report

學號：311512049　姓名：陳緯翰

Task1

```
============================ Task 1 : Forward Kinematic ============================

900
- Testcase file : fk_testcase.json
- Your Score Of Forward Kinematic : 10.000 / 10.000, Error Count :    0 /  900
- Your Score Of Jacobian Matrix   : 10.000 / 10.000, Error Count :    0 /  900

====================================================================================
- Your Total Score : 20.000 / 20.000
====================================================================================
```

- 1.1

  利用老師上課ppt中的modified dh conventions來完成此次的任務，原本使用classic dh conventions一直無法達成目標，之後聽完老師上課才完成此次任務。

```
final_matrix = np.eye(4, dtype=np.float64)
    for i in range(7):
        joint_angel_matrix =  np.array([[math.cos(q[i]), -math.sin(q[i]), 0, 0],
                                        [math.sin(q[i]), math.cos(q[i]), 0, 0],
                                        [0, 0, 1, 0],
                                        [0, 0, 0, 1]])
        link_offset_matrix = np.array([[1, 0, 0, 0],
                                    [0, 1, 0, 0],
                                    [0, 0, 1, DH_params[i]['d']],
                                    [0, 0, 0, 1]])
        link_length_matrix = np.array([[1, 0, 0, DH_params[i]['a']],
                                    [0, 1, 0, 0],
                                    [0, 0, 1, 0],
                                    [0, 0, 0, 1]])
        link_twist_matrix = np.array([[1, 0, 0, 0],
                                    [0, math.cos(DH_params[i]['alpha']), -math.sin(DH_params[i]['alpha']), 0],
                                    [0, math.sin(DH_params[i]['alpha']), math.cos(DH_params[i]['alpha']), 0],
                                    [0, 0, 0, 1]])
        tranform_matrix =  link_twist_matrix@ link_length_matrix@  joint_angel_matrix@ link_offset_matrix
        final_matrix = final_matrix@ tranform_matrix

    A = A@ final_matrix

    # jacobian = ? # may be more than one line 6* 7的矩陣
    Jacobian = np.zeros((6,7))
    point_end = final_matrix[0:3, 3]

    T_0_i  = np.eye(4, dtype=np.float64)

    # Calculate partial derivatives for each joint
    for i in range(7):

        T = np.array([[math.cos(q[i]), -math.sin(q[i]), 0, DH_params[i]['a']],
                    [math.sin(q[i])* math.cos(DH_params[i]['alpha']), math.cos(q[i])* math.cos(DH_params[i]['alpha']), -math.sin(DH_params[
                    [math.sin(q[i])* math.sin(DH_params[i]['alpha']), math.cos(q[i])* math.sin(DH_params[i]['alpha']), math.cos(DH_params[i
                    [0, 0, 0, 1]])
        T_0_i = T_0_i@ T

        z_i = T_0_i[0:3, 2]                 # gets the vectors p_i and z_i for the Jacobian from the last two coloums of the transformation
        p_i = T_0_i[0:3, 3]
        r = point_end - p_i
        Jacobian[0:3, i] = np.cross(z_i, r) # linear portion
        Jacobian[3:6, i] = z_i              # angular portion             ## each time the loop is passed, another column of the Jacobi mat
```

- 1.2

  Classic D-H convention和Craig's convention主要的差別在於定義a和alpha定義的軸不同。

- 1.3

i	d	α(rad)	a	θ_i(rad)
1	d_1	0	0	θ_1
2	0	$-\pi/2$	0	θ_2
3	d_3	$\pi/2$	0	θ_3
4	0	$\pi/2$	a_3	θ_4
5	d_5	$-\pi/2$	a_4	θ_5
6	0	$\pi/2$	0	θ_6
7	d_7	$\pi/2$	a_6	θ_7

Task2

```
============================ Task 2 : Inverse Kinematic ============================

- Testcase file : ik_testcase_easy.json
- Mean Error : 0.000711
- Error Count :   0 / 100
- Your Score Of Inverse Kinematic : 10.000 / 10.000

- Testcase file : ik_testcase_medium.json
- Mean Error : 0.000885
- Error Count :   0 / 100
- Your Score Of Inverse Kinematic : 10.000 / 10.000

- Testcase file : ik_testcase_medium_2.json
- Mean Error : 0.000880
- Error Count :   0 / 100
- Your Score Of Inverse Kinematic : 10.000 / 10.000

- Testcase file : ik_testcase_hard.json
- Mean Error : 0.001346
- Error Count :   0 / 100
- Your Score Of Inverse Kinematic : 10.000 / 10.000

====================================================================================
- Your Total Score : 40.000 / 40.000
====================================================================================
```

- 2.1

  利用下面此pseudo-inverse method的公式來完成inverse kinematics的計算

$$\Delta\boldsymbol{\theta} = \alpha J^T(\boldsymbol{\theta})\left(J(\boldsymbol{\theta})J^T(\boldsymbol{\theta})\right)^{-1}\Delta\mathbf{x} = J^{\#}\Delta\mathbf{x}$$

```
dh_params = get_panda_DH_params()
    pose, jacobian = your_fk(robot, dh_params, tmp_q)
    iters = 0
    # if step size 設1的話最後一個task只得到3分
    step_size = 0.05

    while((iters<=max_iters)):
        # pseudo jacobian inverse formula
```

```
        delta_matrix = get_matrix_from_pose(new_pose)@ inv(get_matrix_from_pose(pose))
        delta_x = get_pose_from_matrix(delta_matrix, 6)
        A = jacobian@ jacobian.T
        delta_q = step_size* jacobian.T@ inv(A)@ delta_x

        # joint limitation
        tmp_q = tmp_q + delta_q
        for i in range(7):
            if (tmp_q[i]<joint_limits[i][0]):  # 如果更新過後的q比joint limint小就用limit最小直去做更新
                tmp_q[i] = joint_limits[i][0]
            elif(tmp_q[i]>joint_limits[i][1]):
                tmp_q[i] = joint_limits[i][1]
        # 更新的jacobian and pose
        pose, jacobian = your_fk(robot, dh_params, tmp_q)
        # delta_matrix = get_matrix_from_pose(new_pose)@ inv(get_matrix_from_pose(pose))
        # delta_x = get_pose_from_matrix(delta_matrix, 6)
        Norm = norm(delta_x)

        # set 中止條件
        if (Norm < stop_thresh):
            break

        iters += 1
```

- 2.2

  在原本沒有設定stepsize時(stepsize = 1)，在最後一個hard case中沒有辦法得到號的分數，會產生21次的error，所以我之後將stepsize設定成0.01來得到滿分。且在限制limit方面，我的設計是當所計算出來的tmp_q出過limit值時，我以limit的值來取代原本算出來的值，使機械手臂能夠順利的運動。

Task3

- 3.1
  - get_src2dst_transform_from_kpts

    得到不同坐標系中的keypoint之間的轉換，透過不同視角的mug還有mug的template來去進行疊合及找出轉換關係，之後得到不同座標間的轉移矩陣後再套到下面的程式去進行mug的移動。如何實現是利用轉換到world frame的其中一個相機的3D座標扣掉另一個相機的3D座標(ground truth)座標來得到matching error，利用SVD得到轉矩陣。
  - template_gripper_transform

    object frame到gripper frame的轉移矩陣。之後將其做inverse (template_obj_transform = gripper2obj)得到此轉移矩陣後就可以求出物件在grasp frame中的pose，進而得出物件在世界座標系的座標，以用在robot_dense_action函式中進行機械手臂夾取。
- 3.2

  keypoints最少需要3個點以上，因為matching是用在一個三維空間。
- 3.3

  目前皆以grasping.json來得到固定的夾取點位，所以只能夠得到已知物件的grasping pose，對整個pipeline的robustness有很大的限制，若能夠使用6dof graspnet或其他能夠得到grasping pose的方法來完成此pipeline我覺得能夠更robustness。但若使用此方法就無法使用keypoint來進行配對找到符合的轉移矩陣，必須只單純使用6d pose estimation來完成路徑規劃，可能在最後擺放到hook上會有一些問題產生藥仔想辦法解決。
- 3.4

  In my file