IVR Assignment
B156771 & [exam no. here]

1 Vision

1.1 Estimating Joint Angles

1.1.1 Methodology

The algorithm employed makes use of the two orthogonal projections provided by the cameras to provide a 3D point/vector for each joint, and therefore requires both `image1.py`, and `image2.py` to be run simultaneously. The two cameras estimate the point of the joint in their respective projection using the joints colour and a blob-detection approach. In the event that a joint is hidden from one of the camera, its previous position is sent on the stream instead. Furthermore, the estimated points are transformed so that the origin is given by joint 1 – the yellow joint at the base – and then translated to metres from pixel coordinates. These estimates along with the plane it represents are sent to a third program (`consume_images.py`), which reads the points off these streams and combines them using the value for the z position, which is the common axis. Care had to be taken due to the multiple streams and asynchronous nature of the processing. For this, a thread lock was used to minimise the possibility of race conditions affecting the readings. With the joint positions collected and merged effectively, the angle is calculated using the definition of the dot product of vectors. I.e., given two joint's positions $\vec{j_1}$ and $\vec{j_2}$, where the vector given by the link between them is given by $\vec{l_{1,2}} = \vec{j_2} - \vec{j_1}$; its angle with respect to $\vec{u_k}$, the unit vector along the k axis, is given by

$$\theta = \arccos\left(\frac{\vec{l_{1,2}} \cdot \vec{u_k}}{||\vec{l_{1,2}}||}\right) - \frac{\pi}{2}.$$

The angle is shifted by a factor of $\pi/2$ to account for the range of motion of the joints. The results of this process is given below (estimates are in blue, actual angles are in red):

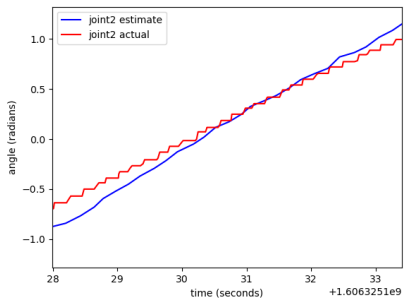

Figure 1: Joint 2 estimates vs actual position

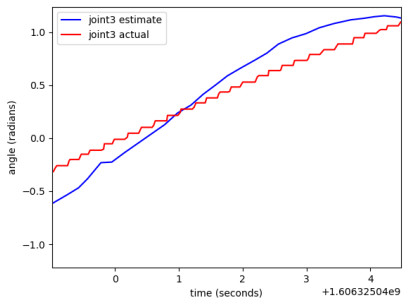

Figure 2: Joint 3 estimates vs actual position

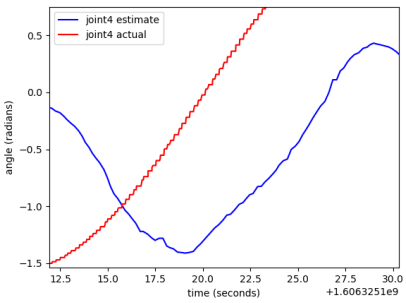

Figure 3: Joint 4 estimates vs actual position

1.1.2 Potential Improvements

From the figures above, it is clear that the estimate of the fourth joint leaves much to be desired. We would hypothesise that the primary source of error for this joint's estimated angle would be the handling of the joint being out of sight of the camera. Given its positioning on the robot, it would be the most likely out of the four joints to be hidden from view at any given point. Perhaps instead of assuming the previous position of the joint when it disappears from view, we could instead perform a linear interpolation between the last recorded position, and the next recorded position – effectively taking the average between the two. Taking this a step even further, we could potentially use a more advanced interpolation technique and estimate the trajectory while taking position measurements, substituting these approximations when an actual value cannot

be determined. While this would lead to potentially smaller errors, we would need to consider the requirements of the application. Should speed of processing be more important, the method employed would be more suitable as the number of operations to perform per measurement would be minimal. If, instead, accuracy were more critical, then this method may prove to be more fruitful despite being more computationally expensive.

1.2 Target Detection

For this task we extended the work done in the previous part. We isolate the two potential objects using the orange colour range and blob detection. From there, we use a template of the orange sphere taken from one of the cameras to isolate the sphere's position. The centroid of the two potential matches are taken, and then a cropped sample of each retrieved from the frame. From there, each sample is compared with the template. The sample with the smaller distance is assumed to be the sphere, and its coordinates, along with a time stamp is returned. After normalizing the position with respect to joint 1 or the base of the robot, the positions estimated from each camera is sent on the stream `target_est` which is read from third script, `track_target.py`. Much like in the previous part, the two coordinates are joined along the z position where the difference in time stamp is used to determine whether or not two points from each camera correspond to the same reading. In this case as well as in the above part, a difference of less than 10 milliseconds is needed for two readings to be considered separate parts of the same point. For the z position, the former of the two readings is used; and a thread lock prevents any potential race conditions. Lastly, the actual positions are read from their respective streams: `target/x_position_controller/command`, and its counterparts. We note that the script `target_move.py` was not used. This script caused the target to move erratically leading to illegible plots as well as inaccurate readings. Since the target and its decoy were already following a sinusoidal movement around the robot, it was decided that the script was not needed. Below we plot the estimated positions per dimension along with the actual reading as given by their respective streams. We can see that the readings for the x and y estimates seem

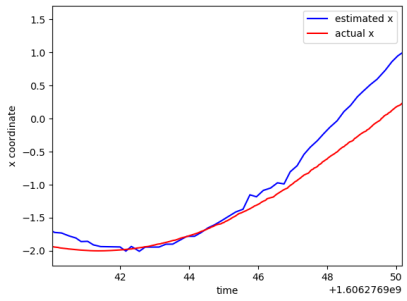

Figure 4: Estimated x coordinate

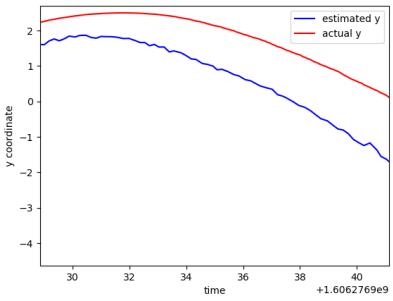

Figure 5: Estimated y coordinate

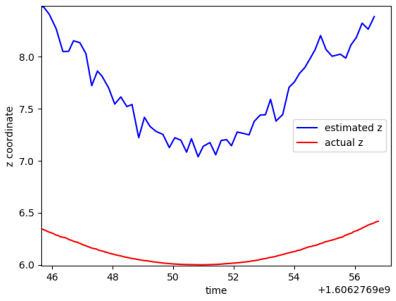

Figure 6: Estimated z coordinate

to be decently accurate, with an error of less than a metre. The z estimate; however, is once again difficult. Having an average error of about three metres, this reading is clearly too inaccurate. Nevertheless, the path that the estimate takes is similar if not equal to that of the actual readings, meaning that the majority of the error is likely caused by a miscalculation in either the normalization of the position or the conversion from pixel to metres. Given that the other estimates seem to line up fairly decently, we would expect that the error would lie in the normalization process.

2 Joint 1 Movements

The algorithm here will follow very similarly to that detailed in section 1. The difference here will be in determining the angle of joint 1 as it is along the z axis, and so the same method will not be as fruitful. The key observation here is that if we look at the robot top-down, we notice that the link between joints 2/3 and 4 show the angle by which joint 1 is rotated. The important assumption is that joint 1 is precisely at the origin; however, we insured this was the case as explained in section 1. We used the data as it was collected in section 1, and this time took the x, y coordinates of joint 4 and passed it into the $\arctan 2$ function. We then used the

same method as above in determining the angles for the remaining joints. This process yielded the following plots:

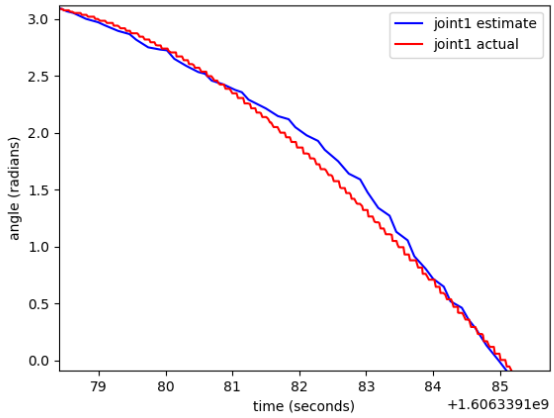

Figure 7: Estimated joint 1 angle

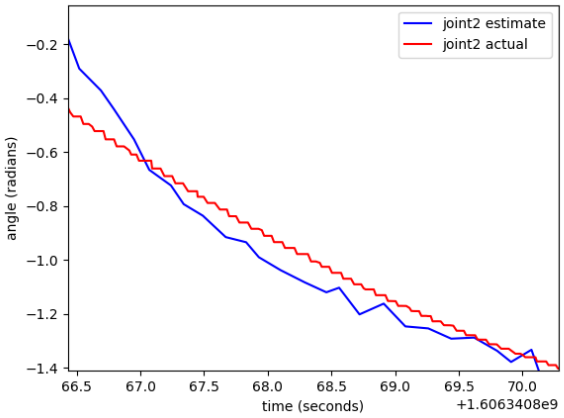

Figure 8: Estimated joint 2 angle

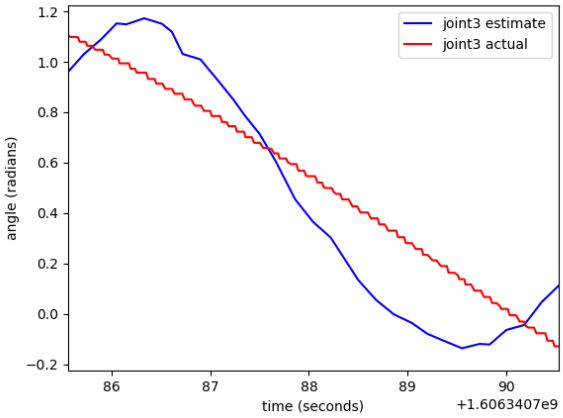

Figure 9: Estimated joint 3 angle

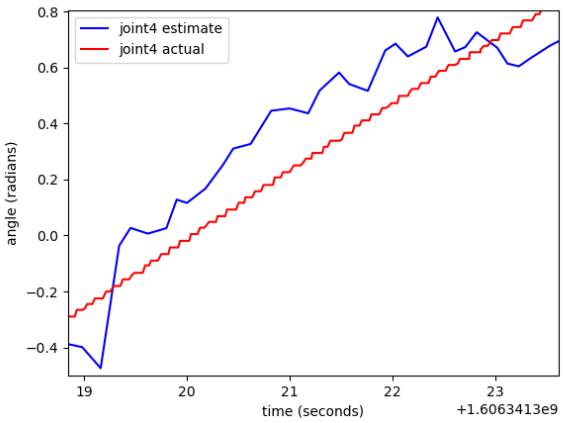

Figure 10: Estimated joint 4 angle

The results we found for joint 1 were exactly as we had hoped. The remaining joints, while some suffered larger errors, were able to retain sufficient accuracy for short periods of time. In the case of joint 4, compared to the plot in section 1, we would argue that it is now more accurate than before, which is an unexpected, but not unwelcomed result. We should note; however, that for extended periods of observation, errors around the local maximum and minimum were much more pronounced. This is certainly due to the increased degree of freedom in the robots movements, as well as the inherent loss of information given by the projection from 3 to 2 dimensions. While the angle between the joints may be moving as determined by their respective sinusoidal functions, because multiple joints are moving freely, when projected onto two dimension, the angle would give the appearance of either not changing at all, or changing only slightly. This would result in the more flattened areas near the local extremes. Furthermore, by rotating around the z axis, the changes in the angle may not always be visible, depending on the degree to which joint 1 is rotated. These two facts combines make it significantly more challenging to accurately determine the angle of the joints.