AIM: WAP for coordinate transform of SCARA Robot.

Theory:

SCARA robot is a kind of special cylindrical coordinate type of industrial robots, which relies on two rotary joints for rapid positioning within the horizontal plane as well as a linear joint and a rotary joint for rotary and stretchy movement within the Z direction. The structure of the SCARA robot studied. It's movement is smooth and reliable in the horizontal direction, and it has a greater stiffness in the vertical direction [1]. In order to improve the accuracy of the robot, kinematic parameters calibration must be done first. Furthermore, kinematic modeling is the theoretical foundation of calibration. Through a function relationship between the end-effector position, orientation matrix and the joint variables, we can determine the parameters of the adjacent coordinate system to obtain a homogeneous transformation relationship. Finally, we can get the kinematic model of the robot using multiplication. The purpose of the establishment of the kinematic model is to find the original error of each parameter transfer coefficient for the results influence, as well as synthetic relationship between the various errors.

SCARA robot movement usually needs to be high-speed and high-precision. Therefore, the analysis of the robot kinematics must be careful, accurate and efficient[3]. Due to the DH model has clear physical meaning and easy programming, this paper established the kinematic model with DH model method.

$$T_{0,1}=\begin{bmatrix}\cos\theta_1 & -\sin\theta_1 & 0 & L1\cos\theta_1\\ \sin\theta_1 & \cos\theta_1 & 0 & L1\sin\theta_1\\ 0 & 0 & 1 & d_1\\ 0 & 0 & 0 & 1\end{bmatrix},\quad T_{1,2}=\begin{bmatrix}\cos\theta_2 & -\sin\theta_2 & 0 & L2\cos\theta_2\\ \sin\theta_2 & \cos\theta_2 & 0 & L2\sin\theta_2\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}. \tag{2}$$

$$T_{2,3}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & -d_3\\ 0 & 0 & 0 & 1\end{bmatrix},\quad T_{3,4}=\begin{bmatrix}\cos\theta_4 & -\sin\theta_4 & 0 & 0\\ \sin\theta_4 & \cos\theta_4 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}. \tag{3}$$

The pose equation of End-effector of the SCARA robot was obtained by multiplying transformation matrix of each joint, it can be shown in (4):

$$\begin{aligned}T_{0,4}=T_{0,1}T_{1,2}T_{2,3}T_{3,4}&=\begin{bmatrix}\cos(\theta_1+\theta_2-\theta_4) & -\sin(\theta_1+\theta_2-\theta_4) & 0 & L1\cos\theta_1+L2\cos(\theta_1+\theta_2)\\ \sin(\theta_1+\theta_2-\theta_4) & \cos(\theta_1+\theta_2-\theta_4) & 0 & L1\sin\theta_1+L2\sin(\theta_1+\theta_2)\\ 0 & 0 & -1 & d1-d3\\ 0 & 0 & 0 & 1\end{bmatrix}\\ &=\begin{bmatrix}n_x & o_x & a_x & p_x\\ n_y & o_y & a_y & p_y\\ n_z & o_z & a_z & p_z\\ 0 & 0 & 0 & 1\end{bmatrix}.\end{aligned} \tag{4}$$

ume If given joint parameters, we can calculate the robot end-effector position (p_x, p_y, p_z) on the reference coordinate system{$X_0Y_0Z_0$}.

CODE:

```
# -*- coding: utf-8 -*-
import math
import numpy as np

theta = [0]
d = [0, 877, 0, 0, 200]
l1 = 425
l2 = 375

angles = tuple(map(int, input("Enter the four joint angles: ").split()))
theta = [0] + [round(math.radians(i),3) for i in angles]
print("Theta Values are:", theta)

d[3] = int(input("Enter link offset 3: "))
print(d)

T1 = np.array([
      [
      math.cos(theta[1]-theta[2]-theta[4]),
      math.sin(theta[1]-theta[2]-theta[4]),
```

```
      0,
      (l1 * math.cos(theta[1])) + (l2 * math.cos(theta[1] - theta[2]))
      ],
      [
      math.sin(theta[1]-theta[2]-theta[4]),
      math.cos(theta[1]-theta[2]-theta[4]),
      0,
      (l1 * math.sin(theta[1])) + (l2 * math.sin(theta[1] - theta[2]))
      ],
      [0, 0, -1, d[1] - d[3] - d[4]],
      [0, 0, 0, 1]
])

ans = np.around(T1, decimals=4)

print(ans[0][3], ans[1][3],ans[2][3])

T3 = np.array([
      [
      (math.sin(theta[4]) * math.sin(theta[1] + theta[2])) + (math.cos(theta[4])
* math.cos(theta[1] + theta[2])),
      (math.sin(theta[4]) * math.cos(theta[1] + theta[2])) + (math.cos(theta[4])
* math.sin(theta[1] + theta[2])),
      0,
      (l2 * math.cos(theta[1] + theta[2])) + (l1 * math.cos(theta[1]))
      ],
      [
      math.sin(theta[4]) * math.cos(theta[1] + theta[2]) + math.cos(theta[4]) *
math.sin(theta[1] + theta[2]),
      - math.sin(theta[4]) * math.sin(theta[1] + theta[2]) + math.cos(theta[4]) *
math.cos(theta[1] + theta[2]),
      0,
      l2 * math.sin(theta[1] + theta[2]) + l1 * math.sin(theta[1])
      ],
      [0, 0, -1, -d[4] - d[3] + d[1]],
      [0, 0, 0, 1]
])

T3 = T3.round(2)
print(T3)
```

OUTPUT:

```
Enter the four joint angles: 45 -60 0 90
Theta Values are: [0, 0.785, -1.047, 0.0, 1.571]
```

```
Enter link offset 3: 120
[0, 877, 0, 120, 200]
203.7987 662.6806 557.0
[[-2.6000e-01  9.7000e-01  0.0000e+00  6.6284e+02]
 [ 9.7000e-01  2.6000e-01  0.0000e+00  2.0327e+02]
 [ 0.0000e+00  0.0000e+00 -1.0000e+00  5.5700e+02]
 [ 0.0000e+00  0.0000e+00  0.0000e+00  1.0000e+00]]
```