Report DIP: Validating the virtual controller trajectory using camera measured trajectories and Detecting differences between the motions of the 7 recordings.

This report discusses the code for validating the virtual controller trajectory using camera measured techniques. For the analysis, only the first chunk was considered of both the video data and the data from the csv file due to time and resource constraints. Additionally, the comparison in a broad sense between two recordings are discussed on the last pages.

The code consists of 6 parts. I will not go in depth about each part because some of them are quite self-explanatory. These parts are:

- Reading and loading the video data (1)
- Tracking the video and getting the theta values (2)
- Getting the angular velocity values and plotting them (3)
- Importing and plotting the values from the csv file (MATLAB generated) (4)
- Resampling and synchronizing all the values (5)
- Calculating the RMS values between the theta and velocity values (6)

I will discuss parts 2, 5 and 6.

Tracking the video and getting the theta values

A fixed-point A, the reference point, and a fixed-point B, the tracking point are hard coded. The point A represents the fixed handle marked by the red circle and the point B represents the bearing marked by the greed dot, shown in figure 1.

Corners are points in an image where two edges meet, and they exhibit strong intensity variations in multiple directions. Harris Corner Detection is excellent at identifying such points because it analyzes changes in pixel intensities around a small neighborhood. This makes corners ideal for tracking, as they are distinct and easily distinguishable, even when the object moves or rotates across frames.

The goal is to track a single distinct point (Point B) on a rod and calculate its trajectory, Harris Corner Detection provides sufficient accuracy without unnecessary complexity. More advanced techniques like SIFT or SURF are unnecessarily computationally heavy without having a better outcome in our tracking process.

We calculate the difference between the reference point and the tracking point during each frame and calculate the angle between the resulting vector and the negative Y-axis. A wait bar is displayed to show the progress of code.

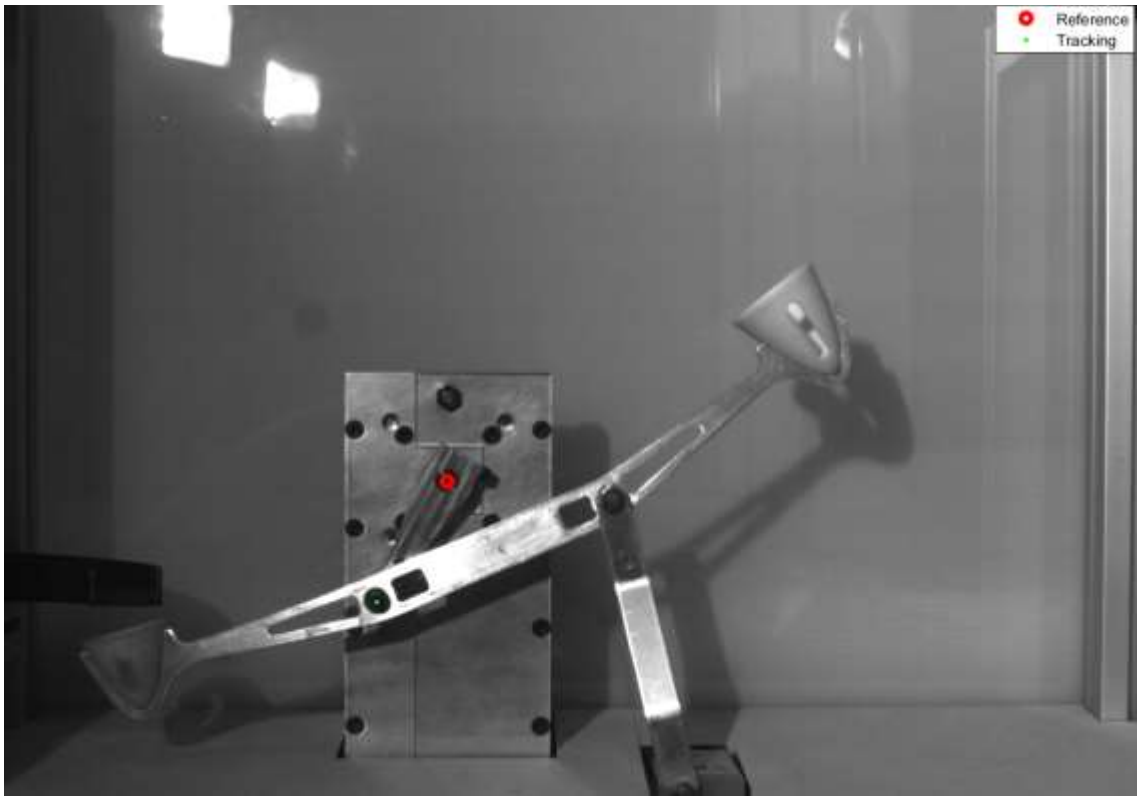

figure 1: reference point (red) and tracking point (green)

We get the following values.

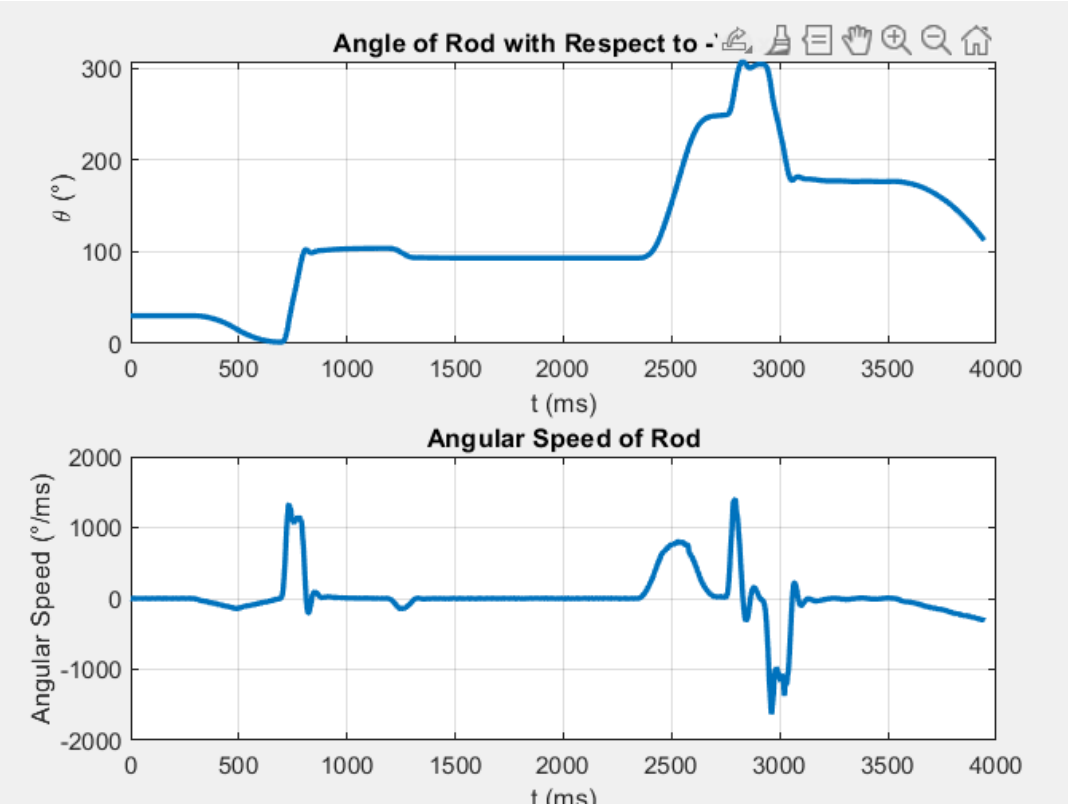

figure 2: theta plotted over time (top) and angular velocity plotted over time (bottom)

We get pretty much the same graphs when getting the values from the csv file and plotting them.

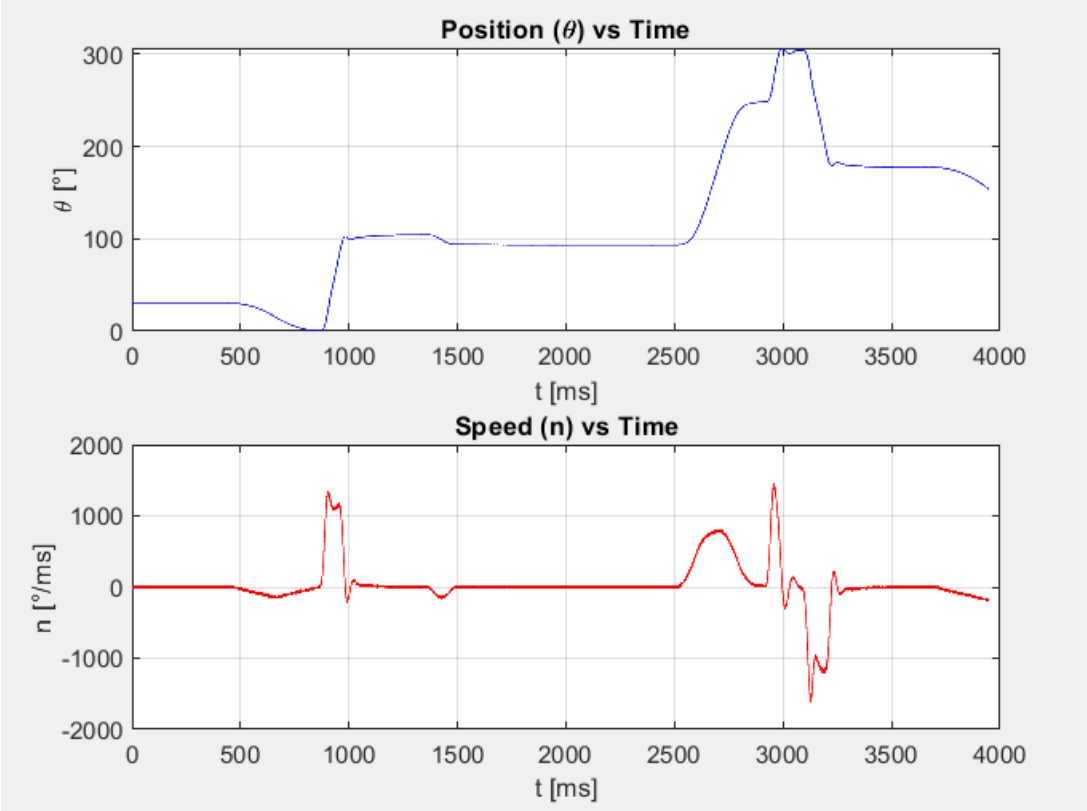

Figure 3: theta plotted over time (top) and angular velocity plotted over time (bottom) [csv]

Resampling and synchronizing all the values

At first glance, you can already see the misalignment. The values from the csv file also need to be resampled. This part is, however, a little tricky. The csv file contains the whole video which is about a minute long. The video is a loop of 8 parts where the mechanical structure repeats the same movement. I have taken the first part to compare with our video. There is a slight problem however, and I will explain as follows.

The video is samples at a rate of 4000 Hz, which means a sample every 0.25 ms. For the whole video, this gives us 143712 samples which comes down to the video being 35.93 seconds long. Dividing this into 8 chunks gives 4.49 s per chunk. Our video, however, is only 3.95 s long, 1500 frames / 380 fps. That is why in the initial comparison of the two theta positions, our theta line ended quicker than the one from the csv file. To resolve this, I did the following.

I am assuming that the controller trajectory and the video start at the same time, but I have nothing to prove this. I took the sample time to get the position values from the csv file as 3947 (ms), which gives us a timevector of 15789, 3947 ms * 4000 Hz. My assumption was confirmed by the *resample* function, that when first given a vector with values corresponding to 4.49 s, it gave me resampled value of 1707, now gave me a resample value of 1500, which was what I needed (because the smaller value is 1500).

I first had to resample the data to match each other. I downsampled the data from the csv file to 1500 values. Then, I had to synchronize the data, but the cross-correlation function did not work properly (or I did). By just looking at the graph, you could see a misalignment of about 100 points, but the correlation function gave a 550. I decided to use a simple method of comparing maxima and finding the difference of the index and this gave a 64-point misalignment. I shifted the video data 64 points to the right by padding zeros at the start, which also meant that the first 64 values after comparison should not be considered.

The resampled and synchronized values are shown below.

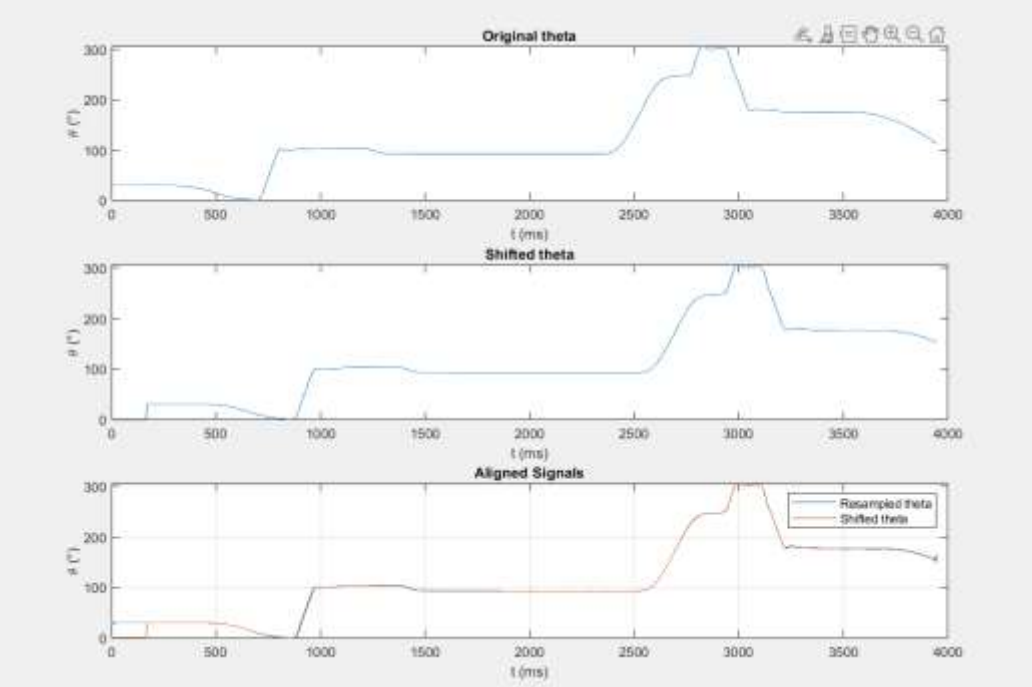

Figure 4: the theta values of the video are shifted 64 points to the right.

Calculating the RMS values between the theta and velocity values

Finally, we can compare the two theta and velocity values. I used Root Mean Squared Error (RMSE) for this task for the following reasons

- Error sensitivity
  RMS penalizes larger errors more than smaller ones due to squaring, which is desirable when validating a trajectory, as large misalignments or deviations are more critical.
- Smoothness of errors
  For continuous signals like angular positions or velocities, RMS provides a reliable and stable measure of how the two signals compare across their entire length.

When plotting the datasets on top of each other, we get the following plots.

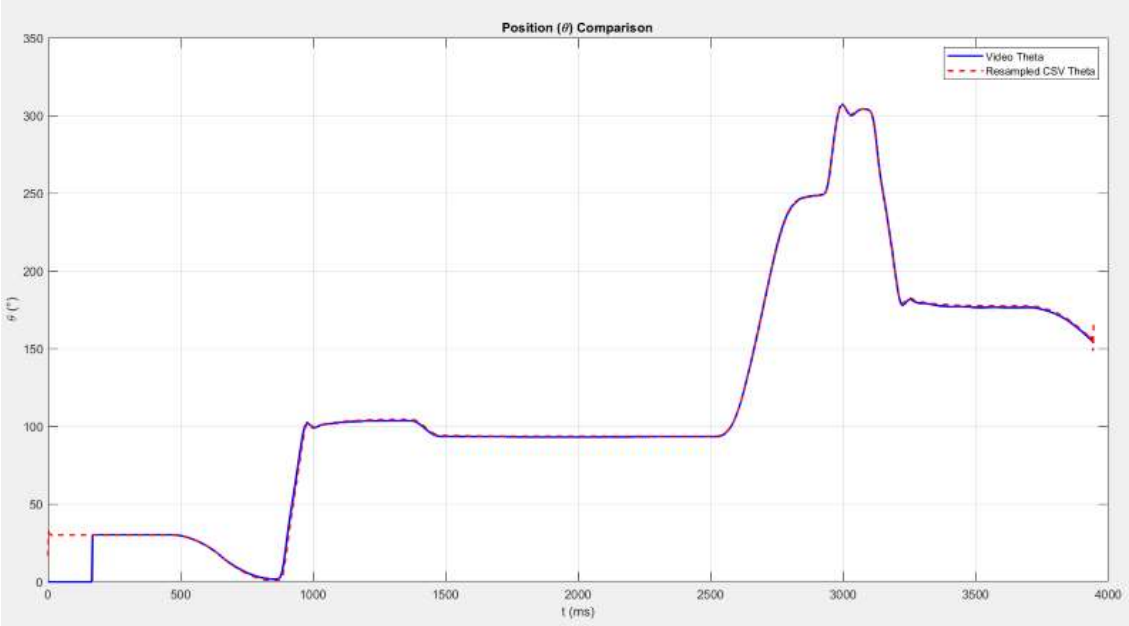

Figure 5: theta comparison

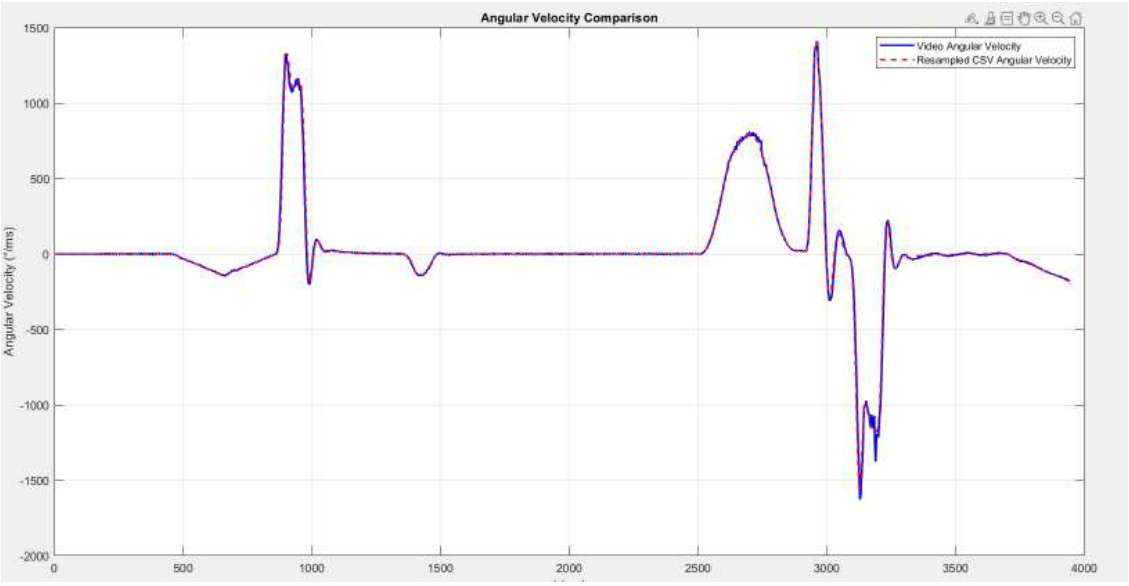

Figure 6: angular velocity comparison

It is not clear on the images, but the values do not align very well, especially at the transitions. More specifically, the resampled values from the csv file do not align with the values from the video values. The reason for this phenomenon is unclear to me, but I am assuming that it has something to do with either the resampling or the synchronization. This is also visible on the error graphs.

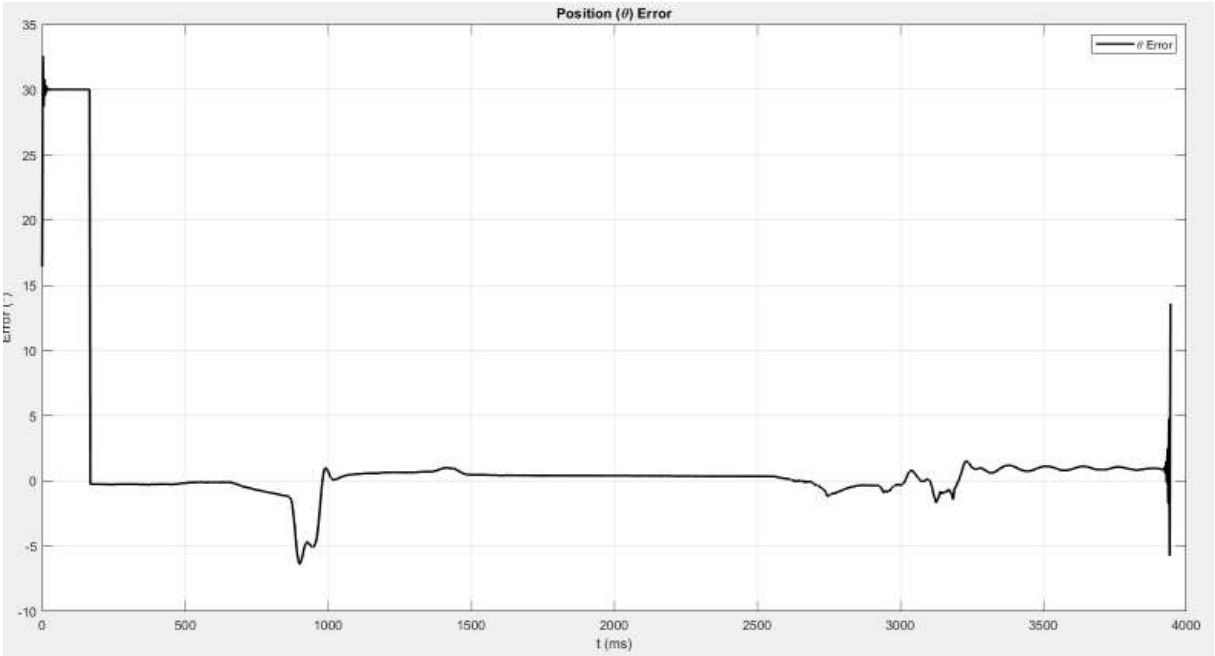

Figure 7: theta error

Not taking into account the first 64 values (because they are zero padded), we can see that the two values align quite well except for the small discrepancies just before the 1000 ms time mark and the discrepancies around the 3000 ms time mark. Just as I discussed above, that is because, at the transition (the slope), the values of the csv file are a little ahead of the video values. The same error can be seen in the angular velocity error graph.

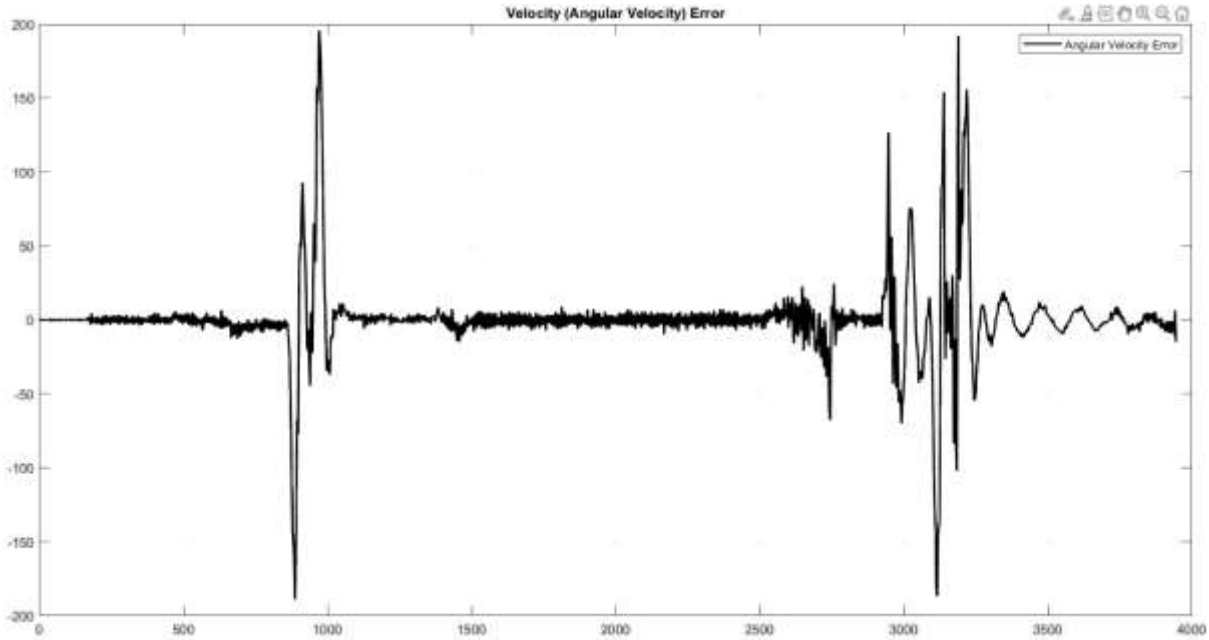

Figure 8: angular velocity error

The error seems way more significant in this graph than in the theta error graph, but they are not. This is a classic example of the RMSE amplifying the bigger value (because of the square).

If, however, we would assume that there was no error in the resampling or synchronization (which could very well be the case), than we can conclude from these graphs the deviation from the virtual controller trajectory. **We can then validate that machine does not perfectly follow the trajectory that it should and that it was given.**

Comparison between two video chunks

We were asked to additionally detect differences between the motions of the 7 recordings. I chose to just compare recording 1 and 3 due to time and resource constraints. The reason for not choosing recording 2 is because it has only 1300 frames for some reason. I compared, just like previous report the, trajectory and the velocity of the tracked point to a reference point. The image with the points for first frames can be seen on *figure 1*. Figure 9 shows the points for frames of recording 3.

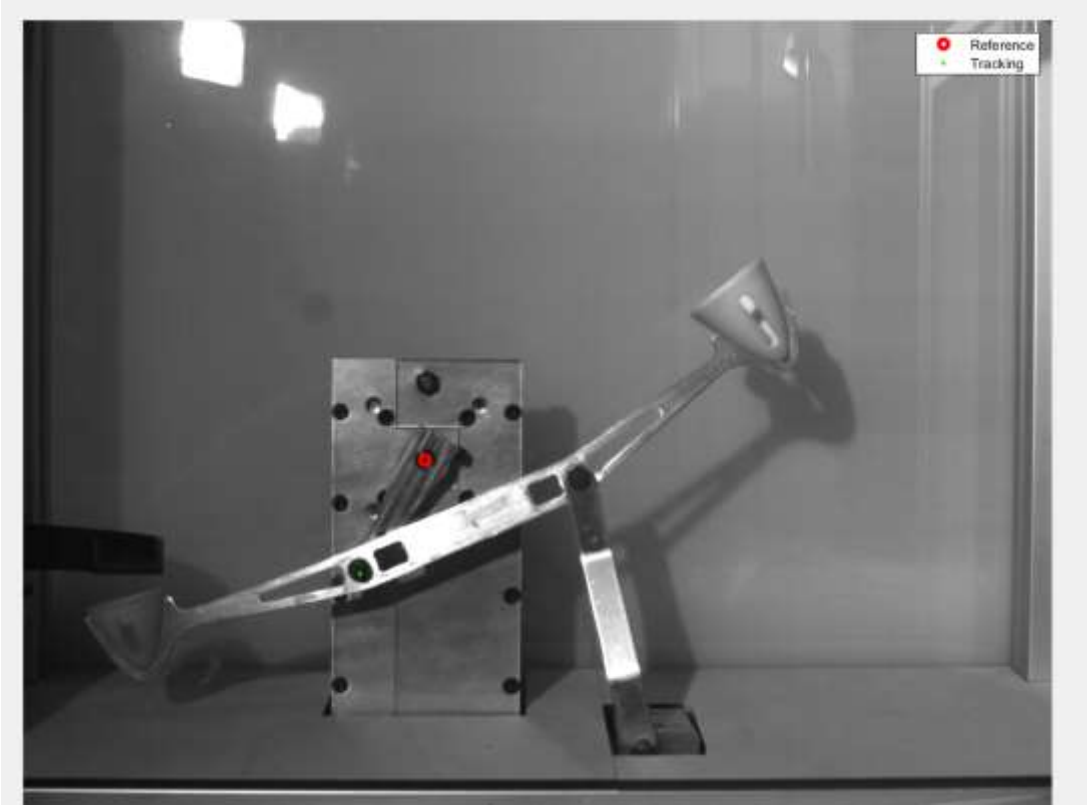

Figure 9: reference point (red) and tracking point (green) of first frame of recording 3

If you look closely, the first frame of the two recording are different. That is the first thing I noticed. Since we hard coded the point coordinates into the code, I had assumed that the points would also work for the other recording but they did not. I then fetched the coordinates by first using *ginput* to select the reference and tracking point and then hard coded them.

The result of recording 3 in terms of position and velocity were as I expected, similar to recording one. The difference was that the movement in recording 3 starting quicker than that of recording 1 with about half a second. The trajectory however remains the same.

In figures 10 and 11 below, the trajectory of two recording are shown as well as a figure (12) with the two positions and velocity graph on top of each other. Here you can see theta3 does indeed start quicker.

The comparison could have been worked out further by synchronizing the two signals, resample if necessary so that they overlap and then calculate the difference between them as done in the previous report. I was, again, not able to achieve this sue to time constraints.

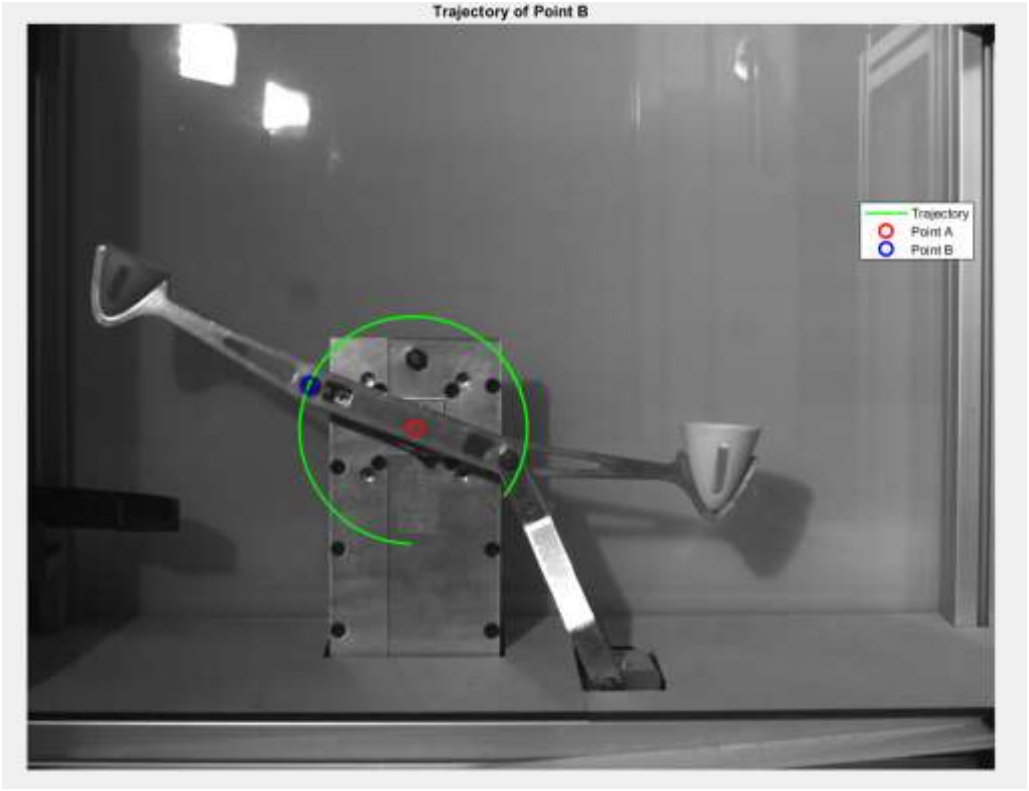

Figure 10: trajectory of point B of recording 1 and end frame

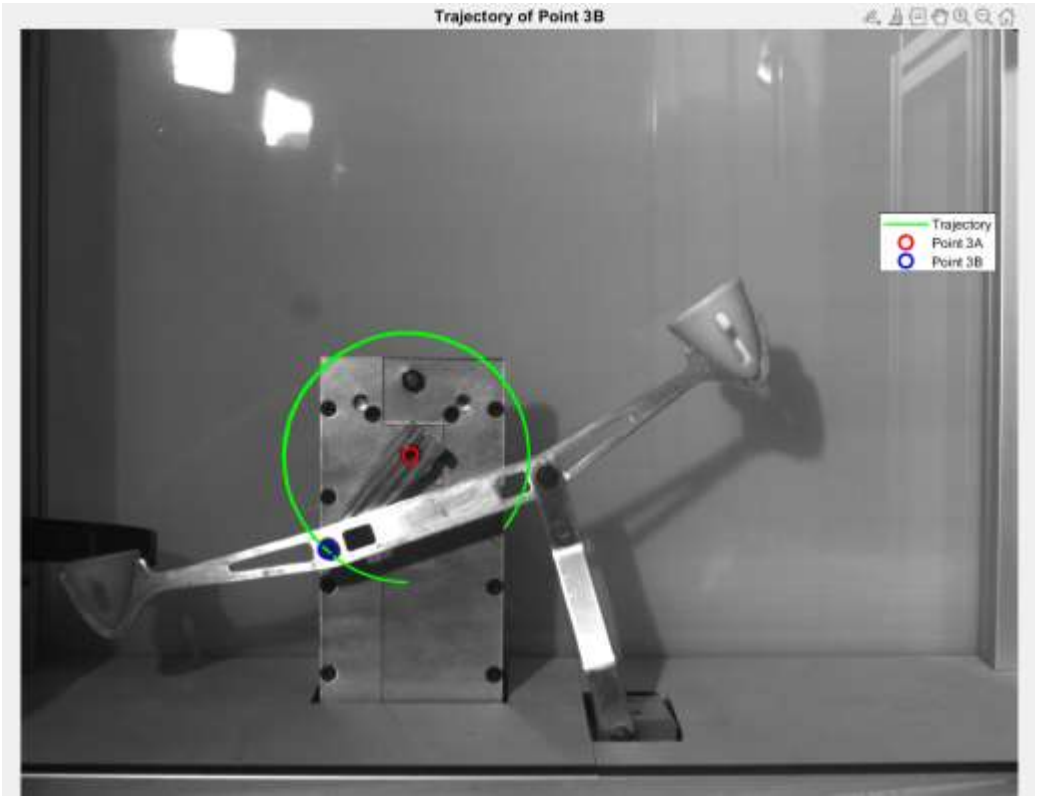

Figure 11: trajectory of point B of recording 3 and end frame

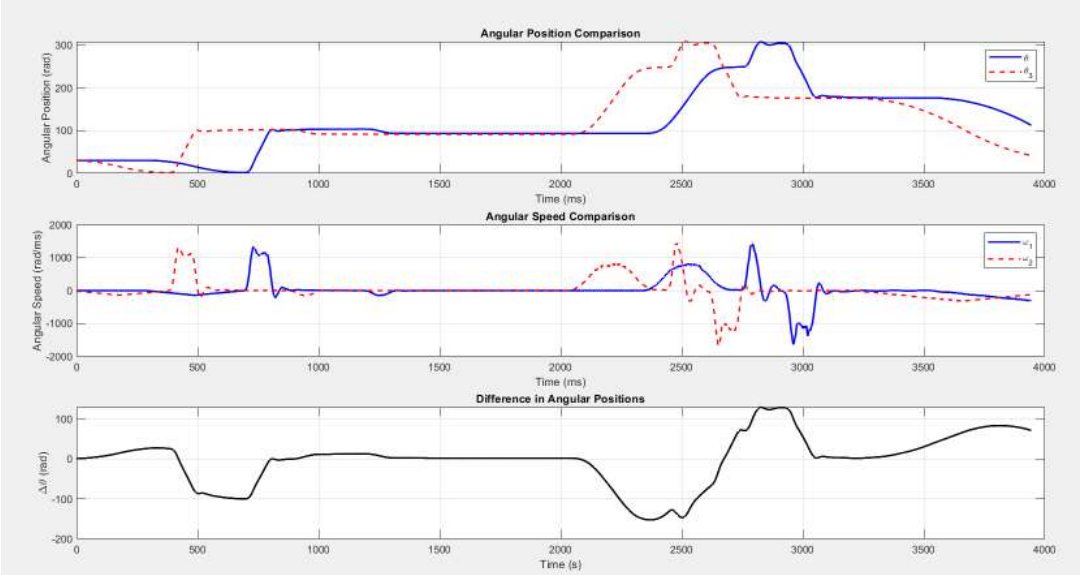

Figure 12: comparison of the position and velocity values