/1cFxhHY6GqsNXkwFIDFyfzEo_GoUgjWYC/view?usp=sharing`. **Note:** This file is quite large (∼4GB) so it may take some time to download.
3. **Open VirtualBox:** Tools → Import Appliance → select `MEAM520F20.ova` → Import. In general, you can leave the defaults. By default the VM will have access to 2 CPU cores and 8 GB of RAM, you can reduce these if you are on a less powerful machine (or increase for a more powerful machine).

[1] If you already have a Ubuntu virtual machine or partition and wish to set up ROS+Gazebo yourself, you can also use the instructions in the handout 'Setup ROS+Gazebo.' I recommend you only do this if you are already an experienced Linux user.

4. **Enter the VM:** Select `MEAM520` and click the green start arrow in the toolbar. Now you should be greeted with a desktop in the VM. If you see a login screen, use the username and password `meam520` to enter the system.

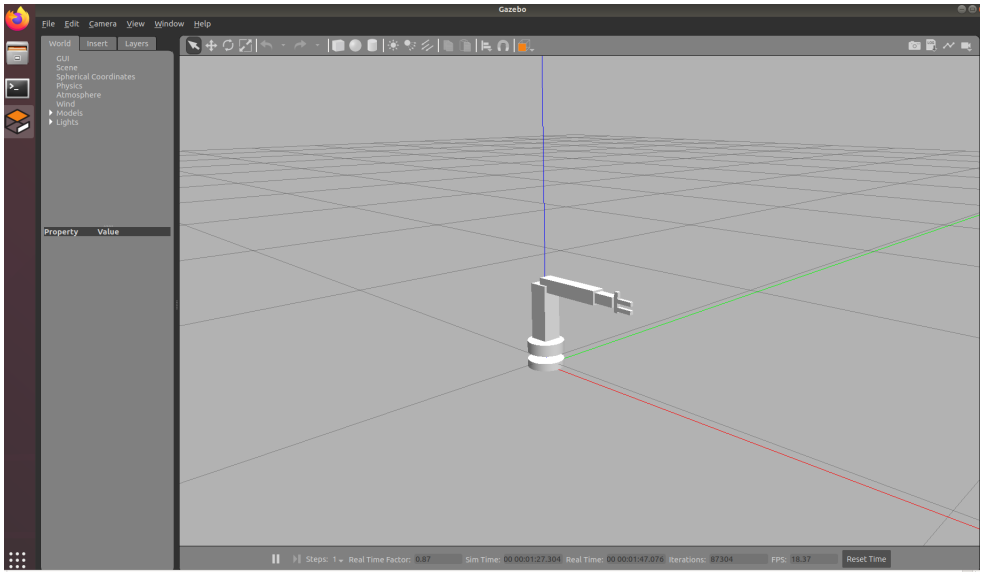
    - **Note:** On Windows or Linux you may get an error about VT-x or AMD-V not being available. If so, you need to enable virtualization for your computer in the BIOS. This is different for every manufacturer, so search `"<make><model> enable virtualization"` and follow the instructions. Your manufacturer should have a similar guide and the instructions will be similar, though with varying details. **DO NOT install Microsoft Hyper-V.**

1.2 Launch Simulator

1. **Start Gazebo:** Open a Terminal window on the virtual machine (press Ctrl-Alt-T, or right click, find "Open Terminal"). Run this command in the Terminal: `roslaunch al5d_gazebo lab0.launch`

   You should see text indicating that the ROS node has been initialized and eventually a Gazebo window with a robot arm as shown below in its zero pose.

   Gazebo is a robot simulation tool that will allow you to test your code and view the robot in action. You can zoom with the scroll wheel, pan by left clicking and dragging, and orbit by holding shift while clicking and dragging. Find out more on the Gazebo website at `http://gazebosim.org/`.



1.3 Control Arm from Python

While the simulator runs in ROS+Gazebo, we can interface with it through code you write in Python. For each lab, we will provide you with stub code containing functions that you must implement.

1. **Get the Code:**

   Choose a directory to store your Python code for the labs, which we will call your workspace. Your workspace must be on the VM so that your python scripts can communicate with the simulator. A good choice is `~/meam520_ws/src/python_code` but any will do. Log into Canvas on that machine. Under `Files > Labs > Code > Python`, download the `.zip` files `Core.zip` and `Lab0.zip` and extract them into your workspace. The resulting directory structure should look like:

> `Workspace`

> `Core`: this folder contains code that will be shared across all labs

> `Lab0`: this folder contains code pertaining to this particular lab

Each time you start a new lab you will download the `Lab#` folder for that lab and place it in this same directory so it can also access the `Core` files.

2. **Python Preparation:** Here are some tips to get you familiar with Python beforehand if you do not have much previous experience.

   - Introduction to Python: `https://developers.google.com/edu/python/introduction`
   - Quick resources for using Numpy: `https://www.dataquest.io/blog/numpy-cheat-sheet/`
   - Additional resources for Numpy: `https://docs.scipy.org/doc/numpy/user/quickstart.html`
   - Tutorials on matplotlib: `https://matplotlib.org/3.2.1/tutorials/index.html`

3. **Install Editor:** Since you will be writing your Python code in the VM, you will need to choose a text editor to run in the VM. Some good options include Atom or `vim`. You can easily find instructions online to install your choice.

4. **Run Demo:**

   Navigate your terminal to the `<workspace>/Lab0/` directory, and run the command

   `python demo.py`.

   You will see the arm in Gazebo move to another configuration. If you need to stop this code while it is running, press Ctrl-Z.

   Take a look at the code you just ran by opening `demo.py`. This code initializes an arm controller which can communicate with the simulator, accepting commands from you for the desired configuration of the robot. You can examine what's going on under the hood by taking a look at `../Core/arm_controller.py`. You will notice the `get_state()` function outputs 2 numpy arrays, the first array shows the robot joint values and the second is the joint velocities.

5. **Examine Different Configurations:**

   We will now move the robot to around to different configurations.

   Edit the `demo.py` file, modifying the input q to:

   `q = [0.1,0.1,0.1,0.1,0.1,0.1]`

   Save the file and run `python demo.py` again in the terminal.

   You will see the arm in Gazebo move to the new configuration that you setup.

   There are 6 inputs, which specify positions for each of the 6 motors/joints (#0-5) on the robot. Units for each of the joints 0-4 are in radians, and the last input is the opening of the gripper in millimeters.

   This command tells all five joints to move to +0.1 radians and the gripper to move to +0.1 millimeters.

   Verify that the robot moved all of its joints by the correct amount in the correct direction. If any of the joints didn't move, that input might have syntax error. You are welcome to try more poses until you have a feeling for how the robot moves.

6. **Turn Off Simulation:**

   Close all running terminals so that Gazebo will shut down automatically. You can turn off the VM as well.

7. **Congratulations on your first simulation run!**

2 Questions

Answer the following questions and submit your answers to Gradescope.

1. Ignoring the gripper, how many degrees of freedom does the simulation robot have?
2. What is the kinematic arrangement (in terms of Rs and Ps) with and without the gripper?
3. Draw the 3D symbolic representation of the robot in the zero configuration.
4. Sketch the reachable workspace of the simulated robot.

The following questions will need to be answered with the simulation.

5. Modify the input variable to:

   ```
   >> q = [0, -pi/4, pi/3, -pi/2, 0, 0]
   ```

   Is the end effector pointing up or down?

6. Modify q to:

   ```
   >> q = [0, 0, 0, 10, 0, 0]
   ```

   The arm controller outputs a message telling you that the target angle for joint 3 is above its upper limit and what that limit is. **Note:** joint numbers are zero indexed.

   Input some additional commands to find out what the upper and lower limits of each of the joints are. Write down the joint limits for each of the joints.

7. What command will move the robot to point straight up but the gripper oriented horizontally facing negative x direction? **Note:** the axes in gazebo are color coded as follows: red - x, green - y, blue - z.
8. Take a look at a real lynx robot in the video we provided on Canvas. Compare the results of the robot simulator and the actual robot motion by inputting the values of `q` specified in the video into the simulation. Do you see any differences between the simulation results and the real world robot? What aspects of the robot in the real world does the simulation capture well? What aspects are not captured by the simulation?

The following questions investigate the framework running underneath the simulation, ROS. These questions will not be graded, but will help provide insight into ROS for those interested.

9. While the arm demo is running, in another terminal window, run `rqt`. Go to Plugins → Introspection → Node Graph. You will now see a graph of what ROS calls "nodes" and "topics". A node amounts to a single piece of software. A topic is a communication line.
10. In a terminal, run `rostopic info /al5d_arm_position_controller1/command`.
    Run `rostopic echo /al5d_arm_position_controller1/command` and wait for around 10 seconds. `rostopic echo /al5d_arm_position_controller1/state` What do these commands do?
11. Take a look at `arm_controller.py` in the Core folder. This file provides the interface between your code and ROS. What is a publisher and subscriber?

Submission Instructions

Submit the assignment. You should submit a **pdf** on Gradescope containing your answers to the Questions section of this handout. If you worked on this assignment as a pair, you may add your partner to your submission on Gradescope.