Lab 0: Run and Characterize the Lynx in Gazebo

MEAM 520, University of Pennsylvania

May 27, 2020

This exercise is due on **Monday, June 1, 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 **Thursday, June 4**, 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 10 points.

You may do the exercise in pairs and add your partner on Gradescope to ensure both students receive credit. 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!

Instructions

The purpose of this mini-lab is to get you familiar with the Lynxmotion robot manipulator ('Lynx') in python + 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. **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`

2. **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.

3. **Download Virtual Image:** Download the `MEAM520_v3.ova` virtual image from `https://drive.google.com/file/d/1_5ZfBvwB32NgHL8rcuaqyzvdER-OaGIR/view?usp=sharing`. **Note:** This file is quite large (∼4GB) so it may take some time to download.

4. **Open VirtualBox:** Tools → Import Appliance → select `MEAM520_v3.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).

5. **Select MEAM520:** On the left side bar, select `MEAM520` and click the green start arrow in the toolbar.

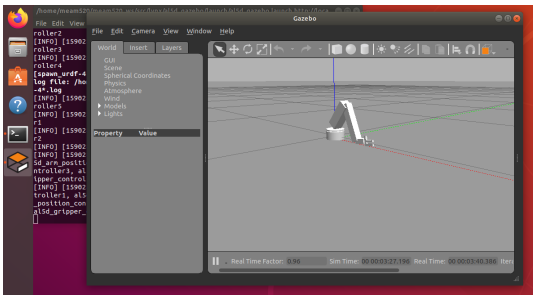
   - **Note:** On Windows or Linux you may get an error about VT-x or AMD-V not being available. If so, you need to enable virtualization for your computer in the BIOS. This is different for every manufacturer, so search `"<make><model> enable virtualization"` and follow the instructions. Your manufacturer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

6. **Enter the VM:** 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.

7. **Start Gazebo:** Open a Terminal window (press Ctrl-Alt-T, or right click, find "Open Terminal"). Run this command in the Terminal: `meam520_arm`

   You should see a bunch of text, and eventually a Gazebo window with a robot arm as shown below.

   We will use gazebo primarily as a viewing window. You can zoom with the scroll wheel, pan by left clicking and dragging, and orbit by holding shift while clicking and dragging. If interested, you can go to the gazebo website at `http://gazebosim.org/`.



8. **Open Spyder:** Click on the "Show Applications" icon (the 3x3 grid of dots in the lower left corner) and search for Spyder to find the Spyder3 application.

9. **Run Demo Code:** There is a file called arm_demo.py under the "code" folder. Open this file in Spyder (Ctrl+O) and run arm_demo.py by hitting the "Run File" button (F5 or the green arrow at the top). With the code running, you will now see the arm in Gazebo moving back and forth!

   The demo code basically changes the robot state in a loop, and it calls the arm_controller.py which communicates with the Gazebo environment.

10. **Stop the Current Test:** Click the little red square icon in the upper left of the IPython window (alternatively, you can exit the open Console and a new one will automatically be generated). Note: it is important that you stop any code actively communicating with the simulation before you start to run a new test.

11. **Examine the zero pose:** Download the file "lab0_v3.py" from Canvas and move this file into the "Code" folder (if you don't move this file to the "Code" folder it will not be able to call the required files). Now, open this file in Spyder. Look at the robot's zero pose by modifying the input q to

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

    Now hit "Run File (F5)" in Spyder and you should see the robot's zero pose. **Note:** Spyder will automatically save the file when you hit "Run File (F5)".

    Take a look at the code and the corresponding output. 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.

    Change the input to:

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

    and run the test with the new values. You will now see the arm in Gazebo moving to the 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.

12. **Turn off simulation:** When you are done, close Spyder and all open terminals and Gazebo will shut down automatically. Then, you can turn off the VM.
13. **Congratulations on your first simulation run!**

Questions

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 robot simulation.

5. Modify q 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]`

   Terminal 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 code 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 in this handout. If you worked on this assignment as a pair, you may add your partner to your submission on gradescope.