Department of Electronic & Telecommunication Engineering,
University of Moratuwa, Sri Lanka.

Robotics Mini Project
Kinematic Analysis of a Robot Arm

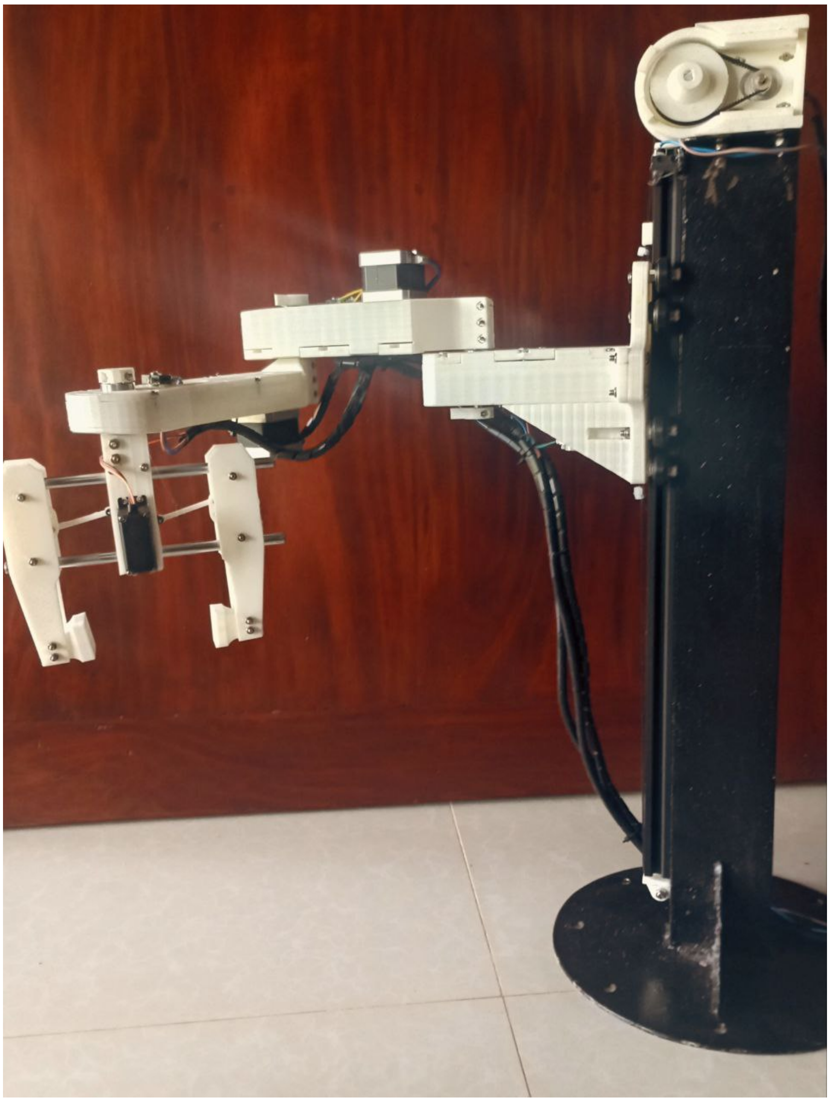
Maduwantha L.H.H 200370K

Summarizing the analysis - Report
In21 S5 EN3563 - Robotics

08th December 2024

Introduction

This mini-project focuses on the kinematic analysis of a SCARA robot with four degrees of freedom (DoF). The robot has a prismatic joint (for vertical movement) and three revolute joints (for horizontal rotation), allowing it to perform pick-and-place tasks. A functional gripper is included to handle round and square objects.



Analysis

The analysis includes:

- Deriving Denavit-Hartenberg (DH) parameters.
- Calculating forward and inverse kinematics.
- Deriving the Jacobian matrix.

Control and Payload

The robot is controlled using an Arduino Uno and stepper motors via a CNC shield. It has a maximum payload of 0.8 kg.

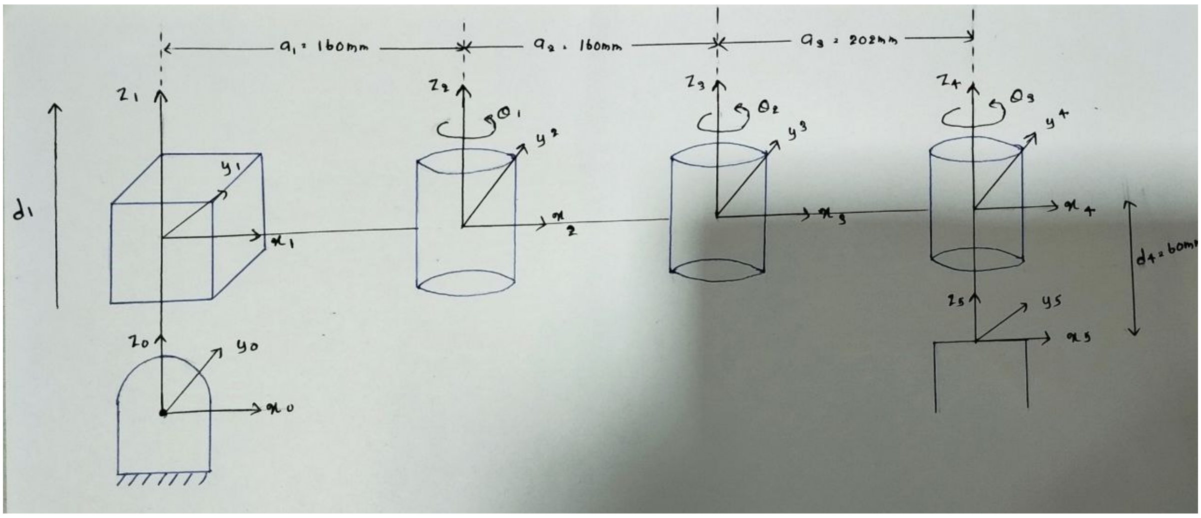
Robot Design and DH Parameters

Robot Configuration

The robot arm consists of the following links and joints:

- $a_1 = 160\,\text{mm}$, $a_2 = 160\,\text{mm}$, $a_3 = 202\,\text{mm}$: These are the link lengths between the revolute joints.
- d_1 (variable): Vertical movement of the prismatic joint.
- $d_4 = 60\,\text{mm}$: Constant offset for the final joint.
- $\theta_1, \theta_2, \theta_3$: The joint angles that control the robot's movement in the horizontal plane.

Scara Robot - Coordinate Frame assignment



Denavit-Hartenberg Parameters

The Denavit-Hartenberg (DH) parameters describe the geometry of the robot arm through a systematic convention. The table below shows the DH parameters for the SCARA robot:

Joint (i)	$\mathbf{a_i}$ (mm)	$\alpha_\mathbf{i}$	$\mathbf{d_i}$ (mm)	$\theta_\mathbf{i}$
1 (Prismatic)	0	0	d_1 (variable)	0
2 (Revolute)	160	0	0	θ_1 (variable)
3 (Revolute)	160	0	0	θ_2 (variable)
4 (Revolute)	202	0	0	θ_3 (variable)
5	0	0	60	0

Table 1: DH Parameters for the SCARA Robot

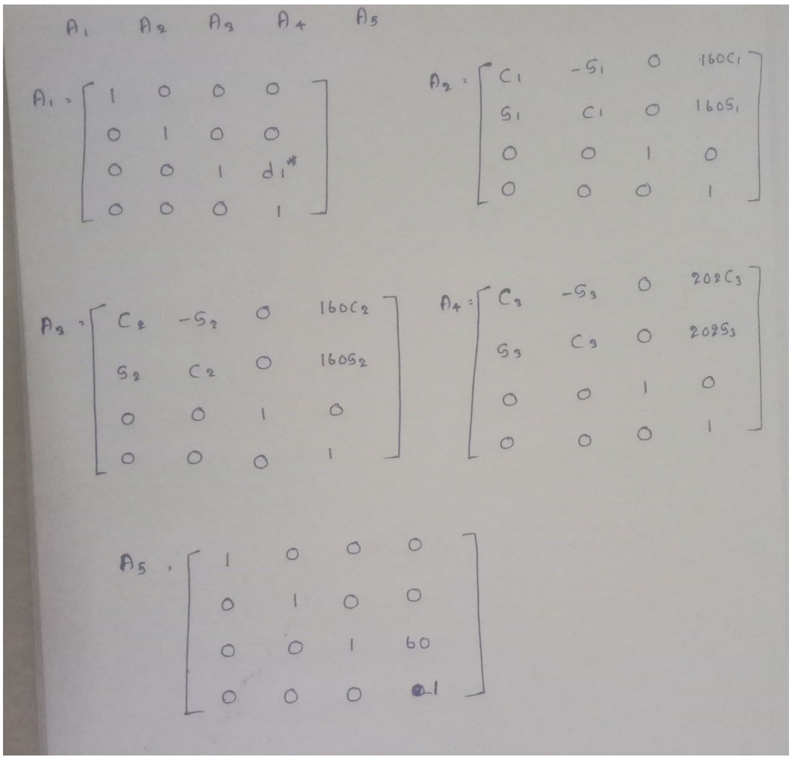
This DH table is crucial for calculating the forward and inverse kinematics.

Forward Kinematics

Forward kinematics calculates the position and orientation of the end-effector based on the joint angles and link lengths. Using Denavit-Hartenberg (DH) parameters, we define the transformation matrix for each joint. The transformation matrix is:

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

By multiplying the individual transformation matrices from base to end-effector, we get the final transformation matrix T_0^4. This matrix provides the position (x, y, z) and the orientation of the end-effector. The angles θ_1, θ_2, and θ_3 are used to determine the robot's orientation in space.



End-Effector Position

Using the link lengths and joint angles, the position of the end-effector is calculated as:

$$x = a_1\sin(90°) + a_2\cos\theta_1 + a_3\cos(\theta_1 + \theta_2)$$

$$x = 160 + 160\cos\theta_1 + 202\cos(\theta_1 + \theta_2)$$

$$y = a_1\cos(90°) + a_2\sin\theta_1 + a_3\sin(\theta_1 + \theta_2)$$

$$y = 0 + 160\sin\theta_1 + 202\sin(\theta_1 + \theta_2)$$

$$z = z_0 + d_1$$

Inverse Kinematics

Inverse kinematics is the process of solving for the joint variables $(\theta_1, \theta_2, \theta_3, d_0)$ given a desired position and orientation of the end-effector. For the SCARA robot, the inverse kinematics is separated into two primary steps:

Step 1: Solve for $\theta_1, \theta_2, \theta_3$

Using the planar geometry of the robot, we can calculate the joint angles. The horizontal position of the end-effector is given by:

$$x = a_1 + a_2 \cos(\theta_1) + a_3 \cos(\theta_1 + \theta_2)$$

$$y = a_2 \sin(\theta_1) + a_3 \sin(\theta_1 + \theta_2)$$

The law of cosines is applied to solve for θ_2 and θ_3 iteratively:

$$\cos \theta_2 = \frac{r^2 - a_2^2 - a_3^2}{2a_3 a_2}$$

Once θ_2 is computed, θ_1 is found by solving for the orientation of the base using inverse trigonometry:

$$\theta_1 = \arctan\left(\frac{y}{x}\right) - \arctan\left(\frac{a_3 \sin \theta_2}{a_2 + a_3 \cos \theta_2}\right)$$

Finally, θ_3 is derived by using the desired end-effector orientation.

Step 2: Solve for d_0

The vertical position is determined by the prismatic joint:

$$d_1 = z - d_4$$

Jacobian Matrix

The Jacobian matrix relates joint velocities to end-effector velocities. It is essential for analyzing the robot's motion and controlling its speed and acceleration. For a PRRR manipulator, the Jacobian matrix has two parts:

- **Linear Velocity:** The part of the Jacobian that maps joint velocities to linear velocities of the end-effector. It involves the cross product between the joint axes and the position vector.
- **Angular Velocity:** The part that maps joint velocities to angular velocities.

The Jacobian for the SCARA robot is given by:

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}$$

Where:

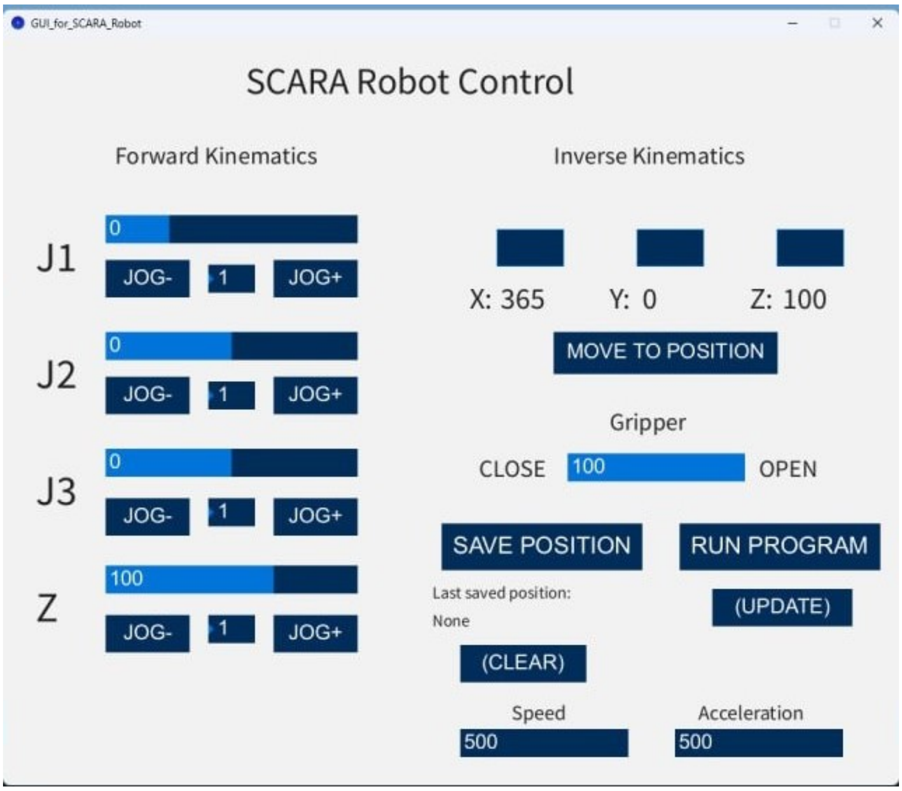
- J_v is the linear velocity Jacobian, accounting for the prismatic and revolute joints.
- J_ω is the angular velocity Jacobian, accounting for revolute joints.

$$J = \begin{bmatrix} (z_0^o) & z_1^o \cdot (t_4^o - t_i) & z_2^o \cdot (t_4^o - t_2^o) & z_3^o \cdot (t_4 - t_3^o) \\ 0 & z_i^o & z_2^o & z_3^o \end{bmatrix}$$

$$J = \begin{bmatrix} 0 & J_{12} & J_{13} & J_{14} \\ 0 & J_{22} & J_{23} & J_{24} \\ 1 & J_{32} & J_{33} & J_{34} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Graphical User Interface (GUI)

The GUI allows users to control the SCARA robot with ease. It was developed using **Processing** and communicates with the robot via an **Arduino Uno**.



Key Features

- **Joint Control:**
  Sliders and buttons enable precise movement of the robot's joints (θ_1, θ_2, θ_3, d_1).
- **Position Control:**
  Users can input the desired end-effector positions, and the robot calculates the corresponding joint angles.
- **Gripper Control:**
  Buttons are provided to open and close the gripper for handling objects.
- **Real-time Feedback:**
  The interface displays the current position and status of the robot in real time.

Reference

- IVProjects GitHub - *Engineering Projects*