RBE500: Foundations of Robotics

Group Assignment Part 1

Student 1: Dhruv Agrawal
Student 2: Shreyas Khobragade
Student 3: Pranjali Rangnekar
Student 4: Roman Henry

1 Questions

1. **Forward Kinematics: Implement a forward kinematics node for this arm that**
   - **Subscribes to the joint values topic and reads them from the robot's joint states topic.**
   - **Calculate the end effector pose**
   - **publishes the pose as a ROS topic (you can choose to do this inside the callback function that reads the joint values, so that each time you receive a new joint value from the robot, you publish the end effector pose)**

   This is a publisher-subscriber implementation. Move the robot to 3 different locations by sending position references to the joints. For each pose provide the screenshot of the robot and print the resulting pose to the terminal by using "ros2 topic echo..." command and include the screenshot of it in your report.
   The robot is shown below with frames assigned to the ground and to each joint.

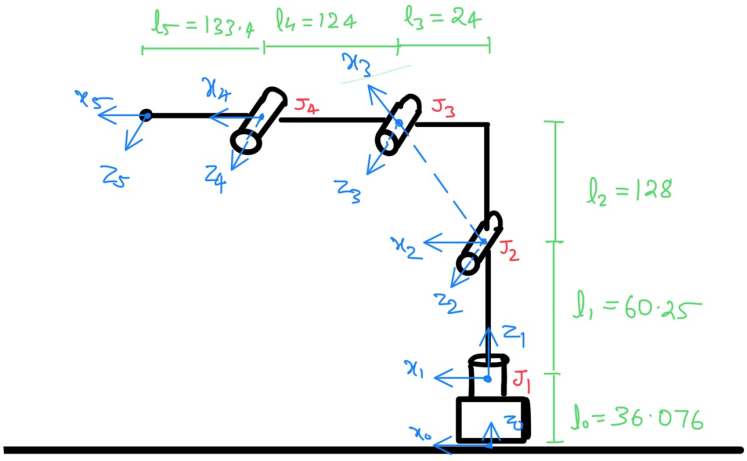


Figure 1: Robot

The DH-parameter table is as follows:

	a	θ	d	α
Link1	0	q_1	l_1	-90
Link2	$\sqrt{l_2^2 + l_3^2}$	$-\arctan\left(\frac{l_2}{l_3}\right) + q_2$	0	0
Link3	l_4	$\arctan\left(\frac{l_2}{l_3}\right) + q_3$	0	0
Link4	l_5	q_4	0	0

Table 1: DH parameters

The Homogeneous Transformation matrix of a frame wrt to the previous frame is:

$$A_i = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where i ∈ [2,5] and

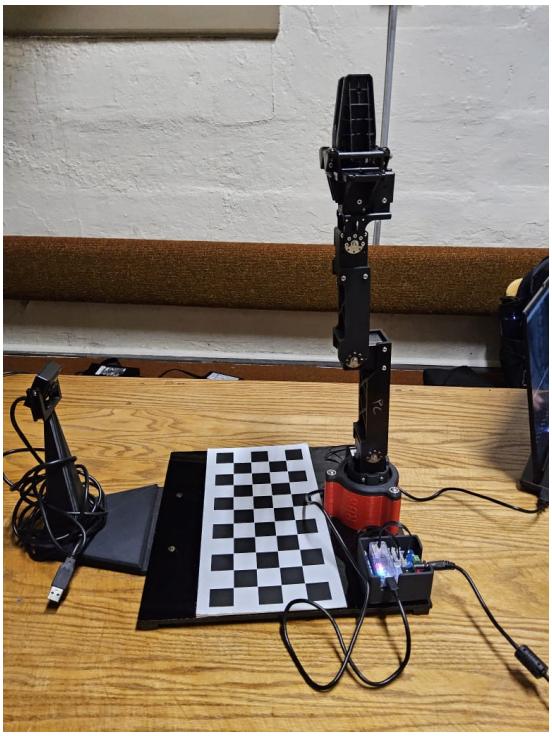
$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since frame 1 is not rotated wrt frame 0, but translated in z_0-direction. The end-effector pose can be calculated as:

$$H_5^0 = A_1 A_2 A_3 A_4 A_5$$

This is implemented in ROS2.

The robot is moved to 3 different positions and the resulted pose and robot configuration is shown below in Figures 2, 3 and 4



(a) Robot Configuration

```
---
position:
  x: 26.556968273270734
  y: -0.04073791220277143
  z: 481.6425908543748
orientation:
  x: 0.5022956620749799
  y: 0.49769374906729297
  z: 0.49846367512124823
  w: -0.5015316187286887
```

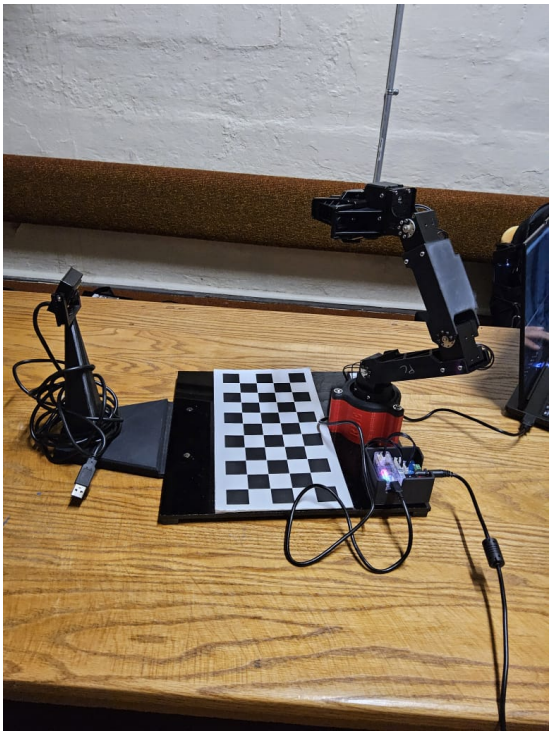
(b) Terminal Pose output

Figure 2: Robot Pose at position 1

```
---
position:
  x: -136.0315406920791
  y: 0.2086699394484568
  z: 368.7771820505268
orientation:
  x: 0.221291421622628
  y: 0.6715877505704196
  z: 0.6719264170805609
  w: -0.22026095893117345
```

(a) Robot Configuration (b) Terminal Pose output

Figure 3: Robot Pose at position 2

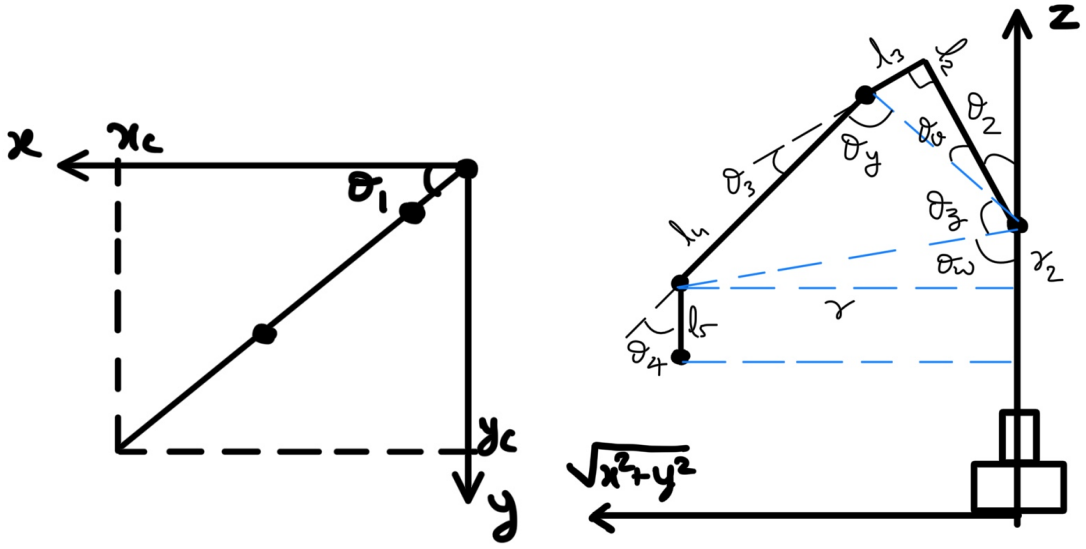


```
---
position:
  x: 57.33674562259804
  y: -89.32722935877747
  z: 287.98237651793676
orientation:
  x: -0.618951361577307
  y: 0.34190526758387
  z: -0.3361941117820062
  w: 0.622071956612021
```

(a) Robot Configuration (b) Terminal Pose output

Figure 4: Robot Pose at position 3

2. **Implement an inverse kinematics node (a separate node) that has a service client that takes a (desired) pose of the end effector from the user and returns joint positions as a response.**
This is a service server-service client implementation.
Test your node with the "ros2 service call" command for three separate end effector positions. Take the screen shot together of the terminal responses and include it to your report.
In Figure 5 the robot is seen from two angles such that complete information of the pose of the robot is obtainable:



(a) Top View (b) Side View

Figure 5: Robot Top-View and Side-View

Here we have 4 joints in a 3D space, so we can have more than one joint configuration, for the same end-effector pose.
Hence we constrain the gripper to be facing down always, so that we can have one-to-one mapping of the end-effector pose and joint angles. From Figure 5, we can calculate the joint angle θ_1 as:

$$\theta_1 = \arctan\left(\frac{y_c}{x_c}\right)$$

The distance of the end-effector from the origin in the X-Y plane is

$$r = \sqrt{x_c^2 + y_c^2}$$

The height difference between joint 2 and joint 4 is:

$$r_2 = l_0 + l_1 - (z_c + l_5)$$

We can calculate θ_y from the equation:

$$\sqrt{r^2 + r_2^2} = \sqrt{\left(\sqrt{l_2^2 + l_3^2}\right)^2 + l_4^2 - 2l_4\sqrt{l_2^2 + l_3^2}cos\theta_y}$$

$$cos\theta_y = \frac{l_2^2 + l_3^2 + l_4^2 - r^2 - r_2^2}{2l_4\sqrt{l_2^2 + l_3^2}} = D$$

$$sin\theta_y = \sqrt{1 - D^2}$$

$$\theta_y = \arctan\left(\frac{\sqrt{1 - D^2}}{D}\right)$$

$$\theta_v = \arctan\left(\frac{l_3}{l_2}\right)$$

Also we can see geometrically that,

$$\theta_3 + \theta_y = \frac{\pi}{2} + \theta_v$$

Thus the value of θ_3 can found as:

$$\theta_3 = \frac{\pi}{2} + \theta_v - \theta_y$$

Similarly we can find θ_z:

$$l_4^2 = r^2 + r_2^2 + l_2^2 + l_3^2 - 2\sqrt{r^2 + r_2^2}\sqrt{l_2^2 + l_3^2}cos\theta_z$$

$$cos\theta_z = \frac{r^2 + r_2^2 + l_2^2 + l_3^2 - l_4^2}{2\sqrt{r^2 + r_2^2}\sqrt{l_2^2 + l_3^2}} = D_2$$

$$sin\theta_z = \sqrt{1 - D_2^2}$$

The value of θ_z is:

$$\theta_z = \arctan\left(\frac{\sqrt{1 - D_2^2}}{D_2}\right)$$

Using Pythagoras Theorem we can find θ_w:

$$\theta_w = \arctan\left(\frac{r}{r_2}\right)$$

From geometry we have:

$$\theta_2 + \theta_v + \theta_z + \theta_w = \pi$$

Since we have calculated all other values, we can find the value of θ_2:

$$\theta_2 = \pi - \theta_v - \theta_z - \theta_w$$

Finally, we calculate θ_4

$$\theta_4 = \frac{\pi}{2} - \theta_2 - \theta_3$$

The Inverse Kinematics node is tested with three different inputs using the **ros2 service call** command and the terminal outputs are shown below in Figures 6, 7 and 8,

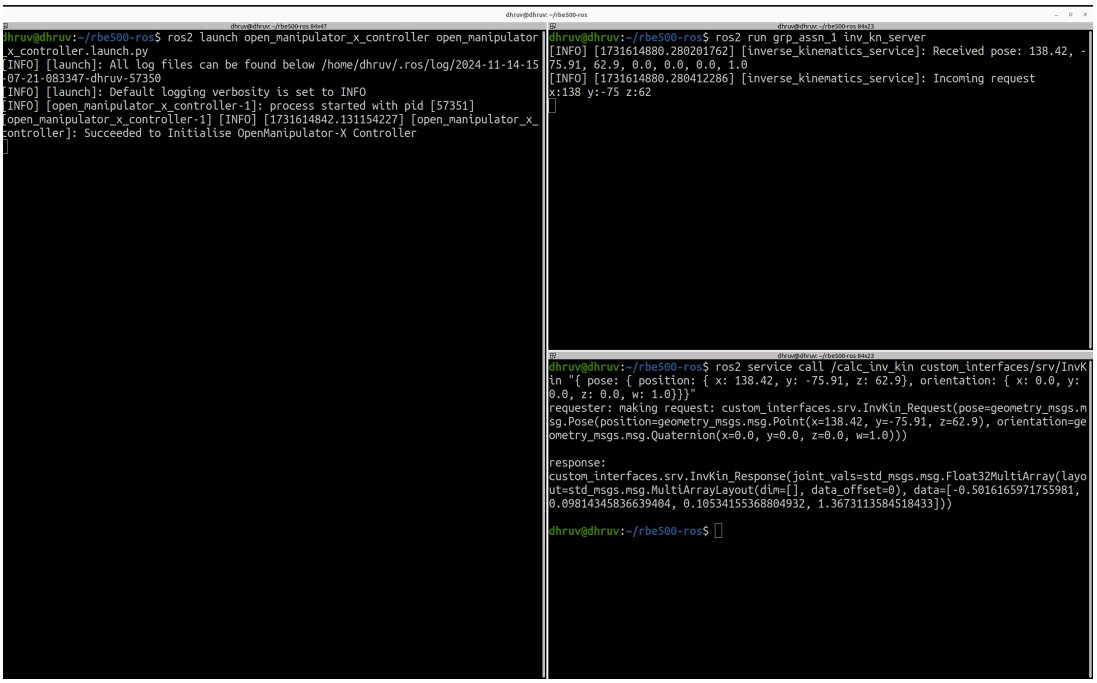


Figure 6: Terminal Output

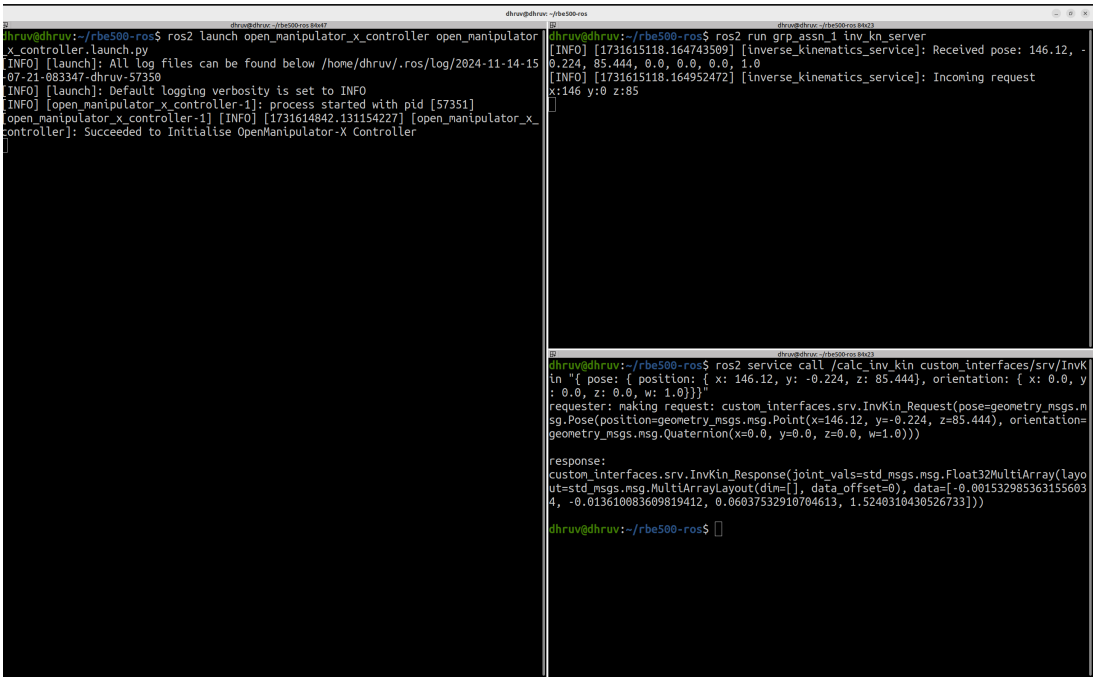


Figure 7: Terminal Output

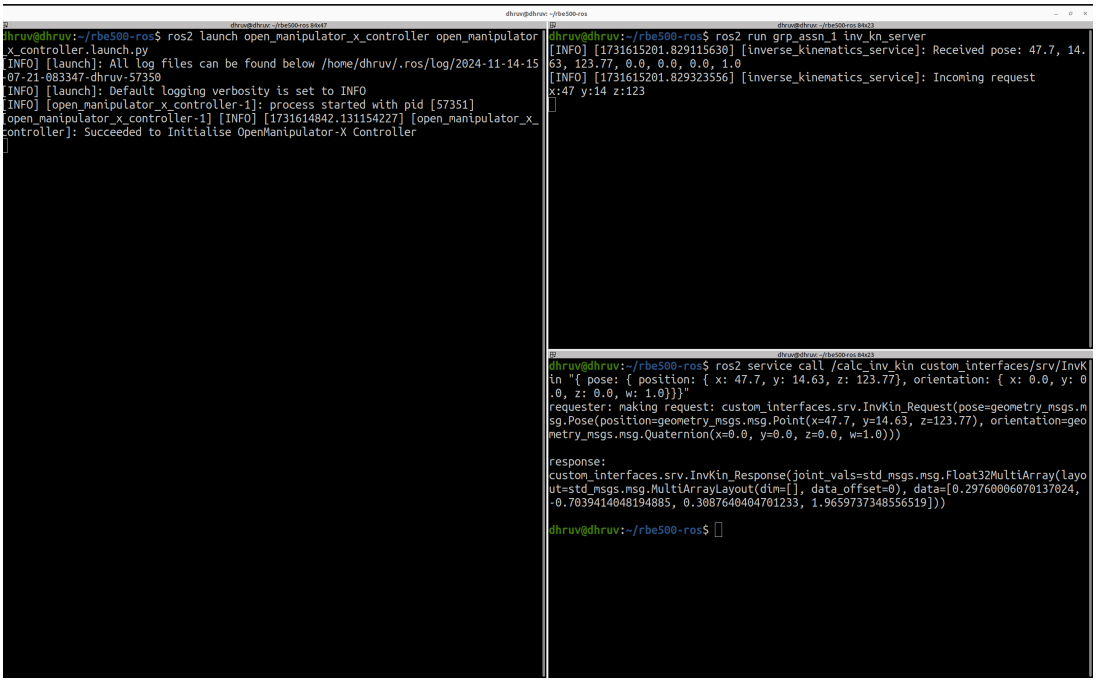


Figure 8: Terminal Output

3. **BONUS: Place an object at a particular (known) location on the chess board. Calculate inverse kinematics for that gripper pose to find the point angles. Move the robot to that position, pick the object and lift the object. Hint: For picking and lifting you need an intermediate position for the gripper right above the object. The gripper should go to that intermediate position and then go down to pick the object and then go up to the intermediate position to lift the object.**

   This has been implemented on the robot whose video is attached with the submission files.
   Code has also been attached in zip format.