UNIVERSIDADE FEDERAL DE ALAGOAS

Instituto da Computação

Engenharia da Computação - 2022.2

Robótica - Prof. Ícaro Araújo

Cinemática inversa com robô SCARA

AB2

Leticia Gabriela Cena de Lima

Karla Sophia Santana da Cruz

Maceió - AL

19 de maio de 2023

Sumário

1 Introdução

A cinemática inversa é uma técnica fundamental em robótica, que permite calcular os ângulos das juntas necessários para posicionar o efetuador final em uma determinada posição desejada. O robô SCARA (Selective Compliance Assembly Robot Arm), é amplamente utilizado na indústria para tarefas de montagem, manipulação e posicionamento. Ele possui um design mecânico que permite movimentos planares, com dois graus de liberdade rotacionais e um grau de liberdade translacional.

A cinemática inversa é especialmente útil para controlar o movimento do robô SCARA, pois, ao fornecer a posição desejada para o efetuador final, é possível calcular os ângulos exatos das juntas para alcançar essa posição. Essa técnica é essencial em aplicações onde é necessário posicionar objetos de forma precisa e controlada.

Neste projeto, utilizaremos o simulador Webots, uma plataforma para simulação de robôs, que permite criar ambientes virtuais realistas e interagir com robôs virtuais. Através da API para Python do Webots, teremos acesso às funcionalidades necessárias para implementar a cinemática inversa do robô SCARA, com o objetivo de utilizar a *"garra"* do robô para pegar o pato posicionado na mesa e colocá-lo dentro da caixa.

No decorrer deste relatório, serão apresentados os passos seguidos para a implementação da cinemática inversa, bem como os resultados obtidos. Serão discutidas as dificuldades encontradas, as soluções adotadas e as conclusões sobre a eficácia da implementação.

Com essa implementação, espera-se obter um controle do movimento do robô SCARA, a fim de atingir o objetivo anteriormente descrito.

2 Experimento

Como falado anteriormente, o objetivo do experimento é usar o simulador Webots e fazer um controlador (em Python) para usar a garra do SCARA para pegar o pato e colocá-lo dentro da caixa. O resultado esperado é o da seguinte figura:

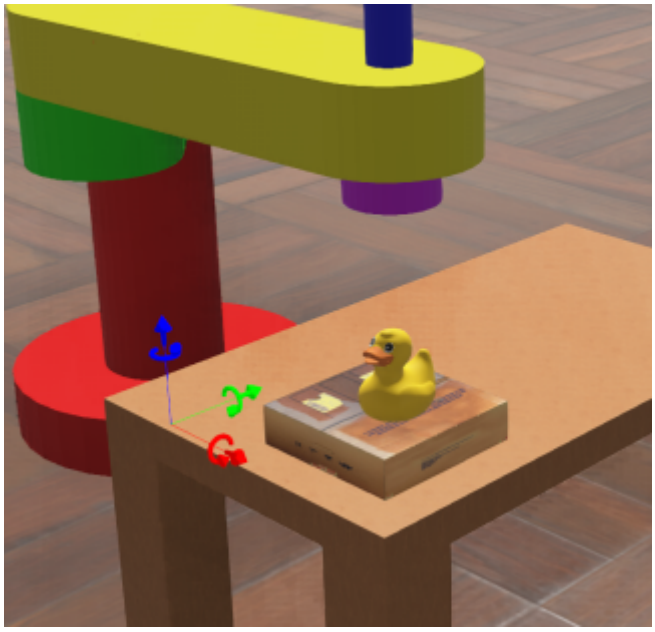

Figura 1: Resultado final desejado.

Para o experimento foi disponibilizado os arquivos da cena (*world*) no simulador e o arquivo python inicial, solicitando o desenvolvimento das funções de cinemática direta e inversa, como também o uso delas no *loop* principal.

2.1 Simulador Webots

2.2 Desenvolvimento das funções

Para realizar o desenvolvimento das funções *fkine* e *invkine* da simulação, foi necessária a construção da modelagem cinemática do robô. Conforme mencionado anteriormente, o robô em questão é um SCARA (do inglês *Selective Compliant Articulated Robot Arm*), o qual consiste de quatro eixos com três juntas rotacionais e uma junta prismática.

2.2.1 Cinemática Direta (*fkine*)

Para que possa encontrar a posição e orientação do elo n com relação ao elo 0 do robô SCARA, é necessário calcular sua cinemática direta. Para isso, é preciso definir o sistema de referência fixo, ou seja, o sistema de referência 0; atribuir as N juntas do SCARA os seus N sistemas de referência; construir sua tabela de parâmetros de elos DH, para, finalmente, calcular as transformações de cada sistema de referência para encontrar a transformação que relaciona os N sistemas, obtendo assim sua cinemática direta.

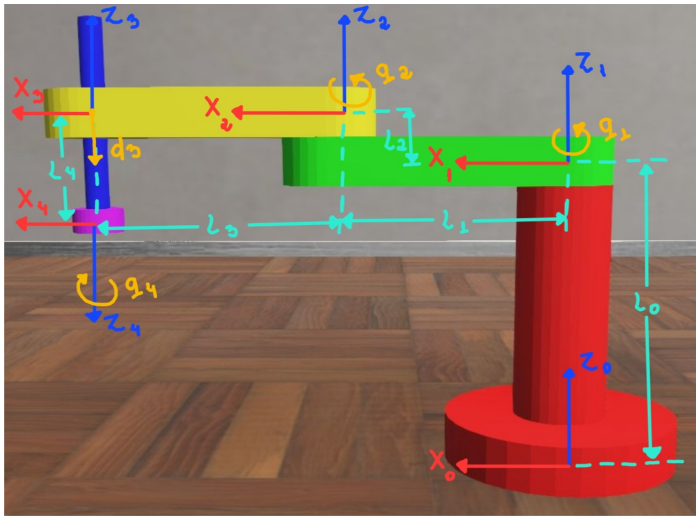

Figura 2: Atribuição dos *Frames* de Referência.

Dessa forma, utilizando a atribuição dos frames de referência dados na documentação desse projeto, representada na figura acima, foi construída a sua tabela de *Denavit-Hartenberg*, demonstrada na tabela abaixo.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	L_0	q_1
2	0	L_1	L_2	q_2
3	0	L_3	0	0
4	π	0	d_3	q_4

Tabela 1: Parâmetros dos elos para o SCARA.

A partir disso, é possível então calcular as transformações que relacionam os sistemas de referência fixados a elos vizinhos, concatená-los e, por fim, encontrar a posição e orientação do elo n com relação ao elo 0, ou seja, sua posição atual.

Assim, baseado na forma geral de transformação dos elos

$$
{}^{i-1}_{i}\mathrm{T} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}
$$

substituindo os parâmetros encontrados na tabela 1, calcula-se as transformações individuais para cada elo, obtendo

$$
{}^{0}_{1}\mathrm{T} = \begin{bmatrix} cq_1 & -sq_1 & 0 & 0 \\ sq_1 & cq_1 & 0 & 0 \\ 0 & 0 & 1 & L_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}
$$

$$
{}^{1}_{2}\mathrm{T} = \begin{bmatrix} cq_2 & -sq_2 & 0 & L_1 \\ sq_2 & cq_2 & 0 & 0 \\ 0 & 0 & 1 & L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}
$$

$$
{}^{2}_{3}\mathrm{T} = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}
$$

$$
{}^{3}_{4}\mathrm{T} = \begin{bmatrix} cq_4 & -sq_4 & 0 & 0 \\ -sq_4 & -cq_4 & 0 & 0 \\ 0 & 0 & -1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}
$$

Por fim, com as transformações individuais, encontra-se a transformação isolada que relaciona o sistema de referência N ao sistema de referência 0:

$$
{}^{0}_{N}\mathrm{T} \;=\; {}^{0}_{1}\mathrm{T}\; {}^{1}_{2}\mathrm{T}\; {}^{2}_{3}\mathrm{T}\; {}^{3}_{4}\mathrm{T} \tag{6}
$$

Logo, a cinemática do SCARA é dada por

$$
{}^{0}_{4}\mathrm{T} = \begin{bmatrix} c(q_1+q_2-q_4) & s(q_1+q_2-q_4) & 0 & L_1\, cq_1 + L_3\, c(q_1+q_2) \\ s(q_1+q_2-q_4) & -c(q_1+q_2-q_4) & 0 & L_1\, sq_1 + L_3\, s(q_1+q_2) \\ 0 & 0 & -1 & L_0 + L_2 - d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}
$$

Assim, baseado na análise da cinemática direta do SCARA acima, foi implementada a função *fkine*, demonstrada abaixo.

```
def fkine(q):
    """
    Implement the forward kinematics of the SCARA robot.
    Input: q - joint angles (list of 4 floats)
    Output: T - transformation matrix (4x4 numpy array)
    """
    # q1 = angulo da junta 1
    # q2 = angulo da junta 2
    # q3 = deslocamento do efetuador
    # q4 = angulo do efetuador

    q1, q2, q3, q4 = q
    d0, d1, d2, d3, d4 = 0.65, 0.1, 0.5, 0.5, -0.2

    # Transformacao da base em relacao a junta 1 (sem rotacao,
       deslocamento em z (altura))
    # vermelho da base em relacao ao vermelho de cima
    T0 = np.array([[1, 0, 0, 0],
```

```
                    [0, 1, 0, 0],
                    [0, 0, 1, d0],
                    [0, 0, 0, 1]])

    # Transformacao da junta 1 em relacao a junta 2 (rotacao em z,
       deslocamento em z (altura) e em x (comprimento))
    # primeiro verde em relacao ao primeiro amarelo
    T1 = np.array([[c(q1), -s(q1), 0, d2],
                    [s(q1), c(q1), 0, 0],
                    [0, 0, 1, d1],
                    [0, 0, 0, 1]])

    # Transformacao da junta 2 em relacao a junta 3 (rotacao em z,
       deslocamento em x (comprimento))
    # primeiro amarelo em relacao ao segundo amarelo
    T2 = np.array([[c(q2), -s(q2), 0, d3],
                    [s(q2), c(q2), 0, 0],
                    [0, 0, 1, 0],
                    [0, 0, 0, 1]])

    # Transformacao da junta 3 em relacao a garra (rotacao em x,
       deslocamento em z (altura))
    # segundo amarelo em relacao ao rosa
    T3 = np.array([[1, 0, 0, 0],
                    [0, c(q3), -s(q3), 0],
                    [0, s(q3), c(q3), -(d4 + q2)],
                    [0, 0, 0, 1]])


    T = np.dot(np.dot(np.dot(T0, T1), T2), T3)

    return T
```

2.2.2 Cinemática Inversa (*inkine*)

Diferentemente da cinemática direta, a cinemática inversa trabalha com a posição desejada e a orientação da ferramenta com relação à estação para obter o conjunto de ângulos de junta que atingirão esse resultado.

O problema de resolver as equações cinemáticas de um manipulador é não linear. Assim, é necessário saber se ao menos existe uma solução e a quantidade de soluções, caso haja. Para o SCARA, serão considerados duas soluções, porém, para essa aplicação, uma solução será escolhida arbitrariamente.

Sabendo disso, foi implementada a função *inkine*, demonstrada abaixo.

```
def invkine(x, y, z, phi):
    """
    Implement the inverse kinematics of the SCARA robot.
    Input: x, y, z, phi - desired end-effector pose
    Output: q - joint angles (list of 4 floats)
    """
```

```
    a1, a2, d0, d1, d4 = 0.5, 0.5, 0.65, 0.1, 0.2

    c2 = (x**2 + y**2 - a1**2 - a2**2) / (2 * a1 * a2)

    if c2 < -1 or c2 > 1:
        raise

    q2_l = np.arctan2(np.sqrt(1 - c2**2), c2)
    q2_ll = np.arctan2(-np.sqrt(1 - c2**2), c2)

    k1_l = a1 + a2 * c(q2_l)
    k1_ll = a1 + a2 * c(q2_ll)

    k2_l = a2 * s(q2_l)
    k2_ll = a2 * s(q2_ll)

    q1_l = np.arctan2(y, x) - np.arctan2(k2_l, k1_l)
    q1_ll = np.arctan2(y, x) - np.arctan2(k2_ll, k1_ll)

    q3 = d0 + d1 - d4 - z

    q4_l = phi - q1_l - q2_l
    q4_ll = phi - q1_ll - q2_ll

    return [q1_l, q2_l, q3, q4_l]
```

2.2.3 mainloop

A função principal do controlador fica responsável por controlar o movimento do braço robótico SCARA e interagir com os objetos *duck* e *box*, inseridos na simulação do Webots. O objetivo final desse *mainloop* é pegar o objeto pato e levá-lo até a caixa.

Para atingir o objetivo, o código segue alguns passos:

1. Cria o objeto com a classe Scara e salva a posição da caixa;
2. Main loop: Executa as etapas até que o Webots pare o controlador;
3. Faz a verificação se o robô não esta segurando o pato, isso diz que o braco esta livre para alcançar o pato;
4. A posição e orientação atual do pato são obtidas usando o método getDuckPose(), que retorna a posição (coordenadas x, y, z) e o ângulo de rotação (yaw) do pato;
5. As posições e orientações desejadas para as juntas do braço robótico são calculadas usando a função invkine(), que implementa a cinemática inversa do robô SCARA. Ela recebe a posição e retorna os angulos das juntas do robô para alcançar o pato;
6. Usa a função de delay para estabilizar o movimento;
7. O método de hold() é chamado para que o braço agarre o pato e não o solte;

8. O próximo loop entrará no else, já que a condição if not scara.grasp: será falsa, isso significa que o robô está segurando o pato e agora irá colocá-lo na caixa;

9. A posição da caixa é atualizada aumentando a coordenada z em 0,1 (movendo a caixa para cima) e mantendo as demais coordenadas;

10. As posições e orientações desejadas para as juntas do robô são recalculadas usando a função invkine() com as novas coordenadas da caixa e são definidos como posições para as juntas do robô usando o método set_position();

11. Novamente usa a função de delay para estabilizar o movimento;

12. Por fim, o braço solta o pato dentro da caixa.

```
if __name__ == "__main__":
    scara = Scara()

    box_pose = [0.85, -0.3, 0.4]

    while scara.step() != -1:
        if not scara.grasp:
            duck_pose = scara.getDuckPose()

            desired_angles = invkine(*duck_pose)

            scara.set_position(desired_angles)

            scara.delay(5000)
            scara.hold()

        else:
            box_pose = [box_pose[0],
                        box_pose[1],
                        box_pose[2] + 0.1,
                        0]

            desired_angles = invkine(*box_pose)

            scara.set_position(desired_angles)

            scara.delay(5000)
            scara.release()

            # Enter here exit cleanup code.
            break
```

3 Conclusão

Neste trabalho, foi desenvolvida a cinemática inversa para um robô SCARA, utilizando o simulador Webots. A cinemática inversa é uma técnica fundamental em robótica, pois

permite calcular os ângulos das juntas necessários para posicionar o efetuador final em uma determinada posição desejada.

Inicialmente, foi feita a modelagem cinemática do robô SCARA, definindo os sistemas de referência e os parâmetros de elos DH. Com base nesses parâmetros, foram calculadas as transformações individuais de cada elo e a transformação total que relaciona o sistema de referência final ao sistema de referência inicial, obtendo assim a cinemática direta do robô.

Com a cinemática direta implementada, foi possível desenvolver a cinemática inversa. Essa técnica permite encontrar os ângulos das juntas a partir de uma posição e orientação desejadas do efetuador final. Para isso, utilizamos equações trigonométricas e geométricas para calcular os ângulos das juntas.

O objetivo final do trabalho foi utilizar a cinemática inversa para controlar o movimento do robô SCARA e posicionar a garra do robô para pegar um objeto e colocá-lo em uma caixa. Através do loop principal do programa, foram realizadas iterações para atualizar a posição desejada do efetuador final e calcular os ângulos das juntas correspondentes.

Durante o desenvolvimento, foram encontradas algumas dificuldades, como a escolha correta dos parâmetros de elos DH e a implementação correta das equações de cinemática inversa. No entanto, essas dificuldades foram superadas com pesquisa, análise e testes.

No final, foi possível alcançar o objetivo proposto e controlar o movimento do robô SCARA de forma precisa, posicionando a garra do robô para pegar o objeto e colocá-lo na caixa. Isso demonstra a eficácia da implementação da cinemática inversa e sua importância na robótica. O trabalho foi uma oportunidade de aprender e aplicar os conceitos de cinemática direta e inversa em um robô SCARA, utilizando o simulador Webots. Além disso, foi possível aprimorar o entendimento sobre o funcionamento e controle de robôs.