PA1 Forward Kinematics Report

Peng Xu

Oct 3, 2017

Overview

Comparison between the Exponential Method and the D-H Method for Forward Kinematics computation.

The following files are included as two python programs and a pdf report.

- ./fk_exp.py
- ./fk_dh.py
- ./report.pdf

Exponential Method

Parameter preparation

The coordinate frame choosen and acccording parameters are shown in Appendix Page 1 and 2.

Code

./fk_exp.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Oct  1 19:44:27 2017

@author: pengxu
"""

import numpy as np
import math


# compute the (x)^, namely the skew symmetric matrix
def skew(x):
    return np.matrix([[0, -x[2], x[1]],
                      [x[2], 0, -x[0]],
                      [-x[1], x[0], 0]])


# compute the rotation matrix by w and theta
def compute_exp_w(w, theta):
    w_hat = skew(w)
    return np.eye(3) + w_hat * math.sin(theta) + w_hat * w_hat * (1 -
math.cos(theta))


# compute transformation matrix
def compute_exp_epsilon(exp_w, q):
    p = np.matmul(np.eye(3) - exp_w, q)
    return np.vstack((np.hstack((exp_w, p)), np.array([0, 0, 0, 1])))


# compute the final gst expression except for gst0
def compute_gst_by_exp(w, q, theta):
    gst = np.eye(4)
    for i in range(len(w)):
        exp_ep = compute_exp_w(w[i], theta[i])
        gst = np.matmul(gst, compute_exp_epsilon(exp_ep, q[i]))
    return gst


if __name__ == '__main__':
    # geometric parameters
    l0 = 13
    l1 = 14.7
    l2 = 12
    l3 = 12
    l4 = 9

    gst_0 = np.matrix([[1, 0, 0, l2+l3+l4],
                       [0, 1, 0, -l0],
                       [0, 0, 1, l1],
                       [0, 0, 0, 1]])

    w0 = np.array([0, 0, 1]).reshape(3, 1)
    w1 = np.array([0, -1, 0]).reshape(3, 1)
    w2 = np.array([0, 0, -1]).reshape(3, 1)
    w3 = np.array([0, 0, -1]).reshape(3, 1)
    w4 = np.array([0, 0, -1]).reshape(3, 1)
    w5 = np.array([1, 0, 0]).reshape(3, 1)

    q0 = np.array([0, -l0, l1]).reshape(3, 1)
    q1 = np.array([0, -l0, l1]).reshape(3, 1)
    q2 = np.array([0, -l0, l1]).reshape(3, 1)
    q3 = np.array([l2, -l0, l1]).reshape(3, 1)
    q4 = np.array([l2+l3, -l0, l1]).reshape(3, 1)
    q5 = np.array([l2+l3+l4, -l0, l1]).reshape(3, 1)

    w_list = [w0, w1, w2, w3, w4, w5]
    q_list = [q0, q1, q2, q3, q4, q5]
    start_theta_list = np.array([-0.003, -0.002, 0.000, 0.000, 0.000,
-1.571]).reshape(-1, 1)
    end_theta_list = np.array([0.122, -0.230, 1.170, -0.023, 0.750,
3.120]).reshape(-1, 1)

    gst_exp = compute_gst_by_exp(w_list, q_list,
theta=end_theta_list-start_theta_list) * gst_0
    print("gst by exp = \n", gst_exp)
```

D-H Method

Parameter preparation

The coordinate frame choosen and acccording parameters are shown in Appendix Page 3.

Code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Oct  1 19:44:27 2017

@author: pengxu
"""

import numpy as np
import math


# DH matrix of each coordinate
def compute_DH_matrix(d, theta, a, alpha):
    Rz_theta = np.matrix([[math.cos(theta), -math.sin(theta), 0, 0],
                          [math.sin(theta), math.cos(theta), 0, 0],
                          [0, 0, 1, 0],
                          [0, 0, 0, 1]])
    Tz_d = np.matrix([[1, 0, 0, 0],
                      [0, 1, 0, 0],
                      [0, 0, 1, d],
                      [0, 0, 0, 1]])
    Tx_a = np.matrix([[1, 0, 0, a],
                      [0, 1, 0, 0],
                      [0, 0, 1, 0],
                      [0, 0, 0, 1]])
    Rx_alpha = np.matrix([[1, 0, 0, 0],
                          [0, math.cos(alpha), -math.sin(alpha), 0],
                          [0, math.sin(alpha), math.cos(alpha), 0],
                          [0, 0, 0, 1]])
    return Rz_theta * Tz_d * Tx_a * Rx_alpha


# multiply all the DH matrice
def compute_gst_by_dh(d, theta, a, alpha):
    A = np.eye(4)
    for i in range(len(d)):
        A = A * compute_DH_matrix(d[i], theta[i], a[i], alpha[i])
    return A


if __name__ == '__main__':
    # geometric parameters
    l0 = 13
    l1 = 14.7
    l2 = 12
    l3 = 12
    l4 = 9

    # zero configuration
    start_theta_list = np.array([0, -0.003, -0.002, 0.000, 0.000, 0.000,
-1.571]).reshape(-1, 1)
    # end configuration
    end_theta_list = np.array([0, 0.122, -0.230, 1.170, -0.023, 0.750,
3.120]).reshape(-1, 1)

    # D-H Parameters
    d = [l1, 0, 0, 0, 0, 0, l4]
    theta = np.array([-math.pi/2, math.pi/2, 0, 0, 0, math.pi/2, 0]).reshape(-1, 1)
    a = [l0, 0, 0, l2, l3, 0, 0]
    alpha = [0, math.pi/2, math.pi/2, 0, 0, math.pi/2, math.pi/2]
    gst_dh = compute_gst_by_dh(d, theta+end_theta_list-start_theta_list, a, alpha)

    # turn pi/2 radius to be identical with the Tool Frame in the Exponential
method
    Rz_halfpi = np.matrix([[math.cos(math.pi/2), -math.sin(math.pi/2), 0, 0],
                          [math.sin(math.pi/2), math.cos(math.pi/2), 0, 0],
                          [0, 0, 1, 0],
                          [0, 0, 0, 1]])

    gst_dh = gst_dh*Rz_halfpi
    print("gst by dh = \n", gst_dh)
```

Verification

1. Equivalence

To prove the two methods above are quivalent, the results of the two programs should be the same.

Considering the zero configuration as below,

```
(-0.003, -0.002, 0.000, 0.000, 0.000, -1.571) radians
```

and the final joint angles as

```
(0.122 -0.230 1.170 -0.023 0.750 3.120) radians
```

we can get the results as following,

```
$ python fk_exp.py
gst by exp =
 [[ -1.91620475e-01  -2.44650559e-01   9.50488136e-01   1.03112376e+01]
 [ -9.78792795e-01  -2.38341458e-02  -2.03461538e-01  -4.24570942e+01]
 [  7.24310517e-02  -9.69318336e-01  -2.34895094e-01   1.31782629e+01]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00000000e+00]]
```

```
```

```
python fk_dh.py
gst by dh =
 [[ -1.91620475e-01  -2.44650559e-01   9.50488136e-01   1.03112376e+01]
 [ -9.78792795e-01  -2.38341458e-02  -2.03461538e-01  -4.24570942e+01]
 [  7.24310517e-02  -9.69318336e-01  -2.34895094e-01   1.31782629e+01]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00000000e+00]]
```

Since they are equal, the equivalence is proved.

1. Belonging to SE(3)

According to the definition, any rigid transformation in 3D can be described by means of a 4 × 4 matrix P with the following structure:

```
P = [  R   p;

     0  0  0  1]
```

where the 3 × 3 orthogonal matrix R ∈ SO(3) is the rotation matrix 1 (the only part of P related to the 3D rotation) and the vector (x, y, z) represents the translational part of the 6D pose.

Obviously, here we only need to prove `R'R = RR' = I` and `|R| = 1`.

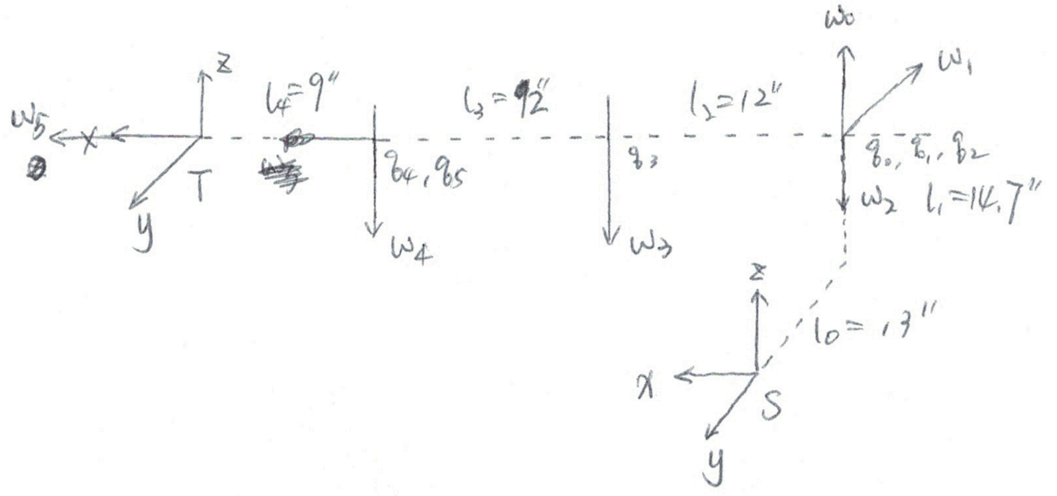
I used the following program to verify,

```
R'R =  [[  1.00000000e+00   4.45895788e-17  -2.77555756e-17]
 [  4.45895788e-17   1.00000000e+00   2.08166817e-17]
 [ -2.77555756e-17   2.08166817e-17   1.00000000e+00]]
RR' =  [[  1.00000000e+00   2.56715640e-17  -2.42861287e-17]
 [  2.56715640e-17   1.00000000e+00   0.00000000e+00]
 [ -2.42861287e-17   0.00000000e+00   1.00000000e+00]]
|R| =  1.0
```

Thus, the result belongs to SE(3).

Appendix

Exponential method

① 

②
$$g_{st}(0) = \begin{bmatrix} & I & & l_2+l_3+l_4 \\ & & & -l_0 \\ & & & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

③
$$\omega_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \omega_1 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \omega_2 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \omega_3 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \omega_4 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \omega_5 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$q_0 = q_1 = q_2 = \begin{bmatrix} 0 \\ -l_0 \\ l_1 \end{bmatrix}, q_3 = \begin{bmatrix} l_2 \\ -l_0 \\ l_1 \end{bmatrix}, q_4 = \begin{bmatrix} l_2+l_3 \\ -l_0 \\ l_1 \end{bmatrix}, q_5 = \begin{bmatrix} l_2+l_3+l_4 \\ -l_0 \\ l_1 \end{bmatrix}$$

Thus,
$$\xi_i = \begin{bmatrix} -\omega_i \times q_i \\ \omega_i \end{bmatrix}, \quad i = 0, 1, \ldots, 5$$

④ Lastly, we obtain

$$g_{st}(\theta) = e^{\hat{\xi}_0\theta_0} e^{\hat{\xi}_1\theta_1} e^{\hat{\xi}_2\theta_2} e^{\hat{\xi}_3\theta_3} e^{\hat{\xi}_4\theta_4} e^{\hat{\xi}_5\theta_5} g_{st}(0)$$

where,

$$e^{\hat{\xi}_i \theta_i} = \begin{bmatrix} & e^{\hat{\omega}_i \theta_i} & & (I - e^{\hat{\omega}_i \theta_i}) q_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

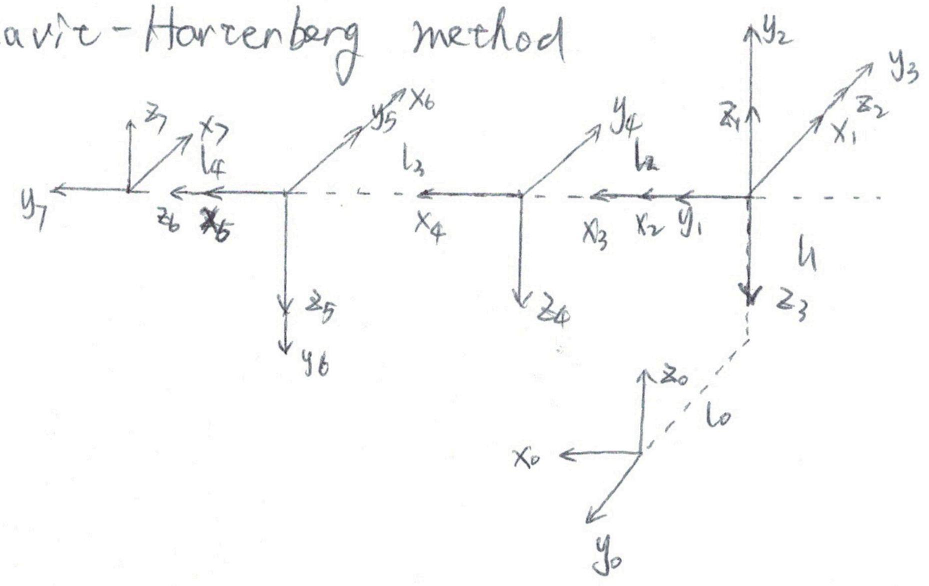
$i = 0, 1, 2, \cdots, 5.$

$$e^{\hat{\omega}_i \theta_i} = I + \hat{\omega}_i \sin\theta_i + \hat{\omega}_i^2 (1 - \cos\theta_i)$$

$i = 0, 1, \cdots, 5$

Denavit-Hartenberg method

①



②

J	d	θ	a	α
0 − 1	l_1	$-\frac{\pi}{2}$	l_0	0
1 − 2	0	$\frac{\pi}{2}$	0	$\frac{\pi}{2}$
2 − 3	0	0	0	$\frac{\pi}{2}$
3 − 4	0	0	l_2	0
4 − 5	0	0	l_3	0
5 − 6	0	$\frac{\pi}{2}$	0	$\frac{\pi}{2}$
6 − 7	l_4	0	0	$\frac{\pi}{2}$