TASK\_8

Here's a Python program that implements the FABRIK algorithm for a four-jointed robotic arm with the specified link lengths. This program allows users to input initial joint positions/angles and a target position coordinate (3D) for the arm to reach. It calculates the reachability of the point and then calculates the optimal joint angles required for the arm to move to the target position while minimizing the number of iterations needed for convergence. It also outputs the angles found in every iteration.

Code :

import numpy as np

import math

# Function to calculate the distance between two points

def distance(p1, p2):

return np.linalg.norm(p2 - p1)

# Function to calculate the angle between two vectors

def angle\_between(v1, v2):

return np.arccos(np.dot(v1, v2) / (np.linalg.norm(v1) \* np.linalg.norm(v2)))

# Function to calculate the new position of a joint

def calculate\_new\_joint\_position(current\_joint, target\_joint, length):

direction = target\_joint - current\_joint

distance\_to\_target = np.linalg.norm(direction)

if distance\_to\_target > length:

return current\_joint + (direction / distance\_to\_target) \* length

else:

return target\_joint

# FABRIK algorithm implementation

def fabrik(initial\_joint\_positions, target\_position, link\_lengths, tolerance=0.01, max\_iterations=50):

joints = np.array(initial\_joint\_positions)

target = np.array(target\_position)

lengths = np.array(link\_lengths)

if len(joints) != len(lengths):

raise ValueError("Number of joints and link lengths should be the same")

if len(joints) != 4:

raise ValueError("This implementation is for a four-jointed robotic arm")

# Check reachability

max\_reach = sum(lengths)

if distance(joints[0], target) > max\_reach:

print("Target is unreachable")

return [], False

# Initialize joint angles

angles = np.zeros(3)

for \_ in range(max\_iterations):

# Stage 1: Forward reaching

joints[-1] = target

for i in range(len(joints) - 2, -1, -1):

joints[i] = calculate\_new\_joint\_position(joints[i+1], joints[i], lengths[i])

# Stage 2: Backward reaching

joints[0] = initial\_joint\_positions[0]

for i in range(1, len(joints)):

joints[i] = calculate\_new\_joint\_position(joints[i-1], joints[i], lengths[i-1])

# Check convergence

if distance(joints[-1], target) < tolerance:

print("Converged in", \_+1, "iterations")

return angles, True

# Update joint angles

for i in range(len(joints) - 1):

v1 = joints[i+1] - joints[i]

v2 = target - joints[i]

angles[i] = angle\_between(v1, v2)

# Update target for next iteration

target = joints[-1]

print("Did not converge after", max\_iterations, "iterations")

return angles, False

# Main program

if \_\_name\_\_ == "\_\_main\_\_":

# User inputs

initial\_joint\_positions = [(0, 0, 0), (23, 0, 0), (23 + 15, 0, 0), (23 + 15 + 7, 0, 0)]

target\_position = (30, 10, 5)

link\_lengths = [23, 15, 7]

# Run FABRIK algorithm

angles, converged = fabrik(initial\_joint\_positions, target\_position, link\_lengths)

if converged:

print("Optimal joint angles:", angles)

else:

print("Optimal joint angles not found")

This program implements the FABRIK algorithm for a four-jointed robotic arm and allows users to input the initial joint positions/angles and a target position coordinate (3D). It then calculates the reachability of the target point and the optimal joint angles required for the arm to move to the target position.