Informe de prácticas

Robótica Computacional

Cheuk Kelly Ng Pante (alu0101364544@ull.edu.es)

27 de diciembre de 2023

Índice general

1. Cinematica Directa

La cinemática directa es una rama de la robótica y la mecánica que se ocupa de la relación entre los movimientos de los eslabones de un robot y las variables que los controlan. En otras palabras, la cinemática directa es el problema de encontrar la posición y orientación del extremo del robot, dado el conjunto de parámetros que definen las posiciones y orientaciones de todos los eslabones.

Para explicar la cinemática directa, se utilizará el sistema de coordenadas de Denavit-Hartenberg (DH). El sistema de coordenadas de Denavit-Hartenberg es un sistema de coordenadas utilizado para modelar cinemática directa e inversa de robots articulados. El sistema de coordenadas de Denavit-Hartenberg se basa en cuatro parámetros asociados a cada articulación. Estos parámetros son:

- d_i: Distancia entre los ejes z_{i-1} y z_i a lo largo del eje x_i.
- θ_i: Ángulo entre los ejes z_{i-1} y z_i alrededor del eje x_i.
- a_i: Distancia entre los ejes x_{i-1} y x_i a lo largo del eje z_{i-1}.
- α_i: Ángulo entre los ejes x_{i-1} y x_i alrededor del eje z_{i-1}.

Los parámetros DH se pueden calcular segun la tabla siguiente:

	A	**B**	**C**
d_i	O_{i-1}	$Z_{i-1} \cap X_i$	Z_{i-1}
θ_i	X_{i-1}	X_i	Z_{i-1}
a_i	$Z_{i-1} \cap X_i$	O_i	X_i
α_i	Z_{i-1}	Z_i	X_i

Cuadro 1.1: Parámetros DH

Para la explicación de la cinemática directa, se utilizará el manipulador 3:

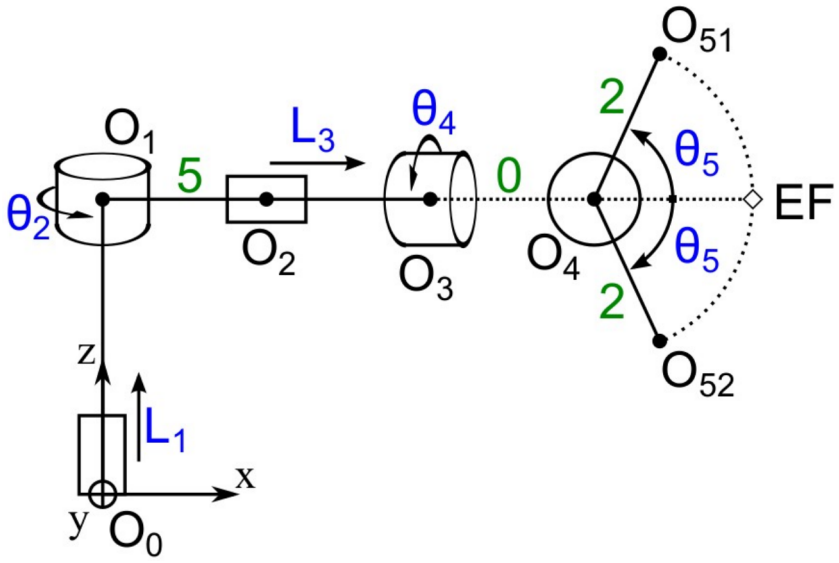

Figura 1.1: Manipulador ejemplo

Para calcular la cinemática directa, primero vamos a calcular los parámetros DH:

```
# Variables de entrada
l1 = p[0]
t2 = p[1]
l3 = p[2]
t4 = p[3]
t5 = p[4]

# Parámetros D-H:
#        1    2     2'   3    4     51     52     EF
d  = [  l1,   0,    0,  l3,   0,     0,     0,    0]
th = [   0,  t2,  -90,   0,  t4, 90-t5, 90+t5,   90]
a  = [   0,   5,    0,   0,   0,     2,     2,    2]
al = [   0, -90,  -90,   0,  90,    90,    90,    0]
```

Figura 1.2: Parámetros DH del manipulador 3

Como se puede observar en la figura 1.2, antes de calcular los parámetros DH, se ha asignado unas variables de entrada, estas corresponden a los ángulos de las articulaciones del manipulador y estas son los valores que se introducen al ejecutar el programa. Por ejemplo, si lo ejecutamos de la siguiente manera:

```
$ python3 ./cinematica_directa 5 0 5 90 45
```

el resultado de la cinemática directa sería la siguiente:

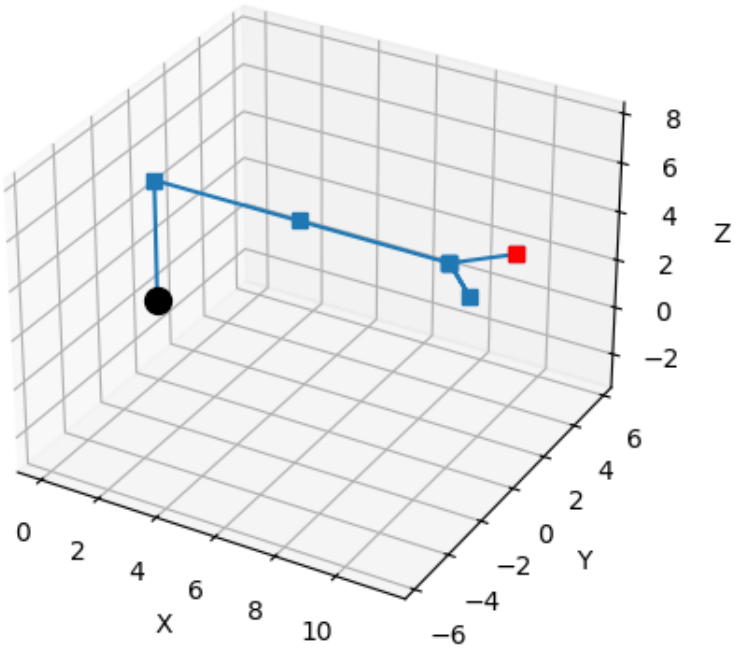

Figura 1.3: Cinemática directa del manipulador 3

Después, se calculado la matriz de transformación de cada articulación para este manipulador:

```
# Cálculo matrices transformación
T01=matriz_T(d[0],th[0],a[0],al[0])
T12=matriz_T(d[1],th[1],a[1],al[1])
T02=np.dot(T01,T12)

T22prima=matriz_T(d[2],th[2],a[2],al[2])
T2prima3=matriz_T(d[3],th[3],a[3],al[3])
T23=np.dot(T22prima,T2prima3)
T03=np.dot(T02,T23)

T34=matriz_T(d[4],th[4],a[4],al[4])
T04=np.dot(T03,T34)

T451=matriz_T(d[5],th[5],a[5],al[5])
T051=np.dot(T04,T451)

T452=matriz_T(d[6],th[6],a[6],al[6])
T052=np.dot(T04,T452)

T4EF=matriz_T(d[7],th[7],a[7],al[7])
T0EF=np.dot(T04,T4EF)
```

Figura 1.4: Matriz de transformación del manipulador 3

Luego, la transformación de cada articulación, especificando los datos de la tabla de Denavit Hartenberg especificadas en la figura 1.2:

```
# Transformación de cada articulación
o10=np.dot(T01, o11).tolist()
o20=np.dot(T02, o22).tolist()
o30=np.dot(T03, o33).tolist()
o40=np.dot(T04, o44).tolist()
o510=np.dot(T051, o5151).tolist()
o520=np.dot(T052, o5252).tolist()
oEF0=np.dot(T0EF, oefef).tolist()
```

Figura 1.5: Transformación de cada articulación del manipulador 3

Y por último, el resultado de la cinemática directa del manipulador 3:

```
# Mostrar resultado de la cinemática directa
muestra_origenes([o00, o11, o22, o33, o44, [[o510], [o520]], oEF0])
muestra_robot   ([o00, o10, o20, o30, o40, [[o510], [o520]], oEF0])
input()
```

Figura 1.6: Cinemática directa del manipulador 3

2. Cinematica Inversa

La cinemática inversa es una rama de la robótica y la mecánica que se ocupa de la relación entre los movimientos de los eslabones de un robot y las variables que los controlan. En otras palabras, la cinemática inversa es el problema de encontrar el conjunto de parámetros que definen las posiciones y orientaciones de todos los eslabones, dado la posición y orientación del extremo del robot.

La cinemática inversa puede ser un problema complejo debido a la presencia de restricciones y limitaciones física del robot, como por ejemplo, limites de movimientos de las articulaciones, colisiones o singularidades.

En la práctica de la cinemática inversa, se ha de calcular la distancia que debe extenderse una articulación prismática situada en el punto O_i, de tal forma que el punto final del robot O_n se acerque tanto como sea posible a la posición objetivo R.

El acercamiento solo puede hacerse en la dirección de extensión L_i, que se puede calcular como un angulo de w que define la rotación respecto al eje x absoluto. El angulo w se puede calcular como:

$$\omega = \sum_{j=0}^{i} \theta_j$$

Usando el producto escalar se puede proyectar el vector que O_n hasta R sobre la dirección de extensión de la articulacion obteniendo así la distancia d:

$$\mathbf{d} = \begin{bmatrix} \cos(\omega) \\ \sin(\omega) \end{bmatrix} \cdot (R - \mathrm{ON})$$

Por tanto, el valor de L_i tras cada iteración pasa a ser:

$$\mathbf{L_i} + \begin{bmatrix} \cos(\omega) \\ \sin(\omega) \end{bmatrix} \cdot (R - \mathrm{ON}), \quad con\ \omega = \sum_{j=0}^{i} \theta_j$$

Para calcular la cinemática inversa en el script en *Python* lo haremos con el algoritmo Cyclic Coordinate Descent (CCD). Y antes de empezar a desarrollar el algoritmo CCD, se ha de definir primero los valores articulares para la cinematica directa. Para ello se han definido unas listas con los valores articulares de cada articulación del robot y estos valores articulares se han definido para que el robot no sobrepase los límites de las articulaciones, limite superior e inferior. Y luego, con otra lista se van a diferenciar entre una articulación rotacional y una prismática.

```
# Valores articulares arbitrarios para la cinemática directa inicial
th = [0.,0.,0.]                 # Ángulos de las articulaciones
a = [5.,5.,5.]                  # Longitud de las articulaciones
tipo_articulacion = [0, 0, 0] # Tipo de articulación - (0) rotacional, (1) prismática
limite_sup = [45, 90, 90]       # Límite superior de las articulaciones, por encima eje X
limite_inf = [-45, -90, -90]    # Límite inferior de las articulaciones, por debajo eje X
```

Figura 2.1: Listas de límites de las articulaciones del robot

Después, ya diferenciando los valores articulares, en el bucle principal calculamos ω que para la articulación rotacional calculamos los vectores V_1 y V_2, que representan las diferencias de posición entre los puntos de interés. Para el vector V_1 es la diferencia entre el punto final del robot y el objetivo y para el vector V_2 es la diferencia entre el punto final del robot y el punto de interés. Una vez calculados los vectores, calculamos los angulos entre los vectores V_1 y V_2 y estos angulos se suman a la variable θ de la articulación rotacional haciendo la diferencia entre los angulos de los vectores V_1 y V_2. Y en el caso de la articulación prismática será la suma de todas las θ hasta la actual (inclusive) y después se calcula la la proyección escalar del vector que va desde el punto final del robot hasta el objetivo sobre la dirección de extensión de la articulación prismática.

Después, hay que prevenir que el robot no sobrepase los límites de las articulaciones, por lo que se ha de comprobar es la posicion actual si es superior o inferior a los límites y si es así la posición pasará a ser la del propio limite, sea superior o inferior.

```
while (dist > EPSILON and abs(prev-dist) > EPSILON/100.):
  prev = dist
  O=[cin_dir(th,a)]
  num_articulaciones = len(th)
  for i in range(num_articulaciones):
    # ===== Cálculo de la cinemática inversa (CCD) =====
    j = num_articulaciones - i - 1  # Articulación actual
    if [tipo_articulacion[j] == 0 for j in range(num_articulaciones)][i]: # Articulacion rotatoria
      v1 = np.subtract(objetivo, O[i][j])
      v2 = np.subtract(O[i][-1], O[i][j])

      alpha1 = atan2(v1[1], v1[0])
      alpha2 = atan2(v2[1], v2[0])

      th[j] += alpha1 - alpha2

      while th[j] > pi: th[j] -= 2 * pi
      while th[j] < -pi: th[j] += 2 * pi

      # Aplicamos los límites a theta
      if (th[j] > np.radians(limite_sup[j])):
        th[j] = np.radians(limite_sup[j])
      elif (th[j] < np.radians(limite_inf[j])):
        th[j] = np.radians(limite_inf[j])
    elif [tipo_articulacion[j] == 1 for j in range(num_articulaciones)][i]: # Articulacion prismatica
      w = np.sum(th[:j + 1])
      d = np.dot([np.cos(w), np.sin(w)], np.subtract(objetivo, O[i][-1]))
      a[j] += d

      # Comprobamos los límites
      if (a[j] > limite_sup[j]):
        a[j] = limite_sup[j]
      elif (a[j] < limite_inf[j]):
        a[j] = limite_inf[j]
    else:
      print("Tipo de articulacion no reconocido, tiene que ser rotatoria (0) o prismatica (1)")
      exit(1)

    O.append(cin_dir(th,a))
```

Figura 2.2: Código de la cinemática inversa con el algoritmo CCD

2.1. Ejemplo de ejecución

Un ejemplo de ejecución del script de la cinemática inversa sería mover el robot al punto x = 5, y = 5:

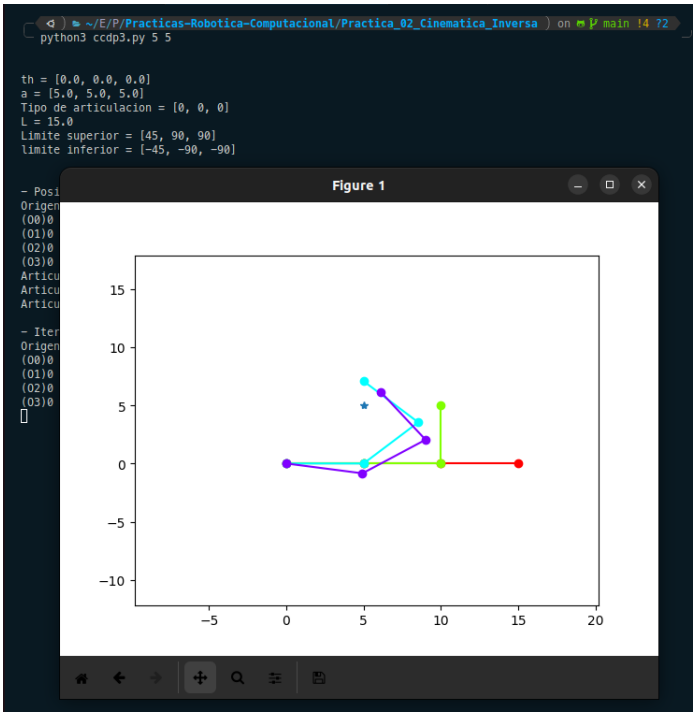

Figura 2.3: Ejemplo de ejecución de la cinemática inversa en la primera iteración

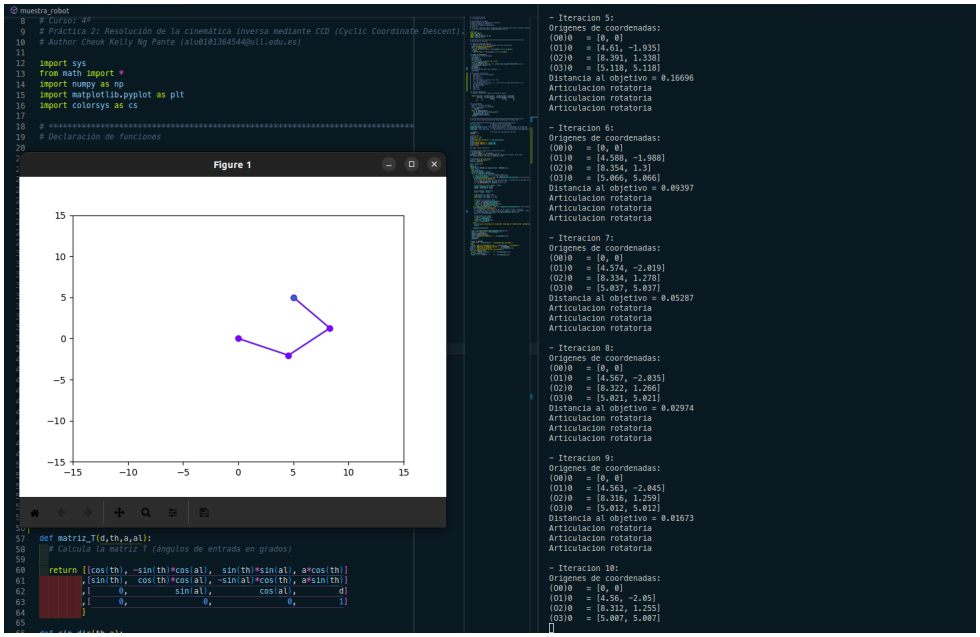

Figura 2.4: Ejemplo de ejecución de la cinemática inversa en la última iteración

3. Localización