Introduction to Vision and Robotics Coursework report

by Ivan Sun s1800640 & Michitatsu Sato s1807428

Part 2 is done by Ivan Sun, Part 3 is done by Michitatsu Sato, Part 4.3 is done by both Ivan Sun and Michitatsu Sato.

GitHub link: https://github.com/IvanSunjg/IVR_Assignment.git

Part 2: Robot Control

Part2.1: Joint State Estimation

Algorithm:(for 2,1 and 2.2)

In this part we used two python files named image1.py and image2.py to receive the image information from the two cameras and then process it. In image1.py we can get the image from camera01 and then get the coordinates of the y axis and z axis. If the joints might not be visible from the camera01, we cannot get the z coordinates(y will be the same as the ones which block the joints) and we will set the y coordinate 0 and we will get the y coordinates in the other py file. And we will publish the result as “/robot/y_z”. The same for image2.py and we will publish “/robot/x_z”. Now we create a python file named state_estimation.py which will subscribe the two previous publishers using message_filters.ApproximateTimeSynchronizer. The same mechanism for the target and end effector . We will use state_estimation.py to publish all the data that we need in the later part.

Joint_calculation:

For the absolute values of three different joints, we perform the same techniques as what we implemented in the labs, using arctan2() or arccos() and the x, y and z coordinates whereas whether the angle is positive or negative would be slightly different. Here we apply some other ways of deciding them.

Joint2&Joint3: these two joint angles are easier to define. If the y coordinate of green is larger than that of blue, then joint2 should be negative. If the x coordinate of green is smaller than that of blue, then joint3 should be negative and vice versa.

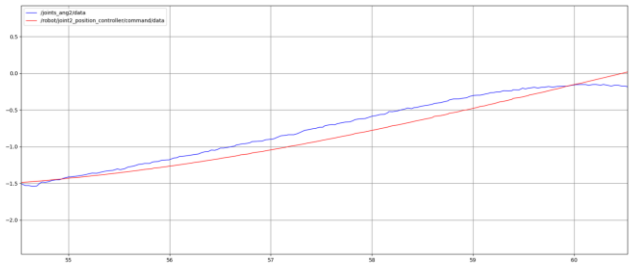
Joint4: joint4 is a bit harder to determine and we did several experiments on joint4 and came up with a table which illustrates all the possible combinations.

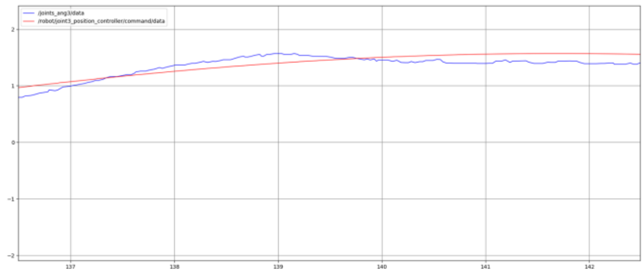
Joint2	joint3	joint4	The position of link 4 compared with the link 3
Positive	Positive	Positive	Down
Positive	Negative	Positive	Down
Negative	Positive	Positive	Up
Negative	Negative	Positive	Up
Positive	Positive	Negative	Up
Positive	Negative	Negative	Up
Negative	Positive	Negative	Down
Negative	Positive	Negative	Down

Therefore, we can get all the negative joint4 by using the table above(use if statements to set the value of joint4).

Validation:

Now we plot the joint that we calculated and joint sent by command.



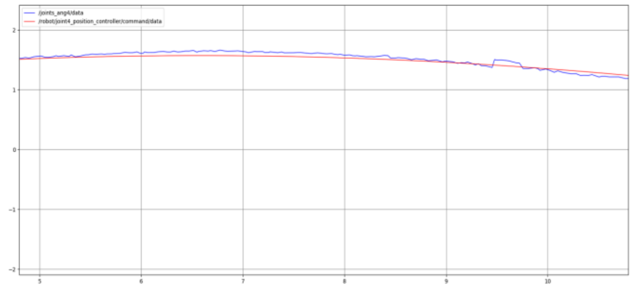


Joint 2: blue line is the estimated values

and red line is the same values.

Joint 3: blue line is the estimated values

and red line is the same values.



Joint 4: blue line is the estimated values and red line is the same values.

Three angles all have some errors. This is because the locations of the three joints are not completely accurate and they might be influenced by the position of the camera as well. Sometimes the sphere might not be a complete circle in the camera because they might be blocked by the links, other joints and the targets.

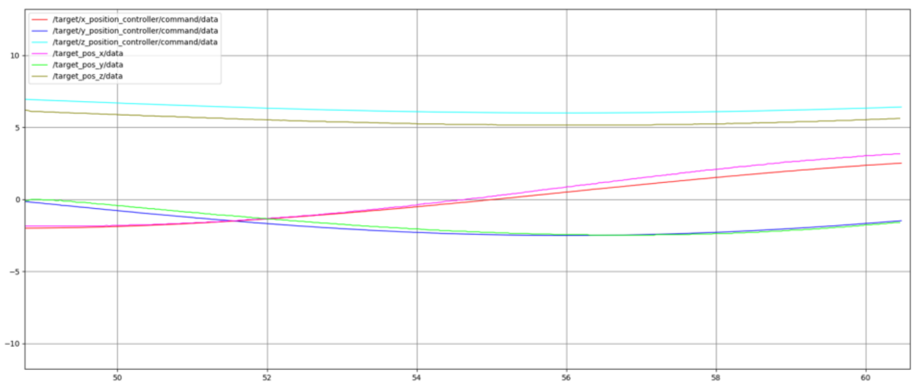
Part2.2: Target detection

The process of subscribing and publishing is the same as the angle calculation and now we only discuss how to distinguish the sphere and the box and the possible errors.

Distinguish: we use the same inRange() function as we used in the joint detection and the range is (5,50,50),(11,255,255). Now we apply cv2.findcontours() to the image. This will normally return 2 contours. One is for the sphere the other is for the box. We can easily distinguish them now since the length of the contour for box will only be four whereas that for sphere will be much larger. Now we apply the same technique to get the centre.

Error: sometimes two objects will become “together” in the sight of the cameras, thus making it hard to distinguish. For this issue, we use a function called cv2.erode() which tries to discard the pixels around the boundary so that we might be able to set two objects apart. Another error comes when the any of two objects is blocked by others. We can not solve this problem completely.

Validation:



We plotted all the three coordinates detected by the camera and sent by the commands. As we can see in the graph that the lines with same coordinates are very close to each other, which means that we have a relatively good detection of the target using the computer vision.

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).

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

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

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

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

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:

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

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

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

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.

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

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

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

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

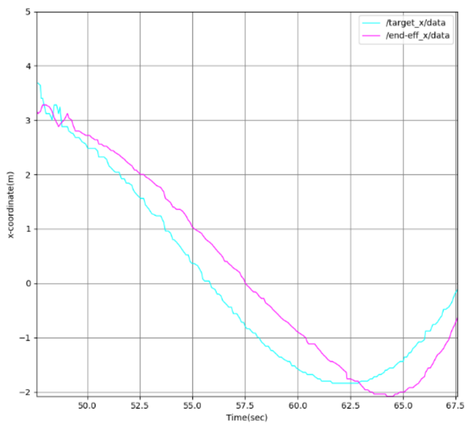
$$\textbf{error} = (\text{target}_{\text{pos}} - \text{end effector}_{\text{pos}}), \quad \boldsymbol{\Delta}\textbf{error}\ (\text{delivative of error}) = \text{error}_t - \text{error}_{t-1}, \quad \textbf{errors}_\textbf{i} = \begin{cases} \int_{t-10}^{t} \text{error dt} \ : \ \text{if } 10 < \left\| \int_{t-10}^{t} \text{error dt} \right\| < 25 \\ [0,0,0]^T \ : \quad \text{otherwise} \end{cases}$$

$$t = current\ time\ step$$

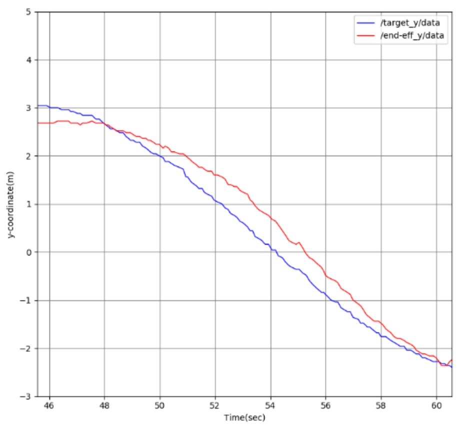
The position of $\text{target}_{\text{camera}}$ and $\text{end effector}_{\text{camera}}$ obtained from Computer vision are smoothed into $\text{target}_{\text{pos}}$ and $\text{end effector}_{\text{pos}}$ in following method

$$\textbf{target}_\textbf{pos} = \text{mean of target}_{\text{camera } t-3} \text{ to target}_{\text{camera } t}, \qquad \textbf{end effector}_\textbf{pos} = \text{mean of end effector}_{\text{camera } t-5} \text{ to end effector}_{\text{camera } t}$$

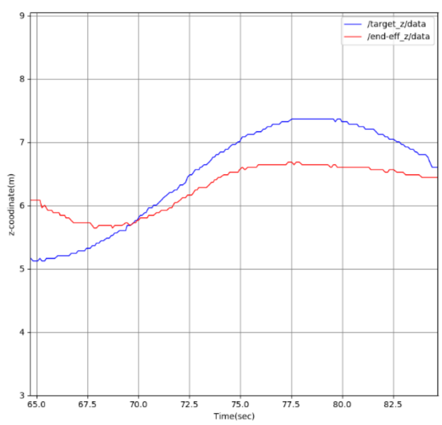
Three plots presented below is comparing x, y, z position of the robot end-effector with the x, y, z position of the target.



X-coordinate of the target and end-effector



Y-coordinate of the target and end-effector



Z-coordinate of the target and end-effector

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.

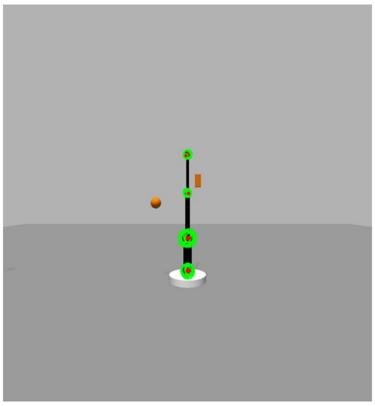


Figure 4.3.1

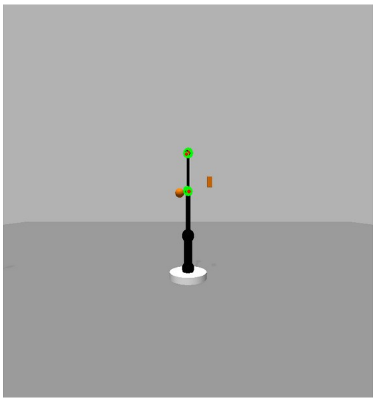


Figure 4.3.2

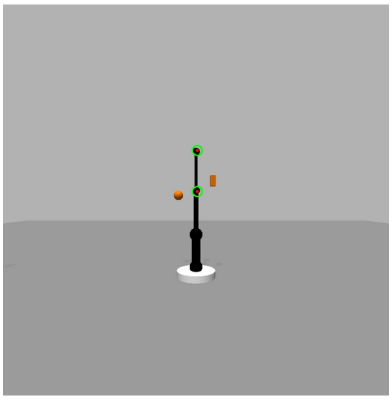
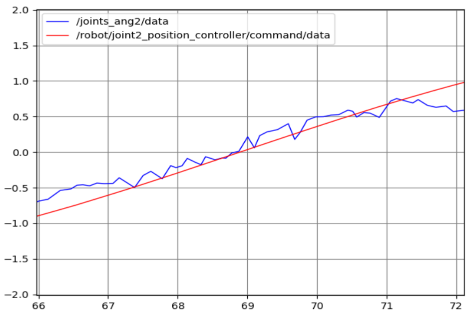
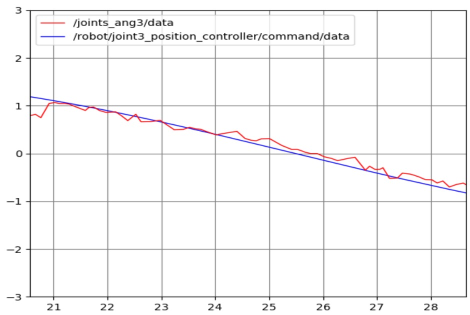
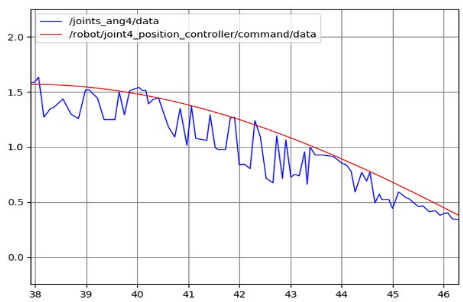


Figure 4.3.3

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).

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

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

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

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

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:

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

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

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

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.

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

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

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

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

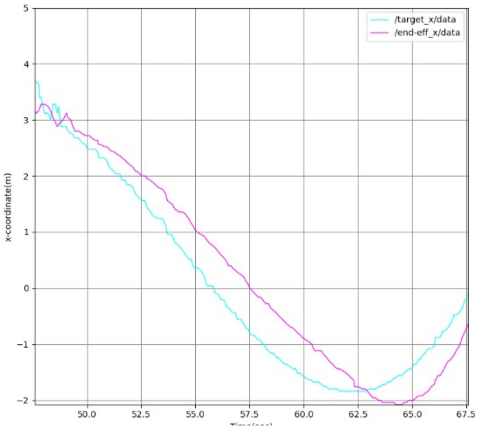
$$error=(target_{pos}-end\ effector_{pos}), \quad \Delta error\ (delivative\ of\ error)=error_t-error_{t-1}, \quad errors_i=\begin{pmatrix} \int_{t-10}^{t} error\ dt & : if\ 10<(\int_{t-10}^{t} error\ dt)<25 \\ (0,0,0)^T & : \quad otherwise \end{pmatrix}$$

$$t=current\ time\ step$$

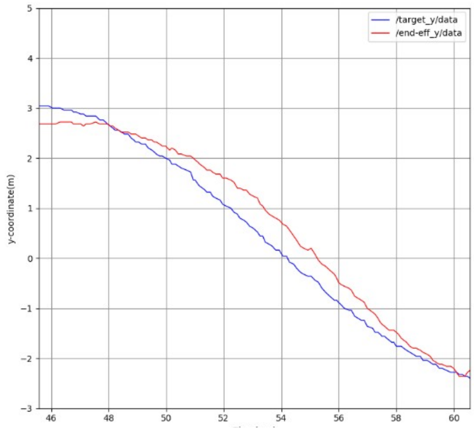
$The\ position\ of\ target_{camera}\ and\ end\ effector_{camera}\ obtained\ from\ Computer\ vision\ are\ smoothed\ into\ target_{pos}\ and\ end\ effector_{pos}\ in\ following\ method$

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

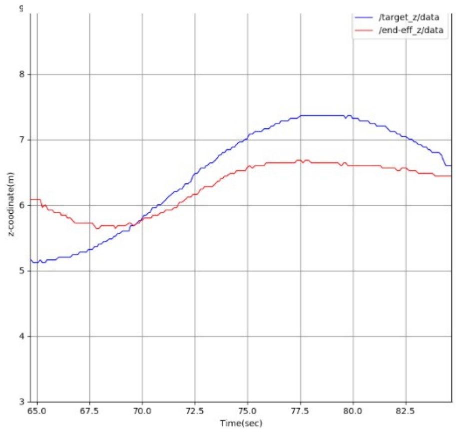
Three plots presented below is comparing x, y, z position of the robot end-effector with the x, y, z position of the target.



X-coordinate of the target and end-effector



Y-coordinate of the target and end-effector



Z-coordinate of the target and end-effector