Práctica 1

Javier Gálvez Obispo

19 de abril de 2021

Ejercicio 1

Represente las posiciones de los puntos respecto de la trama $\{\boldsymbol{A}\}$ cuando:

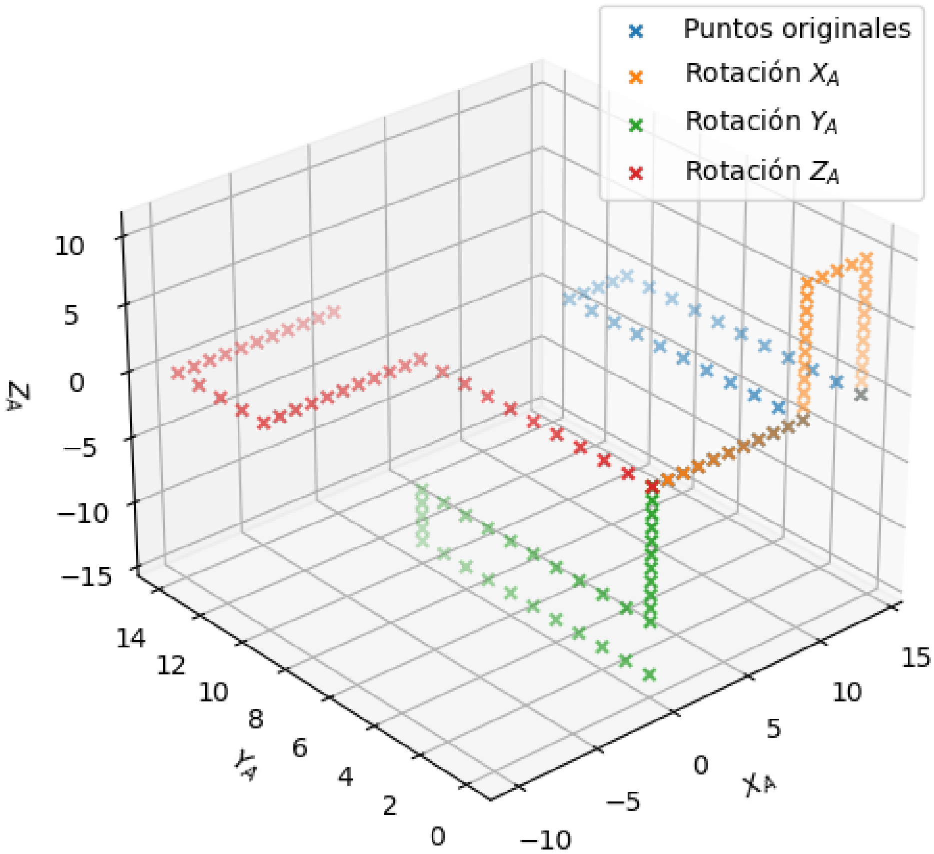
a. La trama $\{\boldsymbol{B}\}$ se gira un ángulo $\alpha = 90°$ alrededor del eje X_A

b. La trama $\{\boldsymbol{B}\}$ se gira un ángulo $\alpha = 90°$ alrededor del eje Y_A

c. La trama $\{\boldsymbol{B}\}$ se gira un ángulo $\alpha = 90°$ alrededor del eje Z_A

Interprete el resultado.

En la siguiente figura se muestran los 3 giros junto a los puntos originales.

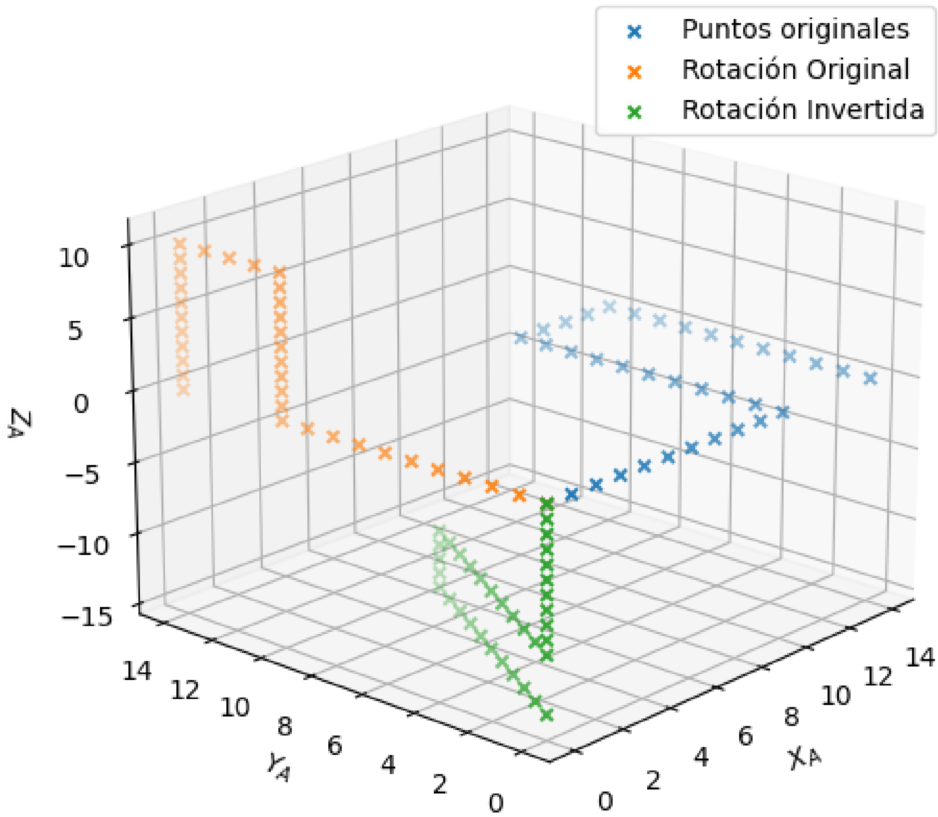


Ejercicio 2

Represente las posiciones de los puntos respecto de la trama $\{\boldsymbol{A}\}$ cuando la trama $\{\boldsymbol{B}\}$ del ejercicio anterior se rota desde la posición original (super-

puesta a la trama $\boldsymbol{\{A\}}$) entorno al eje X_B un ángulo $\gamma = 60°$, a continuación se gira entorno al eje Y_B un ángulo $\beta = 90°$, y después se gira entorno al eje Z_B un ángulo $\alpha = 30°$.
Compruebe como afecta el orden de las rotaciones en las coordenadas respecto de la trama $\boldsymbol{\{A\}}$. Comente el resultado.

El ejercicio nos indica que las rotaciones se hacen respecto a la trama en movimiento, es decir, utilizamos ángulos de Euler. Entonces, para calcular la matriz de rotación descrita ${}^{A}R_B$ multiplicamos las distintas matrices de rotación en el mismo orden que se indica en el enunciado ${}^{A}R_B = R(X_B, 60°) \cdot R(Y_B, 90°) \cdot R(Z_B, 30°)$.
Para comprobar como afecta el orden de las rotaciones a las coordenadas de los puntos invertimos el orden de las matrices en el producto: ${}^{A}R_B = R(Z_B, 30°) \cdot R(Y_B, 90°) \cdot R(X_B, 60°)$.
En la siguiente figura se puede comprobar los distintos resultados.



Ejercicio 3

Diseñe e implemente una función en Python que reciba como entrada una matriz de dimensión n x 4 con los parámetros de Denavit-Hartenberg de un manipulador cualquiera, y devuelva como salida la matriz de transformación homogénea (como un array de dimensión 4 x 4) que relaciona el sistema de coordenas de la base y el sistema de coordenadas del extremo del robot. Contemple la posibilidad de que los ángulos de rotación y los desplazamientos sean variables simbólicas (use el paquete **sympy**). Compruebe el correcto funcionamiento de la función con algunos ejemplos.

Para comprobar el correcto funcionamiento de la función se ha utilizado el ejemplo de la diapositiva 17 del tema 4 en el que los parámetros de Denavit-Hartenberg son los siguientes:

i	θ_i	d_i	a_i	α_i
1	q_1	l_1	0	0
2	$90°$	q_2	0	$90°$
3	0	$l_1 + q_3$	0	0

Además, este ejemplo también muestra como la función implementada puede trabajar indistintamente con valores numéricos y variables simbólicas. La matriz que nos devuelve la función es la siguiente:

Matrix([[-sin(q1), 0, cos(q1), (l3 + q3)*cos(q1)], [cos(q1), 0, sin(q1), (l3 + q3)*sin(q1)], [0, 1, 0, l1 + q2], [0, 0, 0, 1]])

$$^0T_3 = \begin{bmatrix} -sin(q_1) & 0 & cos(q_1) & (l_3 + q_3) * cos(q_1) \\ cos(q_1) & 0 & sin(q_1) & (l_3 + q_3) * sin(q_1) \\ 0 & 0 & 1 & l_1 + q_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Es fácil comprobar el correcto funcionamiento comparando el resultado obtenido con la diapositiva 18 del tema 4 de la asignatura.

Archivos de código

ejercicio1.py

```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d

def rotacionX(angulo):
    r = np.radians(angulo)
    return np.array([[1, 0, 0],
                    [0, np.cos(r), -np.sin(r)],
                    [0, np.sin(r), np.cos(r)]])

def rotacionY(angulo):
    r = np.radians(angulo)
    return np.array([[np.cos(r), 0, np.sin(r)],
                     [0, 1, 0],
                     [-np.sin(r), 0, np.cos(r)]])

def rotacionZ(angulo):
    r = np.radians(angulo)
    return np.array([[np.cos(r), -np.sin(r), 0],
                     [np.sin(r), np.cos(r), 0],
                     [0, 0, 1]])

pxB = np.array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 10,
                10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 12, 13,
                14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14])
pyB = np.array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,
                 2,  3,  4,  5,  6,  7,  8,  9, 10, 10, 10, 10,
                 10,  9,  8,  7,  6,  5,  4,  3,  2,  1,  0])
pzB = np.array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0])
pB = np.array([pxB, pyB, pzB])

rX = rotacionX(90)
rY = rotacionY(90)
rZ = rotacionZ(90)

fig = plt.figure("Ejercicio 1")
```

```
ax = fig.add_subplot(111, projection='3d')
ax.scatter(*pB, marker='x', label='Puntos originales')
ax.scatter(*(rX @ pB), marker='x', label='Rotación $X_A$')
ax.scatter(*(rY @ pB), marker='x', label='Rotación $Y_A$')
ax.scatter(*(rZ @ pB), marker='x', label='Rotación $Z_A$')
ax.set(xlabel='$X_A$', ylabel='$Y_A$', zlabel='$Z_A$')
ax.legend()
plt.show()
```

ejercicio2.py

```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d

def rotacionX(angulo):
    r = np.radians(angulo)
    return np.array([[1, 0, 0],
                    [0, np.cos(r), -np.sin(r)],
                    [0, np.sin(r), np.cos(r)]])

def rotacionY(angulo):
    r = np.radians(angulo)
    return np.array([[np.cos(r), 0, np.sin(r)],
                     [0, 1, 0],
                     [-np.sin(r), 0, np.cos(r)]])

def rotacionZ(angulo):
    r = np.radians(angulo)
    return np.array([[np.cos(r), -np.sin(r), 0],
                     [np.sin(r), np.cos(r), 0],
                     [0, 0, 1]])

pxB = np.array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 10,
                10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 12, 13,
                14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14])
pyB = np.array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,
                 2,  3,  4,  5,  6,  7,  8,  9, 10, 10, 10, 10,
                10,  9,  8,  7,  6,  5,  4,  3,  2,  1,  0])
```

```
pzB = np.array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0])
pB = np.array([pxB, pyB, pzB])

rX = rotacionX(60)
rY = rotacionY(90)
rZ = rotacionZ(30)
m_original = rX @ rY @ rZ
m_invertida = rZ @ rY @ rX

fig = plt.figure("Ejercicio 2")
ax = fig.add_subplot(111, projection='3d')
ax.scatter(*pB, marker='x', label='Puntos originales')
ax.scatter(*(m_original @ pB), marker='x', label='Rotación Original')
ax.scatter(*(m_invertida @ pB), marker='x', label='Rotación Invertida')
ax.set(xlabel='$X_A$', ylabel='$Y_A$', zlabel='$Z_A$')
ax.legend()
plt.show()
```

ejercicio3.py

```
import sympy as sym
from numpy import radians

def denavit_hartenbeg(params):
    t = sym.Identity(4)
    for (theta, d, a, alpha) in params:
        # Rotación Z_{i-1}
        r1 = sym.Matrix([[sym.cos(theta), -sym.sin(theta), 0, 0],
                         [sym.sin(theta), sym.cos(theta), 0, 0],
                         [0, 0, 1, 0],
                         [0, 0, 0, 1]])
        # Traslación Z_{i-1}
        t1 = sym.Matrix([[1, 0, 0, 0], [0, 1, 0, 0],
                         [0, 0, 1, d], [0, 0, 0, 1]])
        # Rotación X_i
        t2 = sym.Matrix([[1, 0, 0, a], [0, 1, 0, 0],
                         [0, 0, 1, 0], [0, 0, 0, 1]])
```

```
        # Traslación X_i
        r2 = sym.Matrix([[1, 0, 0, 0],
                         [0, sym.cos(alpha), -sym.sin(alpha), 0],
                         [0, sym.sin(alpha), sym.cos(alpha), 0],
                         [0, 0, 0, 1]])
        t = t * (r1 * t1 * t2 * r2)
        t = sym.N(t)
        t = t.nsimplify(tolerance=1e-10)
    return t

# Ejemplo diapositiva 17 tema 4
q1, l1, q2, l3, q3 = sym.symbols("q1, l1, q2, l3, q3")
ejemplo = [[q1, l1, 0, 0],
           [radians(90), q2, 0, radians(90)],
           [0, l3 + q3, 0 , 0]]
t = denavit_hartenbeg(ejemplo)
t = sym.N(t)
t = t.nsimplify(tolerance=1e-10)
print(t)
```