Intro to Robotics Report 1

Daniel Farkash, Cedric Hollande

February 22nd 2024

1 Methods

1.1 Strategy employed:

1. Placing the base axis as described in the instructions at $(0, 0, 0)$.
2. I placed dots to represent the locations of the joints and included the distances between them (except for the one distance along the third axis for the second to last joint).
3. I placed z axes corresponding to the directions of rotation on the joints for each joint location. (I later figured out that the second joint rotates in the opposite direction, so I multiplied the rotation variable by negative 1).
4. I found the intersections of each joint z axis and the previous joint's z axis and placed the x axis perpendicular to the plane of the z axes.
5. I placed the y axes at the joint axes intersections according to the right hand rule.
6. I look at my diagram and use the D-H convention to note the variables of the 4 necessary rotations and translations that would be required to move between each of the joint frames.
7. I create homogeneous transformation matrices between each of the frames and continue the necessary calculations in my code (described below).

1.2 Diagram, measurements and parameters:

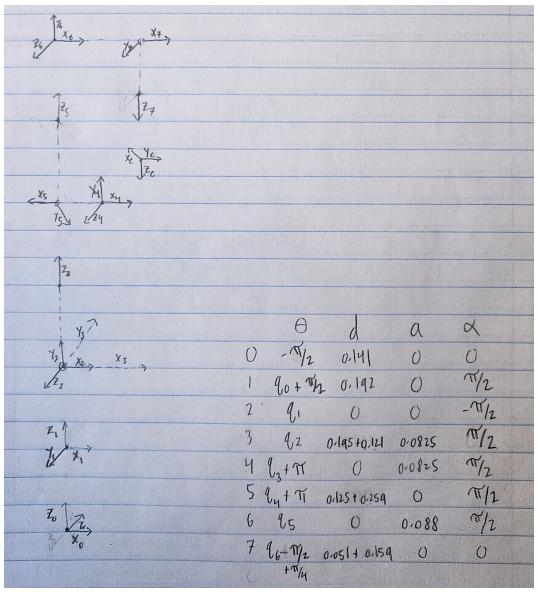

Figure 1: *Diagram showing joint positions and axes.*

1.3 Code structure:

I created a helper function named `make_transformation` that takes in σ, α, d, and a variables and creates a homogeneous transformation matrix according to the D-H method formulation. Then, in the forward function, I created arrays for each of the D-H input variables and filled them with the values I calculated during my visual exploration. I then iterated through all of the joints and right-multiplied the transformation matrices between each of them to get the final transformation matrix from base to end effector. During each step in the iteration, I took the current base to joint transformation matrix and additionally right-multiplied by a correction matrix (distance along joint z axis) to account for when the origin of the joint is not the origin of the joint axes and used the translation vectors of the resulting transformation matrices to fill the joint positions array.

2 Evaluation

2.1 Homogeneous Transformation T_e^0:

When I plug zero angles into our FK code, the terminal outputs the transformation:

$$\begin{bmatrix} 0.70710678 & 0.70710678 & 0 & 0.088 \\ 0.70710678 & -0.70710678 & 0 & 0 \\ 0 & 0 & -1 & 0.823 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2: End effector transformation matrix with **zero** angles

This matches my expectation because once the $\frac{\pi}{4}$ rotation at the last joint is accounted for, we can see by analyzing the rotation matrix that the x axis of the end effector corresponds to the x axis of the base, while the y axis corresponds to the negative y base axis and the z axis corresponds to the negative z base axis, which is the set of relationships between the two frames we can see by observing the diagram. We also see that the distances in the translation vector correspond to the correct distances along the x and z axes as can be calculated in the diagram.

2.2 Configuration Experiments:

We selected three configurations to demonstrate the robot's range of motion and operational flexibility. These configurations were strategically chosen to explore the robot's performance in different scenarios, including its maximum reach and ability to navigate complex maneuvers.

- Configuration 1 (*Default Pose*): Tests the robot's neutral position, serving as a baseline for comparison.
- Configuration 2 (*Extended Reach*): Evaluates the robot's dexterity and precision in executing extended movements to grab a block.
- Configuration 3 (*Complex Extended Reach*): Examines the robot's capability at close to maximum extension, crucial for tasks requiring long reach.

Below, we compare the end effector final pose based on the joint angles in radians, for the simulation and real life results for each configuration:

#	Joint Angles (rad)	Simulation (cm)	Actual (cm)
1	$(0, 0, 0, -\frac{\pi}{2}, 0, \frac{\pi}{2}, \frac{\pi}{4})$	(55.45, 0, 52.15)	(56, 0, 52)
2	$(\frac{\pi}{2}, \frac{\pi}{4}, 0, -\frac{\pi}{4}, 0, \frac{\pi}{2}, \frac{\pi}{4})$	(0, 75.38, 37.06)	(0, 75, 36)
3	$(0, -\frac{\pi}{2}, \frac{\pi}{2}, -\frac{\pi}{4}, \frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{4})$	(-70.81, 35.79, 12.3)	(-69, 36, 12)

Table 1: *Joint angles, updated simulated end effector positions, and measured positions for selected configurations.*

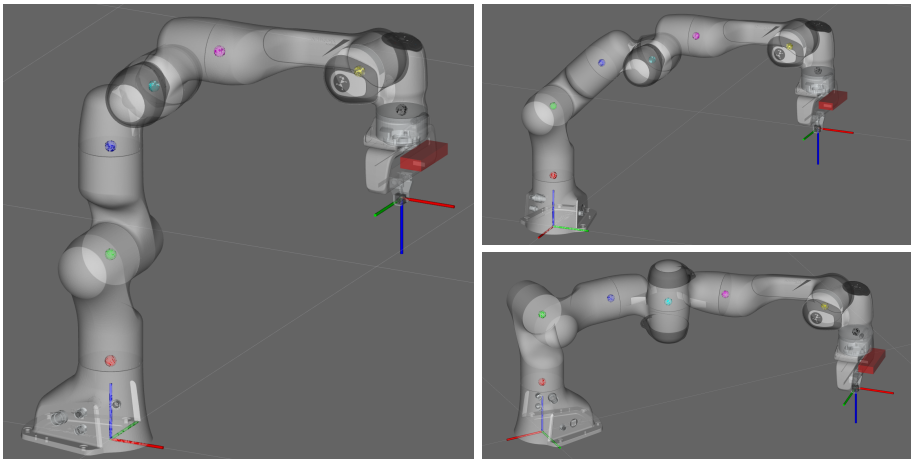

Figure 3: *Robot joints in our 3 configurations.*

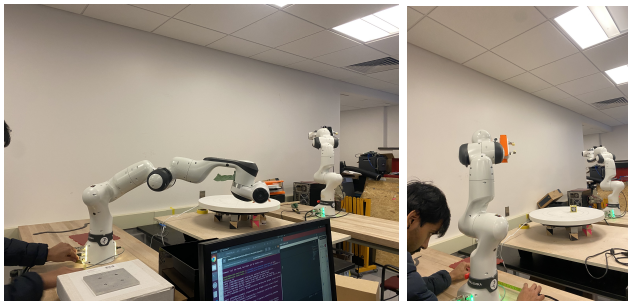

Figure 4: *Real Life Testing & Measuring.*

The detailed comparison above not only validates the accuracy of our simulation model but also illustrates the robot's capability to closely replicate theoretical motions within a practical setting. The minor deviations observed between simulated and measured positions are attributed to the inherent measurement uncertainties and physical constraints encountered during real-world operation.

2.3 Workspace:

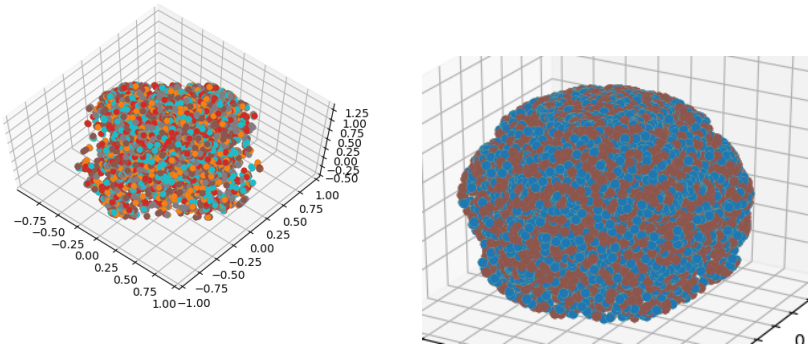

Figure 5: *Visualizations of the robot's workspace.*

(I show two point representations of the workspace, one is more dense than the other, but less interpretable.) My strategy for creating the workspace is to create a recursive helper function that for each joint angle iterates over a certain number of steps that span the minimum to maximum angles of that joint and combines it with a similar angle choice for every other joint into a call of the forward kinematics function from which the translation to the end effector is extracted and then plotted. Some assumptions made were: the joints can move to each of the given angles with relatively high precision (in reality it is not guaranteed that the joints can accurately move to all possible angles within their range), there will be no collisions with the robot (either with itself or with other objects/surfaces), the step increments made in the workspace plotting are dense enough to properly represent the workable area. Other simple configurations also worked. As you can see below, the joint positions are correctly computed and follow the robot's motion.

3 Analysis

3.1 Results and Expectations:

The core of our evaluation centered around comparing the anticipated outcomes from our theoretical models and simulations with the actual data collected from hardware tests. This comparison is crucial for validating the accuracy and reliability of our robotic control solutions.

Expectations vs. Actual Outcomes: Our initial expectations were based on the assumption that the theoretical models and simulations would closely mirror the real-world performance of the robot. In the majority of cases, this held true, with the simulations providing a reliable forecast of the physical robot's capabilities. Specifically:

- *Zero Joint Angles:* The transformation matrix obtained from inputting zero joint angles aligned perfectly with our expectations, confirming the accuracy of our forward kinematics model.
- *Selected Configurations:* For the chosen configurations, we observed a high degree of similarity between the simulated end-effector positions and those measured in the physical tests. The deviations were minimal and fell within an acceptable range of error (within 3 cm):
    1. *Default Pose* demonstrated precision in the robot's neutral position alignment.
    2. *Extended Reach* and *Complex Extended Reach* configurations validated the robot's operational flexibility and reach capabilities.

Differences Between Function Outputs and Simulation Results: The minor discrepancies observed can be attributed to several factors not accounted for in the simulations:

1. *Measurement Uncertainties:* Manual measurements introduced a level of uncertainty, especially in complex configurations where precision is paramount.
2. *Mechanical Limitations:* The physical robot may experience limitations not present in the simulation, including joint friction and response delays.
3. *Environmental Factors:* Real-world tests are subject to environmental variables that simulations can overlook, such as table tilt or unexpected obstacles.

In summary, while our results closely aligned with expectations, the noted differences emphasize the importance of accounting for practical limitations and environmental factors in simulations. These insights will guide future improvements to our models, ensuring a more accurate prediction of the robot's capabilities and performance in real-world applications.

3.2 Theoretical vs. Simulated Discrepancies:

Upon comparing the theoretical and simulated workspaces of our robot, notable differences emerged. While our theoretical model provided an optimistic projection of the robot's capabilities, the simulation and subsequent physical tests revealed certain limitations. Specifically, configurations predicted as reachable in theory were sometimes unattainable in simulation due to the risk of self-collisions not considered in the initial model.

These discrepancies underscore the necessity of incorporating collision detection and avoidance mechanisms into our forward kinematics calculations. By doing so, we can refine our theoretical models to more accurately reflect the practical constraints of the robot's physical design and operational environment.

Furthermore, to bridge the gap between theoretical predictions and practical outcomes, we recommend the following enhancements to our modeling approach:

- Integration of dynamic constraints and motor torque limits to ensure that the predicted configurations are not only geometrically feasible but also dynamically viable.
- Consideration of external factors such as environmental obstacles and interaction with objects to tailor the workspace analysis for specific application scenarios.
- Implementation of a more granular simulation of the robot's mechanical properties, including joint friction and backlash, to improve the fidelity of the simulated workspace.

3.3 Simulated vs Physical:

There are differences between the simulated and physical robot. One such difference is that the real robot motors do not have an infinite theoretical precision and can likely only move between a finite set of positions, meaning that not all configurations that are possible in simulation could be perfectly reflected in the real robot. Another difference is that there will always be an error in the real world for each joint angle and each length of the arm that is not accounted for in the simulation (due to physical construction, motor capabilities, etc.). There will also be additional interference externally, such as the tilt of the table, contact with other items, and possible drift in the motors zero states through repeated use. Something that may cause a solution that works in simulation to fail on hardware could be collision with nearby items that do not exist in simulation or not being able to move a motor to its full range of motion due to imperfections in the physical implementation. By testing our solution on hardware we could find out the real limits of the joints that can be reached as well as the real size of the arm components so that we can calculate more accurate bounds for the reachable workspace.