Introduction to Vision & Robotics

Coursework Report

Qiwen Deng s1810150 & Tu Weiyu s1820742

Part1 – Specification

Qiwen Deng is responsible for coding for question 3.1 and afterwards, Weiyu Tu responsible for the code before 3.1 and all the plots.

Git-hub link: http://github.com/Na2Cl2/IVR_coursework.git

Part2 – Robot Vision

Part2.1 Joint state estimation (Related Filename: image1.py, image2.py, rotates.py, processing.py)

Algorithm:

In question 2, we use 4 python files (*image1.py, image2.py, rotates.py, processing.py*). In those files, *image1.py* and *image2.py* are responsible for receiving data from *camera1* and *camera2*, respectively. Since we are using multiple python files to take the images from cameras, we want to synchronize time which make sure that they will send the image of the same point of time.

Hence, we use the *message_filters.ApproximateTimeSynchronizer* to subscribe the topics in the file *processing.py*. The file *rotates.py* is just the file that used to publish the rotation commands.

From *camera1* we can get the y and z coordinates of the joints and target and from *camera2* we get the x and z coordinates of the joints and target. The joints may be hidden behind another joint from one of the cameras, in this case, the detect function will just simply return [0,0].

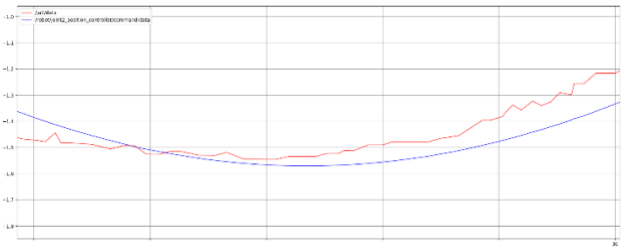
Joint estimation:

As a start of joint estimation, we applied the same way as used in labs, just using the trigonometry formulates. However, this will just work out the absolute values of the angle, so we compare the relative positions to determine the sign. For joint 2 and 3, we compare the x and y coordinates of green joint and blue joint. If the y of green is larger than that of blue, joint2 will be negative and the x of green is smaller than that of blue, then joint3 will be negative and vice versa. For the sign of joint 4, it is slightly more complicated, since we need joint2, joint3 and the relative positions of link3(blue - green) and link4(green - red).

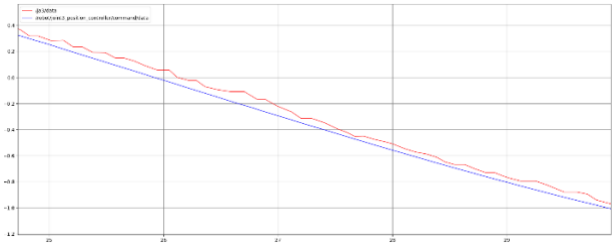
Joint2	Joint3	Relative Position	Joint4
Positive	Positive	Up	Negative
Positive	Positive	Down	Positive
Positive	Negative	Up	Negative
Positive	Negative	Down	Positive
Negative	Positive	Up	Positive
Negative	Positive	Down	Negative
Negative	Negative	Up	Positive
Negative	Negative	Down	Negative

Plots:

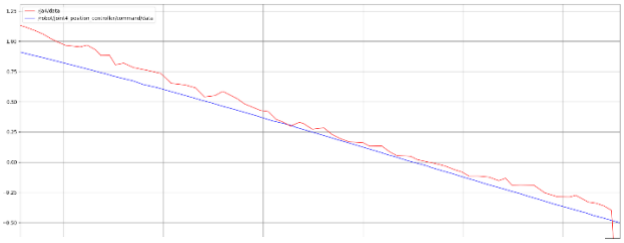
For all of the plots below, RED lines shows FK-predicted angles and BLUE lines shows the detected joint angles.



Joint2



Joint3



Joint4

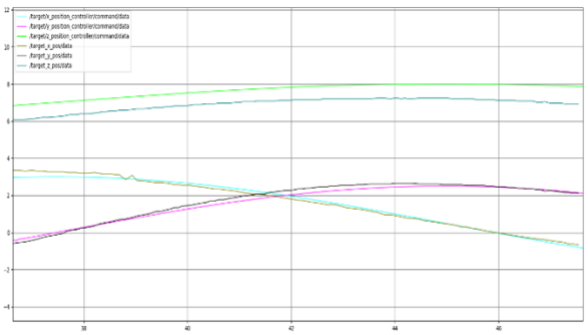
These three graphs show that actual angles are not perfect compare to the theoretical one. This is because the locations of the three joints are not exactly precise due to the noises of images, or the position of the camera as well. Moreover, the joints might not go a complete circle in the camera because they might be blocked by the links, other joints or the orange targets.

Part2.2 Target Detection (Related Filename: image1.py, image2.py, rotates.py, processing.py)

For this question, we are going to work out the position of the orange sphere. We did the job by the following steps.

1. Isolate the orange objects by mask.
2. Use *cv2.findContours* to recognize the objects and crop 2 images as templates (if we get 2 objects).
3. Since the length of the contour for box will only be 4, we just take the object which publish the result and combine them.

Plot:



The error may come from the sphere bumping around when it is moving, or it may be blocked by some other object. More importantly, image from the camera does not exactly represent the physical location of the sphere in 3d space, since the focal length is unknown. The error becomes even larger when we try to combine information from 2 cameras.

Part3 – Robot Control

Part3.1: Forward Kinematics (Related Filename: image1.py, image2.py, rotates.py, processing.py, question3_1sub.py)

We carried out the detected angles by editing the angles of each joint in *rotates.py* file each time. The Forward Kinematics prediction is calculated by functions written in *question3_1sub.py*.

The end-effector can be obtained by the sum of the top 3 rows of the Transformation Matrix *T(q)*. To work this out, we used the *Denavit-Hartenberg Convention* table to construct the Homogenous Transformation matrix. (α means rotation in some x-axis and a is the translation in this direction, θ will be the direction of rotation in that Link Li)

Link	α	a	d	θ
L1	$\pi/2$	0	2.5	$\theta\_1 + \pi/2$
L2	$\pi/2$	0	0	$\theta\_2 + \pi/2$
L4	$-\pi/2$	3.5	0	θ_3
L4	0	3	0	θ_4

Secondly, we calculate the Homogenous Transformation matrix:

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

From the formula above, we get 4 Homogenous Transformation matrix:

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

Therefore, we have:

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

By calculating the matrix, we get:

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

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

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

Finally , we carry out the table below, *q* will be the joint angles ($\theta_1, \theta_2, \theta_3, \theta_4$)

#	Angles of Each Joint	Detected Angles	FK Predicted Angles
1	0.1, 0.2, 0.2, 0.2	1.74, - 1.91, 8.52	1.45, - 1.70, 8.56
2	0.3, 0.4, 0.4, 0.4	2.88, - 2.61, 8.19	3.31, - 2.45, 7.35
3	0.5, 0.6,-0.6, 0.6	- 1.32, - 5.26, 5.45	- 0.96, - 5.29, 5.61
4	0.7,-0.8, 0.8, 0.8	2.04, 3.97, 6.72	2.23, 3.57, 6.76

5	0.9,-1.0, 1.0, 1.0	1.76, 4.46, 6.31	1.92, 3.97, 6.12
6	0.2, 0.3, 0.3, 0.3	2.88, - 2.65, 8.27	2.36, - 2.22, 8.65
7	0.4, 0.5, 0.5,-0.5	1.04, 5.03, 6.32	1.21, 4.68, 6.53
8	0.6, 0.7,-0.7, 0.7	- 0.85, - 5.70, 4.46	- 0.63, - 5.68, 4.64
9	0.8, 0.9, 0.9,-0.9	3.63, 3.05, 6.88	3.75, 2.21, 6.41
10	1.0, 1.1, 1.1, 1.1	4.99, 2.86, 1.09	5.01, 1.93, 1.12

As can be seen, the accuracy of z is generally better than that of x and y coordinates, but the maximum error of the x-coordinate (0.43m) is the smallest compare with the maximum error of the z(0.84m) and y(0.93m) coordinate. The errors may come from the following perspectives: Firstly, the observed position of the target is different from its real coordinates in the 3D space since there are noises and errors in reconstruction from two 2D images. Second, we consider the joints as a mass point, not a real object. Volume is ignored which significantly affecting the end effector position. Moreover, the mass center moves as the joint rotates, especially the blue joint. Lastly, even though we got these errors, the FK predictions still matches the observations as expected.

Part3.2 Closed-loop Control (Filename: q3_2.py)

Algorithms:

To obtain the velocity, taking the patrial derivative respect to four joint angles of the matrix obtained in the last question, we can get the Jacobian matrix.

First row will be the velocity of X direction:

θ	$\frac{dx_e}{d\theta}$
θ_1	$3(\cos\theta_1\sin\theta_2\cos\theta_3\cos\theta_4 - \sin\theta_1\sin\theta_3\cos\theta_4 + \cos\theta_1\cos\theta_2\sin\theta_4) - 3.5\sin\theta_1\sin\theta_3$
θ_2	$3(sin\theta_1\cos\theta_2\cos\theta_3\cos\theta_4 - \sin\theta_1 sin\theta_2 sin\theta_4) + 3.5 sin\theta_1 cos\theta_2 cos\theta_3$
θ_3	$3(cos\theta_1 cos\theta_3 cos\theta_4 - \sin\theta_1 sin\theta_2 sin\theta_3 cos\theta_4) + 3.5(cos\theta_1 cos\theta_3 - sin\theta_1 sin\theta_2 sin\theta_3)$
θ_4	$3(\sin\theta_1 cos\theta_2 cos\theta_4 - sin\theta_1 sin\theta_2 cos\theta_3 sin\theta_4 - cos\theta_1 sin\theta_3 sin\theta_4)$

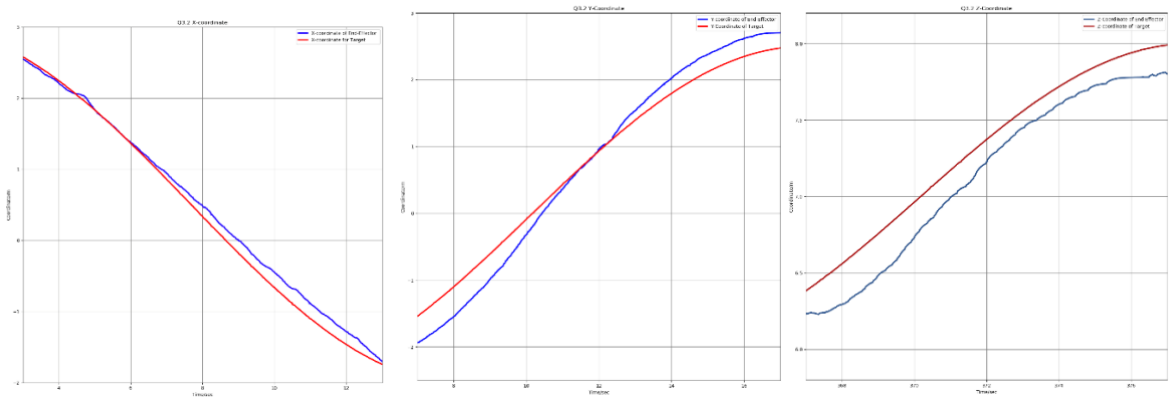
Second row will be the velocity in Y direction:

θ	$\frac{dx_e}{d\theta}$
θ_1	$3(sin\theta_1 sin\theta_2 cos\theta_3 cos\theta_4 + cos\theta_1 sin\theta_3 cos\theta_4 + sin\theta_1 cos\theta_2 sin\theta_4) + 3.5(cos\theta_1 sin\theta_3 + sin\theta_1 sin\theta_2 cos\theta_3)$
θ_2	$3(-\cos\theta_1\cos\theta_2\cos\theta_3\cos\theta_4 + \cos\theta_1 sin\theta_2 sin\theta_4) - 3.5 cos\theta_1 cos\theta_2 cos\theta_3$
θ_3	$3(sin\theta_1 cos\theta_3 cos\theta_4 + cos\theta_1 sin\theta_2 sin\theta_3 cos\theta_4) + 3.5(sin\theta_1 cos\theta_3 - sin\theta_1 sin\theta_2 sin\theta_3)$
θ_4	$3(-cos\theta_1 cos\theta_2 cos\theta_4 + sin\theta_1 sin\theta_2 cos\theta_3 sin\theta_4 - sin\theta_1 sin\theta_3 sin\theta_4)$

Third row will be the velocity in Y direction:

θ	$\frac{dx_e}{d\theta}$
θ_1	0
θ_2	$-3(\sin\theta_2\cos\theta_3\cos\theta_4 + cos\theta_2 sin\theta_4) - 3.5 sin\theta_2 cos\theta_3$
θ_3	$-3 cos\theta_2 sin\theta_3 cos\theta_4 - 3.5 sin\theta_3 cos\theta_2$
θ_4	$-3(\sin\theta_2 cos\theta_4 + cos\theta_2 cos\theta_3 sin\theta_4)$

Plots:



Part4 – Additional Question

Part 4.2 Null-space Control (Filename: q4_2.py)

Algorithms:

The difference of the control algorithm between this and the previous one is that we have added an additional term when returning the next joint angles which will rotate to.

The previous one follows a simple PD control algorithm.

$$\dot{q} = J^{+}[K_d\dot{x}_d + K_p(x_p - x)]$$

Where the $J^{+}$is the pseudo inverse of Jacobian

From the observation, we know that the targets are only reachable by the red effector (the green-red link). Using this redundancy of our robot in the 3D space, we can do the null space control to avoid the orange cube by simply add a term after to maximize the distance between End-Effector and the cuboid.

$$\dot{q} = J^{+}\left[K_d\dot{x}_d + K_p\left(x_p - x\right)\right] + (I - J^{+}J)\dot{q}_0$$

In this project, we make $\dot{q}_0$ be the dot product of the Jacobian and the vector, then we multiply it by a large number to maximize the distance.

As can be seen, after applying *null-space control algorithm*, our End-Effector sticks to the target1 and also keeps away from the cuboid. Moreover, the curve of our end-effector fluctuates much more serious than the previous one.

Plots: