The work of this submission was not split between two people and was done only by the student who's number is in the file name. The repository for this can be found: https://github.com/yanniknelson/ivr\_assignment

# Robot Vision

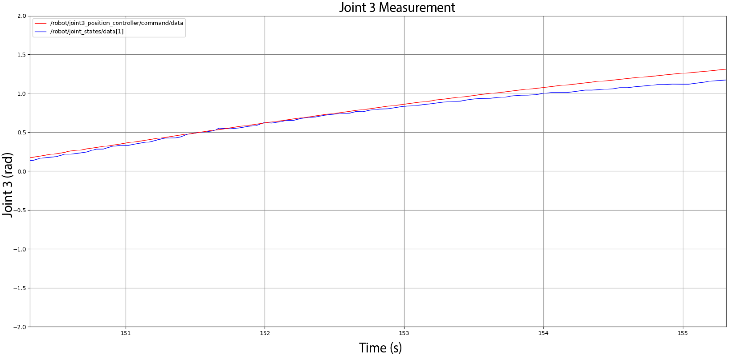
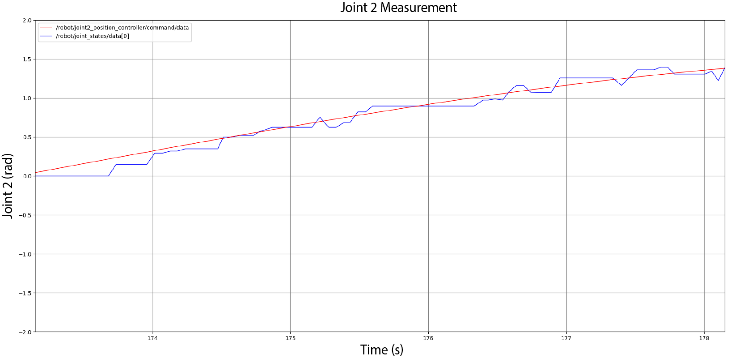
## Joint State Estimation

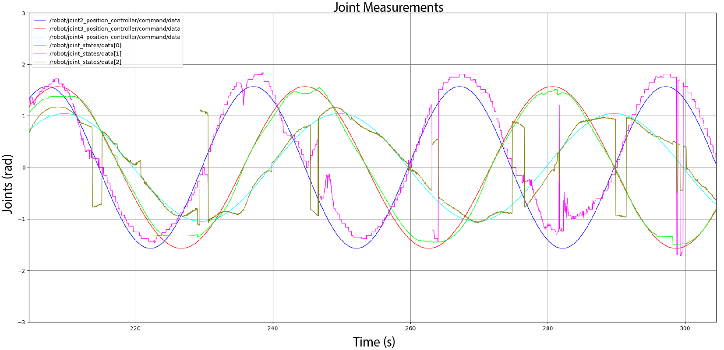
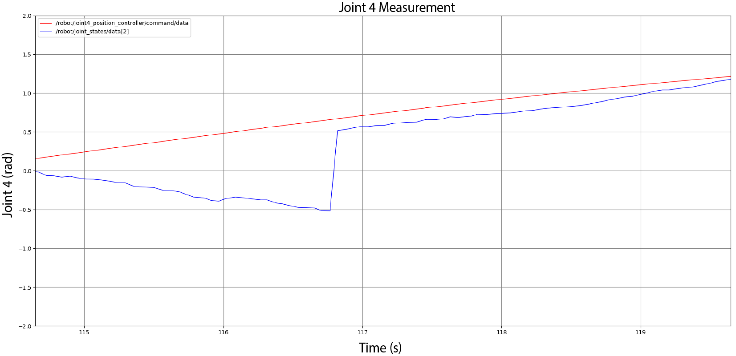
The first step of finding the joint angles involves finding the position of the joint blobs. To do this, I threshold each image to isolate each joint, I then find the moments in the thresholded image (should only have one for each joint) and return the centre of that moment (if there is no moment found then I return the centre of the last moment found for the image plane being viewed and the colour being thresholded).

This then leaves me with in-image coordinates for each joint from each camera, to find the 3D world coordinates relative to the base joint (yellow joint) I first find the pixel to meter scales for each image by taking the distance from the blue joint to the yellow joint and dividing 2.5 by that value. To find the X coordinate I subtract the yellow joint position in the zx image from the position of the desired joint in the zx image, take the image x coordinate of that and time it by the scale for that image. I do the same thing to find the Y coordinate but using the zy image positions and scale instead. Finally to find the Z coordinate perform the same operation for both images taking the image Y coordinate instead and then take the average of the values from the two images (this attempts to deal with perspective but doesn't do very well).

To then find the angle of each joint I attempted multiple methods:

* Joint 2
  + I started by trying to project the vector corresponding the link 3 (the link between the blue and green joints) only the zy plane and then taking the angle between that the [0,0,1] vector (straight up).
  + I then used the kinematic equation describing the position of the green joint which gave the y value = -3.5s(2)c (3), z value = 3.5c(2)c (3) + 2.5. Using these I rearranged them to get 2 this didn't work.
  + I finally took the angle of the vector from where the blue joint should be [0,0,2.5] and the measured position of the blue joint(the blue joint moved as it rotated), this produced the best results.
* Joint 3
  + I take the angle between the link 3 vector and the zy plane, this produces almost exact results.
* Joint 4
  + I take the angle between the link 3 vector and the link 4 vector and then invert the sign based on the relative positions of the blue and green joints and the zx plane. The angle absolute value performs reasonably well, the sign does not, I did not have time to improve this performance.

Here are the plots of the joints in the order of Joint2, Joint3, Joint 4, All for an extended period of time:

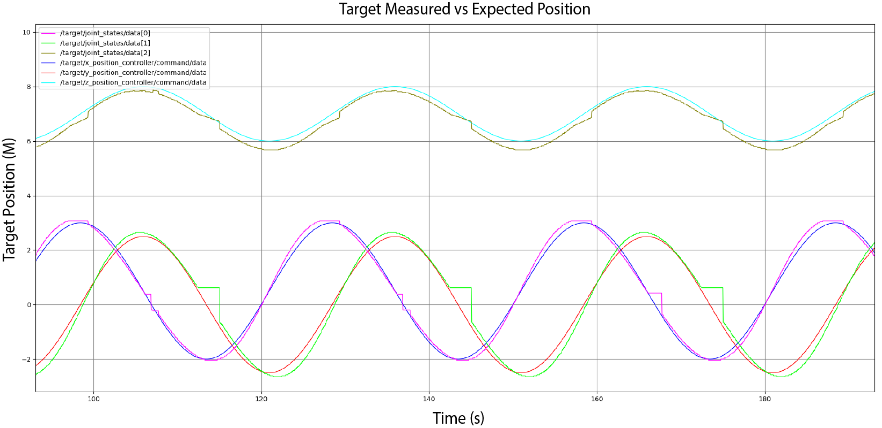


Here you can see that the accuracy of the 3 (measured = green, expected = red) tracking is almost exact save for the times the green joint goes out of view of one of the cameras which produces a flat part in the green plot.\newline\newline

The 2 (measured = pink, expected = blue) tracking performs reasonably well but becomes less accurate and more noisy as 3 gets far from 0, this is due to the perspective causing the Z of the green joint to appear lower than the blue joint, which shouldn't be possible.\newline\newline

The 4 tracking performs reasonably well with the exception of incorrectly jumping sign when 2 is near zero, or when the blue joint is crossing the zx plane, this is because the conditions used to determine the sign of the angle work best when the blue joint is far from the zx plane but become too simple as it gets close, I did not have time to come up with better checks.

## Target Detection

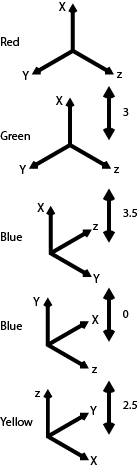
My algorithm works by taking in templates for the target and the box for each view and finding the contours for each at initialisation. My algorithm then thresholds the images received from each camera, uses the cv2.findContours() function to find the contours and the cv2.matchShape() to check each contour against both the box and the target template contours, the result from these comparisons are checked and if the match to the target is greater than the match to the box then the moment for that contour is found, the centre is extracted from that moment and that centre is returned. If the algorithm isn't more sure either of the contours is the target than they are the wall then it will return the last saved position of the target in the respective view. I then find the x y z values from the position in the image as before.

In the plot you can see flat parts and jumps in plots of the measured values. The spikes occur due to instances where partial obstruction of the wall or target causes the shape matching to confuse the target and the orange box. The Flat parts occur due to the obstruction of the target as well, except in these cases the shape matching 'catches' that the box is not the target and so doesn't change the measured position of the target until it's visible again.

# Robot Control

## Robot Kinematics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | α | a | d |  |
| 1 |  | 0 | 2.5 |  |
| 2 |  | 0 | 0 |  |
| 3 |  | 3.5 | 0 |  |
| 4 | 0 | 3 | 0 |  |



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Joint 1 | Joint 2 | Joint 3 | Joint 4 | | |  |  |  |  | | --- | --- | --- | --- | | 1 | 1 | 1 | 1 | | |  |  |  |  | | --- | --- | --- | --- | | -1 | -1 | -1 | -1 | | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.5 | 0.5 | 0.5 | |
| |  |  | | --- | --- | | Expected X | Measured X | | |  |  | | --- | --- | | 5.44 | 5.51 | | |  |  | | --- | --- | | 0.78 | 1.15 | | |  |  | | --- | --- | | 4.42 | 5.31 | |
| |  |  | | --- | --- | | Expected Y | Measured Y | | |  |  | | --- | --- | | 1.64 | 2.57 | | |  |  | | --- | --- | | 5.62 | 6.50 | | |  |  | | --- | --- | | -1.96 | -2.90 | |
| |  |  | | --- | --- | | Expected Z | Measured Z | | |  |  | | --- | --- | | 1.87 | 0.88 | | |  |  | | --- | --- | | 1.87 | 1.99 | | |  |  | | --- | --- | | 6.53 | 7.39 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Joint 1 | Joint 2 | Joint 3 | Joint 4 | | |  |  |  |  | | --- | --- | --- | --- | | -0.5 | -0.5 | -0.5 | -0.5 | | |  |  |  |  | | --- | --- | --- | --- | | 1 | 1 | 0.5 | 0.5 | | |  |  |  |  | | --- | --- | --- | --- | | 1 | 1 | -0.5 | -0.5 | |
| |  |  | | --- | --- | | Expected X | Measured X | | |  |  | | --- | --- | | -0.74 | -0.59 | | |  |  | | --- | --- | | 6.05 | 6.85 | | |  |  | | --- | --- | | 1.59 | 2.41 | |
| |  |  | | --- | --- | | Expected Y | Measured Y | | |  |  | | --- | --- | | 4.78 | 4.88 | | |  |  | | --- | --- | | -0.39 | -0.69 | | |  |  | | --- | --- | | -4.50 | -5.53 | |
| |  |  | | --- | --- | | Expected Z | Measured Z | | |  |  | | --- | --- | | 6.53 | 6.94 | | |  |  | | --- | --- | | 4.20 | 3.97 | | |  |  | | --- | --- | | 6.62 | 7.56 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Joint 1 | Joint 2 | Joint 3 | Joint 4 | | |  |  |  |  | | --- | --- | --- | --- | | -1 | -1 | 0.5 | 0.5 | | |  |  |  |  | | --- | --- | --- | --- | | -1 | -1 | -0.5 | -0.5 | |
| |  |  | | --- | --- | | Expected X | Measured X | | |  |  | | --- | --- | | 4.75 | 5.21 | | |  |  | | --- | --- | | 2.88 | 2.49 | |
| |  |  | | --- | --- | | Expected Y | Measured Y | | |  |  | | --- | --- | | -0.45 | -0.73 | | |  |  | | --- | --- | | 5.34 | 7.20 | |
| |  |  | | --- | --- | | Expected Z | Measured Z | | |  |  | | --- | --- | | 6.62 | 7.52 | | |  |  | | --- | --- | | 4.20 | 4.38 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Joint 1 | Joint 2 | Joint 3 | Joint 4 | | |  |  |  |  | | --- | --- | --- | --- | | 0.75 | 0.75 | 0.75 | 0.75 | | |  |  |  |  | | --- | --- | --- | --- | | -0.75 | -0.75 | -0.75 | -0.75 | |
| |  |  | | --- | --- | | Expected X | Measured X | | |  |  | | --- | --- | | 5.80 | 6.23 | | |  |  | | --- | --- | | 0.12 | 0.53 | |
| |  |  | | --- | --- | | Expected Y | Measured Y | | |  |  | | --- | --- | | -0.53 | -0.88 | | |  |  | | --- | --- | | 5.82 | 6.28 | |
| |  |  | | --- | --- | | Expected Z | Measured Z | | |  |  | | --- | --- | | 4.16 | 3.91 | | |  |  | | --- | --- | | 4.16 | 4.46 | |

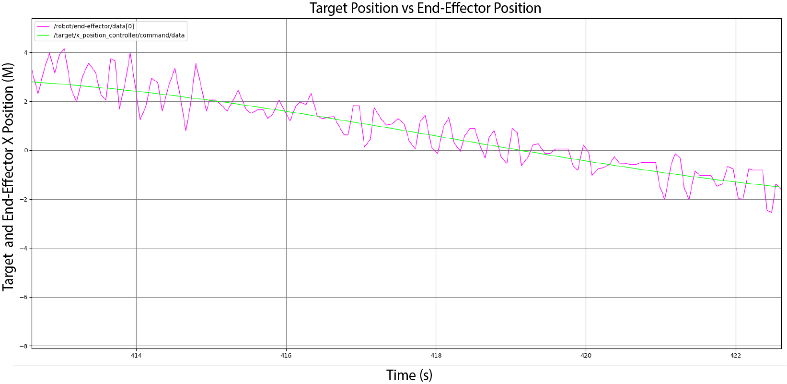
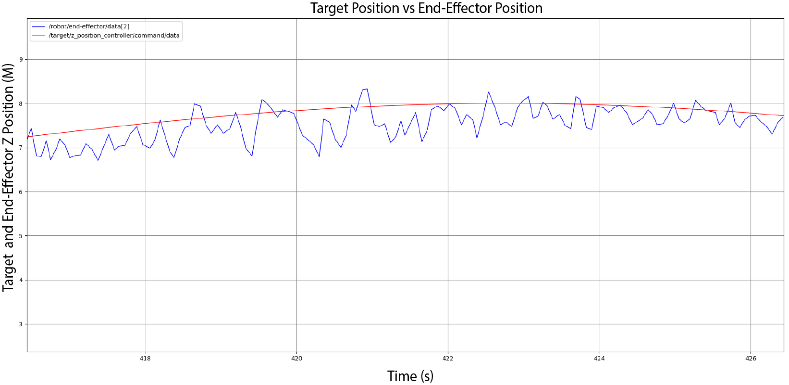
Here you can see that for the most part the FK prediction and measured values of the end-effector are very close, or within 0.5-1 for each coordinate value, this discrepancy occurs due the perspective of the camera causing the pixel to meter conversion to be inaccurate when the robot moves out of the zx and zy planes as well as the perspective causing the z values to be very incorrect when the end-effector is near either of the cameras. Additionally a few of these joint combinations resulted in the end-effector to be obscured to one of the camera, this results in the position being inaccurate as the vision algorithm will simply return the position of the last visible position of the end-effector. I attempted to minimize this error by approaching the position from a visible area in small increments resulting in the last visible position being as close to the actual position as possible (this is the case for the last example).

## Closed Loop Control

But as we aren't measuring joint 1 and don't want to change it we set to 0 to get a new kinematic equation and derive only by to get the Jacobian:

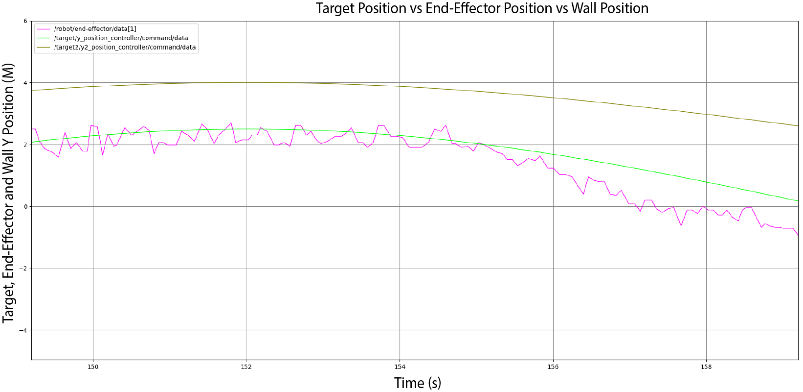
Note, to run this the robot must have at least one joint (2-4) not equal to 0 and all joints must be visible to both cameras when the code starts.

Performance X accuracy, Y accuracy, Z accuracy:



# Null-Space Control

This algorithm works similarly to the closed control in that it performs the same task as the closed control, but it also uses the null-space projection and a secondary task to try and maximise the distance between the 'wall' and the end-effector while still following the target. Note, to run this the robot must have at least one joint (2-4) not equal to 0 and all joints must be visible to both cameras when the code starts

Performance X accuracy, Y accuracy, Z accuracy: