Introduction to Vision and Robotics CW

Arka Purkayastha S1811783
Umut Kartal S1962038

November 2021

1 Workload

`Arka Purkayastha:` Worked on questions 3.1 and 3.2.
`Umut Kartal:` Worked on questions 2.1 and 2.2.
`Github:` https://github.com/arakban/IVR

2 Robot Vision

Given two orthogonal views on the zy plane and the xz plane our mission for this task was to calculate the joint angles for the robot to use. First, we merged the data coming from the two cameras so that we would have a more robust knowledge about the data, also, we could swap between the cameras if one view was blocked, which helped with edge cases. Then, we used OpenCV's "moments" as the vision algorithm to detect the blobs. Assuming that each pixel in image has weight that is equal to its intensity, the algorithm tries to find the center of mass. The formula for the calculations is as follows: $M_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y)$. After this, we used calculus to calculate the joint angles and limited them so that no values outside the ranged would be estimated.

`Vision 1:`

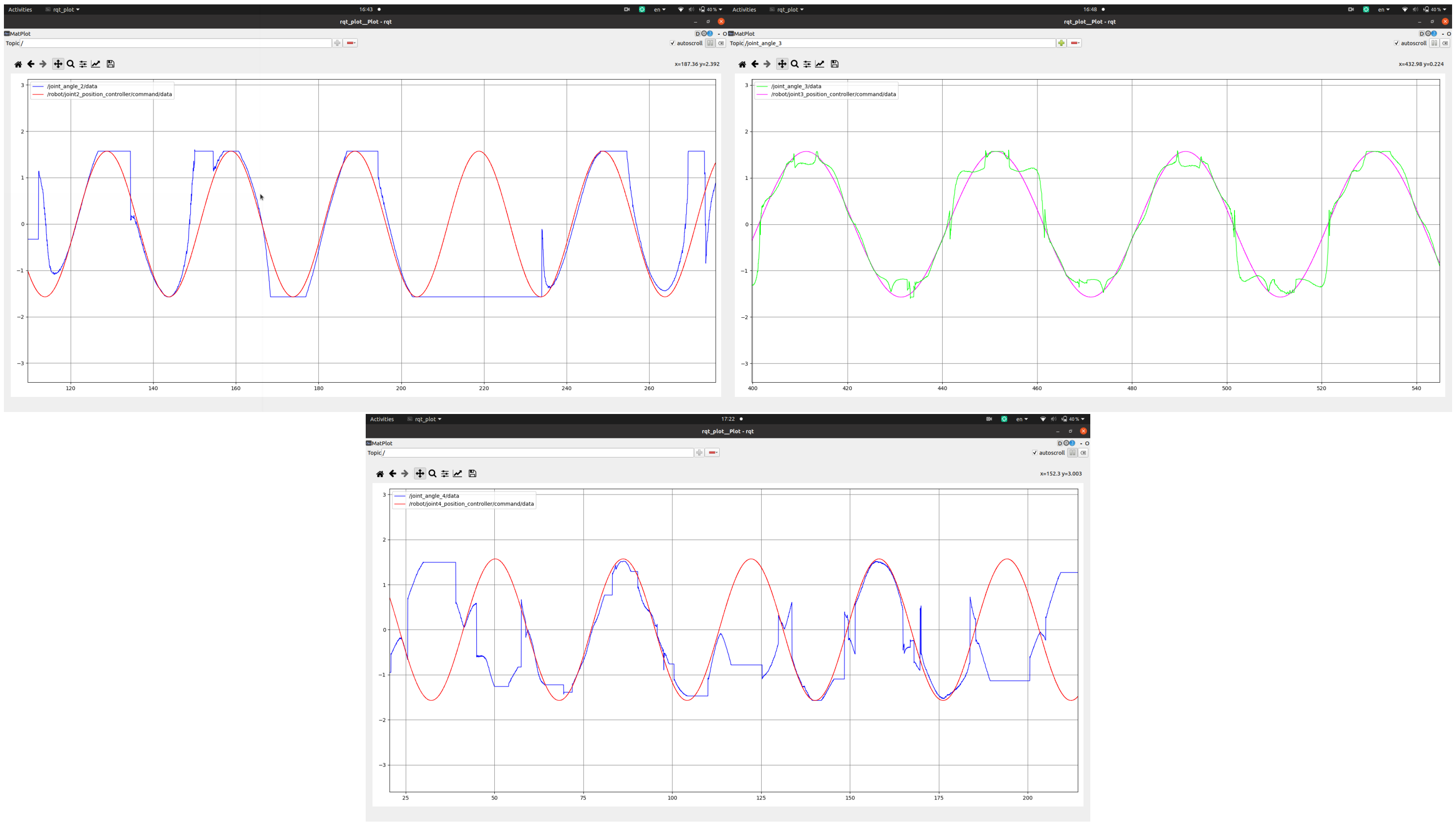


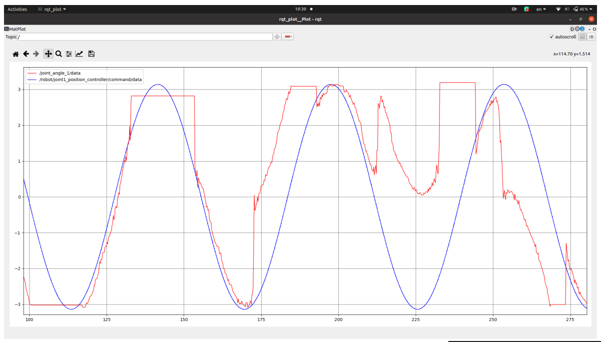
Figure 1: Plots of Joint 2(upper left), Joint 3(upper right) and Joint 4(lower), Perceived vs Correct

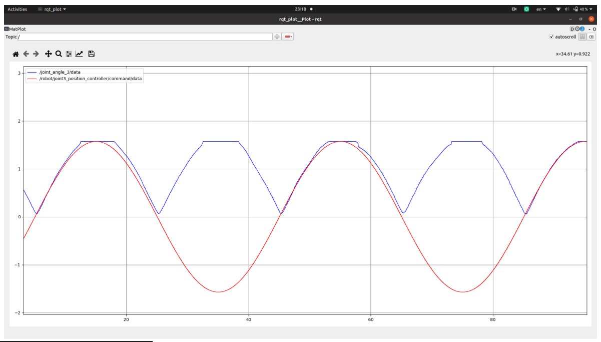
10

For blob detection and perceiving the joints, we used the data coming from the cameras, which triggered the callback. For the variables we needed the vector between between the blue and the yellow centers, and the vector between the red and the blue centers. The first vector is needed to calculate the angle of joint 2 since the joint rotates around the y axis and doesn't get affected by the other rotations. The estimations mostly seem correct, however, as it can be seen in the graph the estimation doesn't rotate, this is due to joint 2 being blocked. For joint 3, the problem was perceiving the sign of the angle, however by using data coming from joint 2 we were able to decide on the sign. Our perceptions are mostly correct and in line with the actual data, however we can see a tangent shape on the peak values. For joint 4 we used the second vector that was mentioned before, the vector between the blue and red centers. Joint 4 was heavily dependant on joint 2 and joint 3, which will lead to the perception being faulty at some points, so while calculating the joint 4 angles we transformed the vector two times to be able to get a clean result for the angle. The graph for the joint 4 angles is the most inconsistent one due to being affected by the limit function, movement of other joints, and being blocked.

Vision 2:

For vision 2, we try to solve a different problem, however, we come across similar approaches and problems. Similar to vision 1, we needed the vector between between the blue and the yellow centers, and the vector between the red and the blue centers. To calculate the joint 1 angles, we had to invert the y-axis of the vector (Blue and yellow) and then get the element-wise arc tangent of the x-axis and the inverted y-axis. The graph was mostly consistent with the actual data, however it had major issues at some points, this could be caused by the effect of the limit function. For joint 3 we had to assume that it was always positive and it was not possible to easily estimate the sign of the angle, thus, even though the estimations were close to perfect, when the real angle went below 0, our estimations gave the positive counterpart, since joint 3 was moving in a 3D space while the other angles are confined to 2D space. For joint 4 we transformed the matrix to get the sign of the actual angle and used that with the actual angle calculations. The graph for joint 4 estimations gave fairly good results, however, similar to joint 1, there some inconsistencies, this could be caused by the false sign estimation of joint 3, since the estimated data is still close to the absolute





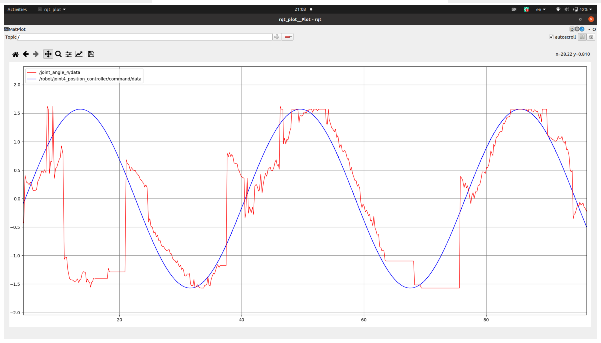


Figure 2: Plots of Joint 1(upper left), Joint 3(upper right) and Joint 4(lower), Estimated vs Correct

value, but in the opposite sign.

9

3 Robot Control

3.1 3.1 Forward Kinematics

For 3.1 we needed to implement the FK formula to estimate the effect of change in joint angles to robot end-effector state. The DH (parameters) table that we had for our robot was

DH parameters

Joint i	θ	α	a	d
1	θ_1	$\pi/2$	0	4
2	θ_2	$-\pi/2$	0	0
3	θ_3	$\pi/2$	3.2	0
4	θ_4	0	2.8	0

with $s(\theta_i) = \sin(\theta_i)$ and $c(\theta_i) = \cos(\theta_i)$ for the forward kinematics calculations we have:

$$x_e = 2.8 * s(\theta_1)s(\theta_3)c(\theta_4) + 2.8 * c(\theta_1)s(\theta_4) + 3.2 * s(\theta_1)s(\theta_3)$$
$$y_e = 2.8 * -c(\theta_1)s(\theta_3)c(\theta_4) - 2.8 * s(\theta_1)s(\theta_4) + 3.2 * c(\theta_1)s(\theta_3)$$
$$z_e = 2.8 * c(\theta_3)c(\theta_4) + 3.2 * c(\theta_3) + 4$$

Results for FK estimated end-effector position for 10 different joint positions:

*J*oint-angle from FK	*J*oint-angled from Images	*E*rror between image and FK
0.796, 0.798, -0.557		-0.095,.1239.994
	-0.891,-0.921,10.55	
0.796,0.8,-0.557		-0.1029,-0.108, 9.995
	-0.898,-0.908,10.55	
1.13, 0.554,-0.911		-5.131,-0.6002,4.0
	-6.261,-1.154,4.91	
0.77, 1.0,-0.794		-0.753, -3.96,5.34,9.994,
	-1.523,-4.961,6.142	
0.796,0.8,-0.557		-0.102,-0.108,9.995
	1.13, 0.554, -0.911	
0.594, 0.922,-0.723		-4.2949, -0.647, 6.57209559
	-4.888,-1.56, 7.295	
1.13,0.554,-0.911		-5.13,-0.60,4.0
	-6.26115281,-1.15427957 4.911	
0.77,1.0,-0.794		-0.75350,-3.96,5.348
	-1.523,-4.961,6.142	
0.708, 1.002, -0.755		-1.813,-3.107,5.876
	-2.521,-4.10910367, 6.631	
0.578,0.752,-0.794		-4.4013,1.526,5.799
	-4.97939,0.7745,6.593	

3.2 3.2 Inverse Kinematics

$$dx_e/dt =$$

$$\begin{bmatrix} 2.8 * c(\theta_1)s(\theta_3)c(\theta_4) - 2.8 * s(\theta_1)s(\theta_4) - 3.2 * s(\theta_1)s(\theta_3) \\ 2.8 * s(\theta_1)c(\theta_3)c(\theta_4) + 3.2 * s(\theta_1)s(\theta_3) \\ 2.8 * c(\theta_1)c(\theta_4) - 2.8 * s(\theta_1)s(\theta_3)s(\theta_4) \end{bmatrix}$$

$$dy_e/dt =$$

$$\begin{bmatrix} 2.8 * s(\theta_1)s(\theta_3)c(\theta_4) + 2.8 * c(\theta_1)s(\theta_4) + 3.2 * s(\theta_1)s(\theta_3) \\ 2.8 * s(\theta_1)c(\theta_3)c(\theta_4) - 2.8 * c(\theta_1)s(\theta_4) + 3.2 * s(\theta_1)s(\theta_3) \\ 2.8 * s(\theta_1)s(\theta_3)s(\theta_4) - 2.8 * c(\theta_1)c(\theta_4) + 3.2s(\theta_1)s(\theta_3) \end{bmatrix}$$

$$dz_e/dt =$$

$$\begin{bmatrix} 2.8 * v(\theta_3)c(\theta_4) + 3.2 * c(\theta_3) \\ 2.8 * -s(\theta_4)c(\theta_4) - 3.2 * s(\theta_3) \\ 2.8 * c(\theta_3)c(\theta_4) \end{bmatrix}$$

$$J = \begin{bmatrix} \mathrm{dx}_e/dt & \mathrm{dy}_e/dt & \mathrm{dz}_e/dt \end{bmatrix}$$

We use this Jacobian matrix to get the relation between joint velocities and end-effector velocities of a robot manipulator i.e. how much each joint needs to move to get to the target position. The error between the current end-effector position and the target position is calculated, and the dot product of the derivative of error and Jacobian is multiplied with dt to get the change in joint angles needed to move towards target.