A Deep Neural Network based approach for solving inverse kinematics problems for 3DOF robotic arm applications

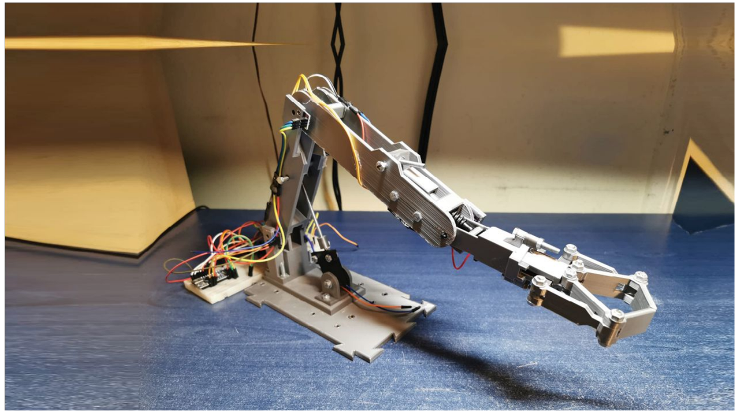
Under the guidance of – **Prof.(Dr.) Debjyoti Chowdhury**

Team members

1. Priyanshu Goswami
2. Sambuddha Basu
3. Arko Bhattacharya
4. Amit Sutradhar

Project Overview

Robotic Arm-Robotic arms, also known as articulated robotic arms, are fast, reliable, and accurate and can be programmed to do an infinite number of tasks in a variety of environments. They are used in factories to automate execution of repetitive tasks, such as applying paint to equipment or parts; in warehouses to pick, select, or sort goods from distribution conveyors to fulfill consumer orders; or in a farm field to pick and place ripe fruits onto storage trays.



Applications :

- Palletizing - Robotic arms can be used to automate the process of placing goods or products onto pallets. By automating the process, palletizing becomes more accurate, cost-effective, and predictable.

- Material Handling - Material-handling robotic arms can help create a safe and efficient warehouse by ensuring goods and materials are properly stored, easy to find, or transported correctly. Automating these processes can help accelerate the delivery of goods to customers, prevent workplace accidents, and improve the efficiency of a facility.

- Welding - Welding is a task that can be performed by robots in advanced industrial settings such as automotive manufacturing. Given its critical impact on product quality, welding is an excellent candidate for advanced robotics with vision and AI augmentation for inline quality inspection.

Materials Required

6 volt dc metal gear server

Power supply (3 cell 11.6 V)

Pi 4 module B

Jumper wires

16 Channel PCA9685 12-Bit PWM/Servo Controller

Dc to dc converter

Big concepts

Inverse Kinematics model

Jacobian matrix

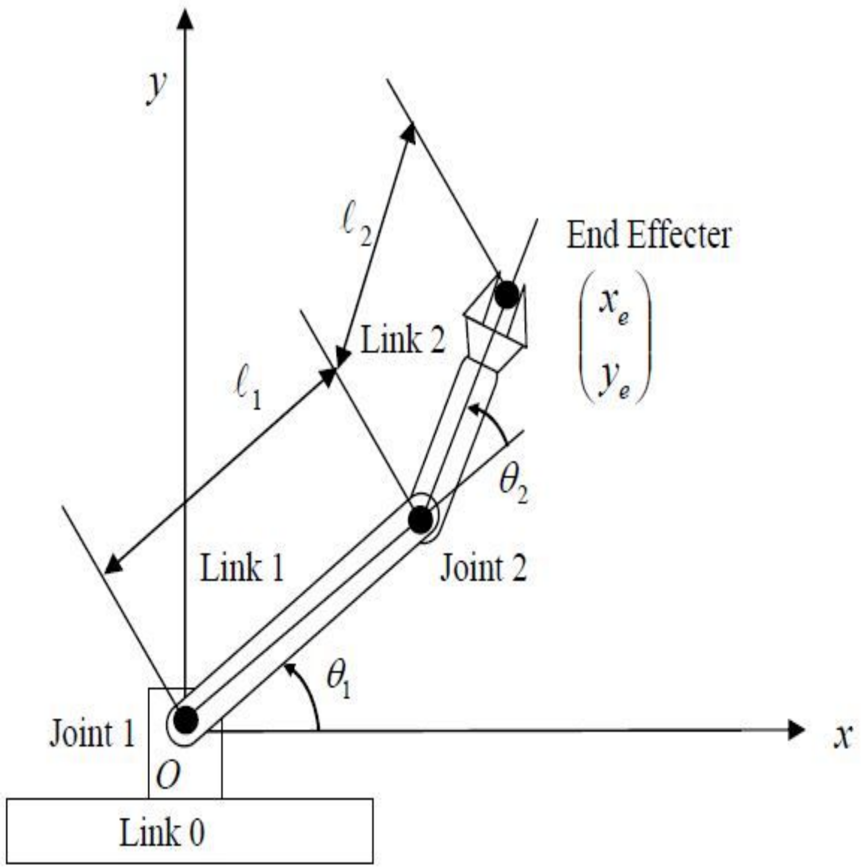
Inverse Kinematics model- As opposed to forward kinematics, which computes the workspace coordinates of the robot given a configuration as input, inverse kinematics (IK) is essentially the reverse operation: computing configuration(s) to reach a desired workspace coordinate. This operation is essential to many robotics tasks, like moving a tool along a specified path, manipulating objects, and observing scenes from a desired vantage point. Because it is so important, inverse kinematics has been studied extensively, with many techniques available to solve it quickly and (relatively) reliably.

Jacobian Matrix-The Jacobian matrix is a fundamental concept in robotics that relates joint velocities to task space velocities. It also has uses in estimating the effects of joint positioning errors in terms of workspace accuracy, as well as in numerical IK methods.

Why ?

The mathematics for real time are just too complex so we are bound to use a neural network where we basically the train the model with a data set and based on that we achieve our objective

In very mathy terms, a Jacobian drops rank and becomes non-invertible. Basically, the math doesn’t work, and the Jacobian doesn’t solve well, and it can cause the controller to tell the manipulator it needs to be at some impossibly high velocity, and the controller will try and follow suit, causing sometimes catastrophic results. It’s reasons like this that massive industrial robots are always behind huge cages and no humans are allowed into the work area



$$x_e(\theta_1, \theta_2) = l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2)$$

$$y_e(\theta_1, \theta_2) = l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2)$$

$$dx_e = \frac{\partial x_e(\theta_1, \theta_2)}{\partial \theta_1} d\theta_1 + \frac{\partial x_e(\theta_1, \theta_2)}{\partial \theta_2} d\theta_2$$

$$dx_e = \frac{\partial y_e(\theta_1, \theta_2)}{\partial \theta_1} d\theta_1 + \frac{\partial y_e(\theta_1, \theta_2)}{\partial \theta_2} d\theta_2$$

$$d\mathbf{x} = \mathbf{J} \cdot d\mathbf{q}$$

$$d\mathbf{x} = \mathbf{J} \cdot d\mathbf{q}$$

And we know that ,

$$d\mathbf{x} = \begin{pmatrix} dx_e \\ dy_e \end{pmatrix} \; d\mathbf{q} = \begin{pmatrix} d\theta_1 \\ d\theta_2 \end{pmatrix}$$

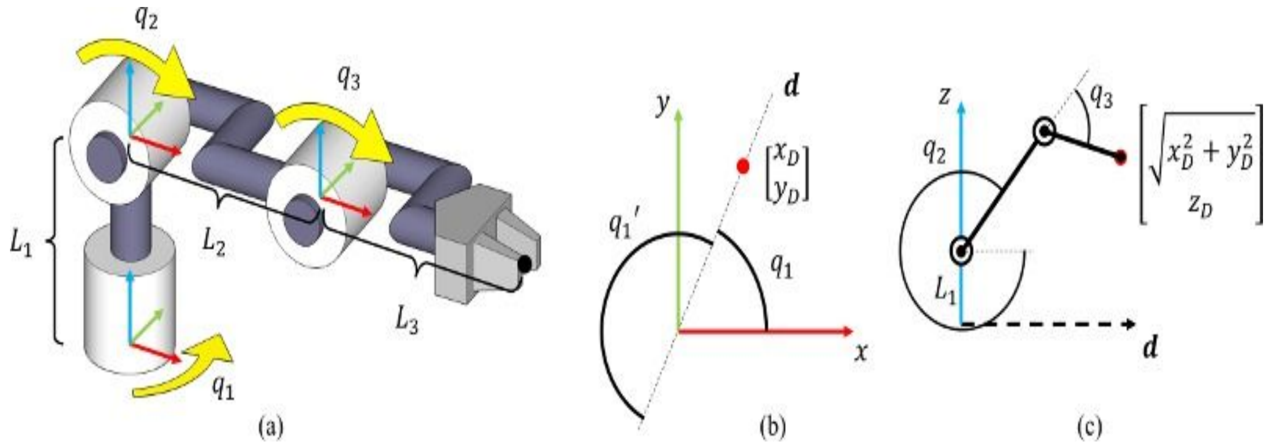
where **q** vector is called the system state and J is the Jacobian matrix , Jacobian matrix is **a matrix of partial derivatives**. Jacobian is the determinant of the jacobian matrix. The main function the jacobian is to transform from one coordinate system to another system.

The Jacobian for the system is:

$$\mathbf{J} = \begin{pmatrix} -l_1 sin(\theta_1) - l_2 sin(\theta_1 + \theta_2) & -l_2 sin(\theta_1 + theta_2) \\ l_1 cos(\theta_1) - l_2 cos(\theta_1 + \theta_2) & l_2 cos(\theta_1 + theta_2) \end{pmatrix}$$

Mathematical model

Explanation-



q_2
q_3
L_1
L_2
L_3
q_1
(a)
y
d
$\begin{bmatrix} x_D \\ y_D \end{bmatrix}$
q_1'
q_1
x
(b)
z
q_3
q_2
$\begin{bmatrix} \sqrt{x_D^2 + y_D^2} \\ z_D \end{bmatrix}$
L_1
d
(c)

a1=L1cosq1+L2cos(q1+q2)+L3cos(q1+q2+q3)
Ye=L1sinq1+L2sin(q1+q2)+L3sin(q1+q2+q3)
θE=q1+q2+q3

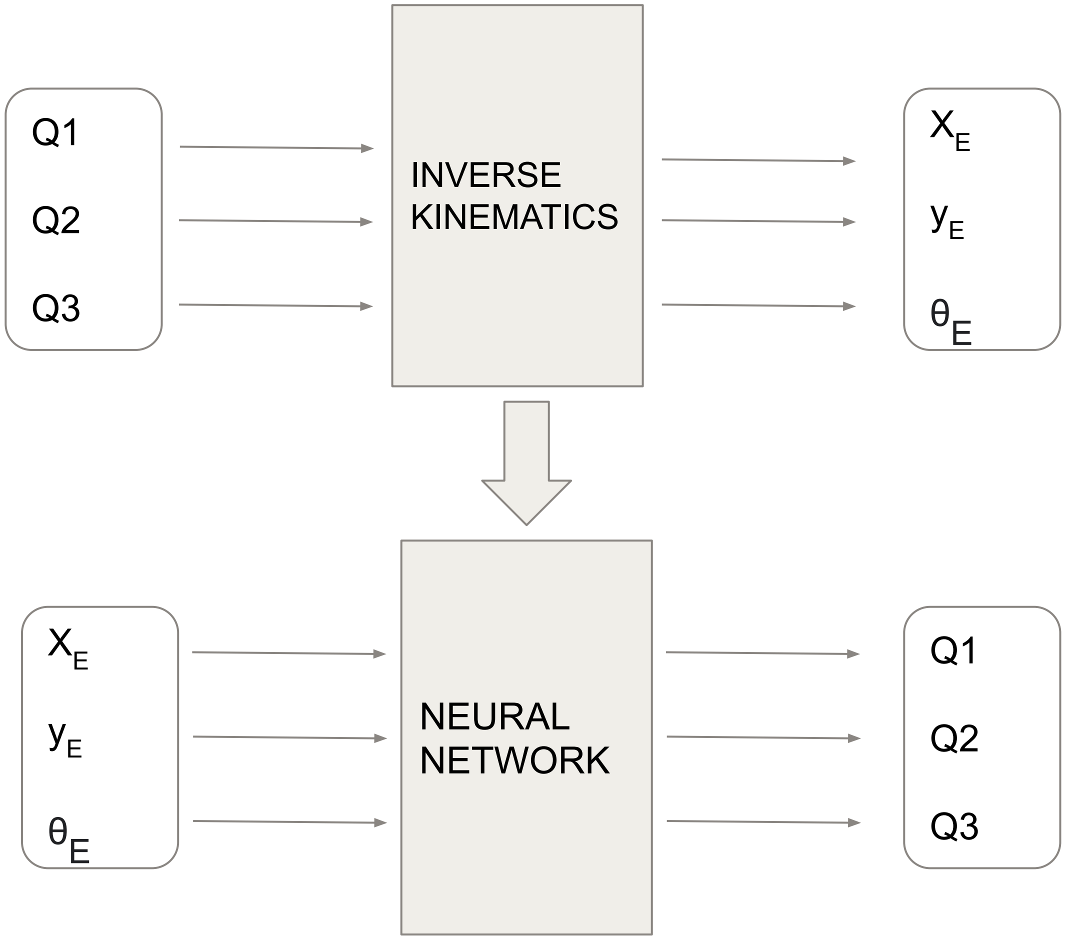
Q1 ∈ **[0,π]**
Q2 ∈ **[-π,0]**
Q3 ∈ **[-π/2,π/2]**

If the robotic arm were to perform a circular trajectory with center (xc,yc) and r radii of circle then

$x_p = x_c + r\cos\Phi \quad \Phi = [0:2\pi] \quad y_p = y_c + r\sin\Phi$

$\theta = \mathrm{atan}(y_p / x_p)$

The model we want to achieve through neural network


Q1
Q2
Q3
INVERSE KINEMATICS
X_E
y_E
θ_E
X_E
y_E
θ_E
NEURAL NETWORK
Q1
Q2
Q3