RBE 500 - HW 7: Forward and Inverse Kinematics for OpenManipulatorX Arm

Objective

The objective is to implement forward and inverse kinematics for the OpenManipulatorX robotic arm:

1. Computing the forward kinematics to determine the end-effector's pose given the joint angles.
2. Solving the inverse kinematics to compute the joint angles required to achieve a desired end-effector pose.

Bonus Objective

Understanding the kinematics of OpenManipulatorX Arm to pick an object off the the ground.

Forward Kinematics Implementation

Methodology

Forward kinematics was implemented using the Denavit-Hartenberg (D-H) parameters. The homogeneous transformation matrices between consecutive coordinate frames were computed and multiplied to determine the end-effector's position and orientation relative to the base frame.

Denavit-Hartenberg Parameters

Three key adjustments were made to account for the offset between joints 2 and 3. The following D-H parameters were used:

Functions Used

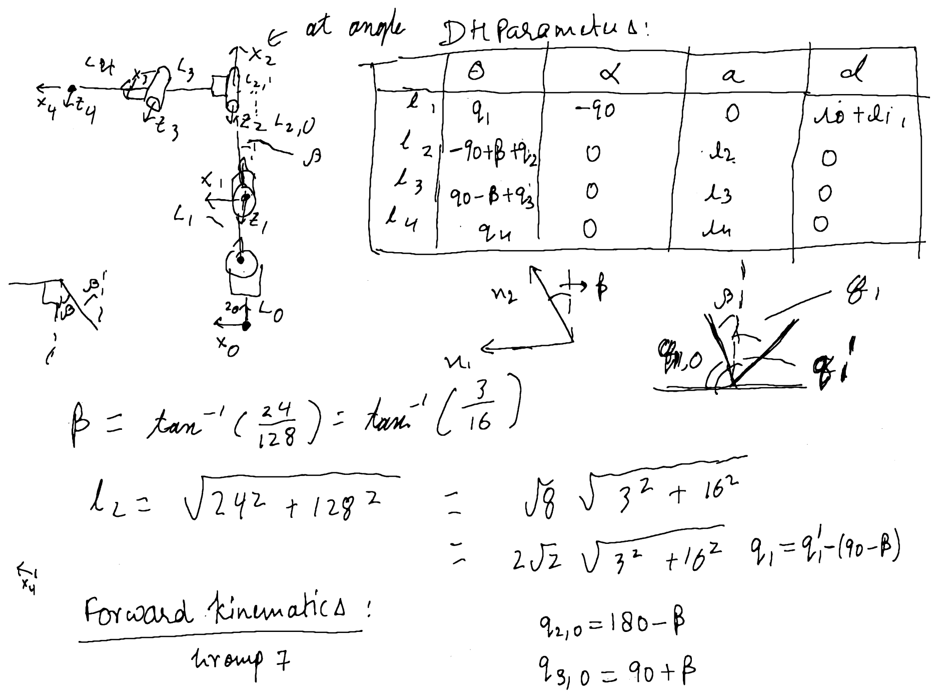
- `A(theta, alpha, a, d)`: Computes the homogeneous transformation matrix using D-H parameters.
- `forward(q1, q2, q3, q4)`: Computes the overall transformation from the base frame to the end-effector frame.

These results indicate the end-effector's pose in the base frame with a position of approximately (Px, Py, Pz).

Inverse Kinematics Implementation

Methodology

Inverse kinematics were derived by solving for joint angles $(q1, q2, q3, q4)$ that achieve a specified end-effector position (x, y, z) and orientation (θ). The approach utilized trigonometric relations and geometric considerations of the manipulator's link lengths and configurations.



DH Parameters:

	θ	α	a	d
l_1	q_1	-90	0	$l_0 + l_1$
l_2	$-90+\beta+q_2$	0	l_2	0
l_3	$90-\beta+q_3$	0	l_3	0
l_4	q_4	0	l_4	0

$$\beta = \tan^{-1}\left(\frac{24}{128}\right) = \tan^{-1}\left(\frac{3}{16}\right)$$

$$l_2 = \sqrt{24^2 + 128^2} = \sqrt{8}\sqrt{3^2+16^2} = 2\sqrt{2}\sqrt{3^2+16^2}$$

$$q_1 = q_1' - (90-\beta)$$

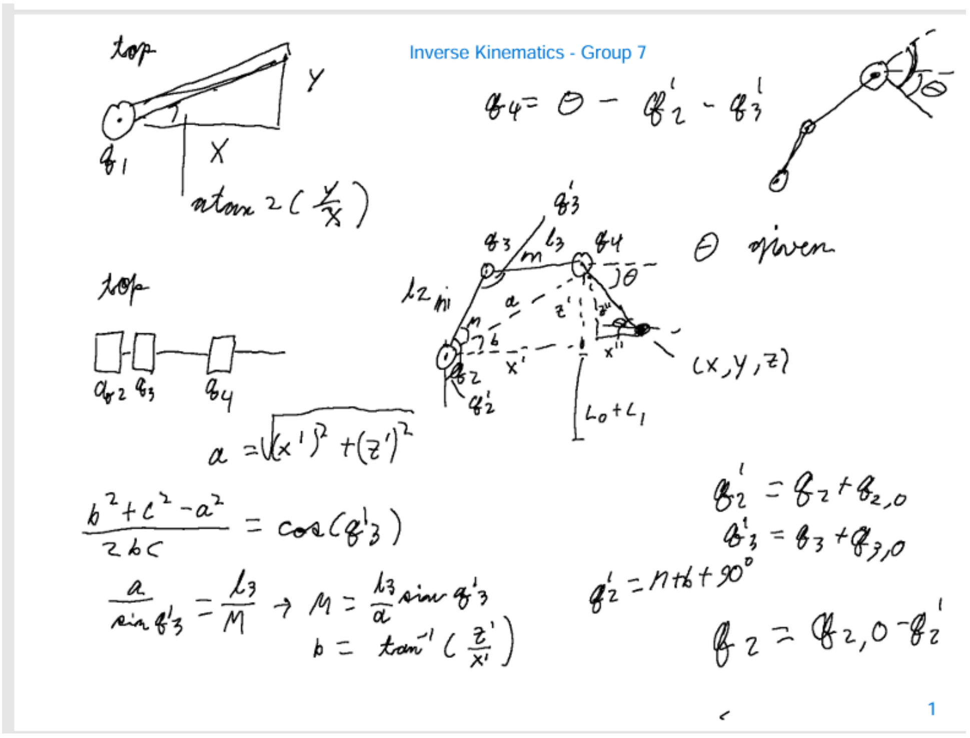
$$q_{2,0} = 180 - \beta$$

$$q_{3,0} = 90 + \beta$$

Forward Kinematics:

Group 7

Figure 1: Detailing the Forward Kinematics


Inverse Kinematics - Group 7

$$\mathrm{atan2}\left(\frac{Y}{X}\right)$$

$$q_4 = \theta - q_2' - q_3'$$

$$a = \sqrt{(x')^2 + (z')^2}$$

$$\frac{b^2 + c^2 - a^2}{2bc} = \cos(q_3')$$

$$\frac{a}{\sin q_3'} = \frac{l_3}{M} \rightarrow M = \frac{l_3}{a}\sin q_3'$$

$$b = \tan^{-1}\left(\frac{z'}{x'}\right)$$

$$q_2' = q_2 + q_{2,0}$$

$$q_3' = q_3 + q_{3,0}$$

$$q_2' = n + b + 90^\circ$$

$$q_2 = q_{2,0} - q_2'$$

1

Figure 2: Inverse Kinematics Process.

Code Integration with ROS 2

Forward Kinematics Node

- Subscribed to joint state messages (sensor$_m sgs/JointState$).

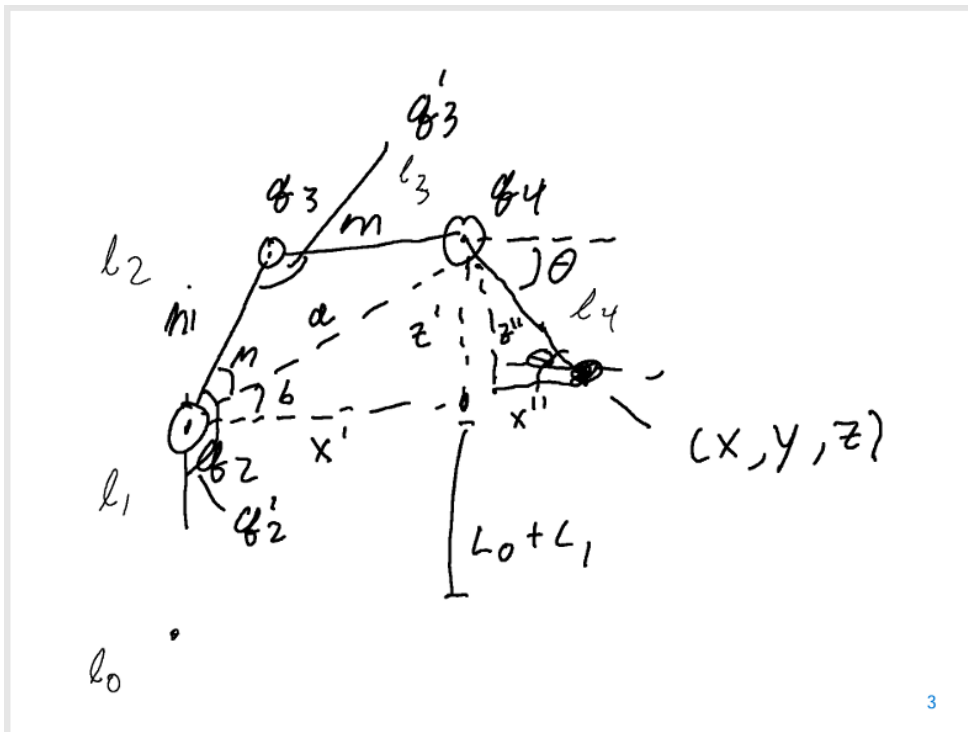
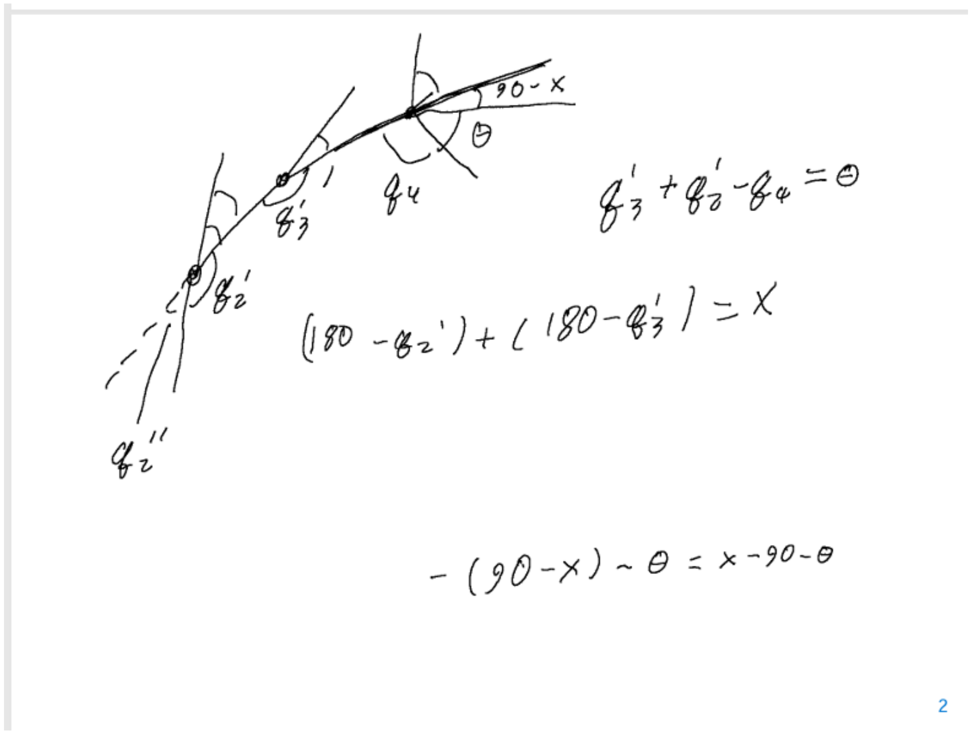


Figure 3: Inverse Kinematics Process.

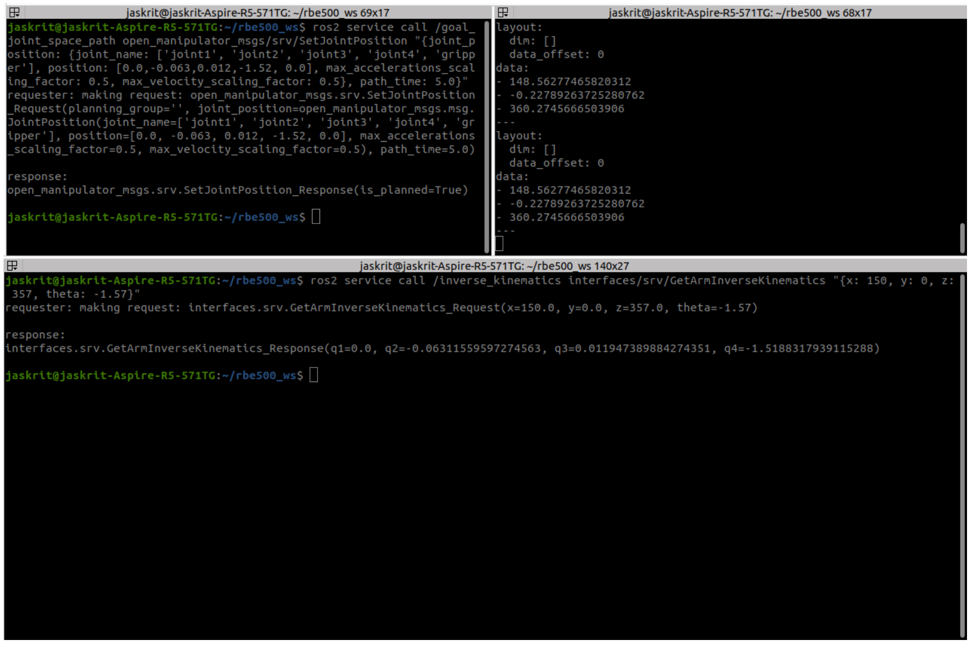


$90 - x$
θ
q_4
q_3'
q_2'
q_2''
$q_3' + q_2' - q_4 = \theta$
$(180 - q_2') + (180 - q_3') = x$
$-(90 - x) - \theta = x - 90 - \theta$
2

Figure 4: Inverse Kinematics Process.

Inverse Kinematics Node

- Implemented as a ROS service server.
- Took the desired end-effector pose as input and returned joint angles.



jaskrit@jaskrit-Aspire-R5-571TG: ~/rbe500_ws 69x17

```
jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$ ros2 service call /goal_
joint_space_path open_manipulator_msgs/srv/SetJointPosition "{joint_p
osition: {joint_name: ['joint1', 'joint2', 'joint3', 'joint4', 'gripp
er'], position: [0.0,-0.063,0.012,-1.52, 0.0], max_accelerations_scal
ing_factor: 0.5, max_velocity_scaling_factor: 0.5}, path_time: 5.0}"
requester: making request: open_manipulator_msgs.srv.SetJointPosition
_Request(planning_group='', joint_position=open_manipulator_msgs.msg.
JointPosition(joint_name=['joint1', 'joint2', 'joint3', 'joint4', 'gr
ipper'], position=[0.0, -0.063, 0.012, -1.52, 0.0], max_accelerations
_scaling_factor=0.5, max_velocity_scaling_factor=0.5), path_time=5.0)

response:
open_manipulator_msgs.srv.SetJointPosition_Response(is_planned=True)

jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$ 
```

jaskrit@jaskrit-Aspire-R5-571TG: ~/rbe500_ws 68x17

```
layout:
  dim: []
  data_offset: 0
data:
- 148.56277465820312
- -0.22789263725280762
- 360.2745666503906
---
layout:
  dim: []
  data_offset: 0
data:
- 148.56277465820312
- -0.22789263725280762
- 360.2745666503906
---
```

jaskrit@jaskrit-Aspire-R5-571TG: ~/rbe500_ws 140x27

```
jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$ ros2 service call /inverse_kinematics interfaces/srv/GetArmInverseKinematics "{x: 150, y: 0, z:
 357, theta: -1.57}"
requester: making request: interfaces.srv.GetArmInverseKinematics_Request(x=150.0, y=0.0, z=357.0, theta=-1.57)

response:
interfaces.srv.GetArmInverseKinematics_Response(q1=0.0, q2=-0.06311559597274563, q3=0.011947389884274351, q4=-1.5188317939115288)

jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$ 
```

Figure 5: Screenshot of pos1.

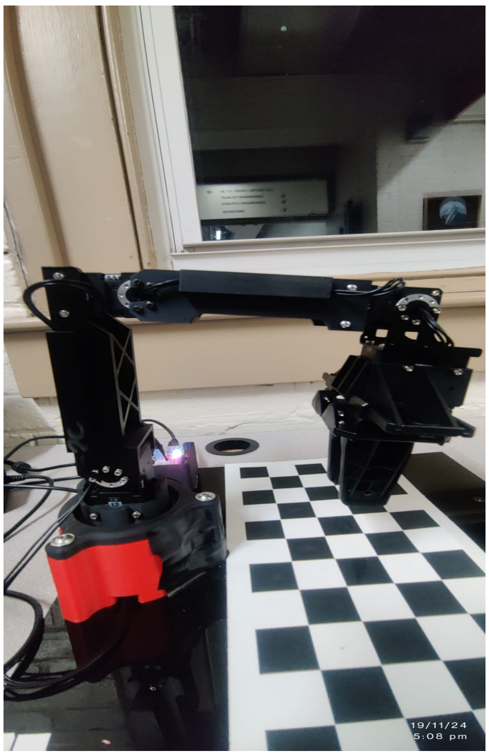
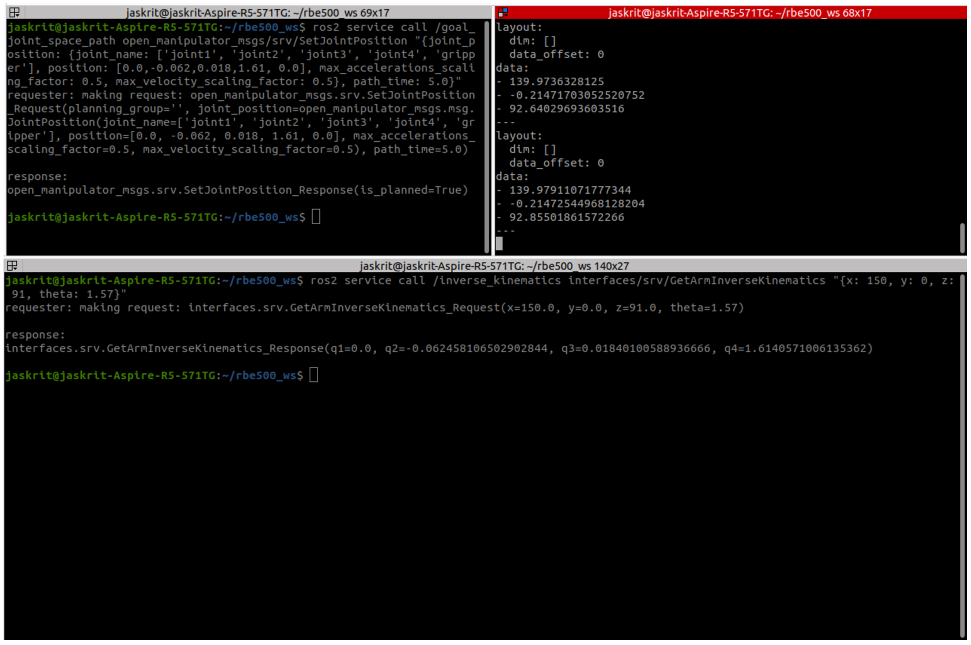


Figure 6: Robot pos1.

Testing

- **Forward Kinematics:** Published joint angles and verified the computed pose using `ros2 topic echo`.
- **Inverse Kinematics:** Provided end-effector poses using `ros2 service call` and validated the returned joint angles.


jaskrit@jaskrit-Aspire-R5-571TG: ~/rbe500_ws 69x17
jaskrit@jaskrit-Aspire-R5-571TG: ~/rbe500_ws 68x17
jaskrit@jaskrit-Aspire-R5-571TG: ~/rbe500_ws 140x27

```
jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$ ros2 service call /goal_
joint_space_path open_manipulator_msgs/srv/SetJointPosition "{joint_p
osition: {joint_name: ['joint1', 'joint2', 'joint3', 'joint4', 'gripp
er'], position: [0.0,-0.062,0.018,1.61, 0.0], max_accelerations_scali
ng_factor: 0.5, max_velocity_scaling_factor: 0.5}, path_time: 5.0}"
requester: making request: open_manipulator_msgs.srv.SetJointPosition
_Request(planning_group='', joint_position=open_manipulator_msgs.msg.
JointPosition(joint_name=['joint1', 'joint2', 'joint3', 'joint4', 'gr
ipper'], position=[0.0, -0.062, 0.018, 1.61, 0.0], max_accelerations_
scaling_factor=0.5, max_velocity_scaling_factor=0.5), path_time=5.0)

response:
open_manipulator_msgs.srv.SetJointPosition_Response(is_planned=True)

jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$
```

```
layout:
  dim: []
  data_offset: 0
data:
- 139.9736328125
- -0.21471703052520752
- 92.64029693603516
---
layout:
  dim: []
  data_offset: 0
data:
- 139.97911071777344
- -0.21472544968128204
- 92.85501861572266
---
```

```
jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$ ros2 service call /inverse_kinematics interfaces/srv/GetArmInverseKinematics "{x: 150, y: 0, z:
 91, theta: 1.57}"
requester: making request: interfaces.srv.GetArmInverseKinematics_Request(x=150.0, y=0.0, z=91.0, theta=1.57)

response:
interfaces.srv.GetArmInverseKinematics_Response(q1=0.0, q2=-0.062458106502902844, q3=0.01840100588936666, q4=1.6140571006135362)

jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$
```

Figure 7: Screenshot of pos2.

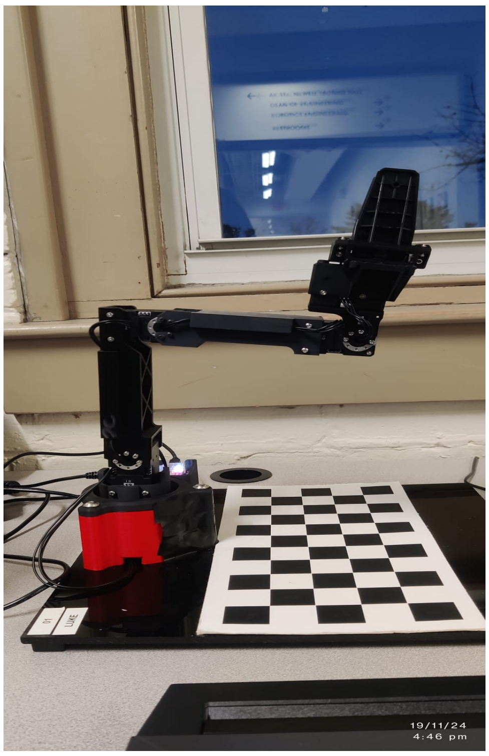
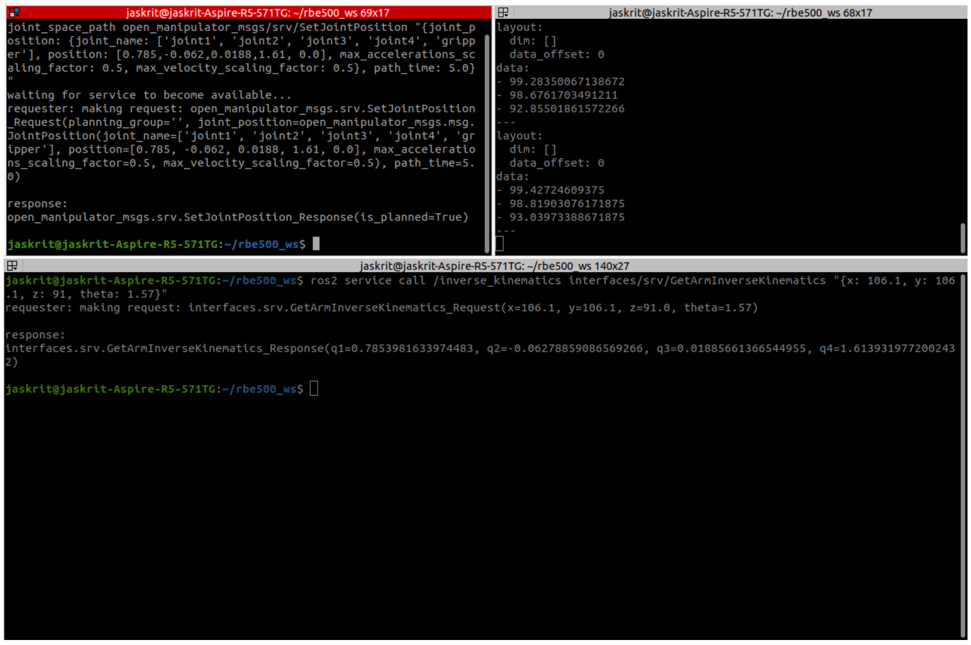


Figure 8: Robot position 2.

Description:

In the above screenshots, we provide a specific end-effector position as input to our inverse kinematics service, which returns the joint values q1,q2,q3 and q4.In the top-left section, the service is called with the goal joint state path defined by q1,q2,q3 q4.On the upper-right, the resulting end-effector position is verified using forward kinematics.



```
jaskrit@jaskrit-Aspire-R5-571TG: ~/rbe500_ws 69x17
joint_space_path open_manipulator_msgs/srv/SetJointPosition "{joint_p
osition: {joint_name: ['joint1', 'joint2', 'joint3', 'joint4', 'gripp
er'], position: [0.785,-0.062,0.0188,1.61, 0.0], max_accelerations_sc
aling_factor: 0.5, max_velocity_scaling_factor: 0.5}, path_time: 5.0}
"
waiting for service to become available...
requester: making request: open_manipulator_msgs.srv.SetJointPosition
_Request(planning_group='', joint_position=open_manipulator_msgs.msg.
JointPosition(joint_name=['joint1', 'joint2', 'joint3', 'joint4', 'gr
ipper'], position=[0.785, -0.062, 0.0188, 1.61, 0.0], max_acceleratio
ns_scaling_factor=0.5, max_velocity_scaling_factor=0.5), path_time=5.
0)

response:
open_manipulator_msgs.srv.SetJointPosition_Response(is_planned=True)

jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$
```

```
jaskrit@jaskrit-Aspire-R5-571TG: ~/rbe500_ws 68x17
layout:
  dim: []
  data_offset: 0
data:
- 99.28350067138672
- 98.6761703491211
- 92.85501861572266
---
layout:
  dim: []
  data_offset: 0
data:
- 99.42724609375
- 98.81903076171875
- 93.03973388671875
---
```

```
jaskrit@jaskrit-Aspire-R5-571TG: ~/rbe500_ws 140x27
jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$ ros2 service call /inverse_kinematics interfaces/srv/GetArmInverseKinematics "{x: 106.1, y: 106
.1, z: 91, theta: 1.57}"
requester: making request: interfaces.srv.GetArmInverseKinematics_Request(x=106.1, y=106.1, z=91.0, theta=1.57)

response:
interfaces.srv.GetArmInverseKinematics_Response(q1=0.7853981633974483, q2=-0.06278859086569266, q3=0.01885661366544955, q4=1.613931977200243
2)

jaskrit@jaskrit-Aspire-R5-571TG:~/rbe500_ws$
```

Figure 9: Screenshot of pos3.

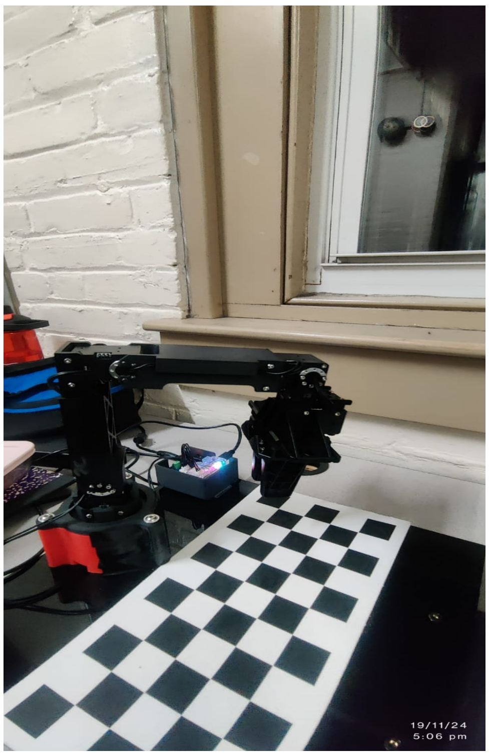


Figure 10: Robot position 3.

Observations

1. The inverse kinematics solution successfully calculated joint angles for given poses.
2. Further experimentation with (θ) provided to `inverse()` resulted in successful pick-and-place operations for the object.

Conclusion

This assignment successfully demonstrates the development and testing of forward and inverse kinematics for the OpenManipulatorX robotic arm. The results confirm the accuracy of the implemented algorithms under the tested scenarios.