ECE 470: Lab 4 - Inverse Kinematics

Name: Ram Reddy
NetID: ramr3
Lab Partner: Kate Lee
TA: Lukas Zscherpel
Section: Monday 9AM
Date Submitted:11/03/2025

1. Introduction

The purpose of this lab is to derive and implement the solution to the inverse kinematics equations for the UR3 robot. We will specifically derive the elbow up inverse kinematics equations and write the python code to move the robot to the specific end effector coordinates. In order to solve the equations we needed to first identify a pose and then analyze and solve for each joint angle using analytical inverse kinematics in respect to that pose. Once we figured out the equation for each angle we could then code these equations in python to move the UR3 end effector to the desired location. We finally tested the derived equations using the specified end effector locations to ensure accuracy of the derived inverse kinematics expressions.

2. Methods

2.1 Establish world coordinate frame to convert to base coordinate frame

The first step of the process is to establish the world coordinate frame with respect to the center of the UR3 base plate, so that any inputted coordinates which are in the world coordinate frame can be converted into the UR3 base coordinate frame. The world frame is centered at the corner of the base UR3 base plate. The base coordinate frame is centered at the center at the top of the UR3 base plate. Looking at Figure 1 we can easily translate the world frame to the base frame. The xGrip and yGrip are simply translated by 150 mm. The zGrip is translated by 10 mm. To get the right sign we simply need to do the translation that will make the x, y, z grips into 0. The calculated translation is shown in code snippet 1 below. There is also a multiplication of 1000 to convert the given world coordinates into millimeters. Lastly, in Code Snippet 2, the link lengths are also shown in millimeters as they are helpful in calculating the other equations in this lab. The link lengths were taken from the Lab 3 Manual.

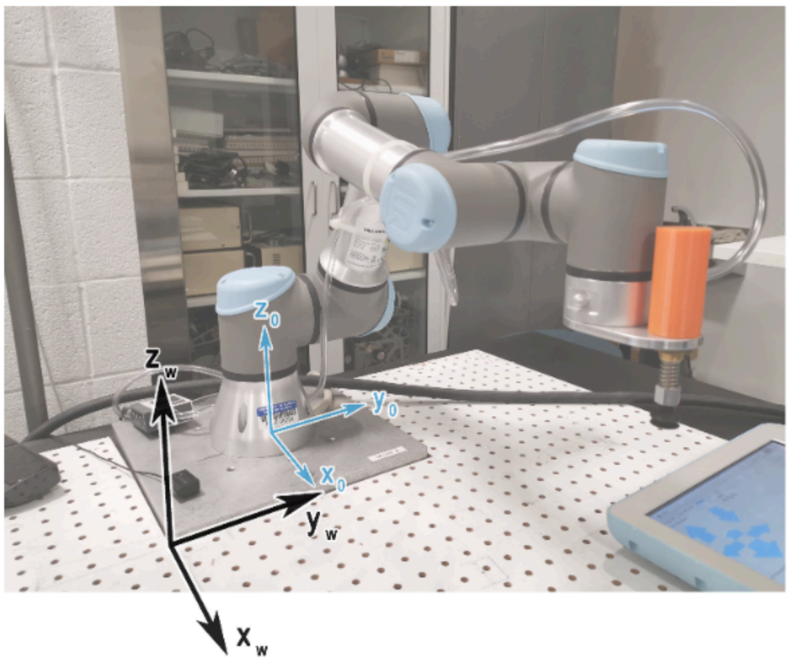


Figure 1: World and Base Frames in relation to UR3

```
xGrip = (xWgrip*1000+150)
yGrip =  (yWgrip*1000-150)
zGrip = zWgrip*1000 - 10
```

Code Snippet 1

```
L1 = 152
L2 = 120
L3 = 244
L4 = 93
L5 = 213
L6 = 83
L7 = 83
L8 = 82
L9 = 53.5
L10 = 59
```

Code Snippet 2

2.2 Finding Wrist Center Point

Now the next step is to get the wrist center point. This is calculated from the xGrip, yGrip, zGrip from step 1 and the inputted yaw angle. Immediately we can see that we only need to care about xGrip and yGrip because the zGrip is the same level as zCen as shown in Figure 2. When we look at the top view we can also see that to get to the center point we need to subtract the x and y components of the gripper depending on the yaw angle. Looking at Figure 3, we see that the center point corresponds to the point under the red point connecting link 9 and 8. Since we already know the gripper location, we simply need to subtract the link 9 components in correspondence with the yaw angle. Looking back the zoom in of the gripper in Figure 3, we can subtract the L9 * cos(yaw) to get the x components and the L9 * sin(yaw) to get the y component. The xCen, yCen, and zCen equations are shown in code snippet 3 below.

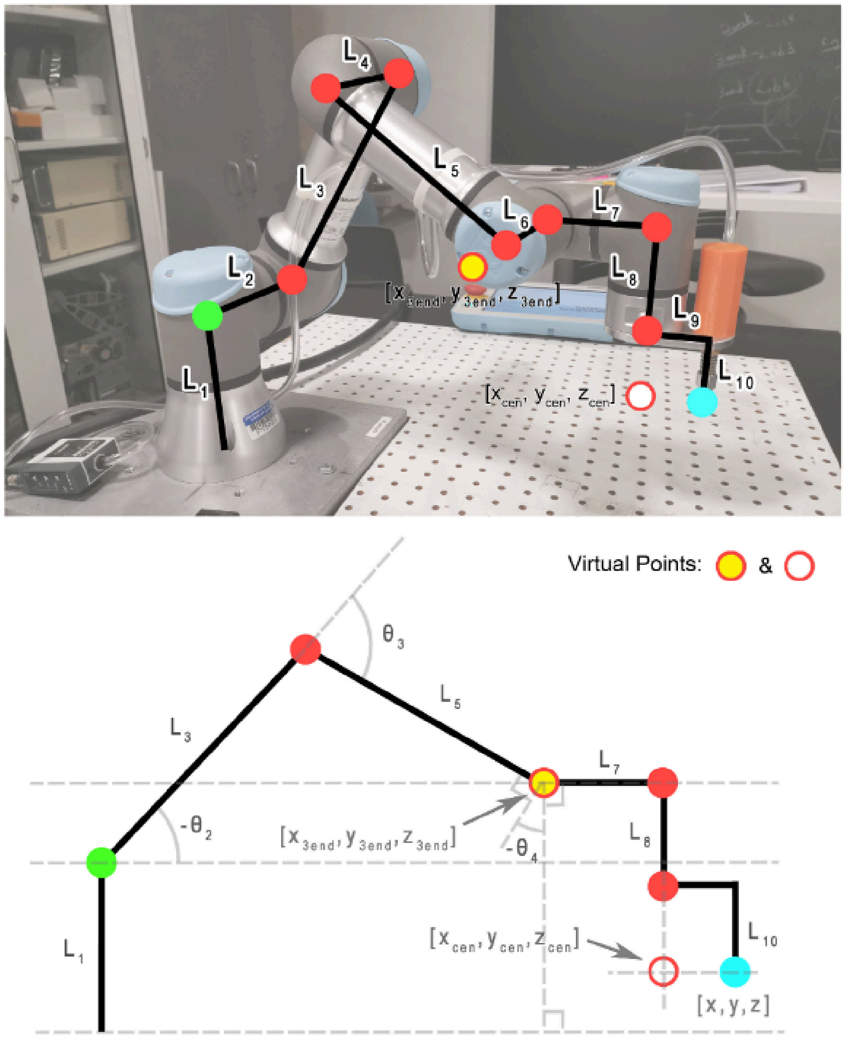


Figure 2: Side View of UR3

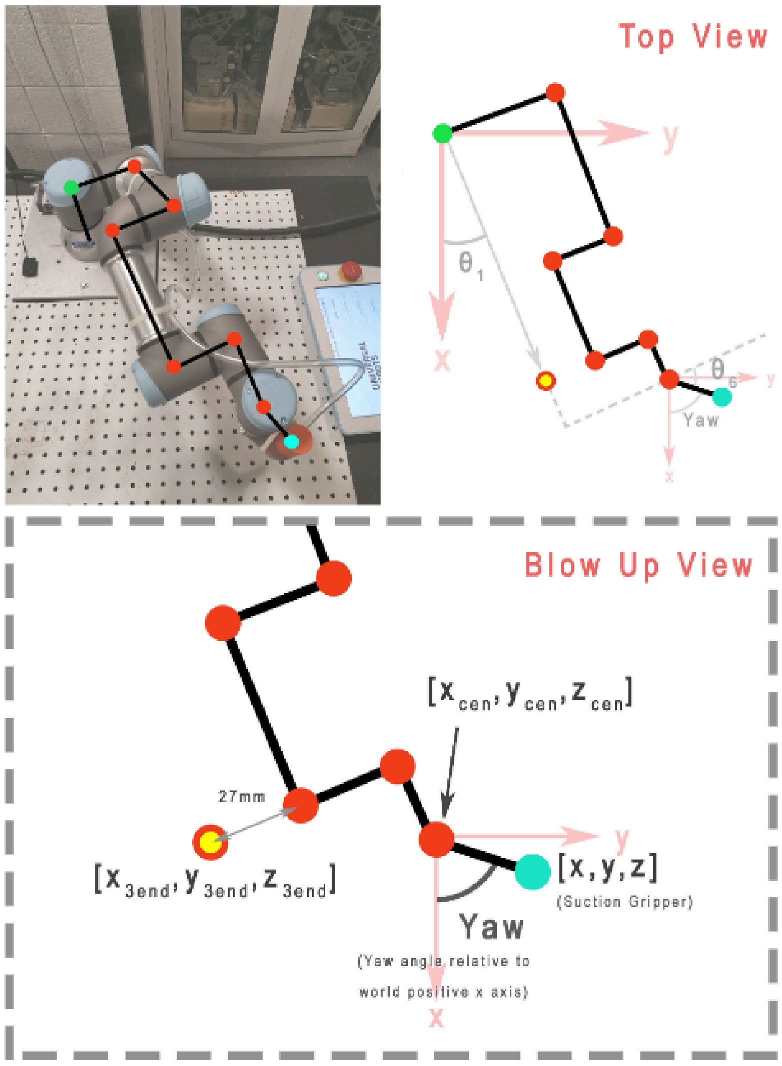


Figure 3: Top View of UR3

```
xCen = (xWgrip*1000+150) - L9*np.cos(np.radians(yaw_WgripDegree))
yCen =  (yWgrip*1000-150) - L9*np.sin(np.radians(yaw_WgripDegree))
zCen = zWgrip*1000 - 10
```

Code Snippet 3

2.3 Finding Theta 1

Now, the next step is to find theta 1. In order to find theta 1, we need to find two angles because theta 1 is not able to be solved with a single observable angle as it's not aligned and depends on the gripper. The two angles are shown in Figure 4 below. We need to first find alpha which is the theta big angle, this can be solved with a simple projection of the top view on the x,y plane. As we already know in this view, alpha would be purely dependent on the yCen and xCen. We can simply do an atan2 function to get the big theta angle, which would just be atan2(yCen, xCen). Now to get the small theta or beta, we can use the L2, L4, L6 sides and use the small angle triangle shown in the figure below. Since we know the hypotenuse and the small side we can use arcsin to calculate the angle. This would like arcsin((L2-L4+L6), sqrt(yCen^2 + xCen^2)). Now theta 1 would just be the big theta - small theta which is shown in code snippet 4 below.

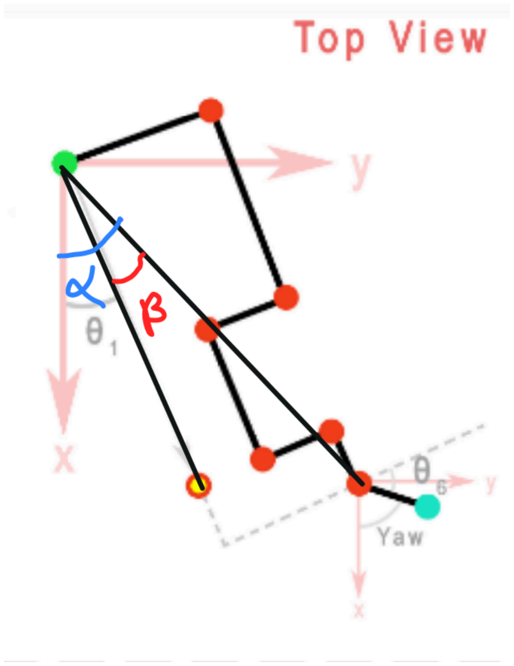


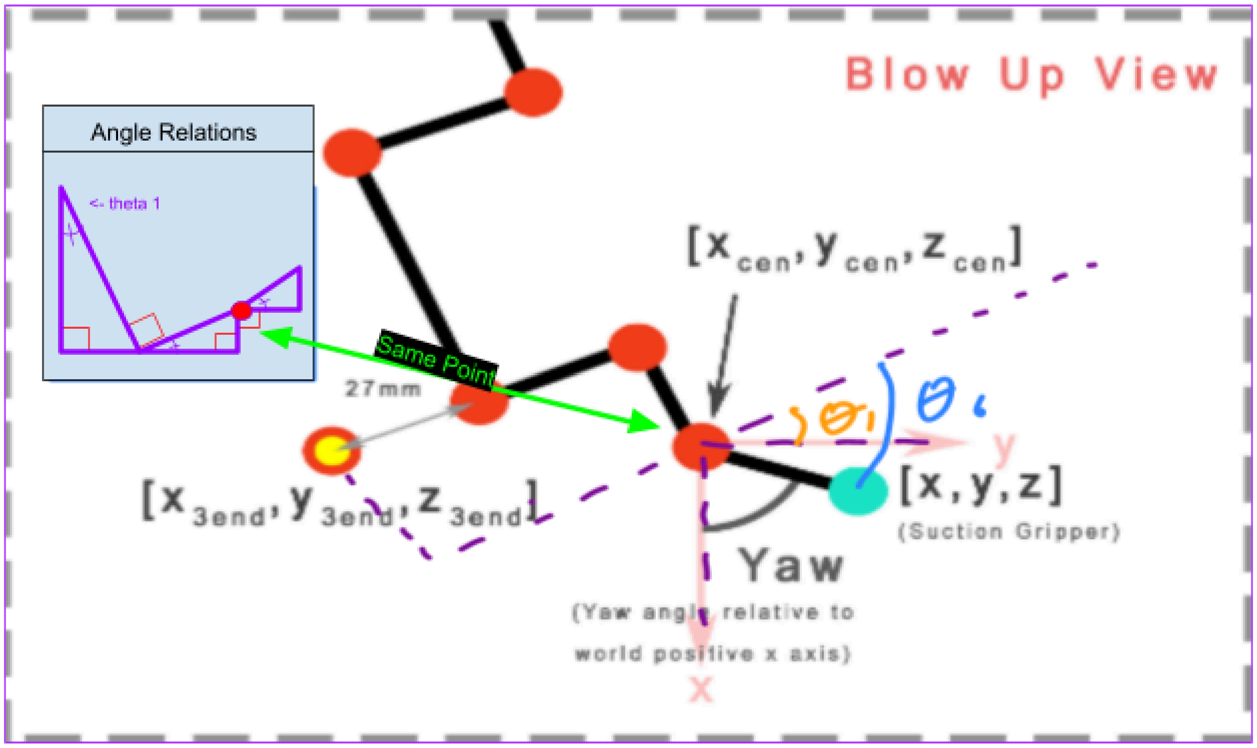
Figure 4: Top View of Alpha and Beta on UR3 Diagram

```
thetaBig = np.arctan2(yCen, xCen)
thetaSmall = np.arcsin((L2-L4+L6)/np.sqrt(xCen**2+yCen**2))
theta1 = thetaBig - thetaSmall
```

Code Snippet 4

2.4 Finding Theta 6

Now that we have found theta 1 we can now solve for theta 6 with the help of the yaw angle. Extending some lines to the top view of the UR3 we can find angle relationships with theta 1 and use that to help solve for theta 6. In Figure 5 below, we can see in the angle relationships diagram that the small angle from the extended purple line and y axis is actually just theta 1. From the angle relations diagram we also see that the other angle marked with 2 lines is 90 - theta 1. Since we know that an angle on a straight line is 180, we can calculate theta 6 from the known angles. This gives us that theta 6 = 180 - (90 - theta 1) - yaw → theta 6 = theta 1 - yaw. The angle represented in code is shown in code snippet 5 below.



Blow Up View
Angle Relations
<- theta 1
Same Point
27mm
$[x_{cen}, y_{cen}, z_{cen}]$
θ_1
θ_6
y
$[x_{3end}, y_{3end}, z_{3end}]$
[x, y, z]
(Suction Gripper)
Yaw
(Yaw angle relative to world positive x axis)
x

Figure 5: Top View of Theta 6 on UR3 Diagram

```
theta6 = 0.5*PI + theta1 - np.radians(yaw_WgripDegree)
```

Code Snippet 5

2.5 Finding Projected Endpoint

Moving on to the projected endpoint, there are multiple things to consider here. For this projected endpoint we need to use the xCen, yCen, zCen and the theta 1 variables. First we need to find the relationships between these two points. Looking at Figure 6 which is shown below, we can see that there are two important factors when looking at the x and y coordinates of x3End and y3End. There is one point connecting the L6 and L7 links and the other point connecting L5 and L6 links. We can also see there are two triangles to help find the x and y components using theta 1. To get y3End we need to first subtract the purple triangle y component which would be L7 * sin(theta 1). Then we need to get blue triangle y component which would be (27 + L6) * cos(theta 1). Subtracting both of these components from yCen gives y3End. Thus y3End = yCen - L7 * sin(theta 1) - (27 + L6) * cos(theta 1). Now we can go to x3End which is calculated very similarly. Getting both the purple and blue triangle x components, we get L7 * cos(theta 1) for the purple triangle and (27 + L7) * sin(theta 1) for the blue triangle. In this case we need to first subtract the x component of the purple triangle and then add in the x component of the blue triangle to get the x3End which gives x3End = xCen - L7 * cos(theta 1) + (27 + L7) * sin(theta 1). Lastly, for z3End we can look at the side view in Figure 2 above to see there is a simple translation. From Figure 2 we can see that L8 and L10 need to be added to zCen to get z3End thus z3End = zCen + L8 + L10. The full equations for these 3 are shown in code snippet 6 below.

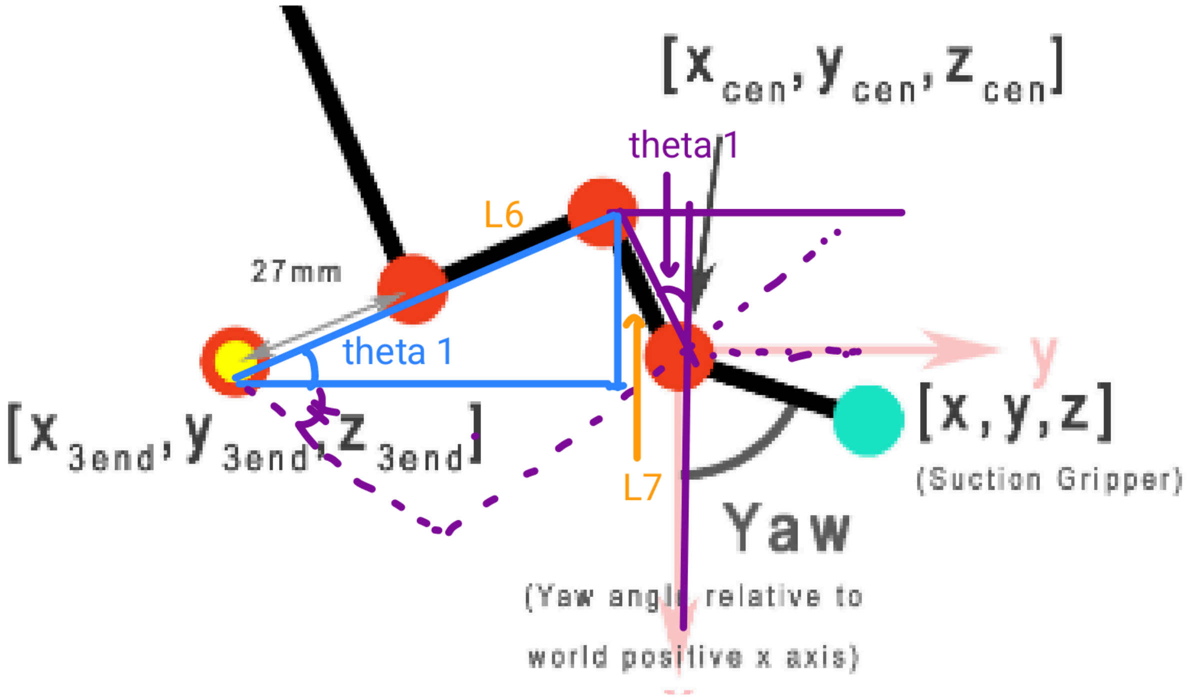


Figure 5: Top View of Endpoint Triangles on UR3 Diagram

```
x3End = xCen - L7*np.cos(theta1) + (27 + L6)*np.sin(theta1)
y3End = yCen - L7*np.sin(theta1) - (27 + L6)*np.cos(theta1)
z3End = zCen + L8 + L10
```

Code Snippet 6

2.6 Calculating Theta 2, Theta 3, Theta 4 and Theta 5

Finally, we are at the last stage of calculations. Theta 5 is given already which is - pi / 2. Theta 2, theta 3 and theta 4 are all similarly calculated. First, to calculate these angles then we need to look at the side view represented below in figure 6. Let's start with calculating theta 3 as that is the easiest. We first need to find the side c. To find c we need to get sides a and b. Side b is simply the z3End - link 1. Side a is projection of the endpoint on the x and y plane. This can be calculated from the sqrt(x3End^2 + y3End^2). Now we have sides a and b. We can use pythagorean theorem to find side c, so c = sqrt(a^2 + b^2). With side c we can use the law of cosines to find the supplementary angle to theta 3. Using the law of cosines 180 - theta 3 = arccos((L3^2 + L5^2 - c^2)/(2*L3*L5)) → theta 3 = 180 - arccos((L3^2 + L5^2 - c^2)/(2*L3*L5)).

With theta 3 we can move on theta 2. Theta 2 is calculated similarly to theta 1 where we find a big angle and small angle to calculate the full angle. First let's calculate the big angle which is shown as alpha in figure 6. To get alpha, we do a similar process to get theta 3. Using law of cosines, alpha = arccos((L5^2 - L3^2 - c^2)/(-2*L3*c)). Then to get the small angle or beta in figure 6, we can simply use an arctan on the two sides b and a, so beta = arctan(b/a)). Combining alpha and beta we get theta 2 =arccos((L5^2 - L3^2 - c^2)/(-2*L3*c)) +arctan(b/a). Also as theta 2 is negative in the lab manual we need to multiple theta 2 by -1 so theta 2 = -1*(arccos((L5^2 - L3^2 - c^2)/(-2*L3*c)) +arctan(b/a)).

Now that we have both theta 2 and theta 3, getting theta 4 is quite simple. To get theta 2 we first need phi which is in the top triangle in figure 6. Theta 2 is a corresponding angle in the top most triangle so phi = 180 - (180 - theta 3) - -theta 2 → phi = theta 3 + theta 2. We also know that theta 4 = 180 - 90 - (90 - phi) due to the upper transversal line. Plugging in everything we get theta 4 = 180 - 90 - (90 - (theta 3 + theta 2)) → theta 4 = 180 - 90 - 90 + theta 3 + theta 2 → theta 4 = theta 3 + theta 2. Also for this angle, it need to be negative to theta 4 = -1*(theta 3 + theta 2). All the equations are represented in coding snippet 6.

```python
a = np.sqrt(x3End**2 + y3End**2)
b = z3End - L1
c = np.sqrt(a**2 + b**2)

theta3 = PI - np.arccos((L3**2 + L5**2 - c**2)/(2*L3*L5))
theta2 = -1*(np.arccos((L5**2 - L3**2 - c**2)/(-2*L3*c)) + np.arctan(b/a))
theta4 = -1*(theta3 + theta2)
theta5 = -1*0.5*PI
```

Coding Snippet 7

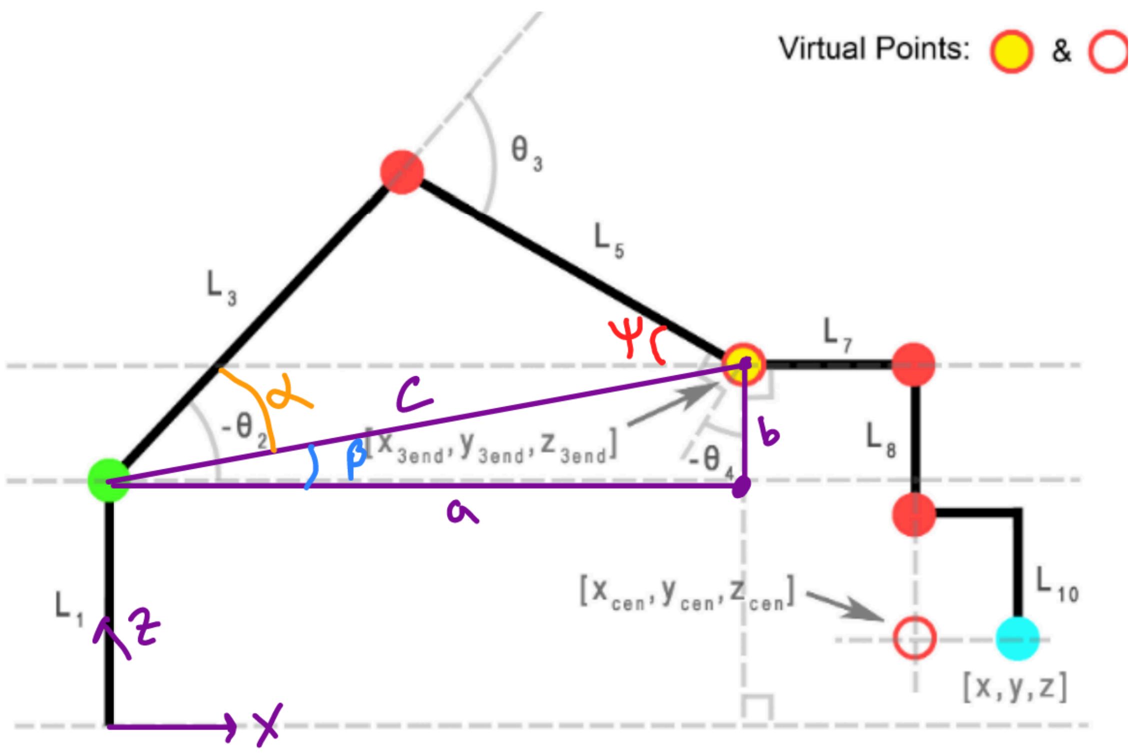


Figure 6: Side View for Theta 2, 3, 4 on UR3 Diagram

3. Data and Results

We were able to successfully complete the lab within the given lab hours. After finishing the lab, we tested the program with 3 different inputs. The program starts with three inputs which are the x, y, z world frame grip coordinates and the yaw angles in degrees which are shown in the Table 1 below. Then the program calculates the theta 1, theta 2, theta 3, theta 4, theta 5, and theta 6 values from the previously derived equations. Once these thetas are calculated they are then put through lab 3 forward kinematics matrices to get motor encoder values. The robot then automatically moves to the inputted position and then we would measure the position by hand using a ruler. After measuring we would find the error between the given x, y, z and the measured through the error formula in Figure 7. The results are shown in table 1 below.

$$error_1 = \|r_1 - d_1\| = \sqrt{(r_{1x} - d_{1x})^2 + (r_{1y} - d_{1y})^2 + (r_{1z} - d_{1z})^2}.$$

Figure 7: Error Formula

Table 1. Measured Position, Calculated Position, and the Error Based on Joint Angles			
D (Given x, y, z in m)	r (Measured Position in m)	yaw (degrees)	$error$ (m)
[0.15, 0,2, 0.15]	[0.183, 0.318, 0.149]	90	0.123
[0.2, 0.4, 0.05]	[0.2, 0.422, 0.046]	45	0.0224
[0.1, 0.25, 0.1]	[0.115, 0.258, 0.099]	105	0.017

The error can arise from many sources. There might be manufacturing tolerances which might change the minute differences between how the robot moves and and is represented in the world. Another possible reason for error is due to human error in actually measuring the position of the end effector. It was quite hard to measure the end effector with the given rulers as they were not big enough to measure most of the locations of the end effector from the origin, so we had to estimate the location of the end effector due to some extent. This could cause some error in the measured position.

4. Conclusion

In conclusion, in this lab we derive and implement the solution to the inverse kinematics equations for the UR3 robot. We calculated the transformation matrix by figuring out the M matrix and screw matrices. We specifically derived the elbow up inverse kinematics equations and wrote the python code to move the robot to the specific end effector coordinate. After calculating the thetas for the UR3 robot and further inputting them into the forward kinematics function we were able to generate the encoder values for each motor. The robot would then move into the correct place in the real world. We also measured the accuracy with three different points and had minimal error. Overall, the lab was a great learning experience in learning about analytical inverse kinematics and how it plays a role in moving a robot.

Additional Resources

Lab 4 & 3 Manual

Appendix A

```
#!/usr/bin/env python
import numpy as np
from scipy.linalg import expm
from lab4_header import *

"""
Use 'expm' for matrix exponential.
```

```
Angles are in radian, distance are in meters.
"""

def Get_MS():
    # =================== Your code starts here ====================#
    # Fill in the correct values for S1~6, as well as the M matrix
    M =
np.array([[0,-1,0,390/1000],[0,0,-1,401/1000],[1,0,0,215.5/1000],[0,0,0,1]])
    S1 = np.array([[0,-1,0,150/1000],[1,0,0,150/1000],[0,0,0,0],[0,0,0,0]])
    S2 = np.array([[0,0,1,-162/1000],[0,0,0,0],[-1,0,0,-150/1000],[0,0,0,0]])
    S3 = np.array([[0,0,1,-162/1000],[0,0,0,0],[-1,0,0,94/1000],[0,0,0,0]])
    S4 = np.array([[0,0,1,-162/1000],[0,0,0,0],[-1,0,0,307/1000],[0,0,0,0]])
    S5 = np.array([[0,0,0,0],[0,0,-1,162/1000],[0,1,0,-260/1000],[0,0,0,0]])
    S6 = np.array([[0,0,1,-162/1000],[0,0,0,0],[-1,0,0,390/1000],[0,0,0,0]])

    S = [S1, S2, S3, S4, S5, S6]

    # ==============================================================#
    return M, S

"""
Function that calculates encoder numbers for each motor
"""
def lab_fk(theta1, theta2, theta3, theta4, theta5, theta6):

    # Initialize the return_value
    return_value = [None, None, None, None, None, None]

    # =========== Implement joint angle to encoder expressions here ===========
    print("Foward kinematics calculated:\n")

    # =================== Your code starts here ====================#
    M, S = Get_MS()
    T =
expm(S[0]*theta1)@expm(S[1]*theta2)@expm(S[2]*theta3)@expm(S[3]*theta4)@expm(S[4]*t
heta5)@expm(S[5]*theta6)@M

    # ==============================================================#

    print(str(T) + "\n")

    return_value[0] = theta1 + PI
    return_value[1] = theta2
    return_value[2] = theta3
    return_value[3] = theta4 - (0.5*PI)
    return_value[4] = theta5
    return_value[5] = theta6
```

```python
    return return_value

"""
Function that calculates an elbow up Inverse Kinematic solution for the UR3
"""
def lab_invk(xWgrip, yWgrip, zWgrip, yaw_WgripDegree):
    # =================== Your code starts here ====================#
    theta1 = 0.0
    theta2 = 0.0
    theta3 = 0.0
    theta4 = 0.0
    theta5 = 0.0
    theta6 = 0.0

    L1 = 152
    L2 = 120
    L3 = 244
    L4 = 93
    L5 = 213
    L6 = 83
    L7 = 83
    L8 = 82
    L9 = 53.5
    L10 = 59

    xCen = (xWgrip*1000+150) - L9*np.cos(np.radians(yaw_WgripDegree))
    yCen =  (yWgrip*1000-150) - L9*np.sin(np.radians(yaw_WgripDegree))
    zCen = zWgrip*1000 - 10

    thetaBig = np.arctan2(yCen, xCen)
    thetaSmall = np.arcsin((L2-L4+L6)/np.sqrt(xCen**2+yCen**2))
    theta1 = thetaBig - thetaSmall

    theta6 = 0.5*PI + theta1 - np.radians(yaw_WgripDegree)

    x3End = xCen - L7*np.cos(theta1) + (27 + L6)*np.sin(theta1)
    y3End = yCen - L7*np.sin(theta1) - (27 + L6)*np.cos(theta1)
    z3End = zCen + L8 + L10

    a = np.sqrt(x3End**2 + y3End**2)
    b = z3End - L1
    c = np.sqrt(a**2 + b**2)

    theta3 = PI - np.arccos((L3**2 + L5**2 - c**2)/(2*L3*L5))
    theta2 = -1*(np.arccos((L5**2 - L3**2 - c**2)/(-2*L3*c)) + np.arctan(b/a))
    theta4 = -1*(theta3 + theta2)
    theta5 = -1*0.5*PI
    # ==============================================================#
    return lab_fk(theta1, theta2, theta3, theta4, theta5, theta6)
```