ENPM662 Project 2

Brendan Neal and Adam Lobo

December 5th, 2023

Contents

1 Introduction

This project is the culmination of all concepts and techniques taught in ENPM662. Students are to design and program a 6+ degree of freedom robot to accomplish a task.

2 Application

Our robot is an automatic table busser. It goes around to different tables in a restaurant and clears the dishes. It first navigates to the desired table. It then individually removes dishes from a table by picking them up and dropping them in a bus basket. It can repeat this process on as needed basis. In theory, this could replace bus boys entirely. In practice, there have been a few real life creations of bus robots. However, these are not widely used or available since the concept seems relatively new. Hopefully someday, the bus robot will revolutionize the restaurant industry. An example design for a robotic table busser comes from: [1].

3 Robot Type

We combined a mobile robot and a manipulator robot into one. Our mobile robot is a basic four-wheel drive cart that can navigate forward and backward and steel left and right. Our manipulator robot is attached to our mobile robot and has five link arms that allow it to rotate in different directions and configurations to reach nearly any spot within a close vicinity. The gripper serves as an end effector, allowing the robot to pick up, transport, and release/put down small-sized objects. This combined robot allows it to navigate, both on a large scale by driving to any location on a map, and on a small scale with the rotation and configuration of the manipulator.

4 DOFs and Dimensions

Our robot consists of seven degrees of freedom, three come from the cart, and four come from the manipulator. The degrees of freedom in the cart are its ability to drive forward linearly and its ability to turn/drive around an axis (hence, X, Y, and Orientation). The degrees of freedom in the manipulator are in each joint. The first joint at the base of the manipulator rotates about the axis parallel to and through the center of links 0 and 1. This controls the manipulator's heading, and allows it to turn 360 degrees, facing any horizontal direction. The next two joints in the middle of the manipulator rotate about an axis perpendicular to that of the arm. The axes of these two joints are always parallel. These two joints control the pitch and height of the end effector (gripper), as well the reach distance (in the direction of the gripper's heading). The final joint closest to the gripper rotates about the axis parallel to and

through the center of links 3 and 4, similar to the first joint. This joint allows for roll/tilt control of the end effector.

The important dimensions relevant to our task are the lengths of each link (L_0 - L_4) of the manipulator, since the orientation of each link influences the position and orientation of our end effector. Each link has a different length, depending of the link's function and components. The remaining dimension values are more just specific to the design of our robot, and do not directly affect the outcome of the task. The dimension values are shown in the two tables below, with the critical link lengths in bold at the top of the manipulator dimensions table.

Variable	Description	Value (in)
L_0	**Link 0 length (base to joint)**	**3**
L_1	**Link 1 length (joint to joint)**	**23**
L_2	**Link 2 length (joint to joint)**	**21**
L_3	**Link 3 length (joint to joint)**	**18**
L_4	**Link 4 length (joint to gripper)**	**5**
w_b	Link 0 base width	6
h_b	Link 0 base height	3
f_b	Link 0 base fillet	0.5
h_l	Link 1-3 main arm height/length	20
w_l	Link 0-4 main arm width	4
f_l	Link 0-4 main arm corner fillet	0.5
h_a	Link 1-2 joint arm height/length	10
w_a	Link 1-2 joint arm width	2
f_{ac}	Link 1-2 joint arm corner fillet	0.25
f_{ab}	Link 1-2 joint arm base fillet	1
gr_l	Gripper length (inner edge)	3
gr_{l2}	Gripper length (outer edge)	2
gr_w	Gripper width	1
gr_t	Gripper thickness	1

Figure 1: Manipulator Dimensions

Variable	Description	Value (in)
wh_r	Wheel radius	6
wh_w	Wheel width	5
ax_r	Axle radius	2
ax_{l1}	Axle (car side) length	11.5
ax_{l2}	Axle (wheel side) length	3.1
j_w	Joint width	4
c_l	Chassis length	42
c_w	Chassis width	24
c_h	Chassis height	24
b_l	Dish basket length	24
b_w	Dish basket width	20
b_d	Dish basket depth	12
b_x	Distance from back of cart chassis to back of dish basket	4
b_y	Distance from side of cart chassis to side of dish basket	2
m_x	Distance from front of cart chassis to center of manipulator	6
m_y	Distance from side of cart chassis to center of manipulator	12

Figure 2: Cart Dimensions

5 CAD Models

Our CAD models were designed from scratch using SolidWorks, specifically for this application. The final assembly of the full robot consists of two sub-assemblies: cart and manipulator.

The cart is very similar to the one designed for our first project, with a few modifications. We made the wheels larger, both in diameter and width, to be able to support a larger load. In addition, we made the wheel axles and joints larger. We also added the basket (an extruded cut from the top of the cart) to hold bussed dishes. This cart does not have the design or aesthetics of a car that our car in project 1 had, since this one serves a different purpose. The steering of this car remains the same as the car in project 1. The front wheels have joints that rotate to allow them to turn. The rear wheels have fixed joints and only rotate linearly (to allow for forward/backward motion).

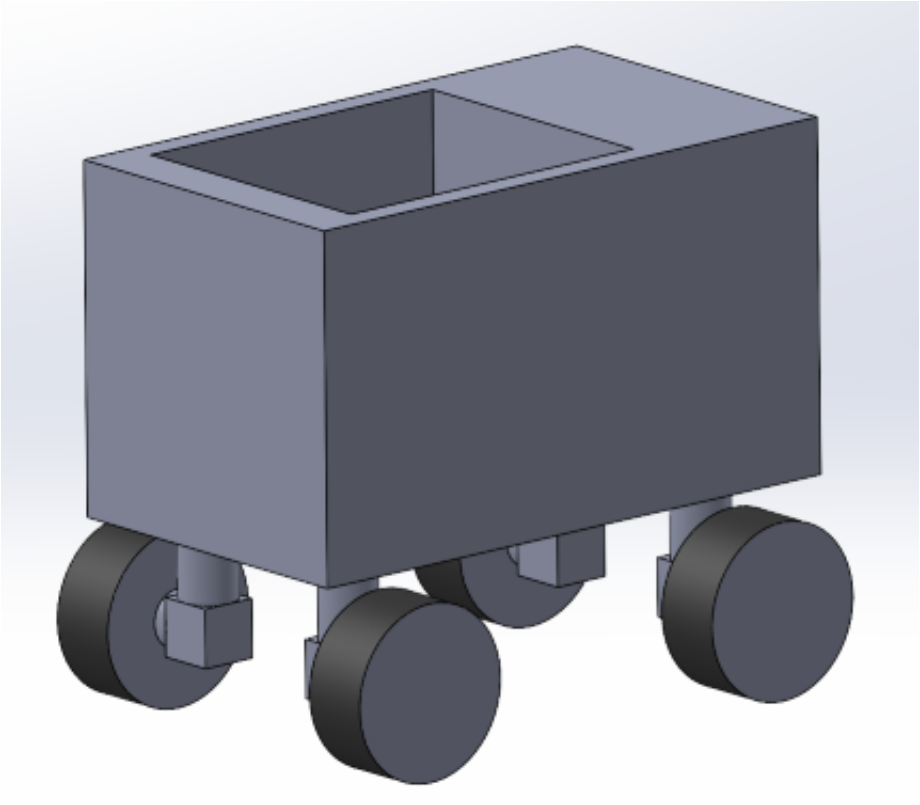

Figure 3: Cart Isometric View

The manipulator consists of five links and five joints. The first link is the base link, L_0, which has a wider lower section filleted to the upper section, which has the width of the rest of the manipulator links. The first joint is a double cylindrical piece on link 0 that fits into an identically shaped hole in link 1, which allows link 1 to rotate with respect to link 0. The main body of links 1-4 are long square extrusions with filleted edges. Link 1 and link 2 have two joint arms at their tops with a cylindrical rod connecting the arms that serves as the joint to the next link. Links 2 and 3 have identical holes at their bottoms that allow them to connect to the previous joint and rotate. Link 3 has a similar joint to link 0 that allows link 4 to rotate similarly. Two grippers are attached to the end of link 4 that serve as the end effector. They are connected through a slot in link 4 that serves as the prismatic joint, allowing the grippers to open and close.

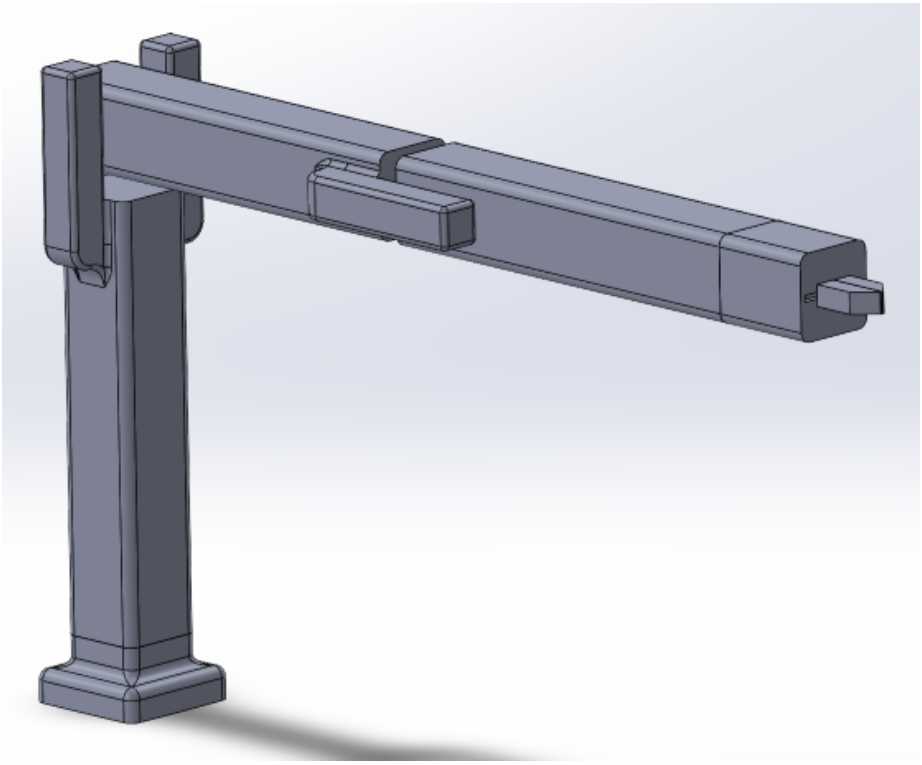

Figure 4: Manipulator Isometric View

The manipulator is positioned on the top of the cart, between the front of the cart and the basket. More specifically, it is centered laterally and positioned 6 inches inward from the front of the cart.

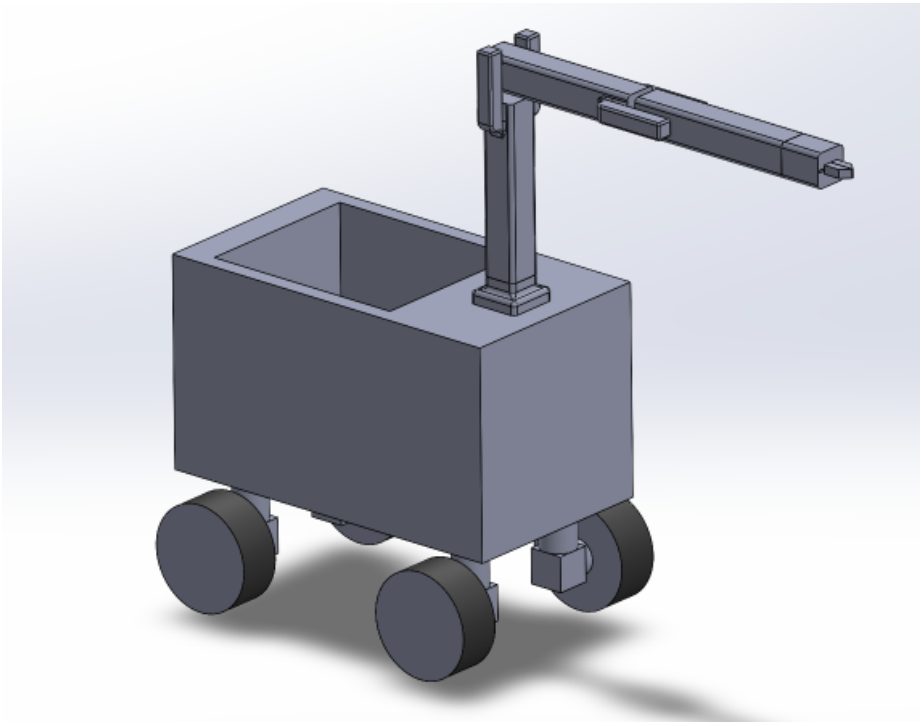

Figure 5: Full Robot Isometric View

6 D-H Parameters

Figure 3 shows our manipulator with each joint in zero state with D-H coordinate frames. Figure 4 is the D-H table with the value for each D-H parameter. There is a total of five coordinate frames (frame 0 to frame 4), with one frame for each joint, plus the final end effector frame. We defined the frames using Spong's convention [2].

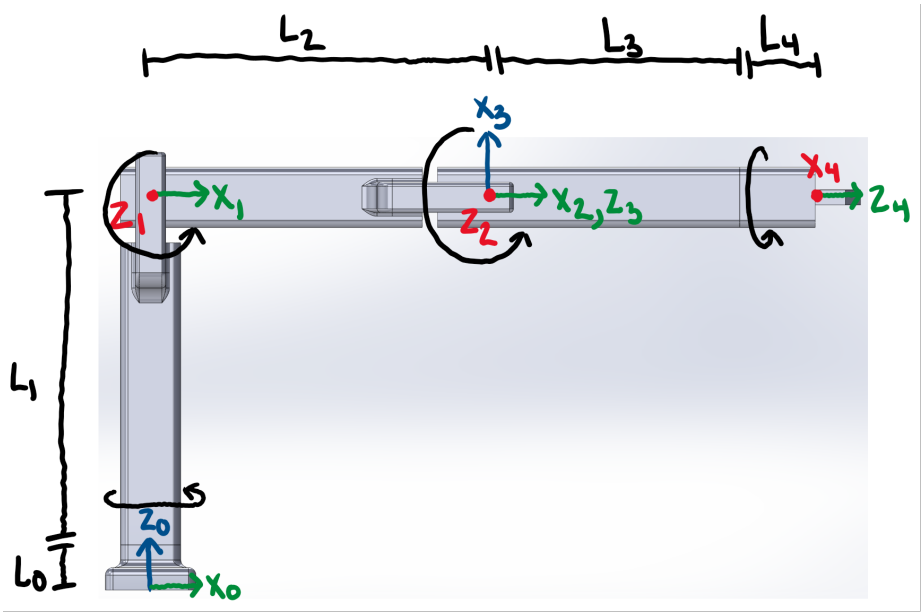

Figure 6: DH Coordinate Frames

	a (in)	α (°)	d (in)	Θ (°)
Frame 0 -> 1	0	90	L_0+L_1	Θ_1^*
Frame 1 -> 2	L_2	0	0	Θ_2^*
Frame 2 -> 3	0	90	0	$90+\Theta_3^*$
Frame 3 -> 4	0	0	L_3+L_4	$90+\Theta_4^*$

Figure 7: DH Table Values

7 Forward Kinematics

For the body of the car, the forward kinematic equations are as follows:

$$\dot{x} = u_s * cos(\theta)$$

$$\dot{y} = u_s * sin(\theta)$$

$$\dot{\theta} = \frac{u_s}{L} * tan(\alpha)$$

Where α is the steer angle.

Using the D-H Table defined in the earlier section, we computed the Forward Positional Kinematics for our robotic manipulator. Each transformation matrix is calculated per row. The multiplication happens as follows:

$$T = R_z(\theta) * T_z(d) * T_x(a) * R_x(\alpha)$$

To calculate the final transformation matrix, we performed:

$$H_4^0 = T_1^0 T_2^1 T_3^2 T_4^3$$

Depicted below in Figure 8 is a screenshot of the final transformation matrix:

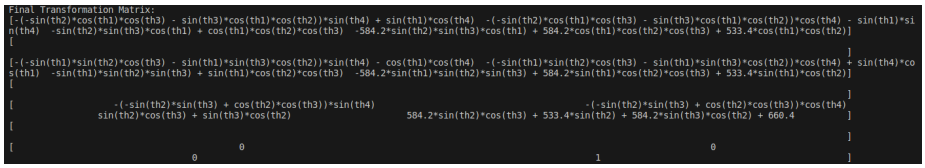

Figure 8: Final Transformation Matrix

8 Inverse Kinematics

In order to perform our manipulator trajectory planning, we had to use Inverse Kinematics. To do so, we first calculated the Jacobian matrix for our manipulator. To find the Jacobian matrix, we have to find each homogeneous transformation matrix of each joint with respect to the base frame. These are given by:

$$H_1^0 = T_1^0$$

$$H_2^0 = T_1^0 T_2^1$$

$$H_3^0 = T_1^0 T_2^1 T_3^2$$

$$H_4^0 = T_1^0 T_2^1 T_3^2 T_4^3$$

Now, to formulate the Jacobian matrix, we have to extract P and the Z components from each respective base-frame transformation matrices. P is the position of the end effector as a function of joint positions (last column, first three rows of H_4^0). Z is the third column, first three rows of each transformation matrix. Then, I can calculate each component of the Jacobian matrix in the following form:

$$J_i = \begin{bmatrix} \frac{\partial P}{\partial q_i} \\ Z_i \end{bmatrix}$$

Once we have each Jacobian component, we can formulate the Jacobian matrix as follows:

$$J = \begin{bmatrix} J_1 & J_2 & J_3 & J_4 \end{bmatrix}$$

Given in Figure 9 is a screenshot of our Jacobian Matrix:

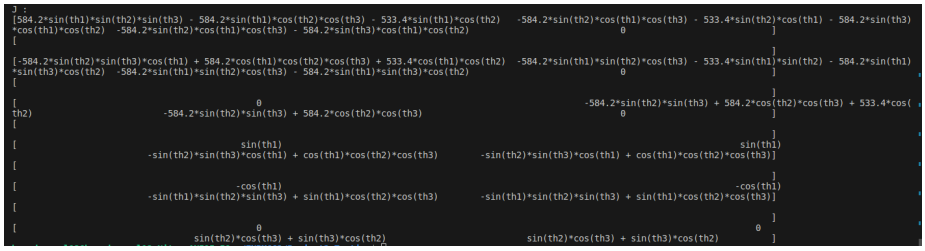

Figure 9: Jacobian Matrix

9 Forward Kinematics Validation

9.1 Method

In order to validate the forward positional kinematics, we substituted known joint angles and configurations into the final transformation matrix and validated that the manipulator moves as intended. We used the Peter Corke Robotics Toolbox in MATLAB to accomplish this. Also, please note that all measurements were converted to mm for these tests and moving forward.

9.2 Tests

Please note that during the tests (and in our D-H Table) that joints 3 and 4's "home position" starts at 90. This has to be set manually using the Peter Corke GUI. Below in Figure 10 is our manipulator's home position:

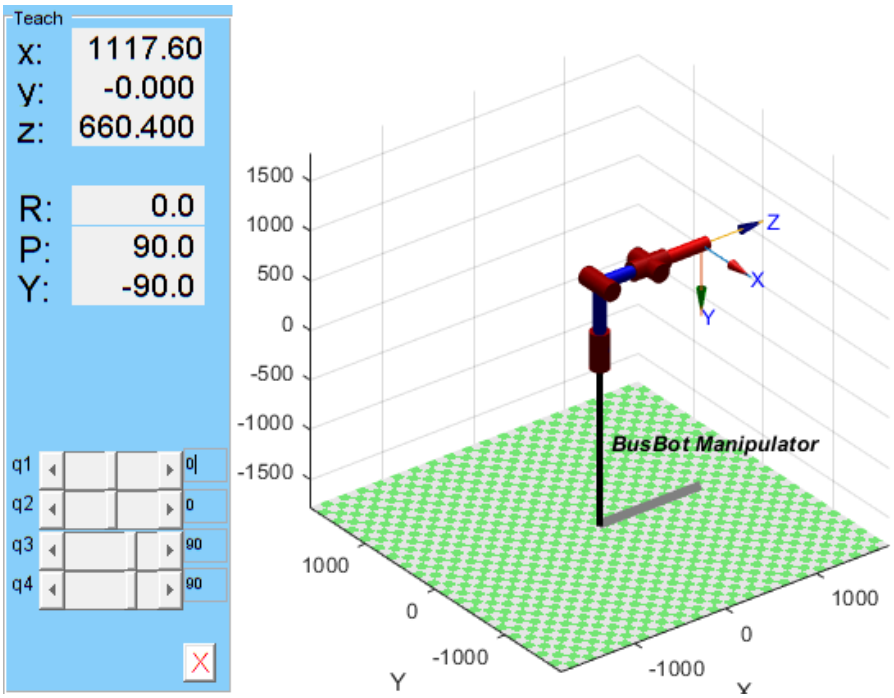

Figure 10: Home Position

9.2.1 Only Joint 1 Rotated by 90

The final homogeneous transformation matrix for q_1 rotated by 90 (all else zero) is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1117.6 \\ 0 & -1 & 0 & 660.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 11 is the visualization of this:

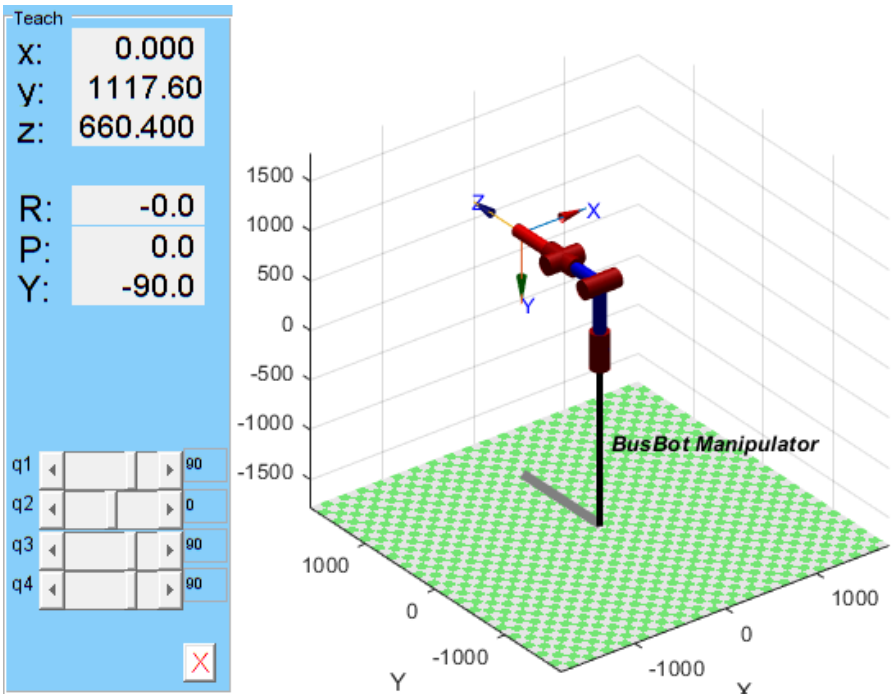

Figure 11: Q1 Rotated by 90

9.2.2 Only Joint 2 Rotated by 90

The final homogeneous transformation matrix for q_2 rotated by 90 (all else zero) is:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1778.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 12 is the visualization of this:

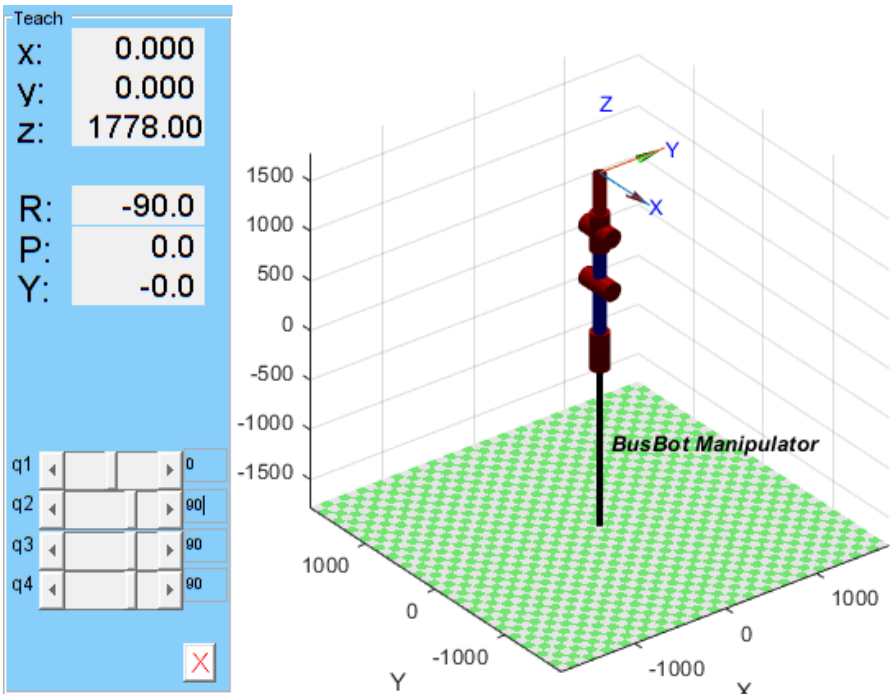

Figure 12: Q2 Rotated by 90

9.2.3 Only Joint 3 Rotated by 90

The final homogeneous transformation matrix for q_3 rotated by 90 (all else zero) is:

$$\begin{bmatrix} 0 & 1 & 0 & 533.4 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1244.6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 13 is the visualization of this:

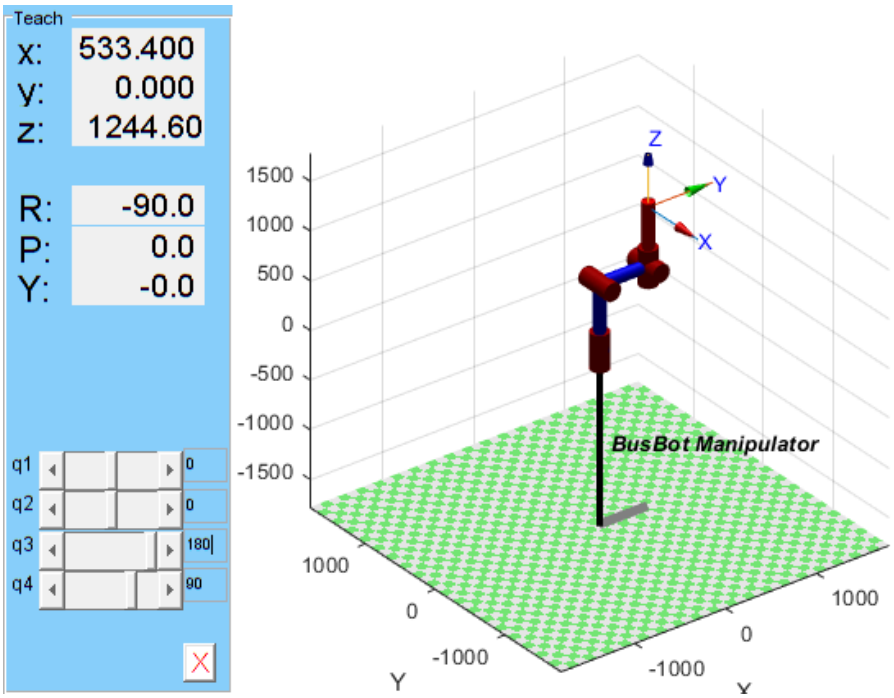

Figure 13: Q3 Rotated by 90

10 Inverse Kinematics Validation

10.1 Method

The validation of our Inverse Velocity Kinematics come in the form of our trajectory planning. Since we are pre-planning our trajectories for this project, we will include those here as validation. First, we define q (the joint position matrix) to be

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

and $\dot{q}$ (the joint velocity matrix) to be

$$\dot{q} = \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \end{bmatrix}$$

We also define $\dot{X}$ (the velocity form of our trajectory) to be

$$\dot{X} = \begin{bmatrix} V_x \\ V_y \\ V_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

With $\dot{X}$, J, and $\dot{q}$, we can write the following equations:

$$\dot{X} = J\dot{q}$$

$$\dot{q} = J^{-1}\dot{X}$$

Because the Jacobian is not square, we had to use pseudo-inverse in this step. In order to obtain the new joint angles, we use numerical integration:

$$q_i = q_{i-1} + \dot{q}_i \Delta t$$

Where $\Delta t = \frac{T}{N}$ and T is the total time of execution and N is a selected number of data points.

10.2 Tests

10.2.1 Straight Line Z Trajectory

The goal of this test was to move the end effector in a straight line down in the Z direction to pick up the cup. In order to lift the cup, we just executed the trajectory in the reverse order. This maneuver starts from the robot's home position. Pictured below are two views of the result:

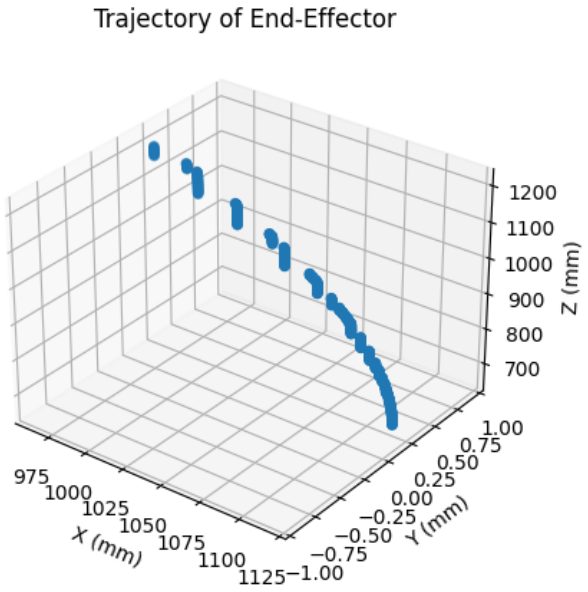

Figure 14: Trajectory 1 XYZ View

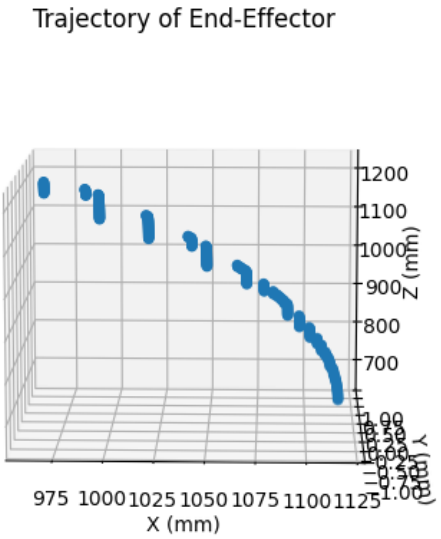

Figure 15: Trajectory 1 XZ View

Due to the geometry of our specific manipulator, this was an extremely hard trajectory to take. There is little-to-no variation in the Y direction, but a lot of variation in the X direction. This is because that our manipulator can move the end effector in -Z using multiple combinations of joint angles. Hence, the disconnected trajectory. However, applying this trajectory in simulation actually served to our benefit, as we could anticipate and plan around the X variation. Regardless, the goal of moving the end effector 60 cm in the Z direction was accomplished, with a known variation in X.

10.2.2 Half Circle in the X-Y Plane

The goal of this test was to move the end effector in a half-circle in the X-Y plane to move the cup over our basket. This maneuver starts from the robot's home position. Pictured below are two views of the result:

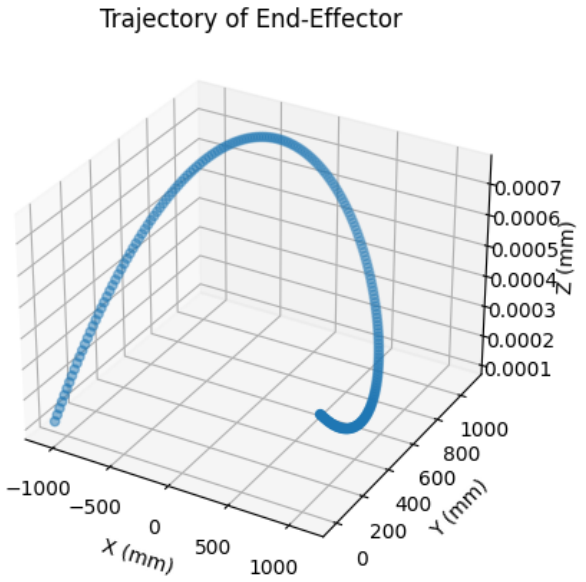

Figure 16: Trajectory 2 XYZ View

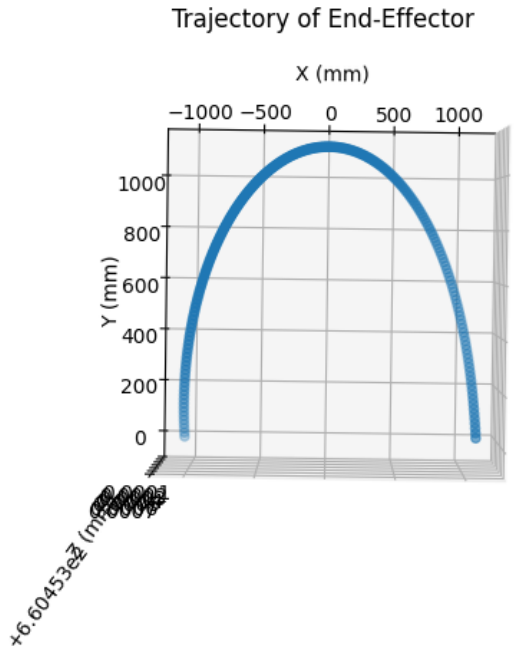

Figure 17: Trajectory 2 XY View

As shown in the figure, the trajectory was executed almost perfectly with a +/- 0.004 mm variation in the Z direction.

10.2.3 Straight Line X Trajectory

The goal of this test was to move the end effector in a straight line in the X direction to move the cup directly over our basket before dropping it in. This maneuver starts from the robot's position after executing the previous trajectory. Pictured below are two views of the result:

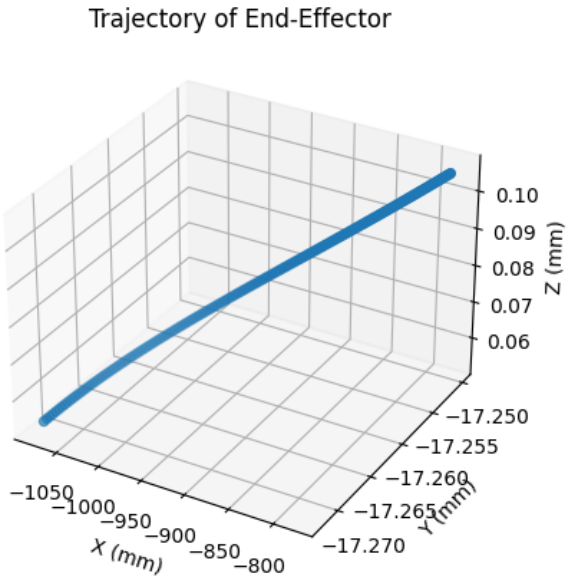

Figure 18: Trajectory 3 XYZ View

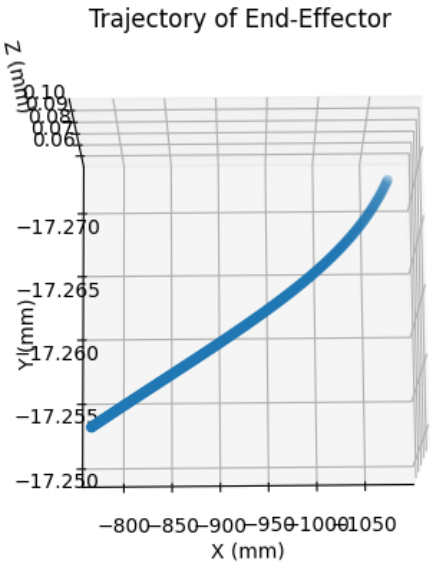

Figure 19: Trajectory 3 XY View

As shown in the figure, the trajectory was executed almost perfectly with a +/- 0.04 mm variation in the Z direction and a +/- 0.01 variation in the Y direction.

10.2.4 Discussion

The trajectories presented above are not perfect (as the position of the end effector varies in unwanted planes). However, because our application does not require an extremely high accuracy, the error was within our personal thresholds. The reasons for these errors was due to using numerical integration, pseudo-inverse, and setting non-perfect zero home position. However, the drawn trajectories were still executed relatively well in our simulation.

11 Workspace Study

The figure below shows a 2D (x/z plane) representation of the workspace of the manipulator arm with respect to the car. From the full arm being vertical (not as shown in the figure), joints 2 and 3 can rotate a maximum of 110 degrees each in either direction. This allows the end effector to reach a radius of 44 inches up to 110 degrees in either direction from the vertical, plus a radius of 23 inches of an additional 110 degrees. In the x/y plane, the robot can rotate

360 degrees (from joint 1) with any of the configurations listed above. The end effector can also rotate 360 degrees axially (from joint 4) to accommodate any roll/tilt orientation.

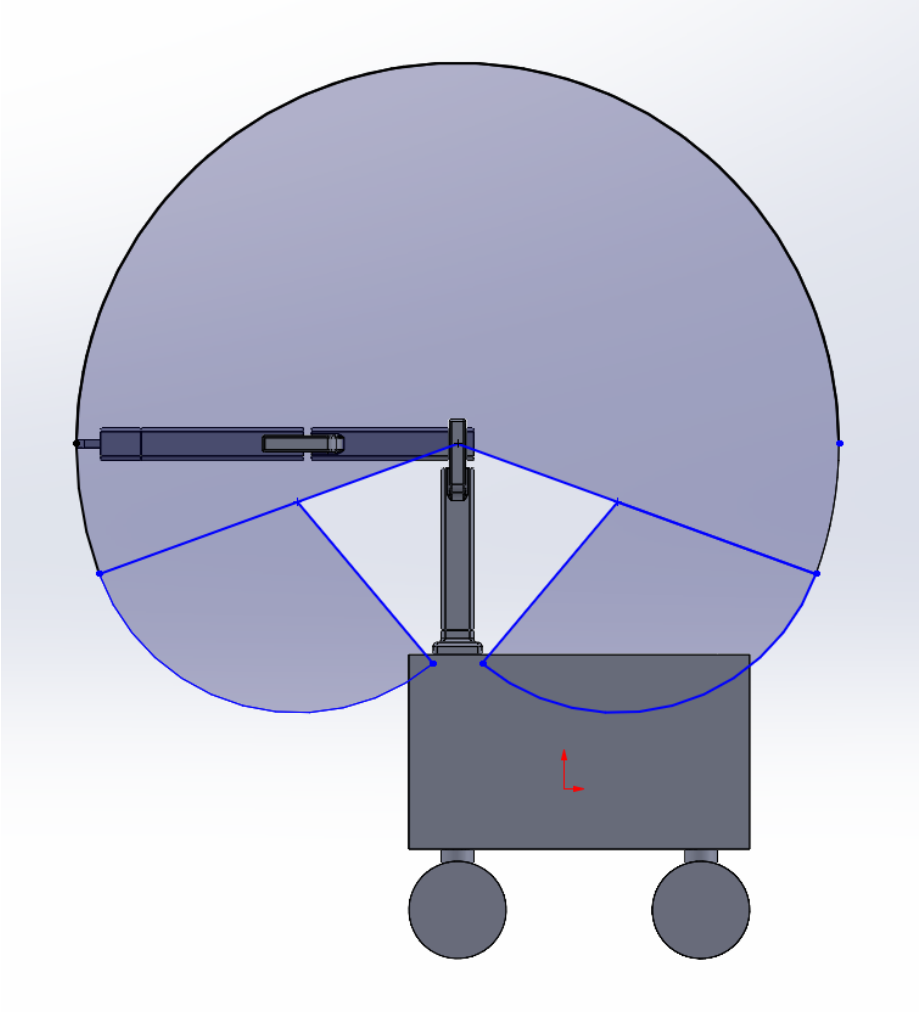

Figure 20: Workspace Study

12 Assumptions

This robot uses very little sensor and perception techniques. This means we have to assume a lot about the environment that would not be applicable in the real world. First, we assume that the cup's pose is known at all times. Since we built the Gazebo world, this information is easy to back out. Secondly, we are

using a vacuum gripper. This means we have to assume the cup can "stick" to the end effector of our robot regardless of orientation, which is not realistic in a real-life scenario.

13 Control Method

The main control method we implemented was open loop control. As mentioned in the previous section, our use of perception techniques is very limited and we could not properly use sensors to close the loop. The reason our robot control system is open loop is because of two things: time constraints and need. In our initial proposal, we wanted to use cameras and object-detection to detect the cup we wanted to pick up and plan the trajectories off of that. However, real-time trajectory planning (especially self-coded) is slow and extremely difficult. Because we had to fall back on hard-coding our trajectories, we no longer had a need to "close the loop" and use any closed loop control techniques. Obviously this methodology would not work in the real world, but this will be further discussed in the "Future Work" section. As an aside, all of the joint controls use ROS2 and Gazebo's in build PID control.

As for executing trajectories, we used only velocity control through inverse kinematics for the completion of this project. The pre-planned trajectories displayed earlier were completed in the Gazebo simulation.

14 Gazebo Visualization

Clicking on this link will take you to a video of our robot driving forward, picking up a cup, and then placing it into it's basket. As seen in the video, the disjointed nature of trajectory 1 is confirmed. There are multiple joint positions that can move the end effector in a straight line in the Z direction, and the calculation of the inverse kinematics swaps between the two.

15 RVIZ Visualization

Depicted below is a screenshot of our robot in RVIZ with associated joint sliders:

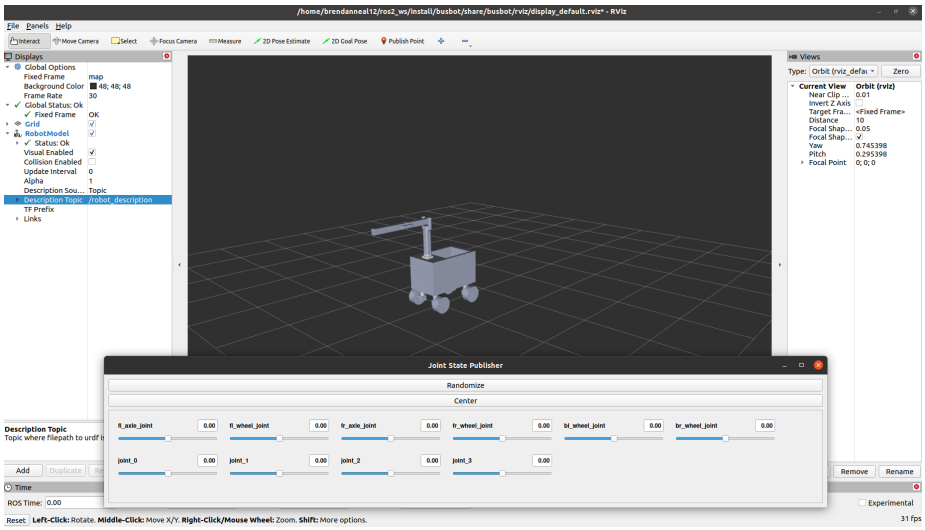

Figure 21: Robot RVIZ View

16 Problems Faced

We faced a multitude of problems during the completion of this project, but thankfully we were able to work through them. Our first problem came in the form of our robot's initial design. The wheel/axle assembly was too small relative to the rest of the robot. As a result, our robot kept falling apart on spawn because it was not able to to handle the weight of the carriage/manipulator on top of it. To fix this, we had to redesign the base.

Another issue that we ran into was the position joints kept falling apart when we attempted to control them. Thanks to the TA's help, the fix was to add a velocity controller to the XACRO file in order to fix the issue.

An issue we ran into was how do we actually publish the joint positions within our timer callback. However, after a few hours of debugging, we were able to get the logic down.

Finally, we had to make our manipulator move slowly because if we had the manipulator execute trajectories too quickly, it would fall apart or the whole vehicle would tip over. This just validates what we learned in class about quasi-static manipulator control.

17 Lessons Learned

We learned multiple things through the course of completing this project. First of all, we learned that when designing a robot in CAD, we have to take into account the weight and balance of individual parts and assemblies and ensure that our design was stable and functional in a physics simulation. SolidWorks won't tell us that our robot will tip over, but Gazebo will, so designing a robot with real-life physics conditions in mind is very important.

Secondly, we learned that real-time trajectory planning and incorporation of sensors/perception is incredibly difficult. Our original plan for this project was to use a camera to detect the tableware we desired to pick up, but once we began "closing the loop" we realized that having our robot control system detect, and then plan the joint trajectories would be very difficult for a pick-and-place application. Hence, we had to fall back on hard coding trajectories and open-loop control.

As a side note, we have both taken ENPM661 Planning for Autonomous Robots and completed a project in that class using MoveIt. After completing this project and the homework throughout this course, we have truly learned how powerful MoveIt is and appreciate it a lot more than we did this past spring.

18 Future Work

Some future work examples include using perception techniques in order to close the loop when it comes to controlling the robotic arm. Our initial plan included using a camera to detect and automatically plan the trajectory to the desired tableware. However, that proved to be much more difficult to do than expected. However, our methods performed in this project would not be applicable to a real-life system. Thus, the need for sensing and perception would be an excellent choice for future work.

Additionally, as mentioned in class, gripping is an extremely tough challenge in robotics today. The vacuum gripper is an idealized version that provides an easy solution for simulations. However, the vacuum gripper does not exist in real life (at least to the extent it is in Gazebo). Thus, more research will have to be performed in both gripper technologies and gripper techniques if this is to be implemented in real life.

19 Conclusions

During the completion of this project, we had many successes. We successfully designed a mobile manipulator using SolidWorks and imported it into a

Gazebo simulation. We successfully defined the D-H table, calculated the forward kinematics and validated them. We calculated forward velocity kinematics, the Jacobian, inverse velocity kinematics, and validated them. We successfully planned and executed four unique trajectories using inverse velocity kinematics in order to simulate picking up a cup from a dinner table.

Even though this system cannot be implemented in the real world, this project proves that mobile manipulators can be used to clear tables in restaurant applications and serves as an excellent starting point for anyone who would like to pursue this project in the future.

References

[1] Chris Albrecht. *Smile Robotics Makes an Autonomous Bussing Robot*. URL: https://thespoon.tech/smile-robotics-makes-an-autonomous-table-bussing-robot/.

[2] Mark W. Spong, Seth Hutchinson, and M. Vidyasager. *Robot Modeling and Control*. Jihn Wiley and Sons, 2006.