4.3. Joint State Estimation for all Black Joints

In Part 4, we choose to deal with the task that all joints are set black so that we can not simply get the coordinates of each joint by filtering the color. A new algorithm which can detect the coordinates will be demonstrated.

Algorithm: the main method we use in this part is cv2.HoughCircles() which tries to find as many circles as it can depending on different parameters. In this task we still first apply cv2.inRange() which isolates the black color in the image as a binary image. This is used to filter the target since the target is also a sphere which might influence the following tasks. Now we will use

cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, dp=0.9, minDist=0.8, maxRadius=17, param1=100, param2=6)

This will return different circles in the image. Under the circumstance of the above parameters the function will regard some regions as circle. An example is shown in the Figure 4.3.1 on which we plot the all the circles that we find in the image. However, this is not enough to distinguish different joints as well because we are unable to recognize which circle belong to which joint. Since joint1 is not rotating in this task, the coordinates of yellow and blue joint are remaining the same. We now calculate the Euclidean distance of all the circle centres to the yellow and blue joint respectively and filter out all the circles whose Euclidean distance is very small in terms of either of two joints. Figure 4.3.2 shows the result after filtering. As we can see in the Figure 4.3.2, only circles that belong to red and green joints are left. But we still have more than two circles(NB: we tried to reduce the numbers of circles we could get by changing the parameters for function cv2.HoughCircles() but this might result in that not even one circle related to the green or red joint will be found, which will cause much bigger error. Therefore, in order to get the centres, we will try to obtain as many circles as we can as long as all the circles are related to the joints not other shape.). To get only circles, we again obtain the Euclidean distance of the circles to the blue joint. Now we will regard the centre which is the closest to the blue joint as green joint and the centre which is the furthest to the blue joint as red joint. Figure 4.3.3 shown the final circles that we get. After we obtain all the coordinates we will use the same algorithm that we implemented in the Part 2 to calculate the angles.

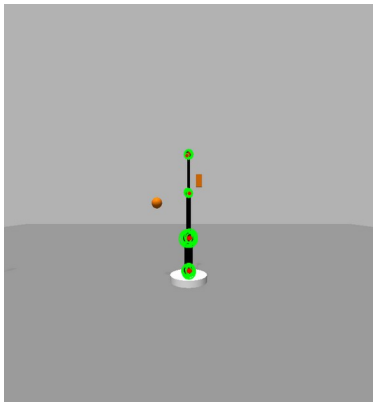


Figure 4.3.1

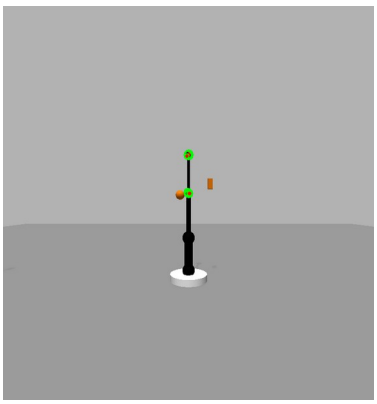


Figure 4.3.2

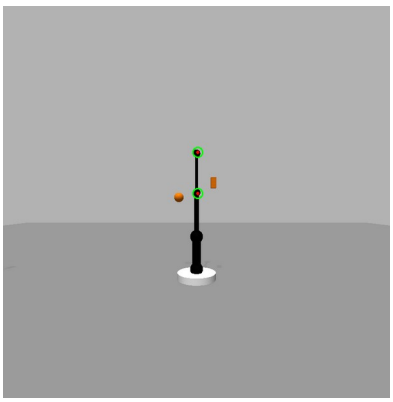
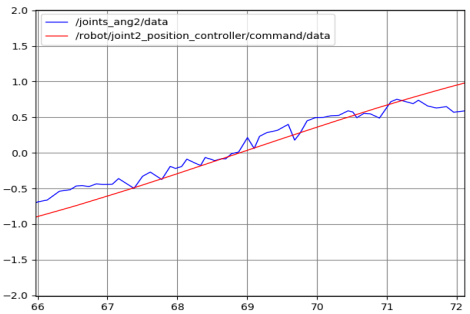
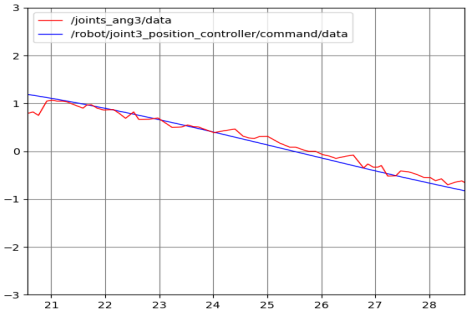
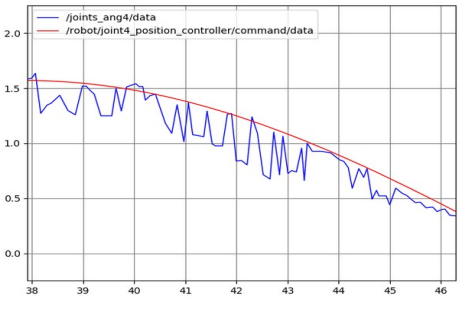


Figure 4.3.3

Validation:







We plotted all the three joint angles detected by the camera and sent by the commands.

3. Robot Control

Part 3.1: Forward Kinematics

The coordinate of the end effector can be obtained by the part of the Transformation Matrix (*T(q)*) for the Robot joints.

To obtain the T(q), first D-H table have to be constructed (q = $[\theta_1, \theta_2, \theta_3, \theta_4]$).

	α	a	d	θ
Link 1	$\frac{\pi}{2}$	0	2.5	$\theta_1+\frac{\pi}{2}$
Link 2	$\frac{\pi}{2}$	0	0	$\theta_2+\frac{\pi}{2}$
Link 3	$\frac{-\pi}{2}$	3.5	0	θ_3
Link 4	0	3	0	θ_4

D-H Table for Robot

Next, from D-H Table construct the transform matrix for each joint (frames).

$$A_i^{i-1} = R_{z,\theta_i}\, Trans_{z,d_i}\, Trans_{x,a_i}\, R_{x,\alpha_i} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ therefore,}$$

$$A_1^0 = \begin{bmatrix} \cos(\theta_1+\frac{\pi}{2}) & 0 & \sin(\theta_1+\frac{\pi}{2}) & 0 \\ \sin(\theta_1+\frac{\pi}{2}) & 0 & -\cos(\theta_1+\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 2.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} A_2^1 = \begin{bmatrix} \cos(\theta_2+\frac{\pi}{2}) & 0 & \sin(\theta_2+\frac{\pi}{2}) & 0 \\ \sin(\theta_2+\frac{\pi}{2}) & 0 & -\cos(\theta_2+\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} A_3^2 = \begin{bmatrix} \cos(\theta_3) & 0 & -\sin(\theta_3) & 3.5\cos(\theta_3) \\ \sin(\theta_3) & 0 & \cos(\theta_3) & 3.5\sin(\theta_3) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} A_4^3 = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & 3\cos(\theta_4) \\ \sin(\theta_4) & \cos(\theta_4) & 0 & 3\sin(\theta_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T(q) = A_1^0 A_2^1 A_3^2 A_4^3 = \begin{bmatrix} \cos(\theta_1+\frac{\pi}{2}) & 0 & \sin(\theta_1+\frac{\pi}{2}) & 0 \\ \sin(\theta_1+\frac{\pi}{2}) & 0 & -\cos(\theta_1+\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 2.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_2+\frac{\pi}{2}) & 0 & \sin(\theta_2+\frac{\pi}{2}) & 0 \\ \sin(\theta_2+\frac{\pi}{2}) & 0 & -\cos(\theta_2+\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_3) & 0 & -\sin(\theta_3) & 3.5\cos(\theta_3) \\ \sin(\theta_3) & 0 & \cos(\theta_3) & 3.5\sin(\theta_3) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & 3\cos(\theta_4) \\ \sin(\theta_4) & \cos(\theta_4) & 0 & 3\sin(\theta_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sin(\theta_1)\sin(\theta_2) & \cos(\theta_1) & -\sin(\theta_1)\cos(\theta_2) & 0 \\ -\sin(\theta_2)\cos(\theta_1) & \sin(\theta_1) & \cos(\theta_1)\cos(\theta_2) & 0 \\ \cos(\theta_2) & 0 & \sin(\theta_2) & 2.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_3)\cos(\theta_4) & -\cos(\theta_3)\sin(\theta_4) & -\sin(\theta_3) & 3.5\cos(\theta_3)+3\cos(\theta_3)\cos(\theta_4) \\ \sin(\theta_3)\cos(\theta_4) & -\sin(\theta_3)\sin(\theta_4) & \cos(\theta_3) & 3.5\sin(\theta_3)+3\sin(\theta_3)\cos(\theta_4) \\ -\sin(\theta_4) & -\cos(\theta_4) & 0 & -3\sin(\theta_4) \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_2^0 A_4^2$$

Coordinate of the end-effector (x, y, z) is corresponding to the T(q)'s last element of 1st, 2nd and 3rd row. Which means to obtain the coordinate of end-effector, whole matrix T(q) does not have to be calculated. By using corresponding row and column of $A_2^0,\ and\ A_4^2$, the desired part of the T(q) can be calculated (corresponding row and column is colored in above).

Estimated end-effector coordinate (x, y, z) is:

$$x_e = \sin(\theta_1)\left(\sin(\theta_2)\cos(\theta_3)\left(3\cos(\theta_4)+3.5\right)+3\cos(\theta_4)\sin(\theta_2)\right)+\cos(\theta_1)\sin(\theta_3)\left(3\cos(\theta_4)+3.5\right)$$

$$y_e = \cos(\theta_1)\left(\sin(\theta_2)\cos(\theta_3)\left(-3\cos(\theta_4)-3.5\right)-3\cos(\theta_2)\sin(\theta_4)\right)+\sin(\theta_1)\sin(\theta_3)\left(3\cos(\theta_4)+3.5\right)$$

$$z_e = cos(\theta_2)cos(\theta_3)(3\ cos(\theta_4)+3.5) - 3sin(\theta_2)\ sin(\theta_4) + 2.5$$

Below Table is result of comparing estimated end-effector coordinate using Forward kinematics with the coordinate obtained by the computer vision implemented in previous part. The average error (difference in Euclidian distance between two points) is 0.81m. This is quite big. However, this does not mean the two point is distant through all the 10 experiments. Actually, the two points are sometimes very near, and sometimes very far away. By observing the image obtained from camera 1 and 2, it can be concluded that when the end-effector is distant from camera, it makes computer vision difficult to calculate the accurate coordinate of the end-effector. The maximum error of the z-coordinate is less than maximum error of the x and y coordinate. This also can be evidence of the conclusion considered above (z-coordinate will be less sensitive to the distance between end-effector and cameras).

$q = [\theta_1, \theta_2, \theta_3, \theta_4]$	Forward Kinematics xyz-coordinate	Computer Vision xyz-coordinate	$q = [\theta_1, \theta_2, \theta_3, \theta_4]$	Forward Kinematics xyz-coordinate	Computer Vision xyz-coordinate
[1.6, 1.6, 1.6, -1.6]	[-0.11, 3.41, 5.5]	[0.16 ,3.96, 5.28]	[0.8, 1.6, 1.6, -1.6]	[2.37, 2.46, 5.5]	[2.51, 3.36, 5.32]
[1.6, -1.6, 1.6, 1.6]	[-0.09, 3.41, 5.5]	[0.0, 3.96, 5.28]	[-0.8, -1.6, -1.6, 1.6]	[-2.39, 2.44, 5.5]	[-2.0, 2.68, 5.32]
[1.6, 1.6, -1.6, -1.6]	[0.09, -3.41, 5.5]	[0.32, -3.4, 5.32]	[1, 1, 1, -1]	[3.14, 3.1, 6.12]	[3.12, 4.32, 6.08]
[-1.6, 1.6, 1.6, -1.6]	[-0.09, -3.41, 5.5]	[0.04, -3.36, 5.32]	[1, 0.3, 1.8, 0.5]	[4.04, 4.51, 0.74]	[4.2, 6.6, 0.16]
[-1.6, 1.6, -1.6, -1.6]	[0.11, 3.41, 5.5]	[0.36, 4.04, 5.32]	[-1, -1, 0.5, 0.5]	[4.75, -0.45, 6.62]	[5.52, -0.44, 6.92]

End-effector position estimated by Forward kinematics and Computer vision with 10 different joints state

Part 3.2: Closed-loop Control

To obtain the formula for Velocity kinematics calculation, Jacobian Matrix have to be calculated first. Jacobian matrix J(q) can be calculated by following formula.

$$J(q)=\begin{pmatrix} \frac{\delta k_1(q)}{\delta\theta_1} & \frac{\delta k_1(q)}{\delta\theta_2} & \frac{\delta k_1(q)}{\delta\theta_3} & \frac{\delta k_1(q)}{\delta\theta_4} \\ \frac{\delta k_2(q)}{\delta\theta_1} & \frac{\delta k_2(q)}{\delta\theta_2} & \frac{\delta k_2(q)}{\delta\theta_3} & \frac{\delta k_2(q)}{\delta\theta_4} \\ \frac{\delta k_3(q)}{\delta\theta_1} & \frac{\delta k_3(q)}{\delta\theta_2} & \frac{\delta k_3(q)}{\delta\theta_3} & \frac{\delta k_3(q)}{\delta\theta_4} \end{pmatrix} \quad where \begin{pmatrix} k_1(q) \\ k_2(q) \\ k_3(q) \end{pmatrix} = \begin{pmatrix} \sin(\theta_1)\left(\sin(\theta_2)\cos(\theta_3)\left(3\cos(\theta_4)+3.5\right)+3\cos(\theta_2)\sin(\theta_3)\right)+\cos(\theta_1)\sin(\theta_3)\left(3\cos(\theta_4)+3.5\right) \\ \cos(\theta_1)\left(\sin(\theta_2)\cos(\theta_3)\left(-3\cos(\theta_4)-3.5\right)-3\cos(\theta_2)\sin(\theta_4)\right)+\sin(\theta_1)\sin(\theta_3)\left(3\cos(\theta_4)+3.5\right) \\ \cos(\theta_2)\cos(\theta_3)\left(3\cos(\theta_4)+3.5\right)-3\sin(\theta_2)\sin(\theta_4)+2.5 \end{pmatrix}$$

Then the formula for updating the joints angles in closed loop control is expressed by following equation.

$$q_{new} = q + \left(J^{+}(q)\cdot\left(K_p\cdot error^T + K_d\cdot \Delta error^T + K_i\cdot errors_i^T\right)\right)*dt$$

$$where \quad J^{+}(q) = pseudo\ inverse\ of\ J(q), \quad K_p=\begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}, \quad K_d=\begin{pmatrix} 0.7 & 0 & 0 \\ 0 & 0.7 & 0 \\ 0 & 0 & 0.7 \end{pmatrix}, \quad K_i=\begin{pmatrix} 0.00001 & 0 & 0 \\ 0 & 0.00001 & 0 \\ 0 & 0 & 0.00001 \end{pmatrix},$$

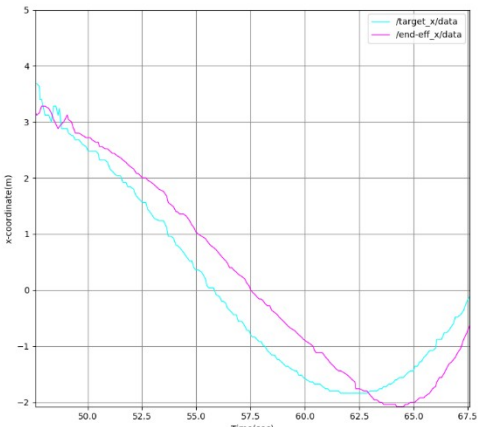
$$error=(target_{pos}-end\ effector_{pos}),\ \Delta error\ (delivative\ of\ error)=error_t-error_{t-1},\ errors_i=\begin{cases}\int_{t-10}^{t} error\ dt & : if\ 10<\left(\int_{t-10}^{t} error\ dt\right)<25 \\ (0,0,0)^T & : otherwise\end{cases}$$

$$t = current\ time\ step$$

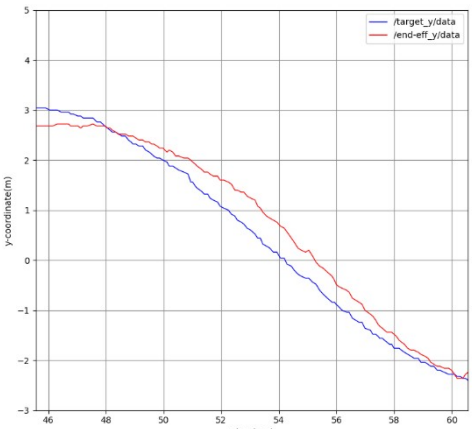
$$The\ position\ of\ target_{camera}\ and\ end\ effector_{camera}\ obtained\ from\ Computer\ vision\ are\ smoothed\ into\ target_{pos}\ and\ end\ effector_{pos}\ in\ following\ method$$

$$target_{pos} = mean\ of\ target_{camera\ t-3}\ to\ target_{camera\ t},\ end\ effector_{pos} = mean\ of\ end\ effector_{camera\ t-5}\ to\ end\ effector_{camera\ t}$$

Three plots presented below is comparing x, y, z position of the robot end-effector with the x, y, z position of the target.



X-coordinate of the target and end-effector



Y-coordinate of the target and end-effector

Z-coordinate of the target and end-effector