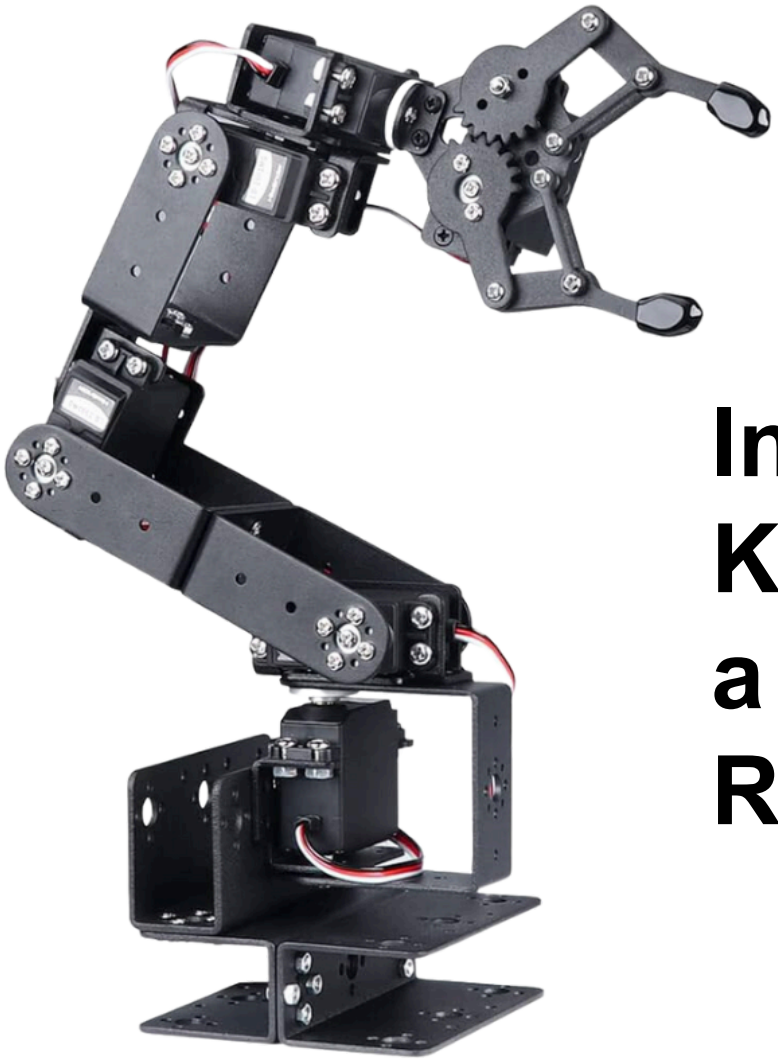


Inverse Kinematics of a 6-DOF Rot3u Robotic Arm

The Cartesian Choreographers Team Members:

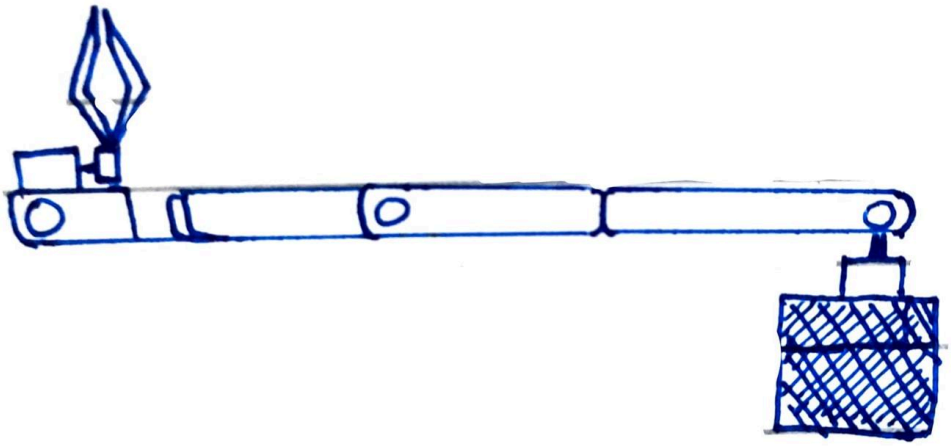
Name	ID
Jaysh Muhammad	450074
Syed Furqan Ali	451802
Zain Ul Abdeen	450682
Rana Talal Ahmad Khan	450436

1. Introduction

This report details the reverse kinematics analysis performed on a 6-DOF Rot3u robot. Reverse kinematics determines the angles of all the joints when given a specific position or orientation of the end-effector. This is particularly important for tasks like path planning, object manipulation, and control.

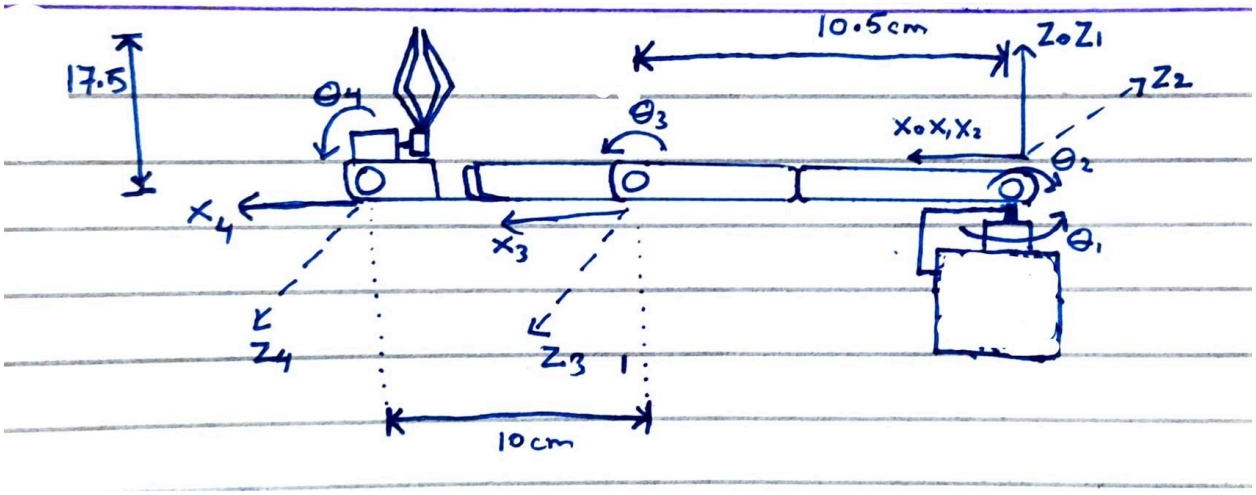
2. Initial Configuration

The initial configuration of the robot, including the joint angles (θ values = 0) for each joint, is specified. This serves as the starting point for the reverse kinematics calculations.



3. Frame assignment (Frames PDF)

New frame assignment was done to increase the WorkSpace.



4. WorkSpace

The Work Space is limited from 10 cm radius around robot base to the 37 cm maximum link length from +37X to -37X. And in Z axis it is limited to +37 Zto -11Z (Table in the Way)

5. Problem Decomposition:

The overall problem is divided into two key sections:

X-Z Plane (3-Link Arm): This section deals with the movement of the three-link arm within the X-Z plane.

X-Y Plane (Turntable): This section handles the rotation of the turntable base in the X-Y plane.

Solution Approach:

The solution for the 3-link arm is broken down further into two stages:

- **Two-Link Arm + End Effector Pose:** The first stage involves calculating the joint angles for the first two links of the arm to position the end effector at the desired location in the X-Z plane.
- **Final Joint Angle:** The final joint angle of the third link is then determined to orient the end effector appropriately.

Inverse Kinematics Algorithm:

The inverse kinematics calculations are performed in Python using both algebraic and iterative methods. The script first determines all possible joint angle solutions using inverse kinematics equations. This results in an array of potential solutions.

Solution Validation and Optimization:

The following steps are taken to validate and select the best solution rom the calculated array:

a. **Invalid Solution Removal:** Solutions with joint angles outside the valid range (0-180 degrees) are discarded, as this is the maximum movement range of the robot's servos.
b. **Closest Solution Selection:** From the remaining valid solutions, the solution requiring the least overall joint angle movement is selected. This minimizes the distance the robot needs to move to reach the target position.

c. **Forward Kinematics Verification**: The selected joint angle solution is then fed into a forward kinematics algorithm to verify that it accurately positions the end effector at the desired coordinates.

6. Calculations for Inverse Kinematics.

Explanation:
First the theta0 is calculated which is the angle for the turntable (X-Z plane) . Then to calculate the other angle of the 3 links of the ARM first we project the Px on to the X,Y plane then we calculate the theta1-3 of the arm,

Px,Py,Pz are the desired Coordinates,
Wx,Wz are the Wrist Coordinates in X-Z plane
C_1 = cos(theta1) , C_2 = cos(theta2) , C_{12} = cos(theta1+theta2), C_{123} = cos(theta1+theta2+theta3)
S_1 = sin(theta1) , S_2 = sin(theta2) , S_{12} = sin(theta1+theta2), S_{123} = sin(theta1+theta2+theta3)
a_1 , a_2, a_3 Are the Link Lengths

Given will Be: P_x, P_y, P_z

$$p_x = a_1c_1 + a_2c_{12} + a_3c_{123}$$
$$p_z = a_1s_1 + a_2s_{12} + a_3s_{123}$$

$$\theta_0 = atan2(P_y, P_z)$$

$$px_{rotated} = p_x \cos(\theta_0) + p_z \cos(\theta_0)$$
$$px = px_{rotated}$$

Loop: $-180° \leq \phi_d \leq 180°$: ϕ+= 10° :

$$\phi = \phi_d \times \frac{\pi}{180}$$

$$p_x = w_x + a_3 \cos(\phi)$$
$$p_z = w_z + a_3 \sin(\phi)$$

$$w_x = p_x - a_3 \cos(\phi)$$
$$w_z = p_z + a_3 \sin(\phi)$$

$$c_2 = \frac{w_x^2 + w_z^2 - a_1^2 - a_2^2}{2a_1a_2}$$

$$s_2 = +\sqrt{1 - C_2^2} \;\; OR \;\; s_2 = -\sqrt{1 - C_2^2}$$

Elbow Up and Down Solutions, so there will be two next angles, based on what Solution You go for. Elbow up or down.

$$\theta_2 = atan2(s_2, c_2)$$

$$s_1 = \frac{w_z(a_1 + a_2c_2) - a_2s_2 \times w_x}{a_1^2 + a_2^2 + 2a_1a_2c_2}$$

$$c_1 = \frac{w_x(a_1 + a_2c_2) - a_2s_2 \times w_z}{a_1^2 + a_2^2 + 2a_1a_2c_2}$$

$$\theta_1 = atan2(s_1, c_1)$$

$$\theta_3 = \phi - \theta_1 - \theta_2$$

6. Code for Inverse Kinematics

The code for the inverse kinematics is provided in the github repository. Links of which are attached below.
Code Link: Github Repository - Inverse Kinematics Python
Code Link: Github Repository - Inverse Kinematics MatLab

7. Reference

References and Special thanks

- Gemini
- Inverse Kinematic Three link Plainer ARM