IVR Coursework 1

Austin Pan (s1870157) & Angus Stewart (s1902147)

1 Contributions

Joint State Estimation: Angus Stewart
Robot Control: Austin Pan
Null-space Control: Austin Pan & Angus Stewart
Github link: `https://github.com/Siliconlad/ivr_assignment`

2 Joint State Estimation

Note: for joint 4 we use $\frac{\pi}{3}\sin(\frac{\pi}{20}t)$ instead of $\frac{\pi}{2}\sin(\frac{\pi}{20}t)$ as permitted in the follow-up to this Piazza post.

First we do joint detection on the images from camera 1 & 2 (in `image1_processor` and `image2_processor`)

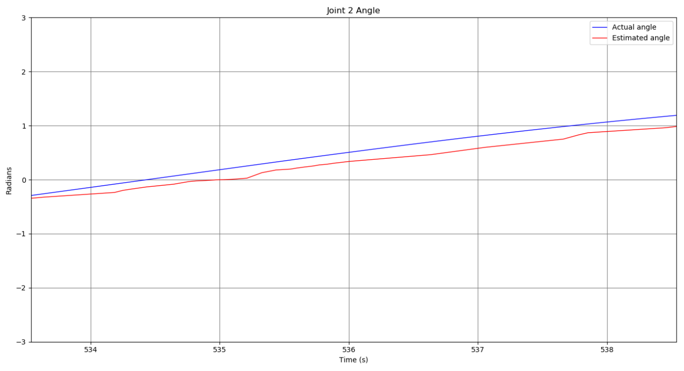
1. Threshold the HSV image based on the joint colour find the contour and its center. Take the average of the centres if there's more than one contour. If there are no contours the joint is hidden.
2. If the yellow joint has not been detected before store the center and calculate `pixel_to_meters`.
3. Shift the centers to be around the yellow center and convert to meters.

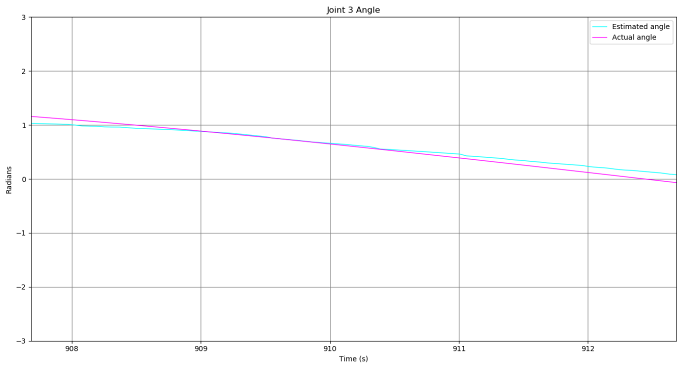
The centers (or `hidden=True`) are published to `fusion` to determine the 3D center of each joint. If:

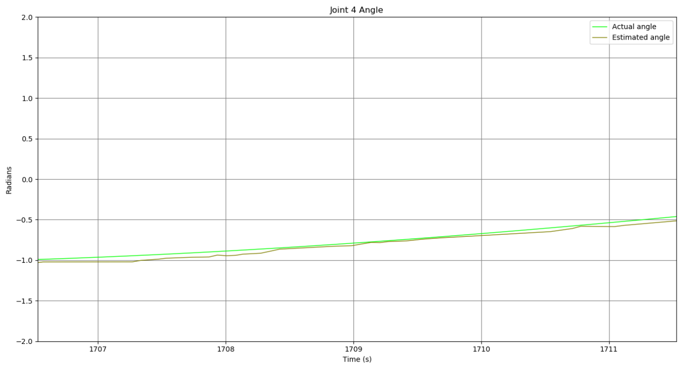
1. **joint centers are received from both images**. The results are combined in the obvious way.
2. **one of the joint centers is hidden**. Assume the green joint is hidden in image 1. We find the closest non-hidden object (out of the other joints and the two orange targets) in image 1 to the position $(y_{prev} z)$ where y_{prev} is the previous y value of the green joint and z is the z value of the green joint from image 2. The new y is the average of the previous y value and the y of the closest object. The x and z values are taken from image 2. The same idea is used for the other objects and for image 2.

3. **both of the joint centers are hidden** then return the previous position of the joint

Finally in the `joint_angles` node we take the 3D positions of the green and red joint from the `fusion` node and using the forward kinematics equations calculate the joint angles.







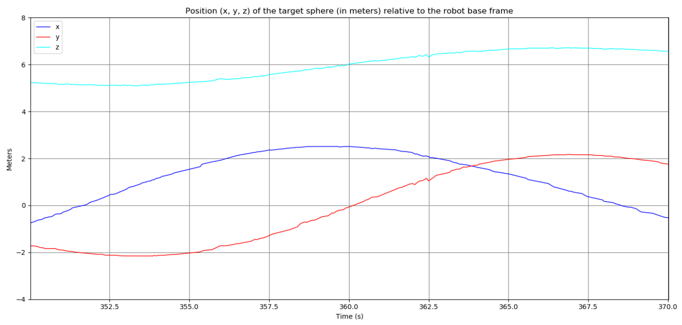
3 Target Detection

To get the position of the target sphere we first detect the sphere in the images from camera 1 and 2:

1. Threshold the image in the same way as in section **??** with the orange color.
2. Use preselected templates of the target sphere and box and match these to the thresholded image. From this we obtain the centre of the areas matched by each template as well as a numerical measure of how well the template matched the area.
3. If the centers are within a certain distance of each other we consider the templates to have matched the same object hence the object corresponding to the template with the lower score must be hidden.
4. The centers are then shifted and converted to meters as described in section **??**.

The results are then combined to form the 3D position in the same way as described in section **??**.
Some sources of error for our position estimate of the target sphere include:

- **Template matching errors** - Because the shape of the sphere is fixed in the template as the sphere grows or shrinks in size template matching may find it harder to match the template accurately. This can be seen by the wobbling of the box around the target in the image pop-ups.
- **Perspective error** - Our algorithm assumes that the target is perfectly orthogonal to both cameras at all times. However this is not realistic (because the target noticeably grows and shrinks) and hence will introduce some amount of error into our estimates.
- **Processing delays** - There may be small errors due to delays from the processing time of our algorithm.



4 Forward Kinematics

Our forward kinematics equation is given by

$$\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} = \begin{bmatrix} 3.5s(\theta_1)s(\theta_2)c(\theta_3) + 3.5c(\theta_1)s(\theta_3) + 3s(\theta_1)c(\theta_2)s(\theta_4) + 3s(\theta_1)s(\theta_2)c(\theta_3)c(\theta_4) + 3c(\theta_1)s(\theta_3)c(\theta_4) \\ 3.5s(\theta_1)s(\theta_3) - 3.5c(\theta_1)s(\theta_2)c(\theta_3) + 3s(\theta_1)s(\theta_3)c(\theta_4) - 3c(\theta_1)s(\theta_2)c(\theta_3)c(\theta_4) - 3c(\theta_1)c(\theta_2)s(\theta_4)) \\ 3.5c(\theta_2)c(\theta_3) + 3c(\theta_2)c(\theta_3)c(\theta_4) - 3s(\theta_2)s(\theta_4) + 2.5 \end{bmatrix} \tag{1}$$

Below is a table of 10 different configurations of the robot joints and the corresponding estimates of the end-effector position using forward kinematics and computer vision (from section **??**).

Joint Configuration				Forward Kinematics			Computer Vision		
θ_1	θ_2	θ_3	θ_4	x	y	z	x	y	z
0.5	0.5	0.5	0.5	4.422	-1.962	6.534	5.259	-2.788	6.691
0.2	1.0	0.2	1.0	2.107	-5.274	3.087	3.250	-6.304	2.050
1.0	1.0	0.4	0.4	5.934	-0.911	4.634	6.575	-1.315	4.196
0.2	0.4	0.6	0.8	3.844	-3.076	5.911	4.989	-4.108	5.859
0.8	0.6	0.3	0.1	4.022	-1.235	7.444	4.599	-1.586	7.793
-0.9	0.6	-0.7	0.3	-5.276	1.049	6.018	-5.298	0.696	5.801
-0.2	-0.8	-0.6	0.2	-2.889	4.052	6.631	-2.592	3.518	6.323
1.2	-1.0	0.7	-0.3	-2.779	5.481	4.385	-2.425	4.834	4.351
-0.3	-1.2	0.8	-0.7	5.290	3.035	2.162	4.602	4.834	1.354
1.1	1.1	-1.1	-1.1	-1.294	-4.202	5.883	-1.508	-4.177	5.879

On average, the distance of the forward kinematics prediction from the image processing prediction was about 1.036 meters.

5 Closed-loop Control

$J = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ B_{11} & B_{12} & B_{13} & B_{14} \\ 0 & C_{12} & C_{13} & C_{14} \end{bmatrix}$ (2) where

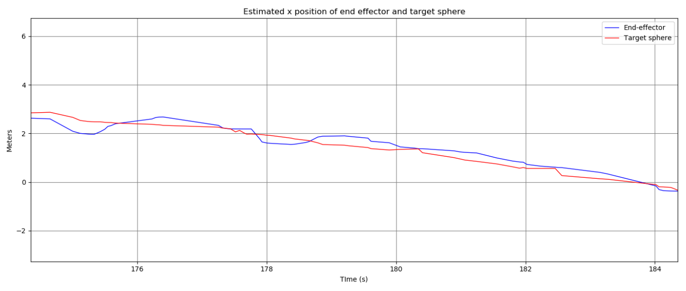
$A_{12} = (7c(\theta_2)c(\theta_3)s(\theta_1))/2 - 3s(\theta_1)s(\theta_2)s(\theta_4) + 3c(\theta_2)c(\theta_3)c(\theta_4)s(\theta_1)$

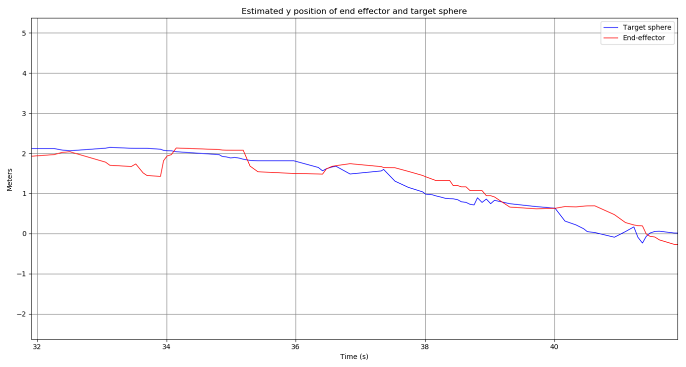
$A_{13} = (7c(\theta_1)c(\theta_3))/2 + 3c(\theta_4)(c(\theta_1)c(\theta_3) - s(\theta_1)s(\theta_2)s(\theta_3)) - (7s(\theta_1)s(\theta_2)s(\theta_3))/2$

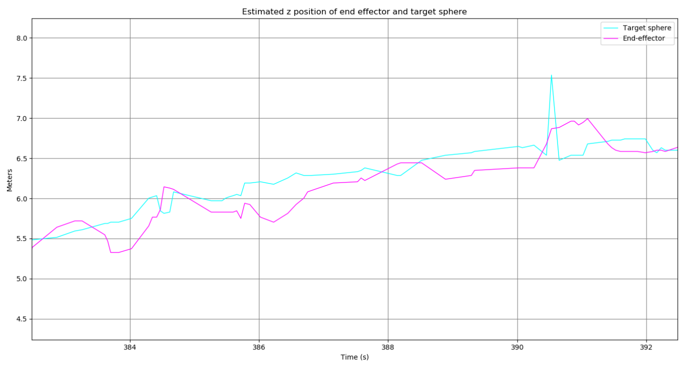
$A_{14} = 3c(\theta_2)c(\theta_4)s(\theta_1) - 3s(\theta_4)(c(\theta_1)s(\theta_3) + c(\theta_3)s(\theta_1)s(\theta_2))$

$B_{11} = (7c(\theta_1)s(\theta_3))/2 + 3c(\theta_4)(c(\theta_1)s(\theta_3) + c(\theta_3)s(\theta_1)s(\theta_2)) + (7c(\theta_3)s(\theta_1)s(\theta_2))/2 + 3c(\theta_2)s(\theta_1)s(\theta_4)$

$$B_{12} = 3c(\theta_1)s(\theta_2)s(\theta_4) - (7c(\theta_1)c(\theta_2)c(\theta_3))/2 - 3c(\theta_1)c(\theta_2)c(\theta_3)c(\theta_4)$$
$$B_{13} = (7c(\theta_3)s(\theta_1))/2 + 3c(\theta_4)(c(\theta_3)s(\theta_1) + c(\theta_1)s(\theta_2)s(\theta_3)) + (7c(\theta_1)s(\theta_2)s(\theta_3))/2$$
$$B_{14} = -3s(\theta_4)(s(\theta_1)s(\theta_3) - c(\theta_1)c(\theta_3)s(\theta_2)) - 3c(\theta_1)c(\theta_2)c(\theta_4)$$
$$C_{12} = -(7c(\theta_3)s(\theta_2))/2 - 3c(\theta_2)s(\theta_4) - 3c(\theta_3)c(\theta_4)s(\theta_2)$$
$$C_{13} = -(7c(\theta_2)s(\theta_3))/2 - 3c(\theta_2)c(\theta_4)s(\theta_3)$$
$$C_{14} = -3c(\theta_4)s(\theta_2) - 3c(\theta_2)c(\theta_3)s(\theta_4)$$







6 Null Space Controller

To utilize the robot's redundancy to make sure the robot doesn't hit the orange box while the end effector still follows the sphere, we use a secondary objective: the distance to the box.

1. Find changes in angle and changes in distance between end-effector and box based on previous time step.

2. Take numerical derivative of distances with respect to changes in angle to get q0.

3. Multiply by null space projection matrix and add to end-effector control input.