Lab 1: Kinematic Characterization of the Lynx

MEAM 520, University of Pennsylvania

September 9, 2020

This lab consists of two portions, with a pre-lab due on **Wednesday, September 16, by midnight (11:59 p.m.)** and a lab (code+report) due on **Wednesday, September 23, by midnight (11:59 p.m.)**. Late submissions will be accepted until midnight on Saturday following the deadline, 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. This assignment is worth 50 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. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

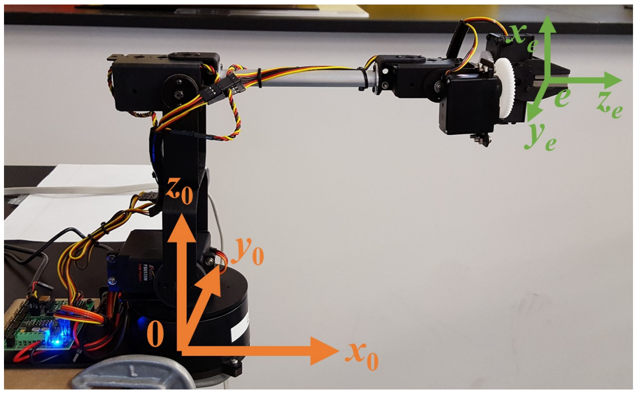
Work closely with your partner throughout the lab, following these guidelines, which were adapted from "All I really needed to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Resources.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, *even if one partner is much more experienced than the other.* You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

1 Pre-lab Tasks (due September 16, 5 pts)

Directions: For the pre-lab component of this lab, you turn in your own **individual** work. Show your work to receive full credit. These calculations should be typed or written legibly. Submit a **pdf** on Gradescope containing your work.

The Lynx is a small robot arm with a parallel-jaw gripper. Specifically, we are using a Lynxmotion AL5D with the heavy-duty wrist rotate upgrade and the SSC-32U Servo Controller, controlled via serial commands sent over a USB-to-serial cable. The robot is mounted to a base plate that is clamped to the table so that it has a large reachable workspace both above and below the table surface. Now that we have a qualitative understanding of the arm from Lab 0, we can do a more formal analysis.



1. You should already have drawn the symbolic representation for the Lynx in Lab 0. Label your drawing with axes of rotation, joint angles, and relevant dimensions. The spec sheet at `http://www.lynxmotion.com/images/jpg/al5dbd.jpg` contains some of this data that you can refer to. However, because you don't have access to the physical robot, we've given you the distance between each pair of joints to avoid any confusion.

$L_1 = 76.2$ mm	*distance between joint 1 and joint 2*
$L_2 = 146.05$ mm	*distance between joint 2 and joint 3*
$L_3 = 187.325$ mm	*distance between joint 3 and joint 4*
$L_4 = 34$ mm	*distance between joint 4 and joint 5*
$L_5 = 34$ mm	*distance between joint 5 and e, the point in the center of the gripper jaws*

2. Let the orange frame 0 be the base frame (at joint 1) and the green coordinate frame labeled e be the end effector frame (at the center of the jaws). Using your geometric intuition, what is the homogeneous transformation matrix for transforming from e to 0 when the robot is in the zero configuration?

3. What is the homogeneous transformation matrix T_e^0 when the joints are at angles $\theta_1 = \pi/4$, $\theta_2 = \theta_3 = \theta_4 = \theta_5 = 0$?

4. What is the homogeneous transformation matrix T_e^0 when the joints are at angles $\theta_1 = -\pi/2$, $\theta_2 = 0$, $\theta_3 = \pi/4$, $\theta_4 = 0$, $\theta_5 = \pi/2$?

2 Lab (due September 23, 45 pts)

The remainder of the lab should be done with a partner. You may work with anyone you choose, but you must work with them for all parts of this assignment. You will both turn in the same report and code (see Submission Instructions), for which you are jointly responsible and you will both receive the same grade.

2.1 Methods

Compute the forward kinematics for the robot using the following steps:

1. Make any corrections needed to your symbolic representation. Remember that all coordinate frames for each of the links, joint angles, and relevant dimensions should be defined.
2. Write down homogeneous transformation matrices for transforming between the coordinate frames for each pair of consecutive links. (This will be cleaner if you use symbolic notation as we did in class.)
3. Write an expression for the position of the center of the gripper in the base frame in terms of these transformation matrices. *(To avoid any ambiguity, this point sits halfway between each of the jaws, lying on the axis of the last revolute joint in the arm, located halfway along the length of the jaws.)*

2.2 Simulation Environment Updates

We've made a few tweaks to the simulation environment, and you must take the following steps to have an up to date simulator. **These steps are mandatory for all students!**

1. **Update the Gazebo simulator:** On the Virtual Machine, open a terminal and run the command

   ```
   cd ~/meam520_ws/src/meam520_sim && git pull
   ```

   This will update the code on your machine to match the current version.
2. **Update the Core:** From Canvas, redownload the `Core.zip` file for your respective language and extract it in the same location as for Lab0.
3. **(Gazebo Pro Tip:)** Right-click on the robot and select 'Move To' to automatically zoom your view into the robot. Hold shift, click the robot's base, and drag to reorient the view.

2.3 Coding

Write a function that takes in a vector of 6 inputs (the five joint angles and the grip distance) and computes the locations of all the joints and the gripper in 3D space.

1. Download the `Lab1.zip` starter code for this assignment from Canvas in your chosen programming language. This zip contains three files: `TestFK_Sim`, `calculateFK`, and `computeWorkspace`. Extract these files into a folder called `Lab1` in the same directory as your `Core` and `Lab0` folders from last time. The files `calculateFK` and `computeWorkspace` are skeleton files. You will fill them in for this assignment.
2. When you call `TestFK_Sim`, it uses the file `calculateFK` to predict the gripper's location, then sends the same command to the Gazebo simulation and pulls the joint locations. The function `calculateFK` has the following structure:
   - Inputs: a vector of 6 inputs (the five joint angles and the grip distance).
   - Outputs: 1) a 6×3 matrix that contains the locations of each of the joints and the center of the gripper, and 2) a 4×4 transformation matrix representing the **end-effector frame** expressed in the base frame.

   Note: if you move to a faraway configuration, the script may terminate before fully reaching the location and will report the simulated position when terminating. Just run the script again to reach the final location.
3. Fill in the `calculateFK` function to output the actual joint positions and transformation matrix for the robot.

Side note: It is bad practice to use `syms` (symbolic math) for building modeling and simulation code since it will significantly slow down your calculations. Instead, use numerical computation. To enforce this idea, in this class, any use of symbolic variables in any submitted code will incur a grade penalty.

2.4 Evaluation

1. Plug in zero joint angles into your matrices above. What homogeneous transformation T_e^0 do you obtain for the zero pose? Does it match your expected transformation matrix from the pre-lab?

2. Plug in the joint angles (1) $\theta_1 = \pi/4$, $\theta_2 = \theta_3 = \theta_4 = \theta_5 = 0$ and (2) $\theta_1 = -\pi/2$, $\theta_2 = 0$, $\theta_3 = \pi/4$, $\theta_4 = 0$, $\theta_5 = \pi/2$. Do the results match your expected transformation matrices from the pre-lab?

3. Choose at least 3 other inputs and experimentally compare the simulation results with the resulting configuration of the physical robot. Choose configurations that convey information about the robot and explain why you chose them. In other words, think about configurations that tell you something about the simulation that your calculation could not tell you.

4. Use your `calculateFK` to predict the reachable workspace of the robot. Create a plot of the positions that the gripper is able to reach.

   - Use the joint limits you identified in Lab 0 to detect the workspace limits. Ignore any other geometric considerations such as self-collision.
   - Describe your strategy for how to compute the set of achievable 3D positions.
   - Implement this strategy as code in the file `computeWorkspace`.

5. Compare the results of your evaluation against the ROS simulator. Use the same procedure you used in Lab 0 but with the Lab 1 launch file, i.e.,:

   ```
   $ roslaunch al5d_gazebo lab1.launch
   ```

   Then run `TestFK_Sim` (using the command for your language) to compare your solution to the simulator output.

2.5 Analysis

1. Discuss the results of your evaluation. Were the results what you expected? Account for any differences between your function outputs and those from the simulation.

2. Account for any differences between the workspaces of the predicted and simulated robot. Were any points predicted as reachable that were not in simulation? Were any predicted as unreachable that were reachable in simulation? Explain what additional information you would need to incorporate into the simulation environment to predict these differences.

3 Submission Instructions

Submit the assignment. One person from each pair should submit code and a pdf copy of the report to the Gradescope assignment for Lab 1. After selecting the files and uploading them, the website will take you to the next page, where in the top right corner you should add your group members. If you do not add your group members they will not get credit for the assignment.

3.1 Report

The format of the report is up to you, but you should make sure that it is clear, organized, and readable. The report should include:

1. Your answers to the conceptual questions in the Methods section, typed or legibly hand-written.
2. A short 1-pg description of how the concepts are incorporated into your code. Include pointers to important line numbers of subfunctions in your code. This will help the graders understand and provide feedback on your work. (This description can be bulleted. No need to use full sentences.)
3. Your experimental results, including a description of your experimental setup (i.e., what were your inputs) and collected data.
4. Your analysis comparing expectations against reality and extrapolations to general conclusions you would make from this lab.

3.2 Code Submission

Your code should be cleaned up so that it is easy to follow. Remove any commented-out commands that you are not using, and add comments to explain the tricky steps. Clearly indicate which parts of the code correspond to which parts of the lab. Your code submission should include **only**:

1. Your `calculateFK`
2. Your `computeWorkspace`
3. Any additional functions needed to run your code not included in the original code.

Each file should be attached separately to Gradescope. Do **not** zip them into a single file attachment.