Inverse Kinematics

Introduction

This lab uses the dimensions of the UR3 robot used in lab and calculates the angles of each joint using inverse kinematics concepts. Inverse kinematics is the process of finding the angles of each joint in the robot, given the dimensions of each joint and the location of the end effector in the world frame. This lab makes some assumptions about the positioning of the robot to make the equations easier to solve, such as having the fifth joint always at -90^o, ensuring that the end effector is always perpendicular to the surface of the table. Most of the challenge of this lab is to use simple geometry concepts in creative ways and keeping in mind the dependencies that certain angles and dimensions have on others as the robot repositions itself.

Method

The first step to develop a set of equations is to convert the inputted x_w, y_w, z_w, and yaw values to (x_{cen}, y_{cen}, z_{cen}). The inputted coordinates describe the position of the end effector in the world frame and need to be converted to the body frame of the robot to make future calculations easier. Figure 1 shows the origin of the world frame and of the body frame for the robot. Figure 2 then shows some dimensions describing the dimensions of the square base of the robot. Figure 3 depicts a sketch of the robotic arm with joints and points of reference marked with circles, along with labeling each link of the arm. These labels correspond to real world dimensions that will be plugged in when programming. The coordinate transform was calculated using this information and with methods shown in Figure 4.

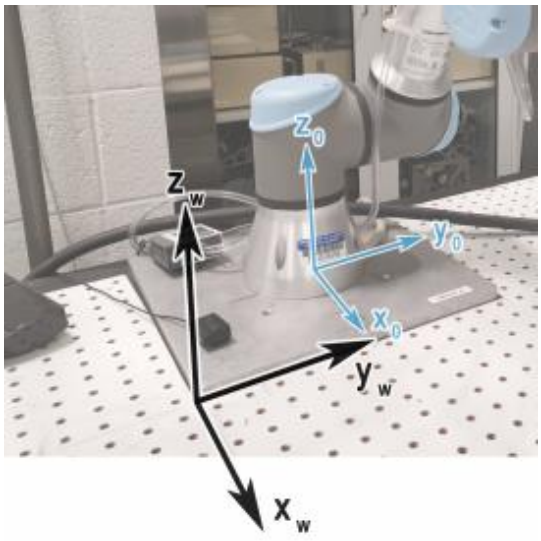

Figure 1: Labeled world frame and body frame of UR3 robot.

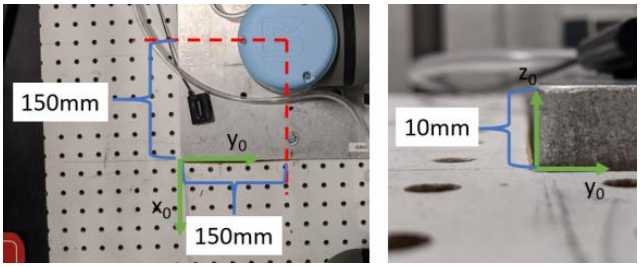

Figure 2: Labeled dimensions of the UR3 base. In these pictures, x_o, y_o, and z_o are labeling the world coordinate frame.

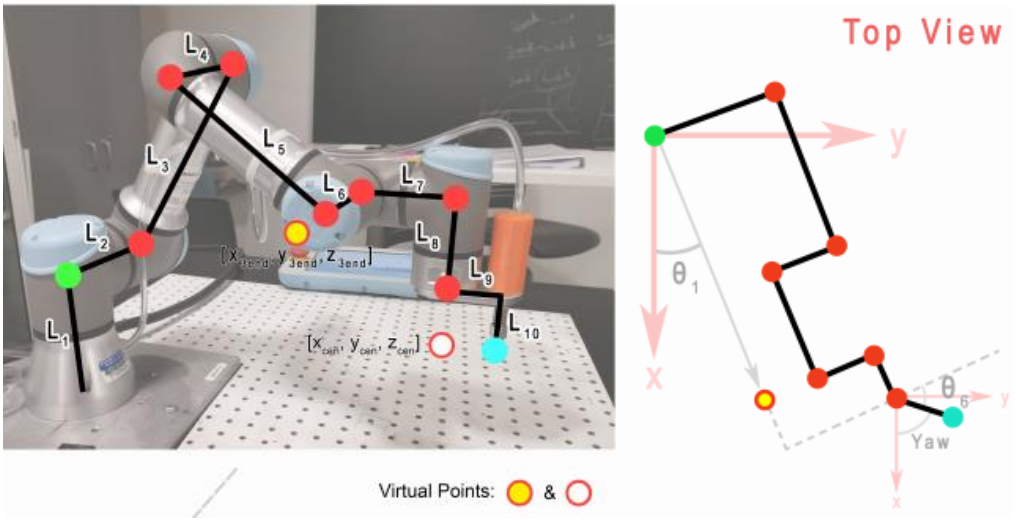

Figure 3: UR3 robot with joints, reference points, and links labeled.

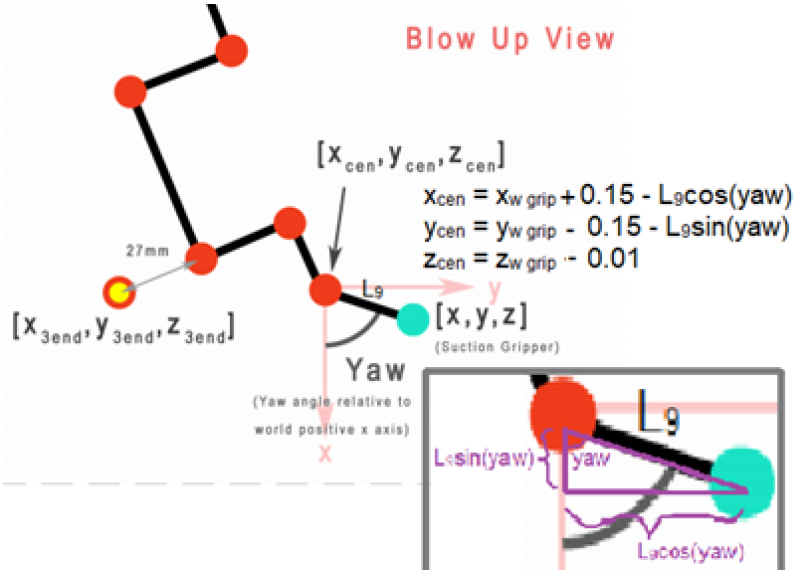

Figure 4: Process and final equations to solve for the coordinates of (x_{cen}, y_{cen}, z_{cen}) in terms of (x_w, y_w, z_w, yaw).

Next, the equations for (x_{cen}, y_{cen}, z_{cen}) were used as parameters for the equation for θ_1. Figure 5 shows the method used to create the equation for θ_1. θ_6 can be written as just a function of θ_1 and yaw, as shown in Figure 6. The rest of the angle equations will be set to be dependent on a virtual point (x_{3end}, y_{3end}, z_{3end}) that is labeled with a yellow dot in Figure 3 and 4.

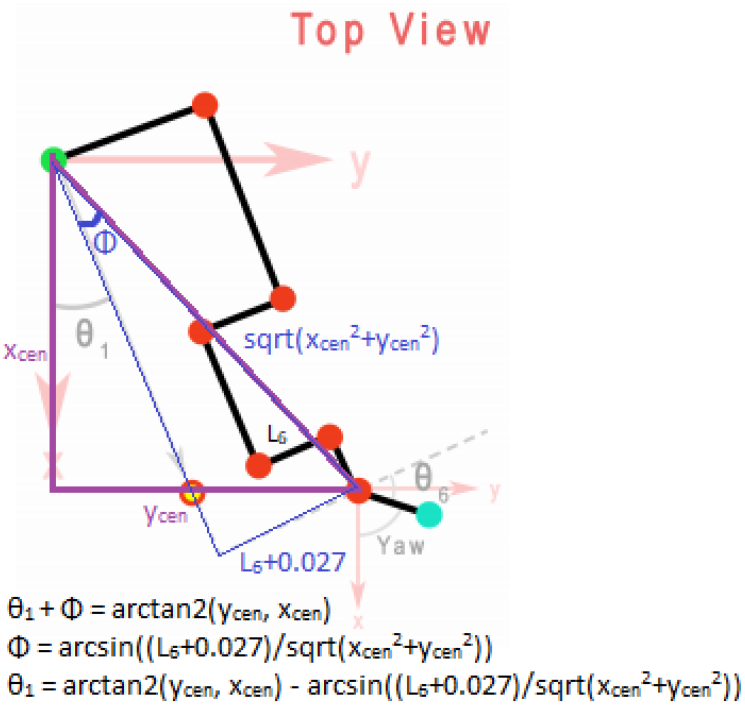

Figure 5: Process and final equations to solve for θ_1 given (x_{cen}, y_{cen}, z_{cen}).

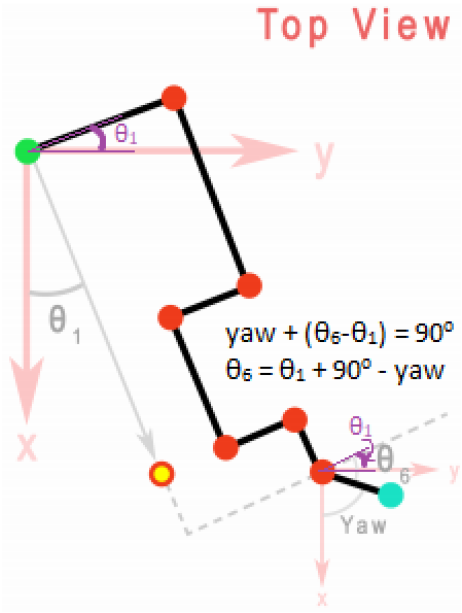

Figure 6: Process and final equations to solve for θ_6 given θ_1 and the yaw angle.

At this point, the rest of the angles will be defined in terms of a virtual point on the end of one of the joints, labeled by a yellow circle on the previous figures. This was the hardest set of equations for me to derive because it was very difficult to not make (x_{end}, y_{end}, z_{end}) dependent on links closer to the base of the arm. The process to determine these equations began by looking at what they should equal at various simple configurations such as when θ_1 was 0 or 90 degrees. My thought process while I created my equations is shown in Figure 7. The equation for z_{end} can easily be solved for by looking at the side view in Figure 8 and noting that z_{end} would always be z_{cen} + L_8 + L_{10}.

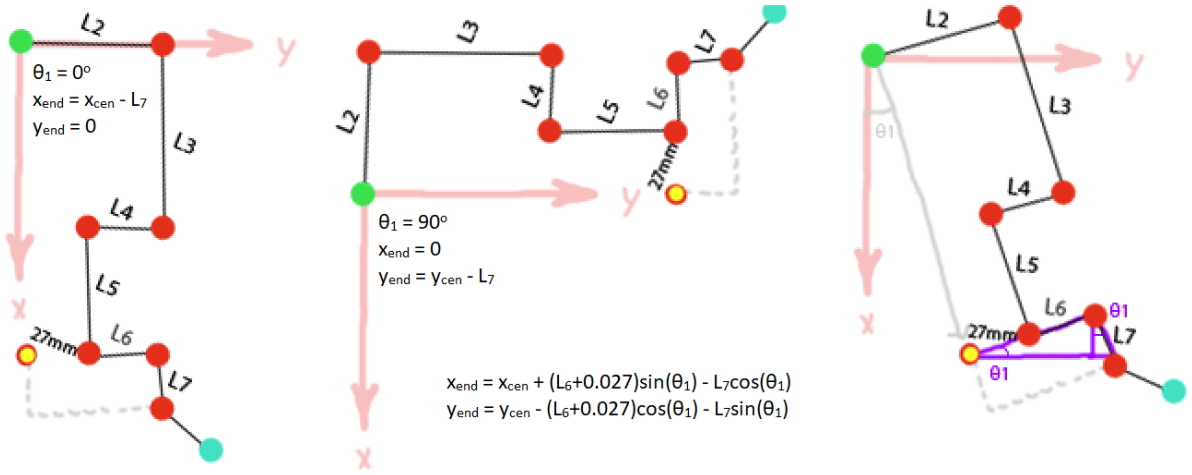

Figure 7: Process of finding the equations for x_{end} and y_{end} in terms of (x_{cen}, y_{cen}, z_{cen}) and θ_1.

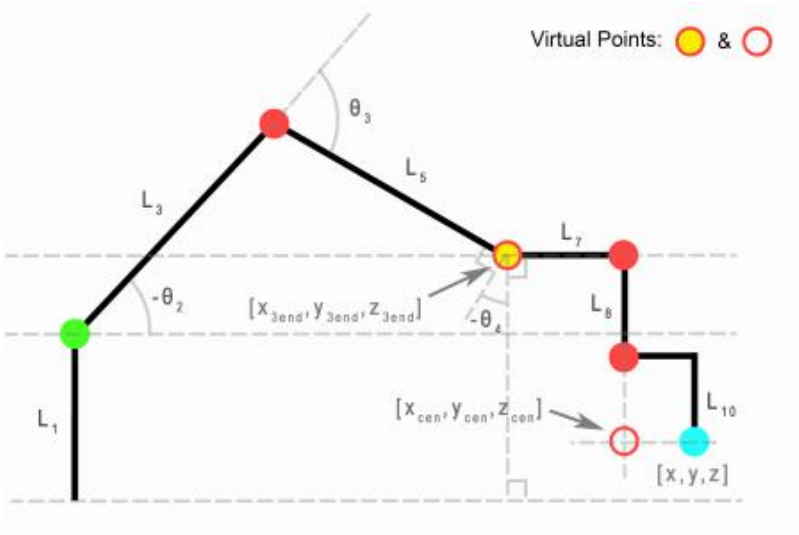

Figure 8: Side view of the UR3 arm with labeled links and points of interest.

Finally, the equations for the rest of the joints can be solved for in terms of (x_{end}, y_{end}, z_{end}), primarily using the side view shown in Figure 8. Figure 9 shows an annotated view of Figure 8 with labeled intermediate angles and lengths between joints. These can largely be

manipulated with simple geometry and with the law of cosines to find the shown equations for θ_2, θ_3, and θ_4. θ_5 can be assumed to always be -90° to keep the end effector parallel to the surface of the table. With all the equations solved for, the last step was to implement it in python as shown in the appendix. Several test values were used and checked with the last lab's forward kinematics program before solving for the points given below. The distances in the first table are in meters and the angles in the second table are all in degrees.

Test Point	x_w	y_w	z_w	θ_{yaw}
1	0.15	-0.10	0.25	-45°
2	0.20	0.40	0.05	45°

Test Point	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	-58.018	-84.526	88.264	-3.738	-90	76.982
2	-58.018	-84.526	88.264	-3.738	-90	76.982

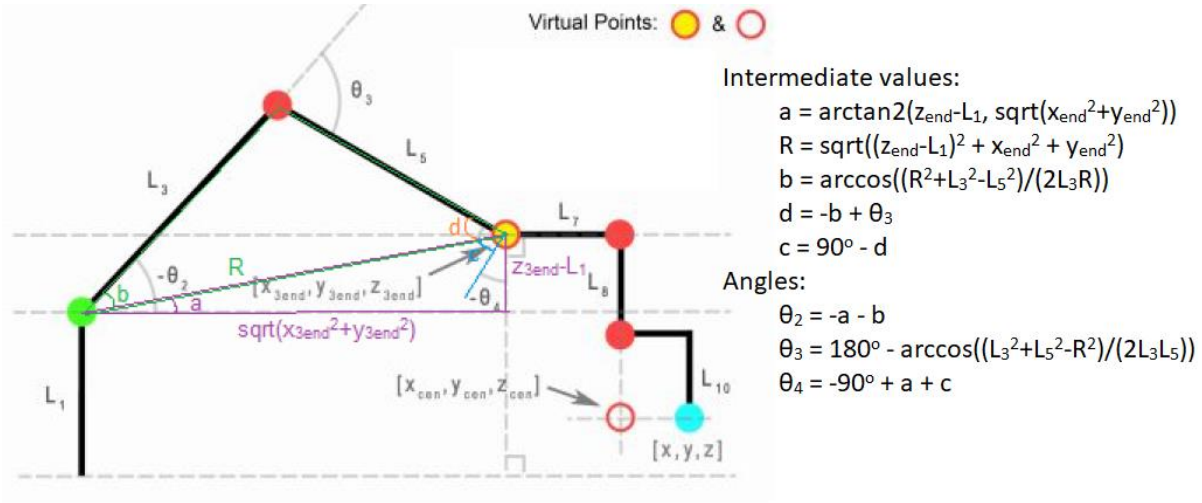

Figure 9: Side view of the UR3 arm with intermediate values for lengths and angles that are used to compute the final values of the desired joint angles.

Results

The most challenging part of this lab was to be able to consistently try to create each equation without making them interdependent or dependent on links that would not quite reliable. Looking from the top view of the robot arm it was very tempting to use L_3 and L_5 to determine distances but as I thought about it, those joints don't measure distance accurately from the top view unless θ_2 and θ_3 were 0°. It was also very difficult to try to consider the whole spectrum of possible values for θ_1 when creating equations. Numpy's arctan2 function helps with that but I'm worried if there is an error it is because of some of the equations not accounting for every angle.

While testing my function, there tends to be a rounding error of at most 0.01 on some of the location values after putting my thetas back through a forward kinematics solver. This could be because of a small difference in labeled dimensions between the link distances we were given for this lab and the link distances I calculated for the last lab.

Conclusion

This lab took a long time to get working and was very difficult to debug. At one point I nearly rewrote all my equations in one day trying to find out why my results were not coming out accurately. Small sign errors ended up causing me the most trouble when trying to debug and when working with my lab partner, I noticed that some distance measurements didn't quite line up as we expected them to. In several of my equations I use L_6+0.027 as the distance between (x_{end}, y_{end}, z_{end}) and the 6^{th} joint in the top view but my lab partner was using L_2-L_4+L_6. Earlier in the checkpoints, I was using just L_2 for this distance. I still feel like all these ways of measuring that distance should be the same but they don't add up to the same values. While this is a small discrepancy, it still adds some confusion to the lab because for a while I was trying to see which one was the truly correct way to denote that distance in case it was causing me to get the incorrect answer.

I think one of the best things about this lab was that it was easy enough to draw on the diagrams and show my thought process to others when I got stuck, or to myself if I forgot why I was solving my equations the way I was solving them. Solving for each equation might be tedious and easily error-prone but the process of solving for each equation can be easily displayed and that makes it very easy to ask anybody for help checking over your work.

References

Block, D., Holm, J., Xu, J., Fan, Y., Cui, H. and Chen , Y. (n.d.). *ECE470 Introduction to Robotics Lab Manual.* 2nd ed.