IVR Assignment 2021
s1911027 and s1915409

Introduction Github repo: `https://github.com/SlipChik/IVR_Assignment`
Q2.1, Q2.2 done by s1911027, debug by s1915409. Q3.1, Q3.2 done by s1915409, debug by s1911027.

Q2.1 Algorithm In this question, we first estimate the centre of colour joints using masking, dilation and moments. We add some variables to record the last seen centre of the colour spheres to prevent obscure of each other. As the link1 is locked, we decide to consider green sphere as the origin. Therefore, we need to convert all the coordinates to suit the setting. Since the Z axis of OpenCV coordinate is top to down, we need to reverse all the Z-axis of all the spheres. If a sphere being obscured in a camera, we return its last known center.

Joint2 We define the transformed x axis to be cross product between unit vector y and yellow_blue_link. By analysing the coordinate, joint2 equals to the angle between the transformed x axis and the original x axis. So we could easily get joint2 by arccos, reversing the dot product of two axis. Angle of joint 2 is positive if the x axis of the yellow_blue_link is greater than 0, vice versa.

Joint3 Similarily, transformed y axis is cross product between unit vector x and yellow_blue_link. Joint3 equals to the angle between the transformed y axis and the original y axis. Since the yellow_blue_link, transformed y and original y are on the same plane, we could get joint3 by calculating the angle between yellow_blue_link and original y axis using arccos and then subtract pi/2 since yellow_blue_link and transformed y are perpendicular to each other.

Joint4 Joint 4 is the angle between yellow_blue_link and blue_red_link, calculated by arccos, reverting the dot product of two links. Since joint4 also rotates around the y axis like joint2. Thus, the blue_red_link is on the same plane as transformed x of joint2. We could use the dot product between transformed x and blue_red_link to determine the sign of joint4. When the dot product is positive, the angle between blue_red_link and transformed x axis is less than pi/2, which means that joint4 has a positive angle, vice versa.

error estimation The two curves are approximately aligned with minor shifts. At time 333 of joint4, sudden misalignment could be spotted. Since the cameras observe in 3D world, they have a perspective effect where the measured position does not equal to the ground truth position. Meanwhile, if a joint is blocked, we use the previous time step in calculation which result in sudden misalignment. Also, positioning errors might also due to variant lighting and noise of the camera. Those are the reasons for errors.

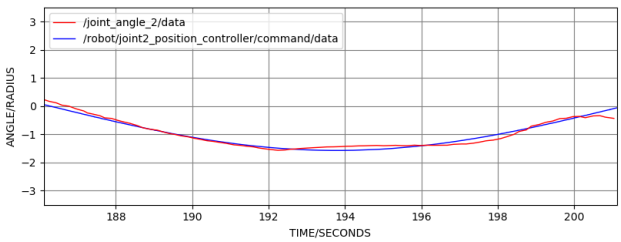

Figure 1: joint2 estimation

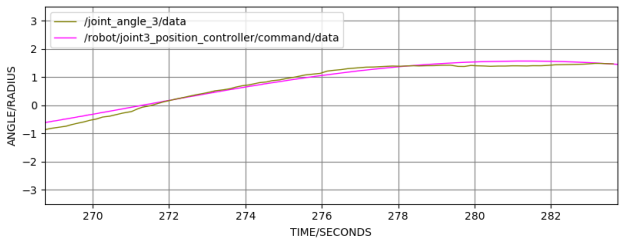

Figure 2: joint3 estimation

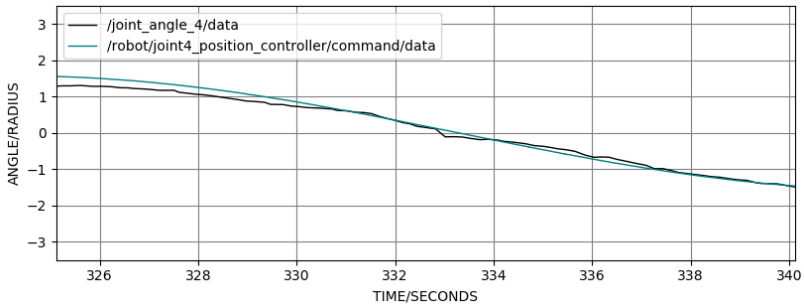

Figure 3: joint4 estimation

Q2.2 Algorithm We use the similar method as Q2.1 to pre-process the data. After determining the coordinates of the joints, links are pulled out to form vectors. So that it is intuitive to calculate angle.

Joint1 The value of joint 1 is first calculated by projecting the yellow_blue_link onto XY plane, then calculate its angle with (0,-1) vector: the original position of Joint 1. Meanwhile, if the vector of yellow_blue_link points to the left of YZ plane, Joint1 value is negative, vies versa. However, when the yellow_blue_link is almost perpendicular to Z axis, I.e. Joint 3 angle close to 0, the projection is close to zero. This not only produce error, but it is also hard to tell which is the correct angle: the measured one or its turned-180-degree counterpart. Since we assume that joint would not make sudden 180 degree flip, we compare the measured joint1 angle with that of the previous time step: if the absolute value between the measured angle is smaller than that of the turned-180-degree version, we would use the measured one. Vise versa.

Joint3 No matter what angle Joint 1 is, the angle between yellow_blue_link and the Z axis is Joint3. So it is calculated using arccos, reverting the dot product of the link and axis. Meanwhile, if the vector of yellow_blue_link points to the left of XZ plane, Joint3 value is negative, vies versa.

Joint4 : Same method as Q2.1, calculate the dot product between blue_red_link and the transformed x, then determine its angle.

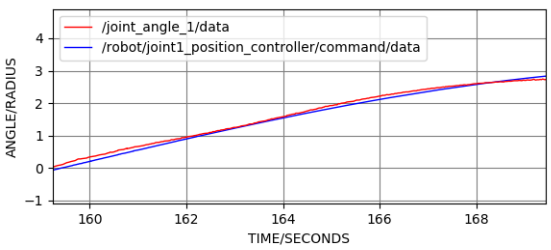

Figure 4: joint1 estimation

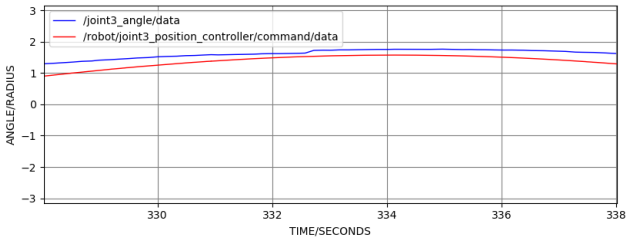

Figure 5: joint3 estimation

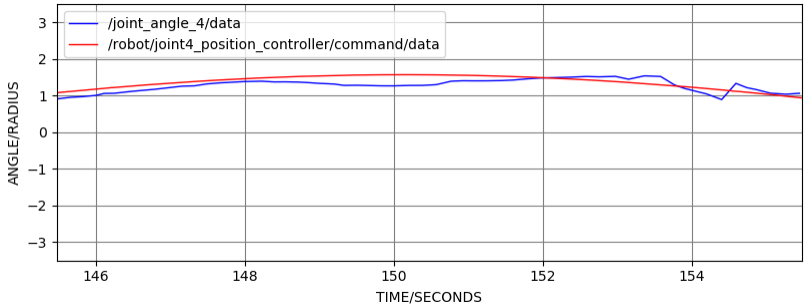

Figure 6: joint4 estimation

Q3.1 DH Parameters & FK equation

Joint	θ	d	α	$r(a)$
1	$\theta_1 + \pi/2$	4	pi/2	0
3	$\theta_3 + \pi/2$	0	pi/2	3.2
4	θ_4	0	0	2.8

$$K(q) = \begin{bmatrix} 2.8 * cos(\theta_1) * sin(\theta_4) + 2.8 * cos(\theta_4) * sin(\theta_3) * sin(\theta_1) + 3.2 * sin(\theta_3) * sin(\theta_1) \\ 2.8 * sin(\theta_1) * sin(\theta_4) - 2.8 * cos(\theta_4) * sin(\theta_3) * cos(\theta_1) - 3.2 * sin(\theta_3) * cos(\theta_1) \\ 4 + 3.2 * cos(\theta_3) + 2.8 * cos(\theta_3) * cos(\theta_4) \end{bmatrix} \quad (1)$$

The estimated positions via forward kinematics and observed positions by computer vision are nearly equivalent, generally with error bellow 2 meters. The prediction of Z value is generally the best. Although in theory, the end effector position calculated by FK is the ground truth, bias in observing joint angles would introduce errors to it. If a joint is not fully visible, the joint centre calculated by computer vision would not be the ground-truth centre (although we mitigated it by returning the last observed position), such as the joint angle(1.6,0,1.5,-1.5); if the value of joint 3 is close to zero, the projection of yellow_blue_link to xy plain would be small so the angle of joint1 would be biased – the end effector position by FK would also be biased – such as the joint angle (2.5,0,0.1,1.2). The end effector captured by CV is also biased: might introduced by masking resolution, variant lighting or camera perspective.

(x,y,z) Estimation by Vision/ meters	(x,y,z) Estimation by FK Matrix/ meters	Joints angle (q1, q2, q3, q4) / radius
(1.254, 4.332, 7.657)	(3.612, 2.304, 7.719)	(-0.75, 0, -0.75, -0.75)
(-2.964, -0.418, 8.379)	(-1.641, -3.480, 8.124)	(-0.75, 0, 0.5, -0.75)
(3.116, -4.826, 6.650)	(4.834, -1.297, 6.423)	(1.0, 0, 1.0, -1.0)
(4.066, -5.510, 3.401)	(5.274, -1.867, 3.303)	(1.0, 0, 1.57, -1.0)
(5.510, 0.038, 3.496)	(4.630, -0.308, 3.517)	(1.0, 0, 1.57, 1.2)
(6.004, -4.750, 6.232)	(4.472, -3.644, 5.578)	(0.8, 0, 1.2, 0.2)
(-4.636, -1.178, 5.396)	(-1.908, -4.840, 4.582)	(2.5, 0, -1.2, 1.2)
(-2.736, 3.876, 4.142)	(-4.795, 3.293, 2.962)	(0.8, 0, -1.57, 0.2)
(2.230, -1.748, 7.467)	(1.042, -2.548, 7.746)	(2.5, 0. 0.1, -1.5)
(4.560, -3.534, 3.762)	(4.160, 2.474, 3.753)	(1.6, 0. 1.5, -1.5)

Figure 7: estimation between vision and FK

Q3.2 Inverse Kinematics Calculation The images shows the inverse kinematics calculation and comapre the x, y, z position

$$\frac{\partial k1}{\partial q1} = -2.8 * \sin(\theta_1) * \sin(\theta_4) + 2.8 * \cos(\theta_4) * \sin(\theta_3) * \cos(\theta_1) + 3.2 * \sin(\theta_3) * \cos(\theta_1) \quad (1)$$

$$\frac{\partial k1}{\partial q3} = 2.8 * \cos(\theta_4) * \cos(\theta_3) * \sin(\theta_1) + 3.2 * \cos(\theta_3) * \sin(\theta_1) \quad (2)$$

$$\frac{\partial k1}{\partial q4} = 2.8 * \cos(\theta_1) * \cos(\theta_4) - 2.8 * \sin(\theta_4) * \sin(\theta_3) * \sin(\theta_1) \quad (3)$$

$$\mathbf{J(q)} = \begin{bmatrix} \frac{\partial k_1}{\partial q_1} & \frac{\partial k_1}{\partial q_3} & \frac{\partial k_1}{\partial q_4} \\ \frac{\partial k_2}{\partial q_1} & \frac{\partial k_2}{\partial q_3} & \frac{\partial k_2}{\partial q_4} \\ \frac{\partial k_3}{\partial q_1} & \frac{\partial k_3}{\partial q_3} & \frac{\partial k_3}{\partial q_4} \end{bmatrix}$$

$$\frac{\partial k2}{\partial q1} = 2.8 * \cos(\theta_1) * \sin(\theta_4) + 2.8 * \cos(\theta_4) * \sin(\theta_3) * \sin(\theta_1) + 3.2 * \sin(\theta_3) * \sin(\theta_1) \quad (4)$$

$$\frac{\partial k2}{\partial q3} = -2.8 * \cos(\theta_4) * \cos(\theta_3) * \cos(\theta_1) - 3.2 * \cos(\theta_3) * \cos(\theta_1) \quad (5)$$

$$\frac{\partial k2}{\partial q4} = 2.8 * \sin(\theta_1) * \cos(\theta_4) + 2.8 * \sin(\theta_4) * \sin(\theta_3) * \cos(\theta_1) \quad (6)$$

$$\frac{\partial k3}{\partial q1} = 0 \quad (7)$$

$$\frac{\partial k3}{\partial q3} = -3.2 * \sin(\theta_3) - 2.8 * \sin(\theta_3) * \cos(\theta_4) \quad (8)$$

$$\frac{\partial k3}{\partial q4} = -2.8 * \cos(\theta_3) * \sin(\theta_4) \quad (9)$$

Figure 8: Inverse Kinematics calculation

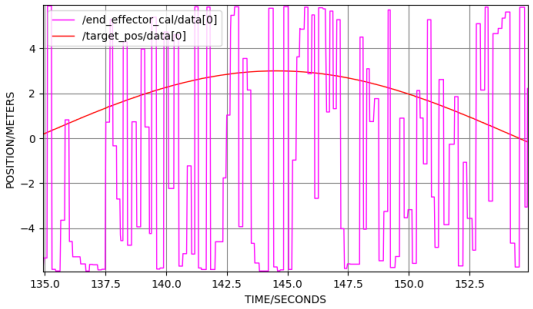

Figure 9: x axis estimation

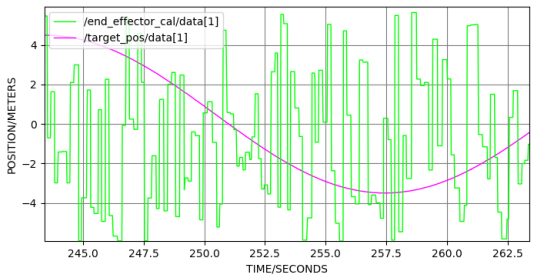

Figure 10: y axis estimation

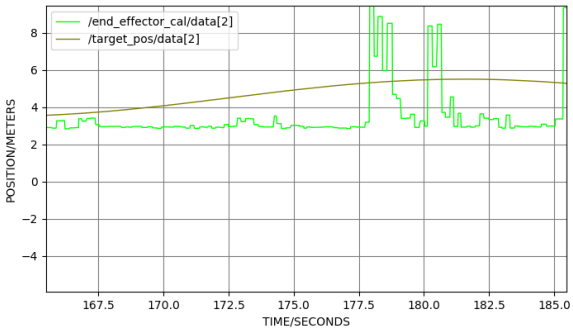

Figure 11: z axis estimation