lab4

June 1, 2021

```
[2]: #!pip install transformations
     from transformations import *
     import os
     import pandas as pd
     from math import *
     import numpy as np
```

0.0.1 Read Taula-DH

```
[10]: taula = pd.read_csv("taula-DH.0", sep=',', names=[0,1,2,3])
```

0.0.2 Find 0_T_Clamp

```
[11]: origin, xaxis, yaxis, zaxis = [0, 0, 0], [1, 0, 0], [0, 1, 0], [0, 0, 1]

      Zero_T_Clamp = identity_matrix()

      for i in range(0,9):
          Zero_T_Clamp = concatenate_matrices(Zero_T_Clamp,
                                              translation_matrix([taula[1][i], 0, 0]),
                                              rotation_matrix(radians(taula[0][i]),␣
       ↪xaxis),
                                              translation_matrix([0, 0, taula[2][i]]),
                                              rotation_matrix(radians(taula[3][i]),␣
       ↪zaxis)
                                              )
      Zero_T_Clamp = concatenate_matrices(Zero_T_Clamp, translation_matrix([1.
       ↪56,0,0]))
      print(Zero_T_Clamp)
```

```
[[ 0.50001511  0.86601668  0.         13.00005415]
 [-0.86601668  0.50001511  0.         -0.49982159]
 [ 0.          0.          1.          0.        ]
 [ 0.          0.          0.          1.        ]]
```

0.0.3 Find position of Clamp from 0

[12]:
```
Clamp_0 = np.matmul(Zero_T_Clamp, np.array([[0, 0, 0, 1]]).transpose())
Clamp_0
```

[12]:
```
array([[13.00005415],
       [-0.49982159],
       [ 0.        ],
       [ 1.        ]])
```

0.1 Exercise 2

0.1.1 Compute T and D, for each row of Taula-DH

[15]:
```
def compute_T_and_D (taula):

    T = []
    D = []

    for i in range(0,9):
        T.append(concatenate_matrices(translation_matrix([taula[1][i], 0, 0]),
                                      rotation_matrix(radians(taula[0][i]),␣
 ↪xaxis),
                                      translation_matrix([0, 0, taula[2][i]]),
                                      rotation_matrix(radians(taula[3][i]),␣
 ↪zaxis)
                                     )
                )
        D.append(np.array([[-np.sin(radians(taula[3][i])), -np.
 ↪cos(radians(taula[3][i])), 0, 0],
                           [np.cos(radians(taula[0][i]))*np.
 ↪cos(radians(taula[3][i])), -np.cos(radians(taula[0][i]))*np.
 ↪sin(radians(taula[3][i])), 0, 0],
                           [np.sin(radians(taula[0][i]))*np.
 ↪cos(radians(taula[3][i])), -np.sin(radians(taula[0][i]))*np.
 ↪sin(radians(taula[3][i])), 0, 0],
                           [0,0,0,0]
                          ]))
    #print(T)
    #print(D)
    return T, D

T, D = compute_T_and_D(taula)
```

0.1.2 Compute J

[16]:
```python
def compute_J (T, D):
    DX = []
    T_8_9 = translation_matrix([1.56,0,0])

    DX.append(D[0] @ T[1] @ T[2] @ T[3] @ T[4] @ T[5] @ T[6] @ T[7] @ T[8] @␣
↪T_8_9)
    DX.append(T[0] @ D[1] @ T[2] @ T[3] @ T[4] @ T[5] @ T[6] @ T[7] @ T[8] @␣
↪T_8_9)
    DX.append(T[0] @ T[1] @ D[2] @ T[3] @ T[4] @ T[5] @ T[6] @ T[7] @ T[8] @␣
↪T_8_9)
    DX.append(T[0] @ T[1] @ T[2] @ D[3] @ T[4] @ T[5] @ T[6] @ T[7] @ T[8] @␣
↪T_8_9)
    DX.append(T[0] @ T[1] @ T[2] @ T[3] @ D[4] @ T[5] @ T[6] @ T[7] @ T[8] @␣
↪T_8_9)
    DX.append(T[0] @ T[1] @ T[2] @ T[3] @ T[4] @ D[5] @ T[6] @ T[7] @ T[8] @␣
↪T_8_9)
    DX.append(T[0] @ T[1] @ T[2] @ T[3] @ T[4] @ T[5] @ D[6] @ T[7] @ T[8] @␣
↪T_8_9)
    DX.append(T[0] @ T[1] @ T[2] @ T[3] @ T[4] @ T[5] @ T[6] @ D[7] @ T[8] @␣
↪T_8_9)
    DX.append(T[0] @ T[1] @ T[2] @ T[3] @ T[4] @ T[5] @ T[6] @ T[7] @ D[8] @␣
↪T_8_9)

    J = [[0 for x in range(9)] for y in range(3)]

    for i in range(0,9):
        J[0][i] = DX[i][0][3]
        J[1][i] = DX[i][1][3]
        J[2][i] = DX[i][0][0]

    return J

def compute_pseudoInv(J):
    J = np.array(J)
    A = J @ J.transpose()
    B = np.linalg.inv(A)

    J_psinv = J.transpose() @ B

    return J_psinv

J = compute_J(T, D)
compute_pseudoInv(J)
```

0.2 Exercise 3

```
[ ]: delta_x = np.array([-0.10, 0, 0])
     taula = pd.read_csv("taula-DH.0", sep=',', names=[0,1,2,3]);

     for i in range(0,90):
         theta = [taula[3][i] for i in range(0,9)];
         T, D = compute_T_and_D(taula);
         J = compute_J(T, D);
         J_pseinv = compute_pseudoInv(J);
         delta_theta = J_pseinv @ delta_x
         theta = np.add(theta, list(map(degrees, delta_theta)))
         taula[3] = theta
         print(taula[3])
         taula.to_csv("taula-DH", sep=',', header=False, index=False)
         os.system("povray jcb.pov")
         command = f"mv jcb.png jcb_{i}.png"
         os.system(command)
```

0.3 Exercise 4

0.3.1 a)

```
[ ]: delta_x = np.array([-0.10, 0, 0])
     taula = pd.read_csv("taula-DH.0", sep=',', names=[0,1,2,3]);

     for i in range(0,90):
         theta = [taula[3][i] for i in range(0,9)];
         T, D = compute_T_and_D(taula);
         J = compute_J(T, D);
         J[0][4] = 0
         J[1][4] = 0
         J[2][4] = 0
         J_pseinv = compute_pseudoInv(J);
         delta_theta = J_pseinv @ delta_x
         theta = np.add(theta, list(map(degrees, delta_theta)))
         taula[3] = theta
         print(taula[3])
         taula.to_csv("taula-DH", sep=',', header=False, index=False)
         os.system("povray jcb.pov")
         command = f"mv jcb.png jcb_{i}.png"
         os.system(command)
```

0.3.2 b)

[ ]:
```python
Nf = 90
d = 9

x = (lambda i: 13 + ((8*i*i - 21*i*Nf + d*i*Nf) / (Nf*Nf)))

taula = pd.read_csv("taula-DH.0", sep=',', names=[0,1,2,3]);

for i in range(0,90):
    delta_x = np.array([x(i+1) - x(i), 0, 0])
    theta = [taula[3][i] for i in range(0,9)];
    T, D = compute_T_and_D(taula);
    J = compute_J(T, D);
    J_pseinv = compute_pseudoInv(J);
    delta_theta = J_pseinv @ delta_x
    theta = np.add(theta, list(map(degrees, delta_theta)))
    taula[3] = theta
    print(taula[3])
    taula.to_csv("taula-DH", sep=',', header=False, index=False)
    os.system("povray jcb.pov")
    command = f"mv jcb.png jcb_{i}.png"
    os.system(command)
```