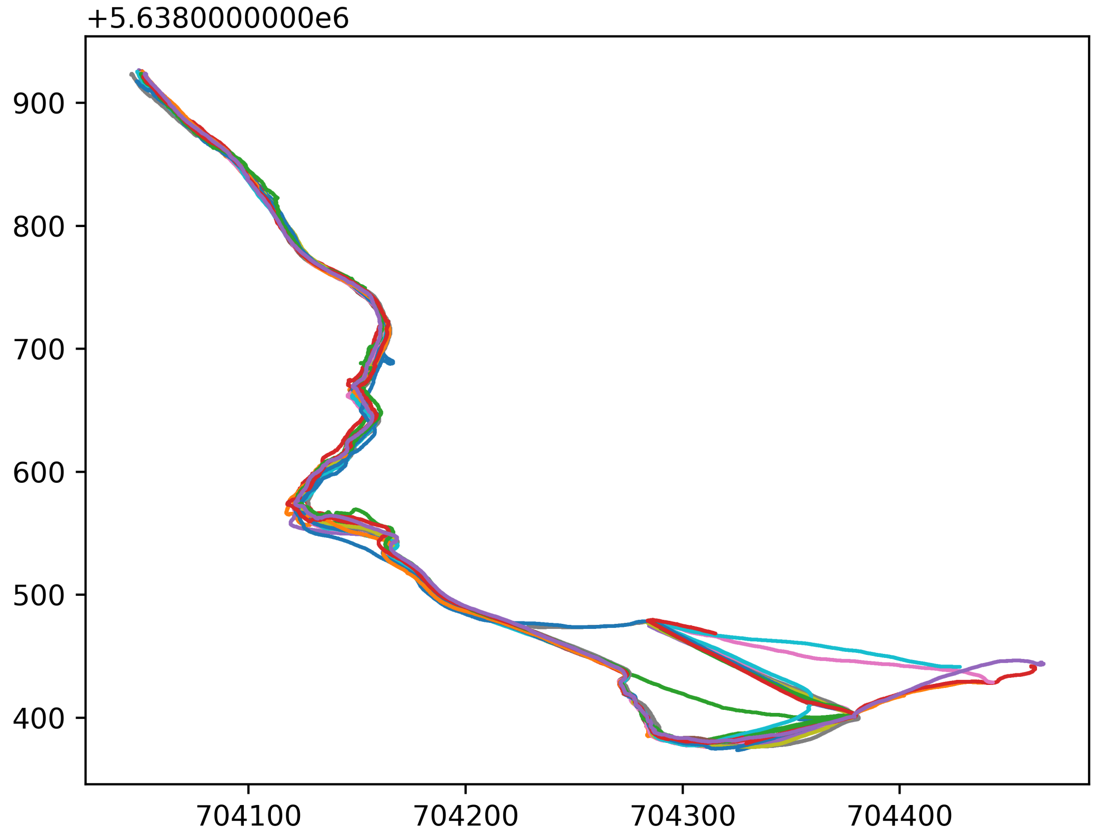

Figure 1: Trajectories as recorded by GPS without any processing applied. All 15 datasets are shown.

1 Basic examination and processing

Within this strand of work we are looking at the GPS and IMU data from the RC car "long trajectories" in Stanmer Park. The overall goal is to examine this data, clean up any issues in it and make it ready for working on assessing visual navigation algorithms.

First, we examine the data by plotting the x and y cordinates of GPS locations taken along the 15 trajectories. The data is plotted in figures 1 to 3.

The first basic observations are:

1. It looks like the coordinates are in m - not in mm as stated in the csv files. If true, the trajectories are some 700m or so long. I will describe everything below under the assumption of units being m.
2. The trajectories start with the vehicle standing still but we can see some drift. Some trajectories have more, some less but on the order of tens of cm (see figure 2.
3. The trajectories presumably started at the same physical location but the recorded starting positions appear offset between trajectories by on the order of $4-8$ m. Furthermore, normalising them to start from the same location does visually not seem to improve the overlap of trajectories, suggesting it is not just a global offset between them (see more about this below)
4. The end of the trajectories is fairly inconsistent, see also figure 3. To do meaningful comparisons of visual navigation algorithms between trajectories, it appears sensible to cut the trajectories to a common, fairly consistent part.

Based on the observations I have made the following first processing steps (`zero_chop.py`):

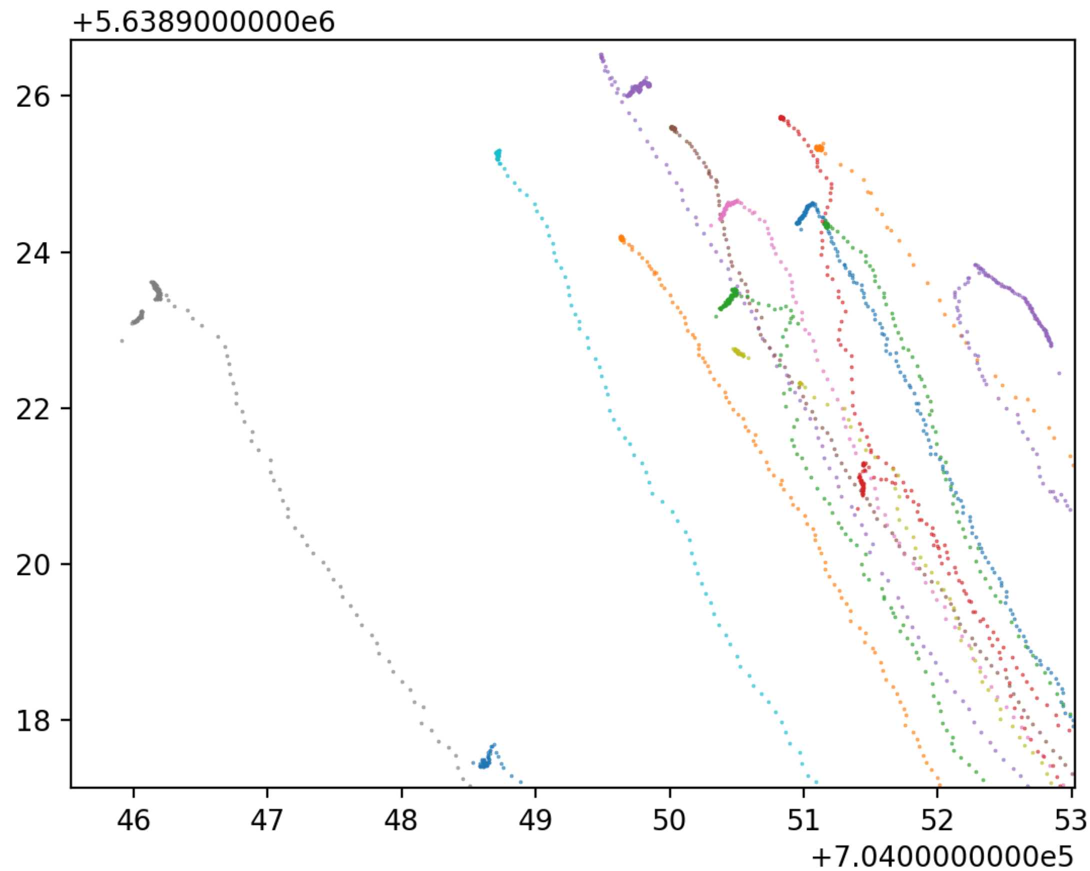

Figure 2: Detail of previous figure focussing on the trajectory start.

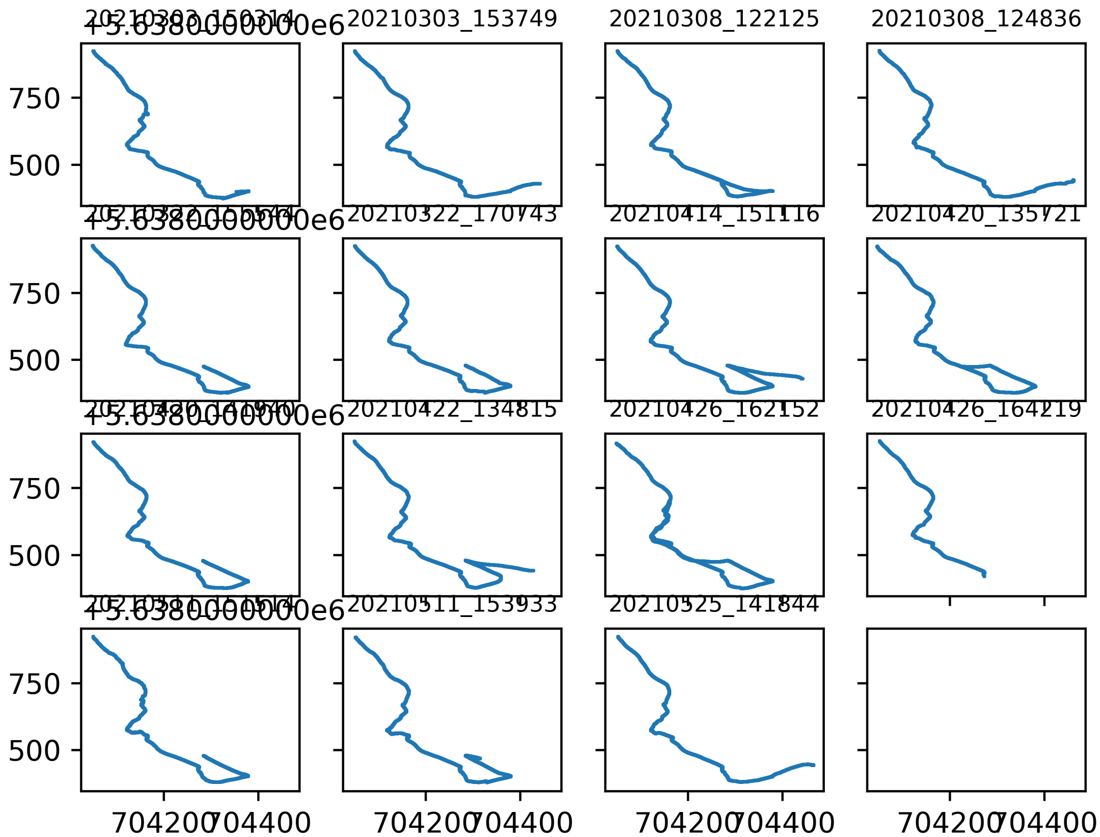

Figure 3: Trajectories as recorded by GPS without any processing applied. All 15 datasets are shown. Mainly included here to be able to identify trajectories and the corresponding data set names.

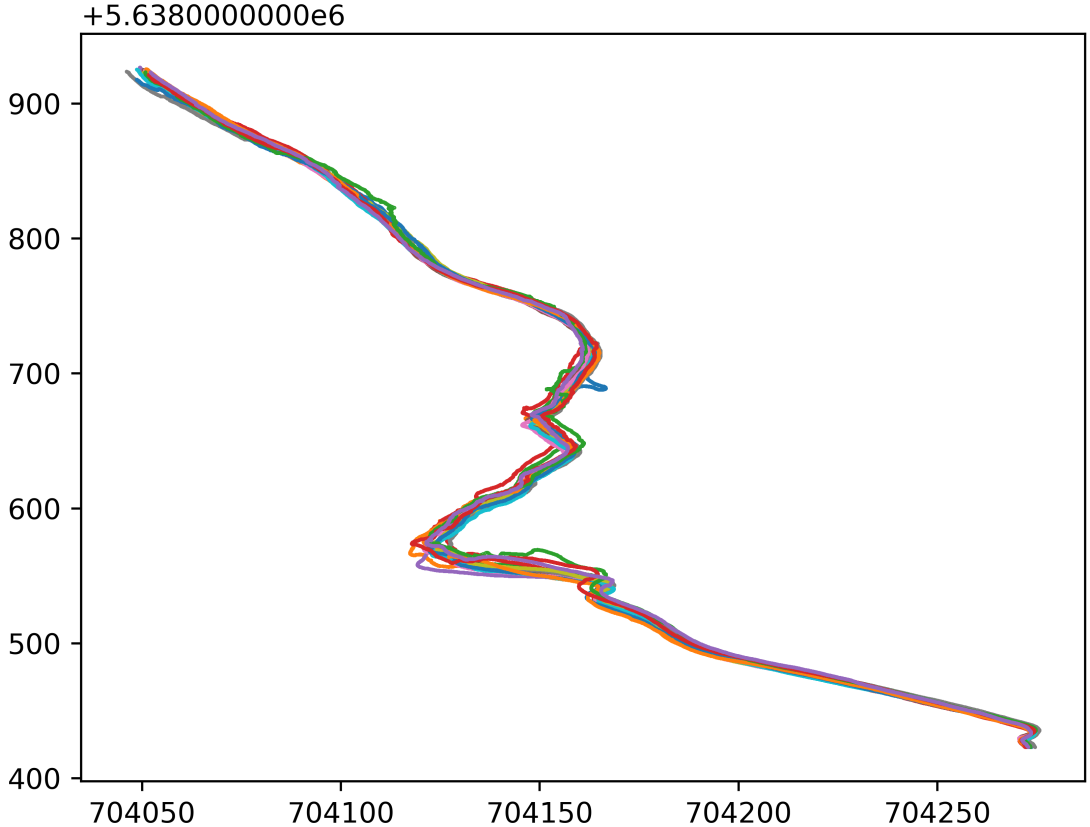

Figure 4: Minimally processed trajectories with initial non-movign phase cut and everything byond the stated y threshold cut.

1. Remove the initial non-moving phase: Here I removed all points where the car did not move more than `min_move` m within `pt_dist` data points (or the next non-NaN point if that is a NaN point). For now I have used `min_move`$= 0.5$ m and `pt_dist`$= 20$.

2. Remove any part of the trajectory once the y coordinate has passed below 5638423 for the first time. This cut-off was chosen by eyeballing the common part of all trajectories.

The resulting trajectories are shown in figure 4.

Before doing any more analysis, it is very inconvenient that about every third GPS position is recorded as NaN. This is presumably due to some mismatch of recording frequencies for GPS and other data and other GPS failures (there are also larger gaps at (thankfully) rare times). Also, there is some obvious noise in the GPS readings. To get somthing more consistent I have taken an approach of fitting pieces of the trajectory with a polynomial function and recombining all fitted pieces by averaging to one interpolated trajectory. This is done with the script `fit_trajectories.py`. For now, I have included results with polynomial degree 1 (simple linear regressions) and on pieces of trajectory of 75 data points each.

Next, we examine the heading information from fitted trajectories and from the IMU recordings. There are jumps in the IMU recordings, which are some kind of artefact of unknown origin. I was not able to find a simple explanation (such as power outage, with reset to 0 or such). I have removed these jumps by the following method (`correct_heading.py`):

1. Whenever the heading reading of the IMU jumps by more than 15 degrees in a single timestep, the jump is subtracted from the subsequent headings.

2. I shift the heading data globally so that the average IMU heading matches the average GPS-based heading.

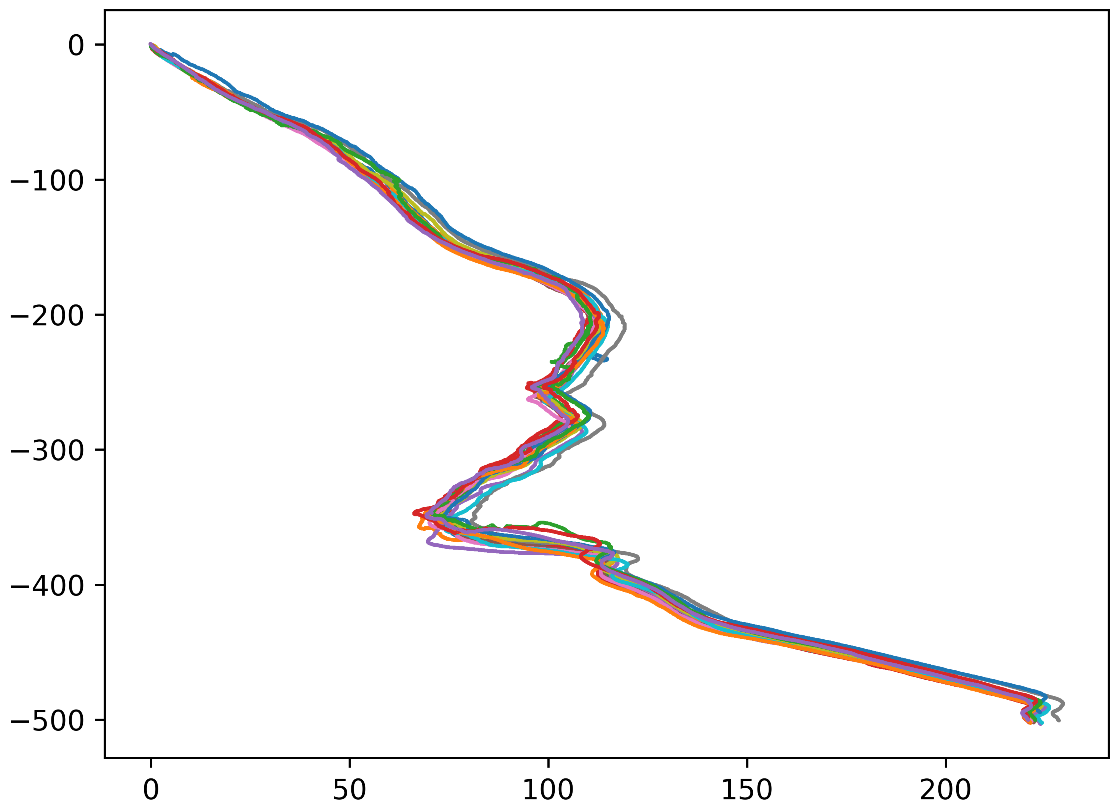

Figure 5: Same a previous figure but with the initial position of the initial trajectory subtracted. Note how these do *not* appear to overlap better.

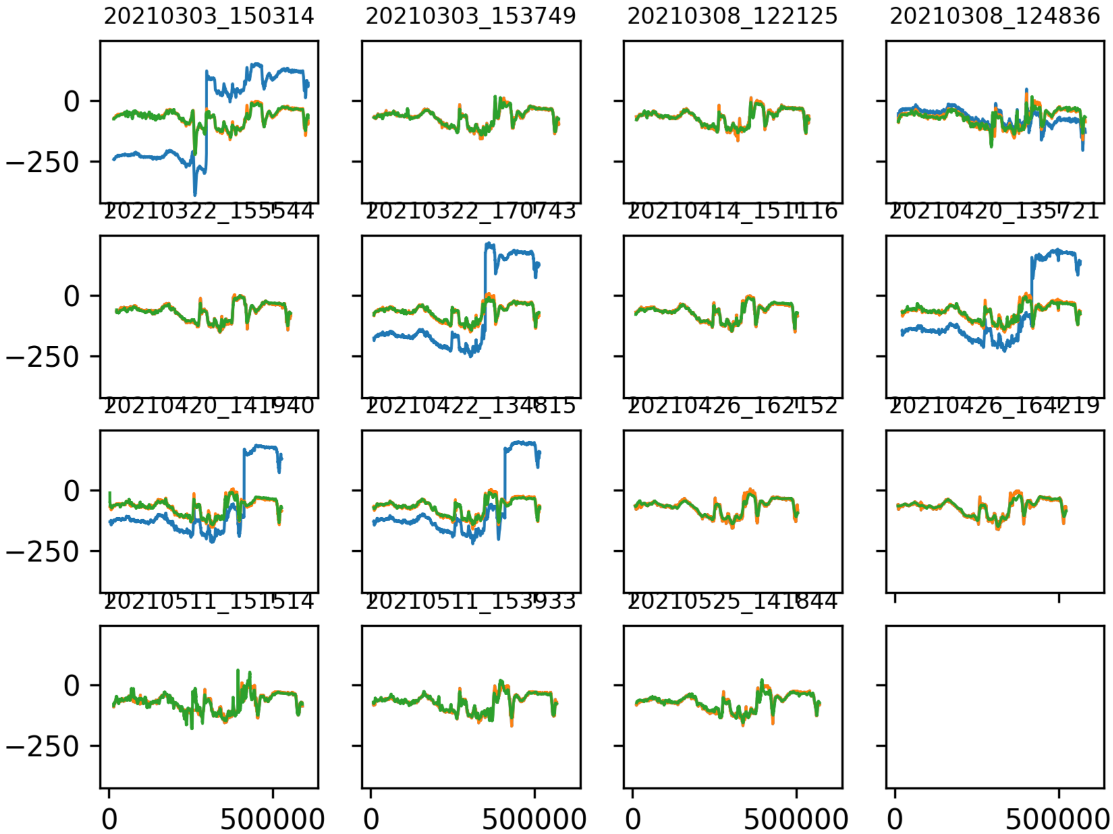

Figure 6: Heading shown as a function of time. For the purposes of this plot I have "unrolled" the angle so that there are no "jumps" of 360 degree.

> GPS-based heading was calulculated simply based on straight line connections between subsequent data points on the interpolated GPS trajectory.

Figure 6 shows the results. The original IMU data, also shifted to match means, is shown in blue, the corrected IMU data in orange and the GPS derived heading in green. The match is arguably sensible even though not perfect. For one, the GPS heading seems to undershoot in sharp turns. This could potentially be improved by taking less points for the trajectory regressions or a higher degree of the fitted polynomials. There are also a couple of sections in individual trajectories where the match is slightly less good.

To give a better impression of the quality of the fitted trajectory and heading fit, I have made movies that move a "magnifying glass" along the trajectories and display the original GPS locations, the fitted trajecory points and GPS derived and IMU derived headings (as a quiver plot). (`make_movie_frames.py` and then using `ffmpeg` like so: `>> ffmpeg -framerate 10 -pattern_type glob -i "frame_*.png" output.mp4`),

Figures 7 and 8 show two individual example frames of teh movie. In the former, the agreement of the headings is excellent which is the case in most locations along the trajectories. In the second example, there is a bit more disagreement. However, this is much more rare and this is one of the more extreme examples. It is worth noting that the IMU headings after jump removal do not appear to degrade gradually along the trajectory but appear to be quite reliable to the very end. This suggests to me that they can be trusted to provide ground truth along the trajectory (and more so than the GPS-derived headings if there happens to be a discrepancy). Please note that I have here displayed the heading arrows next to the original GPS waypoints rather than the extrapolated points; this can be slightly confusing where it is not obvious which extrapolated point corresponds to which original waypoint.

Figure 9 shows the interpolated trajectories altogether. The original shape and spread of the trajectories is well-preserved (compare 4).

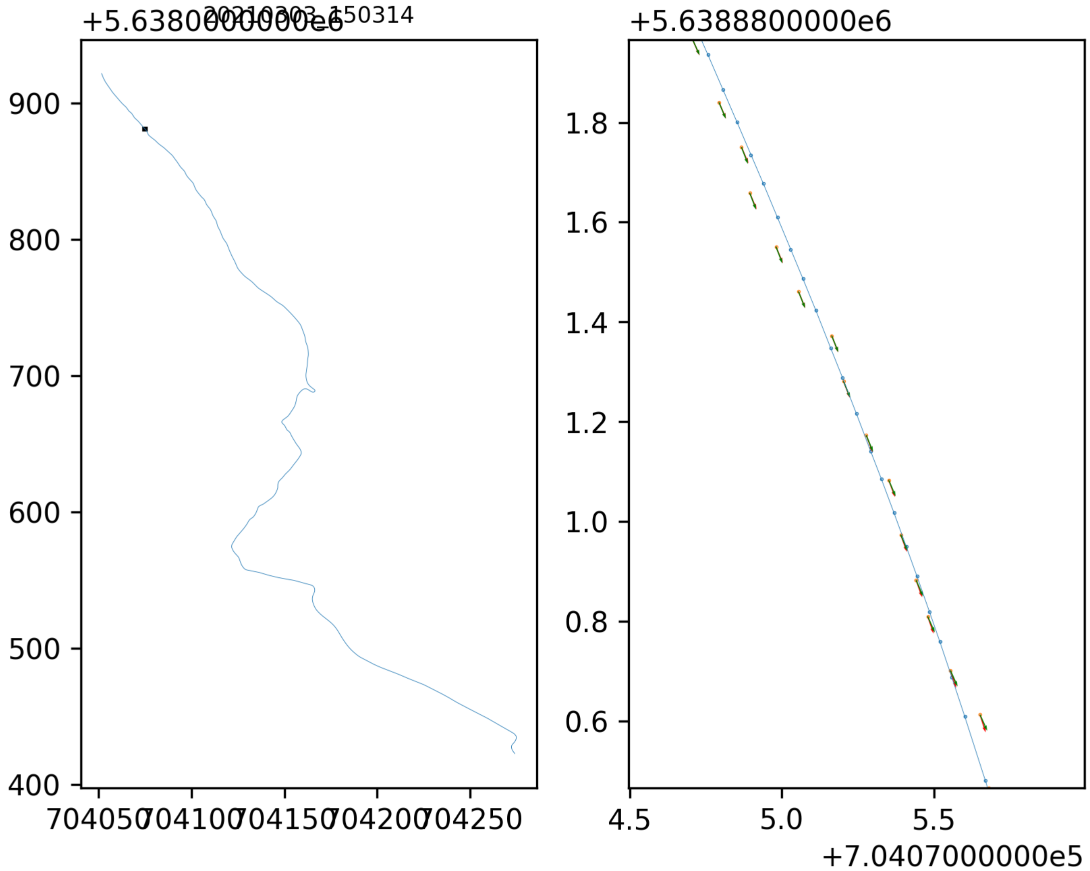

Figure 7: Example of the cleaned data. left: full trajectory, right: enlargement of the area marked by the small rectangle in the left panel. Orange dots are original GPS positions, blue dots and line the interpolated trajectory, and red arrows GPS heading as well as green arrows showing the IMU heading. In this example they coincide very well (which is true most of the time).

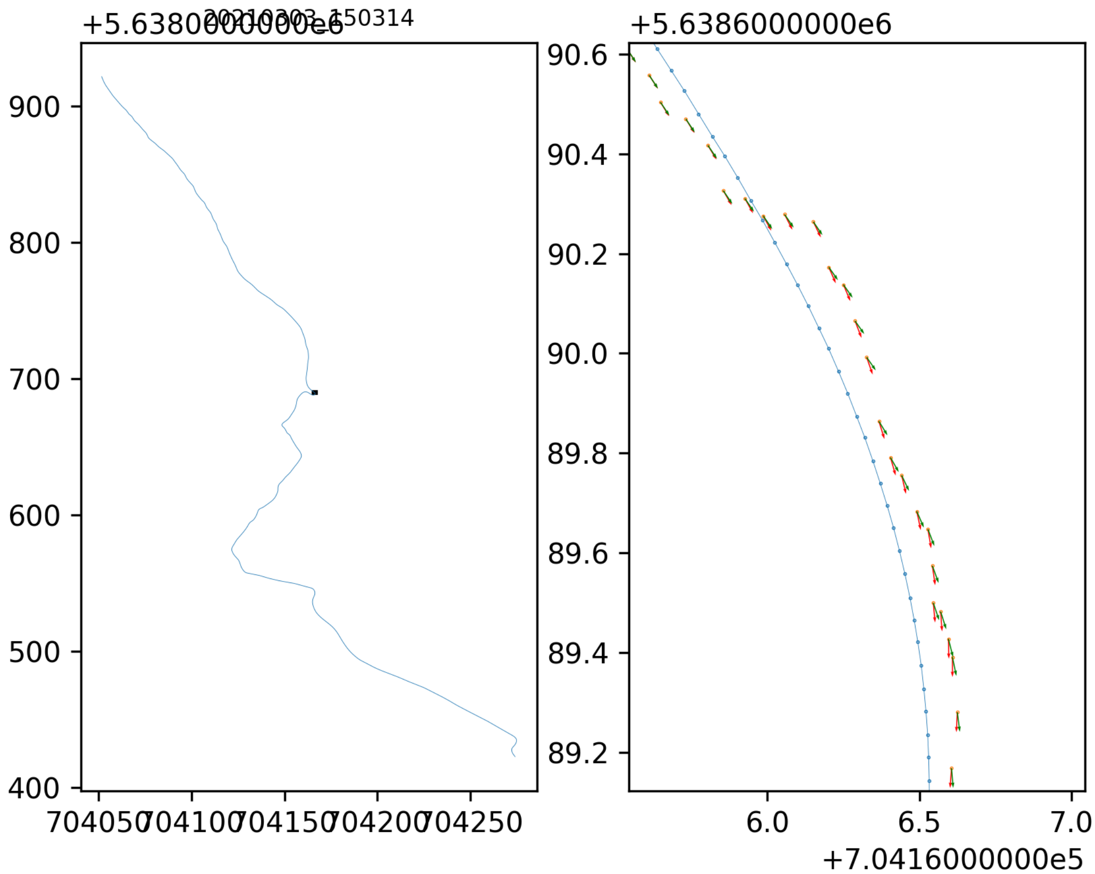

Figure 8: Example of the cleaned data as above. In this example the headings diverge more (this is one of the most extreme examples on this trajectory).

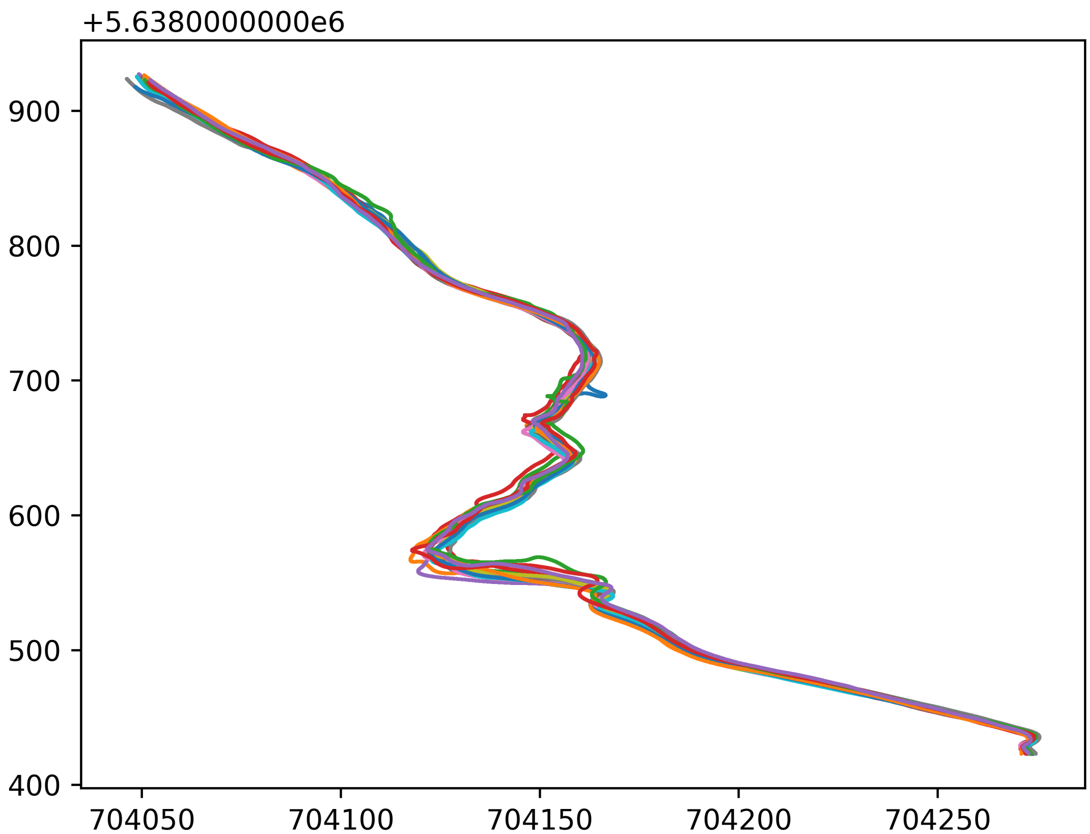

Figure 9: Fitted trajectories plotted together. The spread of the original data is reproduced quite accurately.

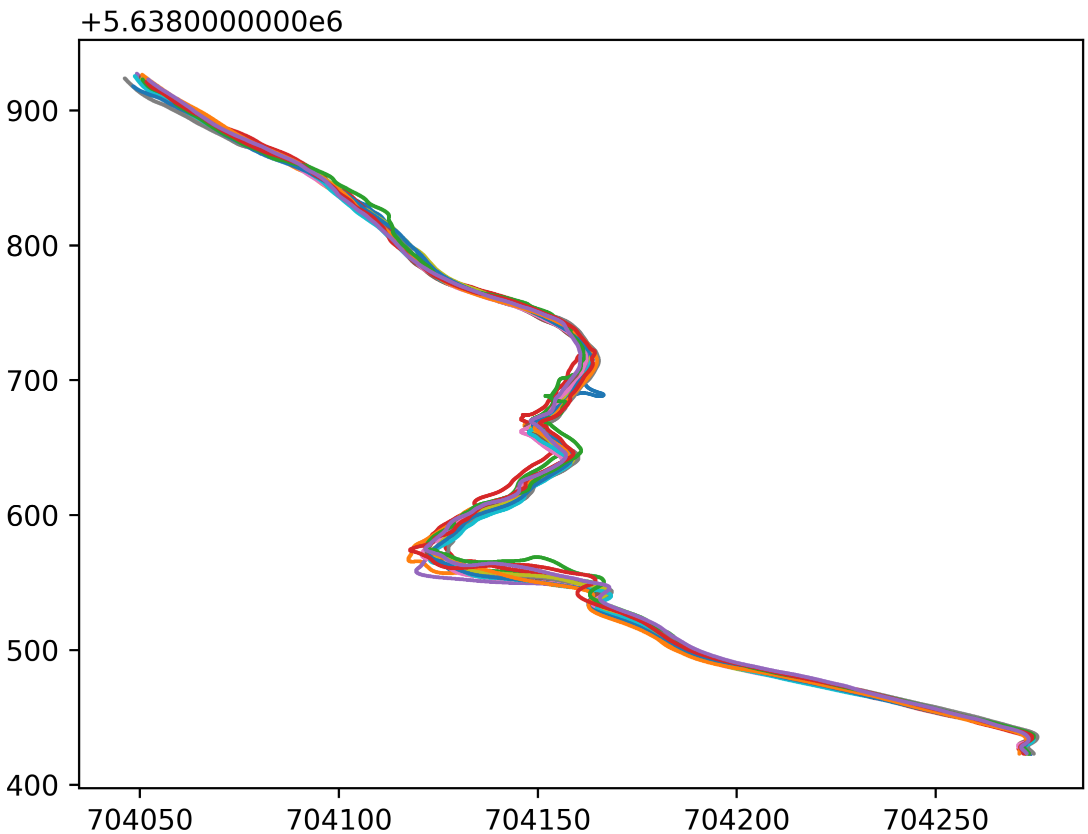

Figure 10: Same as previous figure but with starting points collapsed onto the same point $(0, 0)$.

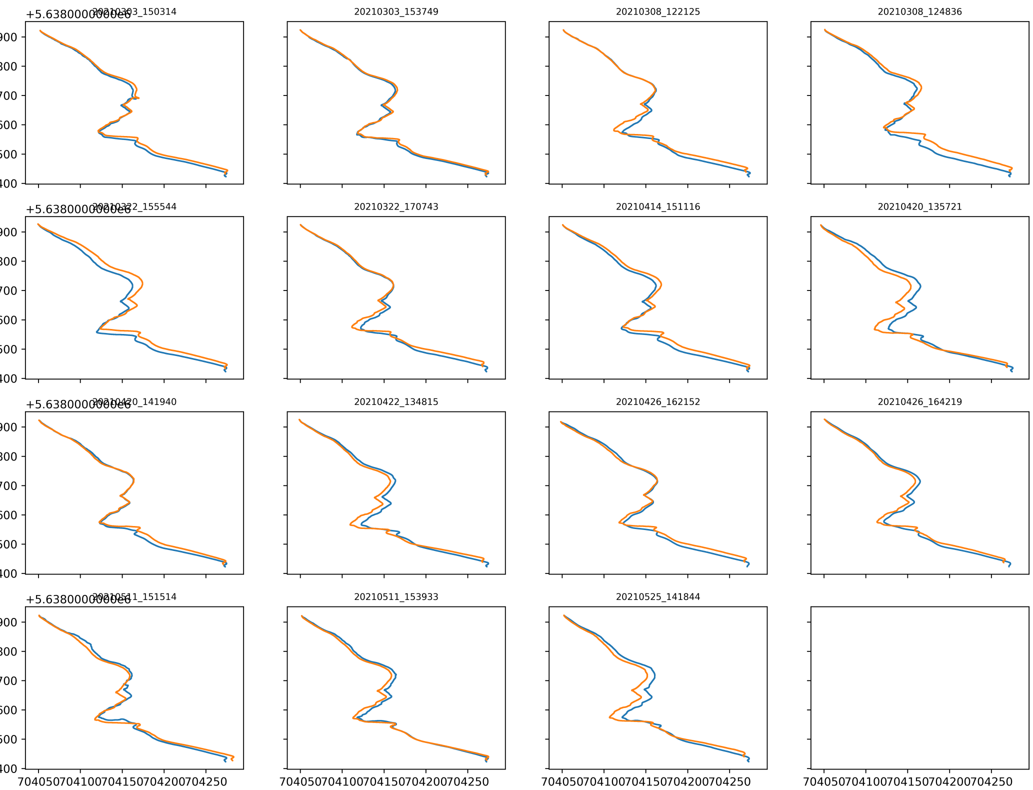

Figure 11: Trajectories reconstructed from IMU heading data (orange) compared to the interpolated trajectories (blue).

When I went ahead and measured the pairwise distances of interpolated trajectories with `measure_0.py`, the trajectories "as is" were closer together than if one makes the starting locations match (by subtracting them), compare figure 9 and 10. On average the summed distances were 26194.83 (raw) and 39009.99 (same starting point).

As a fun exercise I used the IMU headings to reconstruct trajectories. To lend a bit of a helping hand I used the interpolated x and y coordinates to determine the size of stepes to make (the speed in other words). Figure 11 shows the results.

Even though there are obviously some discrepancies, overall the trajectories are very similar, confirming once again that the IMU heading data is quite accurate. There is a little bit of cheating involved here through providing "perfect speed information" but that does not take away from the fact that the heading data is overall mostly right.