University *of Ljubljana*
Faculty *of Mathematics and Physics*

Department of Physics

Optimal Filtering of GPS Data

11. Task for Model Analysis 1, 2023/24

Author: Marko Urbanč
Professor: Prof. Dr. Simon Širca
Advisor: doc. dr. Miha Mihovilovič

Ljubljana, August 2024

Contents

1 Introduction

Filtering of data is vital in many applications as the data we gather from various sensors is too noisy to be truly useful directly. We want to make the best use of our data and extract the most information from it, thus we want to filter it optimally. The Kalman filter is a widely used tool for this purpose and is used in many applications, including GPS data filtering, which will be our example today.

Let's start of by introducing the Kalman filter and its basic principles. The Kalman filter is a recursive algorithm that estimates the state of a linear dynamic system from a series of noisy measurements. It is based on a linear dynamical system model and Gaussian noise assumptions. The Kalman filter is optimal in the sense that it minimizes the mean squared error of the estimated state. The Kalman filter is a two-step process: prediction and update. Lets consider a state vector $\boldsymbol{x}_n$ which represents the state of the system at time n. $\mathrm{P_n}$ is the corresponding covariance matrix. We get the vector $\boldsymbol{x}_0^+$ and the covariance matrix P_0^+ from the initial conditions/first raw measurements. For values where we do not have a specified covariance matrix, we can set it to a large value. The prediction step is then given by:

$$\boldsymbol{x}_{n+1} = \mathrm{F_n x_n} + \mathrm{B_n u_n} + \mathrm{w_n}\,, \tag{1}$$

where $\mathrm{F_n}$ is the state transition matrix, $\mathrm{B_n}$ is the control-input matrix, $\boldsymbol{u}_n$ is the control vector, and $\boldsymbol{w}_n$ is the process noise, with covariance matrix $\mathrm{Q_n}$. We then predict the new state and its covariance matrix as:

$$\boldsymbol{x}_{n+1}^- = \mathrm{F_n x_n^+} + \mathrm{B_n u_n}\,, \tag{2}$$

$$\mathrm{P_{n+1}^-} = \mathrm{F_n P_n^+ F_n}^\mathsf{T} + \mathrm{Q_n}\,. \tag{3}$$

Next we can update this prediction using new measurements. Generally we acquire measurements from sensors which also have their own transition matrix $\mathrm{H_n}$, and noise $\boldsymbol{r}_n$, with the covariance matrix $\mathrm{R_n}$. Thus instead of getting a new state $\boldsymbol{x}_n$, we get:

$$\boldsymbol{z}_n = \mathrm{H_n x_n} + \mathrm{r_n}\,. \tag{4}$$

We can then update the state and its covariance matrix as:

$$\text{Residual} = \mathrm{z_{n+1}} - \mathrm{H_{n+1} x_{n+1}^-}\,, \tag{5}$$

$$\mathrm{K_{n+1}} = \mathrm{P_{n+1}^- H_{n+1}^\mathsf{T}} (\mathrm{H_{n+1} P_{n+1}^- H_{n+1}^\mathsf{T}} + \mathrm{R_{n+1}})^{-1}\,, \tag{6}$$

$$\boldsymbol{x}_{n+1}^+ = \boldsymbol{x}_{n+1}^- + \mathrm{K_{n+1}}\ \text{Residual}\,, \tag{7}$$

$$\mathrm{P_{n+1}^+} = (\mathbb{1} - \mathrm{K_{n+1} H_{n+1}})\mathrm{P_{n+1}^-}\,, \tag{8}$$

$$\tag{9}$$

where $\mathrm{K_{n+1}}$ is known as the **Kalman gain**, which determines how much the new measurement should affect the state estimate. This is the basic Kalman filter algorithm.

2 Task

2.1 Filtering of Cartesian GPS Data

We've been provided a dataset of GPS data in the form of Cartesian coordinates with matching control data. We want to filter this data using the Kalman filter so we can best track the movement of Mr. Čopar as he travels from the Faculty of Mathematics and Physics to his weekend retreat near Litija. We can use the control data to see how well our filter is working. In this example our state vector will be $\boldsymbol{x}_n = (x, y, v_x, v_y)^\mathsf{T}$, where x and y are the position coordinates, and v_x and v_y are the velocities in the x and y directions. The control vector contains information about the acceleration in the x and y directions:

$$\boldsymbol{c}_n = \mathrm{B_n u_n} = \begin{bmatrix} 1/2\, a_x \Delta t^2 \\ 1/2\, a_y \Delta t^2 \\ a_x \Delta t \\ a_y \Delta t \end{bmatrix} \tag{10}$$

where Δt is the time step between measurements. The state transition matrix is given by:

$$\mathrm{F_n} = \begin{bmatrix} \mathbb{1}_{2\times 2} & \Delta t \mathbb{1}_{2\times 2} \\ 0_{2\times 2} & \mathbb{1}_{2\times 2} \end{bmatrix} . \tag{11}$$

Noise due to time evolution is given by the error of acceleration measurements as:

$$\mathrm{Q_n} = \sigma_\mathrm{a}^2 \begin{bmatrix} 1/4\, \mathbb{1}_{2\times 2}\Delta t^4 & 1/2\, \mathbb{1}_{2\times 2}\Delta t^3 \\ 1/2\, \mathbb{1}_{2\times 2}\Delta t^3 & \mathbb{1}_{2\times 2}\Delta t^2 \end{bmatrix} , \tag{12}$$

while the nose from the measurements is given by the GPS data error:

$$\mathrm{R_n} = \mathrm{diag}(\sigma_\mathrm{x}^2,\, \sigma_\mathrm{y}^2,\, \sigma_{\mathrm{v_x}}^2,\, \sigma_{\mathrm{v_y}}^2) , \tag{13}$$

where we can assume that $\sigma_x = \sigma_y$ and $\sigma_{v_x} = \sigma_{v_y} = \sigma_v$ are the standard deviations of the GPS data and velocity measurements. We can simply assume that $\mathrm{H_n} = \mathbb{1}_{4\times 4}$. The instructions specify that for our case $\Delta t = 1.783$ s, $\sigma_x = \sigma_y = 25$ m, $\sigma_a = 0.05\ \mathrm{m/s^2}$ and that the error for the velocity is given by:

$$\sigma_v = \min\{0.277\ \mathrm{m/s},\, 0.01\|\mathrm{v}\|\} , \tag{14}$$

where $\|\boldsymbol{v}\|$ is the magnitude of the velocity vector. We see that the error in the velocity measurements is constrained downwards to be at least 1 km/h. The task instructs us to use the provided data to filter the GPS data and to plot the results. We should also have a look at what happens if we only take every 5th and 10th measurement.

2.2 Missing Pair of Data

In the second part of the task we do the same as in the first subtask but where we assume we only have measurements for the location and acceleration and not for the velocity. We can also switch this around to only have measurements for the position and acceleration but not for the velocity.

2.3 Filtering of Relative GPS Data

In the third subtask the instructions confront us with the fact that smartphones do not provide us with absolute GPS data but only relative data. That means that the accelerations we measure are actually the radial and tangential accelerations which depend on the orientation of the car. This means we must take into account another linear transformation:

$$\mathrm{B_n} = \begin{bmatrix} \mathbb{1}_{2\times 2} & 0_{2\times 2} \\ 0_{2\times 2} & B_n^{\mu\nu} \end{bmatrix} , \quad \text{where} \quad \mathrm{B_n^{\mu\nu}} = \frac{1}{\|\mathrm{v_n}\|} \begin{bmatrix} v_x & -v_y \\ v_y & v_x \end{bmatrix} . \tag{15}$$

We apply this transformation to the control vector which is now given as:

$$\boldsymbol{u}_n = \begin{bmatrix} 0 \\ 0 \\ a_r \Delta t \\ a_t \Delta t \end{bmatrix} . \tag{16}$$

The matrix Q now gets a tad more complicated, having the form:

$$\mathrm{Q_n} = \sigma_\mathrm{a}^2 \begin{bmatrix} \boldsymbol{\sigma}_a^2 & 0_{2\times2} \\ 0_{2\times2} & \mathrm{Q_n^{\mu\nu}} \end{bmatrix}, \quad \text{where} \quad \mathrm{Q_n^{\mu\nu}} = \Delta\mathrm{t}^2 \left\{ \sigma_\mathrm{a}^2 \, \mathbb{1}_{2\times2} + \frac{\mathrm{v_n^\perp P_n^{\mu\nu} v_n^{\perp\,T}}}{\|\mathrm{v_n}\|^4} \left[\left(\mathrm{B_n^{\mu\nu} a_n^\perp}\right) \otimes \left(\mathrm{B_n^{\mu\nu} a_n^\perp}\right)^\mathsf{T} \right] \right\}, \quad (17)$$

where $\boldsymbol{v}_n^\perp = (-v_y, v_x)^\mathsf{T}$ and $\boldsymbol{a}_n^\perp = (-a_t, a_r)^\mathsf{T}$. We've again been provided a data set with the relative GPS measurements which we should filter and plot.

3 Solution Overview

Solving this task was quite standard. I didn't use any fancy approaches or optimizations or distributed computing etc. I simply wrote a `KalmanFilter` class which I could then modify for each subtask. I created some synthesized data to test the filter. The results of that test are plotted in Figure 1. We can see that despite being provided the wrong initial conditions and noisy data, the filter quickly locks back onto the true values, albeit with some lag.

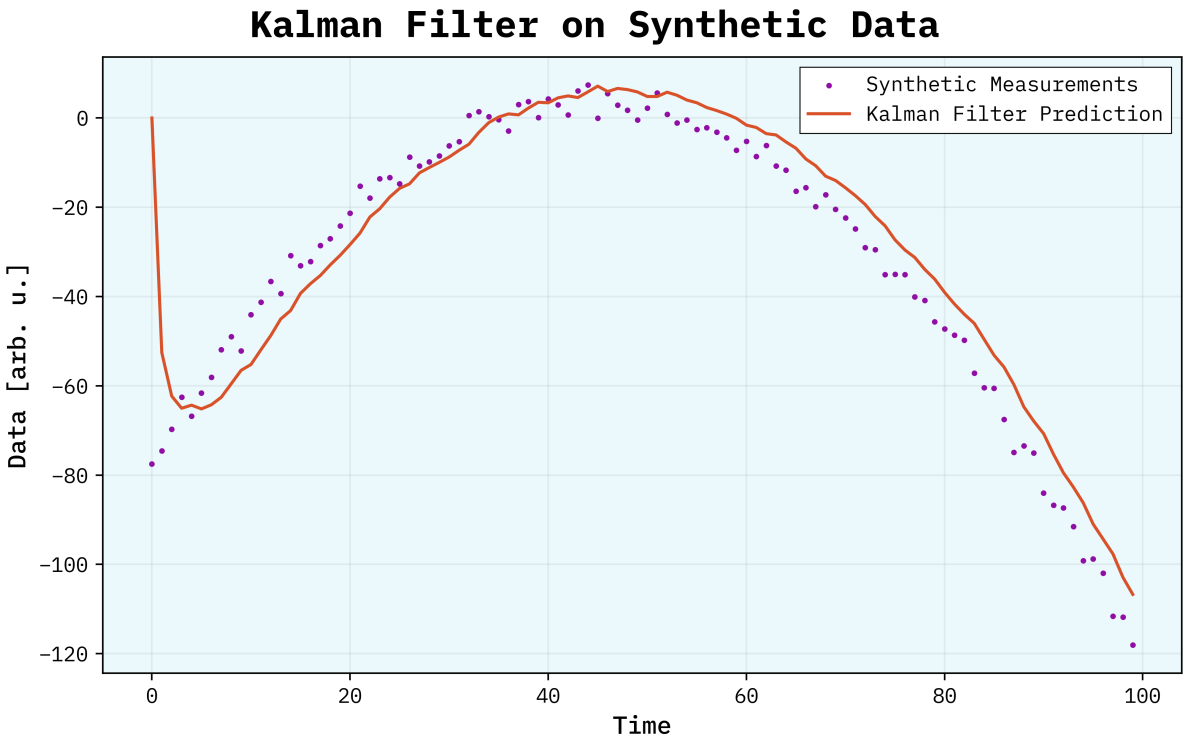

Figure 1: Test plot of the Kalman filter.

4 Results

4.1 Filtering of Cartesian GPS Data

Let us not waste any more words and get right to the results. Figure 2 shows the dataset we've been provided. Both the measurements and the control data are plotted, althought they might be a bit difficult to discern as they are very close. The results of filtering this data are shown in Figure 3.

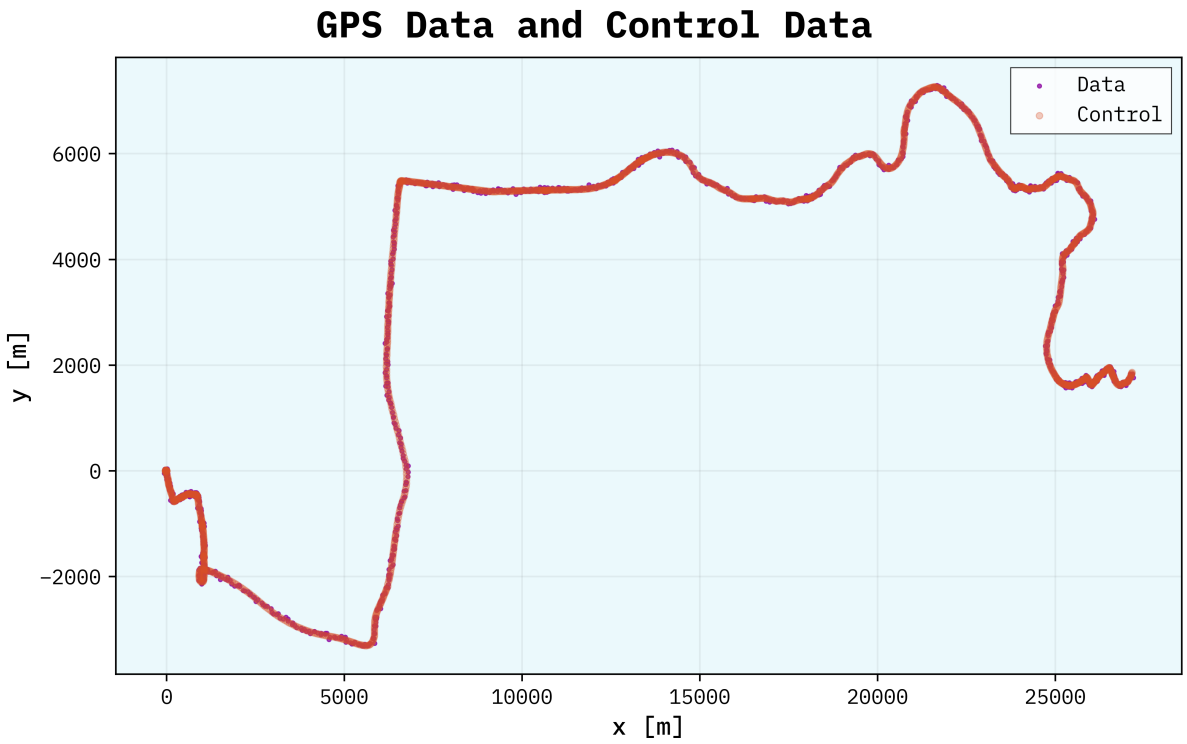

Figure 2: Cartesian GPS data.

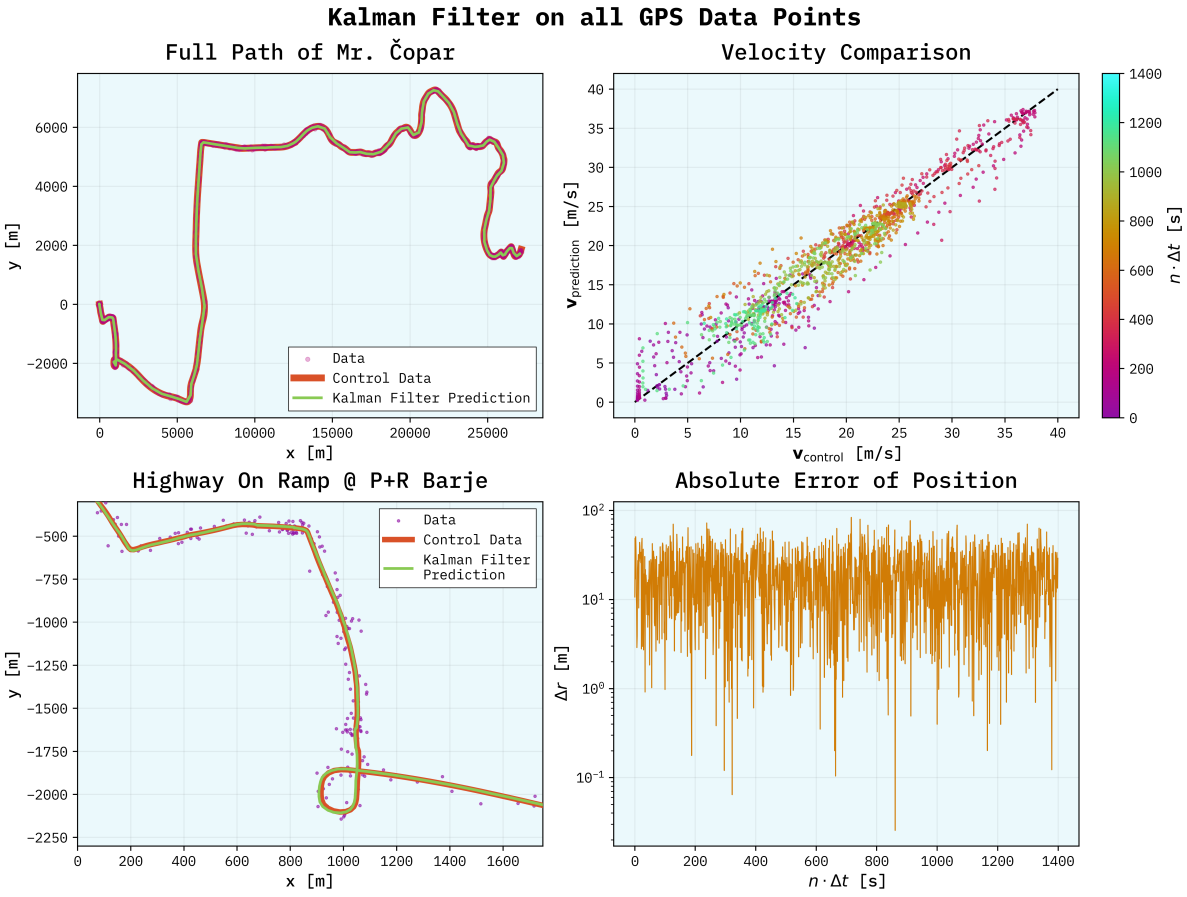

Figure 3: Filtered Cartesian GPS data.

I've tried to make the plot more entertaining and informative by adding some extra information. On the top left side we can see the path Mr. Čopar takes. The filter honestly does quite a remarkable job

at tracking the path. On the bottom left I've included a zoom plot of the Highway On Ramp located at P+R Barje. The filter does a good job at tracking the path even in this more complicated area although it does veer ever so slightly in the loop. It does however correct itself quickly afterwards. On the lower right we have a simple logarithmic plot of the absolute error in position with respect to the control data. The error is about what I expected, about 25 m. On the top right I have a more interesting plot, comparing the control velocity to the filters predicted velocity. Were the filter absolutely perfect we'd expect to see a straight line at $y = x$, however due to the errors in the filters predictions we see a bit of a scatter. I colored the scatter points sequentially so we can see which points correspond to which part of Mr. Čopar's journey. It seems that low velocities are more difficult for the filter to predict. With a bit of imagination one can see that the scatter seems to tighten up as the velocity increases.

We can study velocity in more detail however. Figure 4 shows the control velocity and the filters predicted velocity for both the x and y components. The right-hand side displays the absolute error in the velocity predictions.

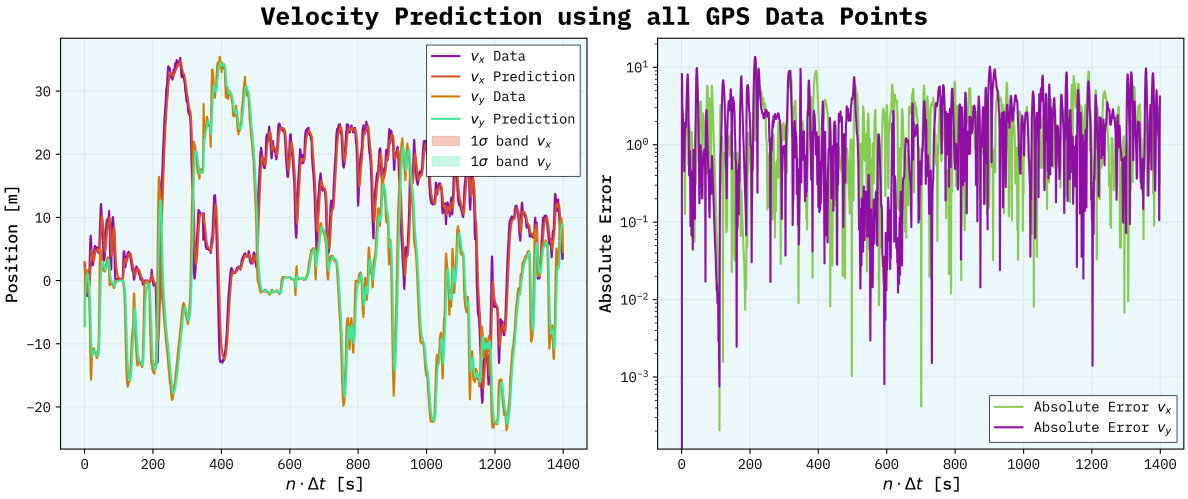

Figure 4: Velocity data and error.

The left-hand plot does actually contain 1-σ error bands for the velocity predictions, however they are so small that they are hardly visible. Despite this, the filter does a decent job of tracking the velocity. The errors in velocity, while not the best, are still quite acceptable.

Next we're interested in what changes if we only take every 5th measurement into account. We can do this by changing the sensor transition matrix $\mathrm{H_n}$ to be non-zero only every 5th measurement. The results are shown in Figure 5.

The plot presents the same information as the plot for the full data set. I thought that would be the best way to compare the two. The filter still does a decent job of tracking the path, although in the zoom plot for the Highway On Ramp we can see that it definitely lacks resolution in the sense that the predicted trajectory is very jagged. The error in position overall is about the same as for the full data set. The velocity plot definitely has a much much larger scatter. I think it's more visible that higher velocities are easier for the filter to predict. We can take this further and only take every 10th measurement into account. The results are shown in Figure 6.

The filter still does an *okay* job of tracking the path, but the resolution is definitely lacking. Tight turns are difficult for the filter to predict with the sparse data. Thus the Highway on Ramp looks completely jagged but what is somewhat surprising is that it still manages to recognize the loop in the path. The error in position is on average higher than before, which is to be expected. Likewise the scatter in the velocity plot is much much larger. It makes sense to take a closer look at the velocity predictions for these two cases. The results are shown in Figures 7 and 8 in the same fashion as before.

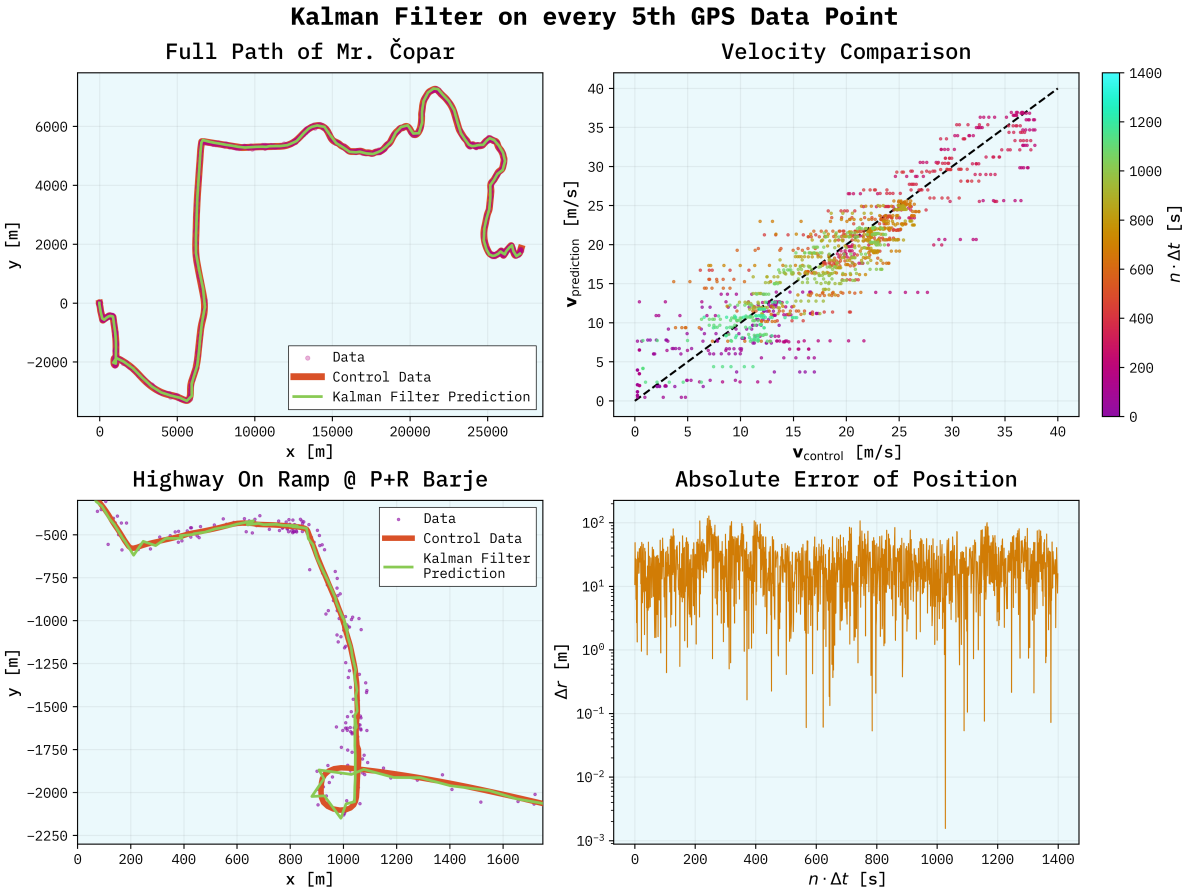

Figure 5: Filtered Cartesian GPS data with every 5th measurement.

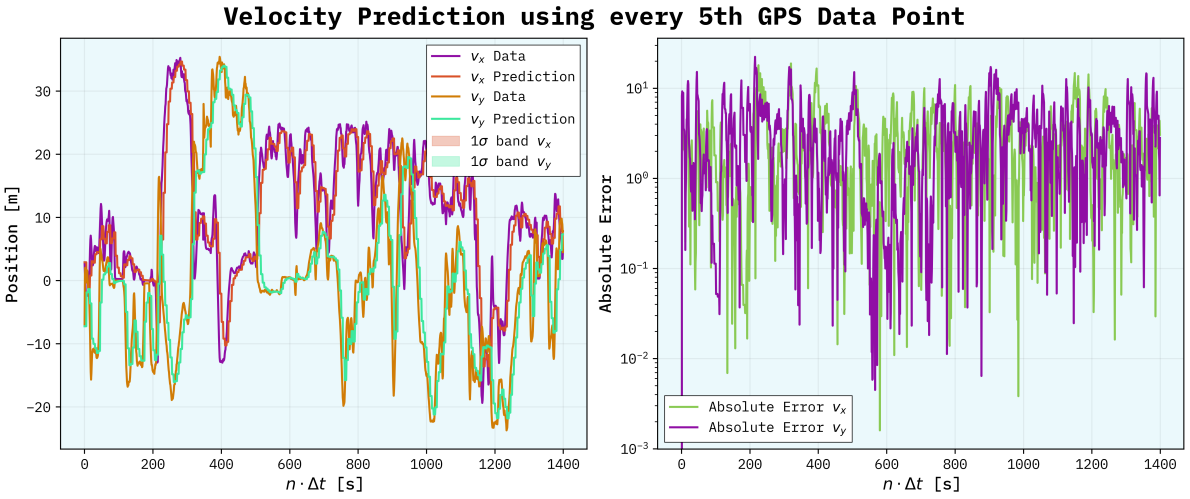

Figure 7: Velocity data and error with every 5th measurement.

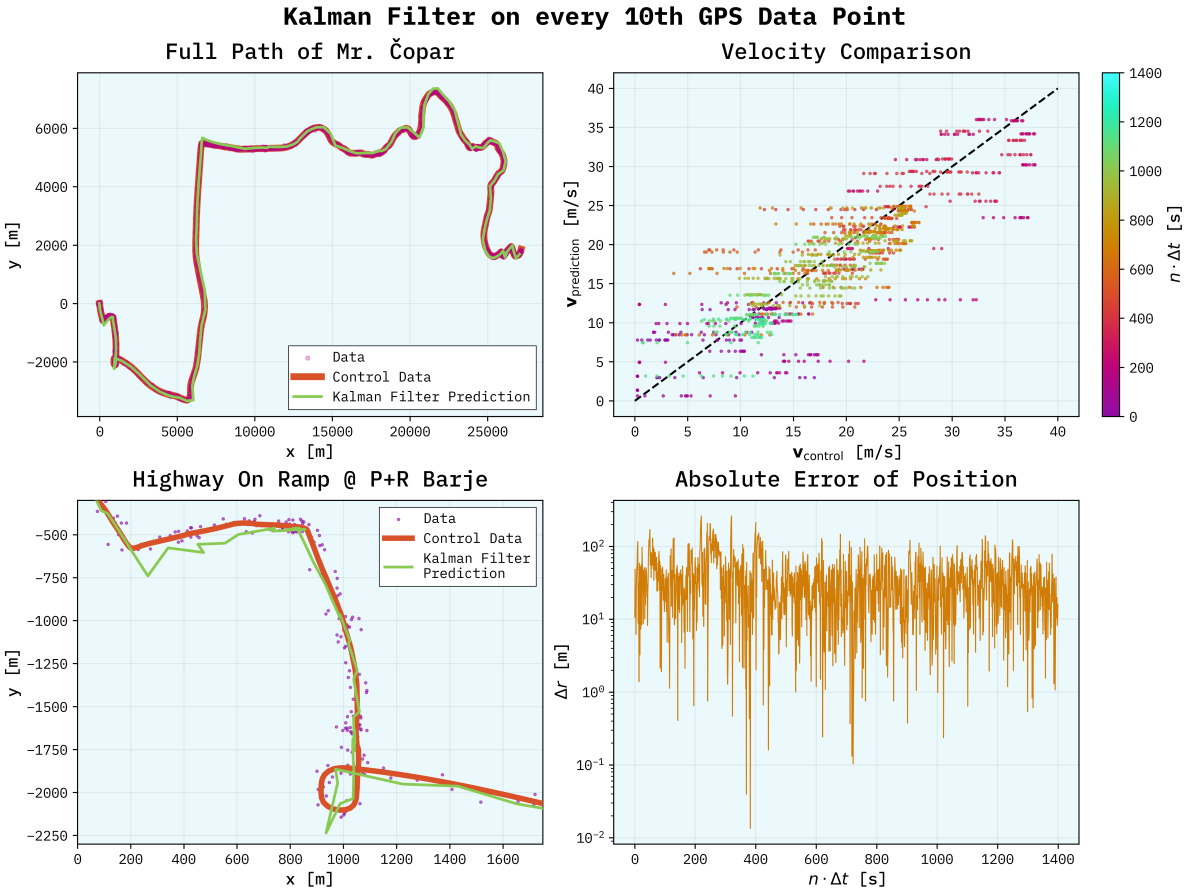

Figure 6: Filtered Cartesian GPS data with every 10th measurement.

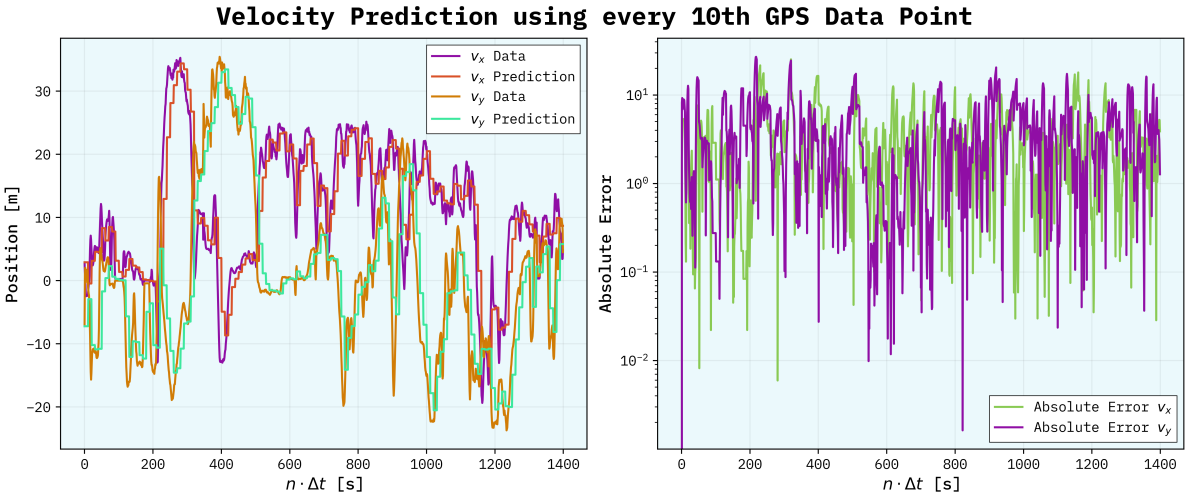

Figure 8: Velocity data and error with every 10th measurement.

The absolute error looks worse than before, which we'll see later on the comprehensive velocity prediction error plot. The filter still does an okayish job of tracking the velocity, but it feels quite a bit laggy. This effect is even more pronounced in the case of every 10th measurement. The error in the velocity predictions is noticeably larger than before. As forecasted above it makes sense to plot all of these errors in a more comprehensive plot which is shown in Figure 9.

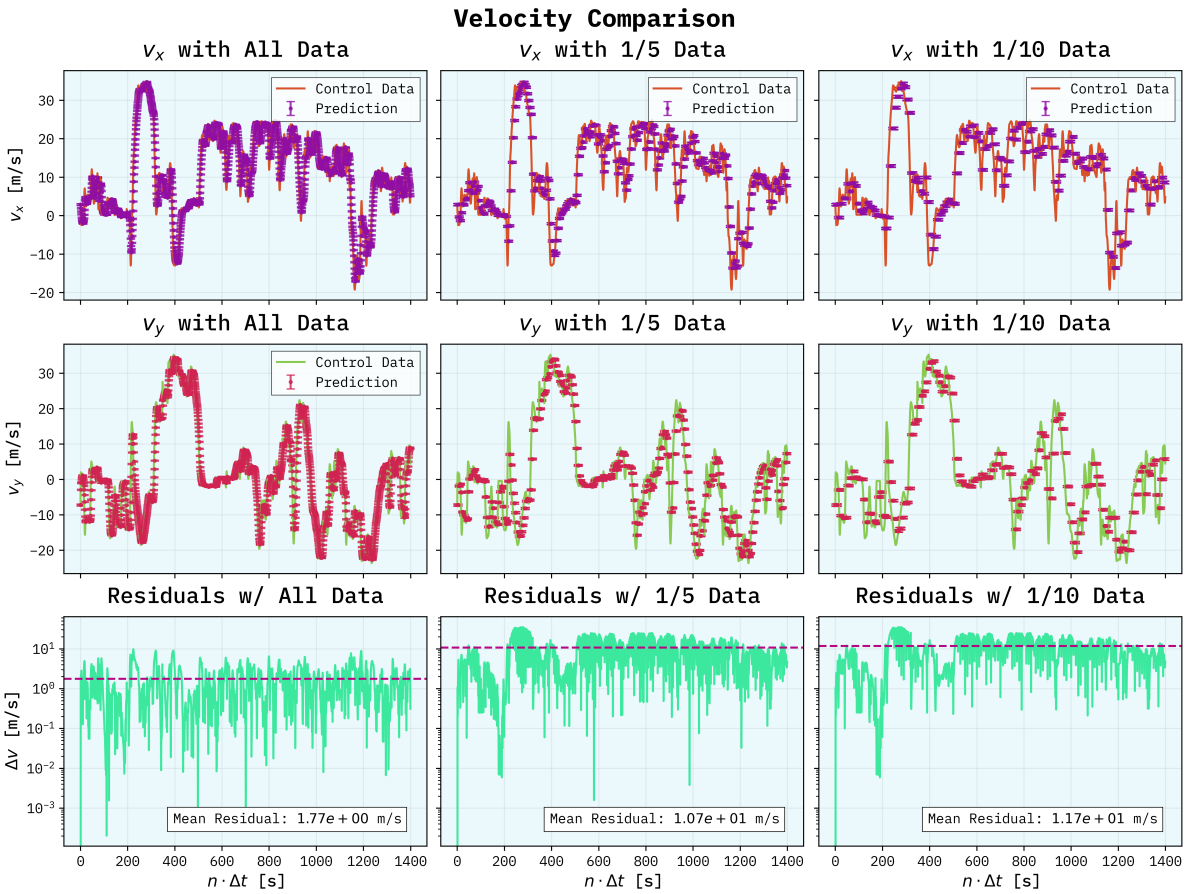

Figure 9: Comprehensive velocity error plot.

The first two rows show the x and y velocity components for all the discussed cases. All points also contain error bars which are very small and thus hardly visible. The third row is more interesting. It shows the absolute error in the velocity predictions for all cases discussed (combined for x and y). I've added a horizontal line which represents the average absolute error. From this we can see that taking every 5th data point produces a sizable jump in the average absolute error. Taking every 10th data point does still increase the error even further but in comparison the jump is not as large. It would make sense then that if we were strapped for processing power it would make sense to take every 10th data point as taking every 5th data point does not provide a significant enough improvement in the error. Although the argument that we need a lot of processing power to run the filter is not very strong. In NASA's Apollo program they used the Kalman filter in the Apollo navigation computer which had roughly 2k of memory and a 100kHz clock speed [1]. Since the missions were successful we have little excuse to not be able to run the filter on a modern computer.

The Kalman filter also provides us with the covariance matrix of the state estimate over time. I've plotted the roots of the diagonal elements of the covariance matrix in Figure 10. Colors represent the various cases.

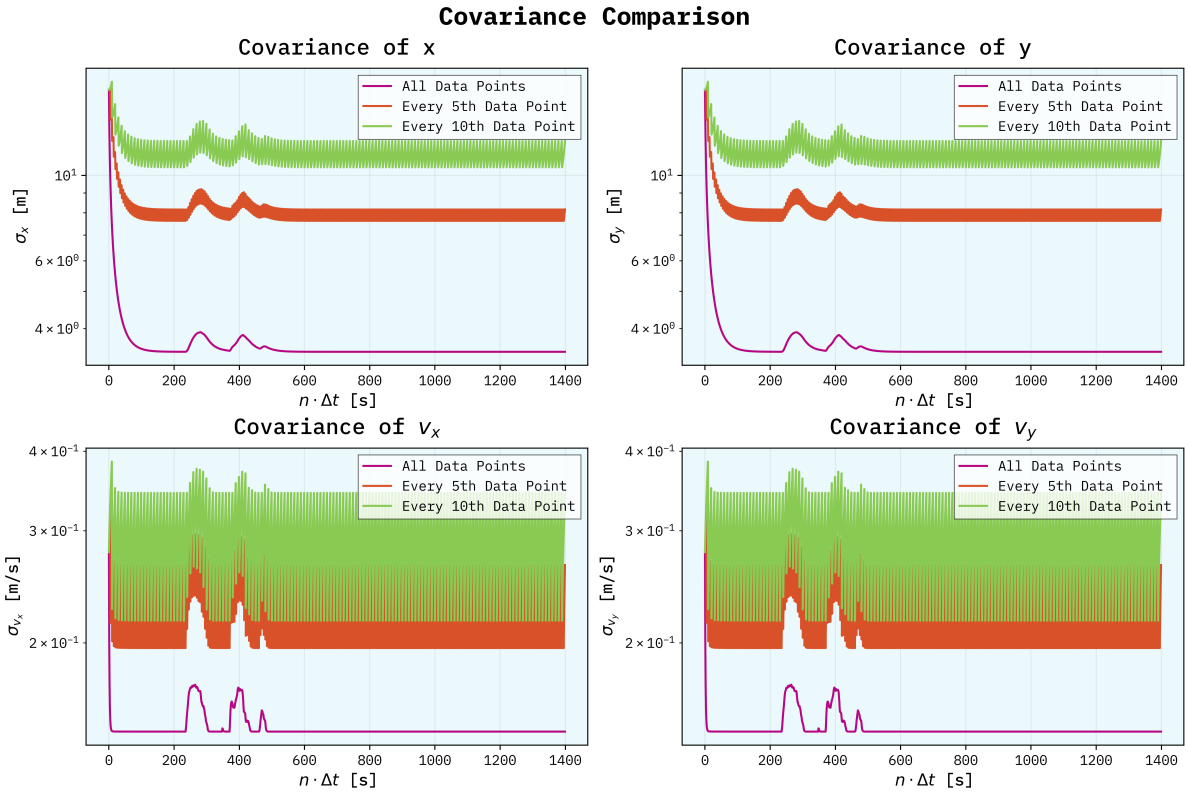

Figure 10: Roots of the diagonal elements of the covariance matrix.

A strange oscillatory pattern emerges in the truncated data cases. This might be due to the covariance rising until the next measurement is given which causes the covariance to drop promptly. Other than that the covariances stay quite stable. They fall from the initial high values to a stable low value. All plots contain some bumps however. These correspond to the Highway on ramp and subsequent turn which cause the filter some trouble.

4.2 Missing Pair of Data

What happens if we're missing data? Assuming the measurements given were made using a mobile phone we shouldn't have access to the velocity data. The results of filtering the data without velocity measurements are shown in Figure 11.

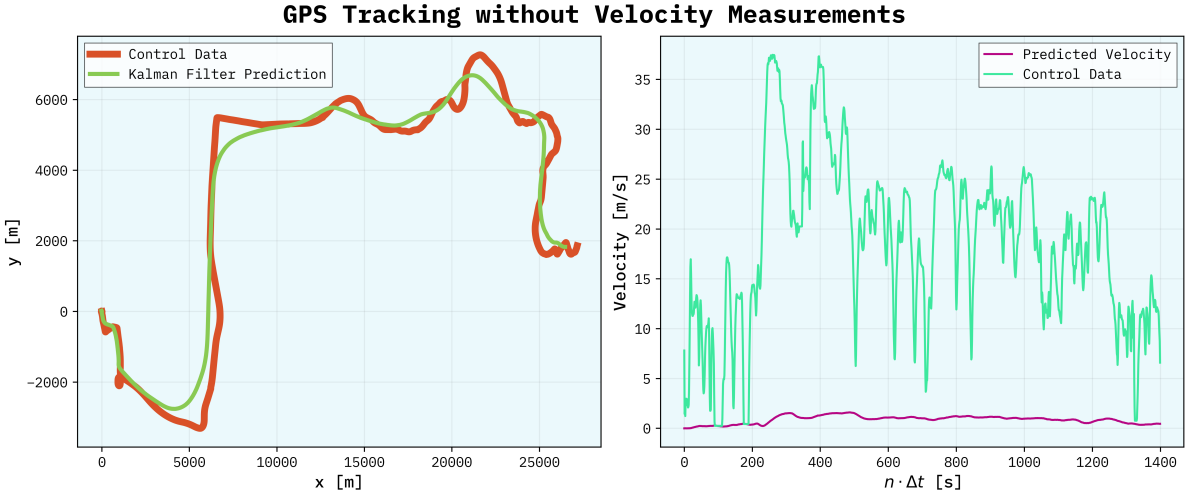

Figure 11: Filtered Cartesian GPS data without velocity measurements.

The filter does an abhorrent job of tracking the path compared to what we've seen before. We can also see that the velocity predictions it makes are very much off. I wasn't able to find a way to make the filter work better without the velocity measurements. So I do not have much to say about this case other than it does not seem to work. That means that my plan of penalizing Mr. Čopar if he speeds via just this data is not going to work or even if it did it would be very inaccurate and thus unfair. Consequently I did not make such a plot. Also since I do not want to antagonize Mr. Čopar. The issue of tracking with only acceleration is not uncommon however and is a real problem. I've read up online on a few novel solutions to this problem however discussing them is unfortunately outside the scope of this report. I do want to direct the reader to the PhD thesis of Oliver J. Woodman from the University of Cambridge [2] which discusses this problem in great detail.

Moving on, if we take switch the data around to where we're lacking acceleration data we get the results shown in Figure 12.

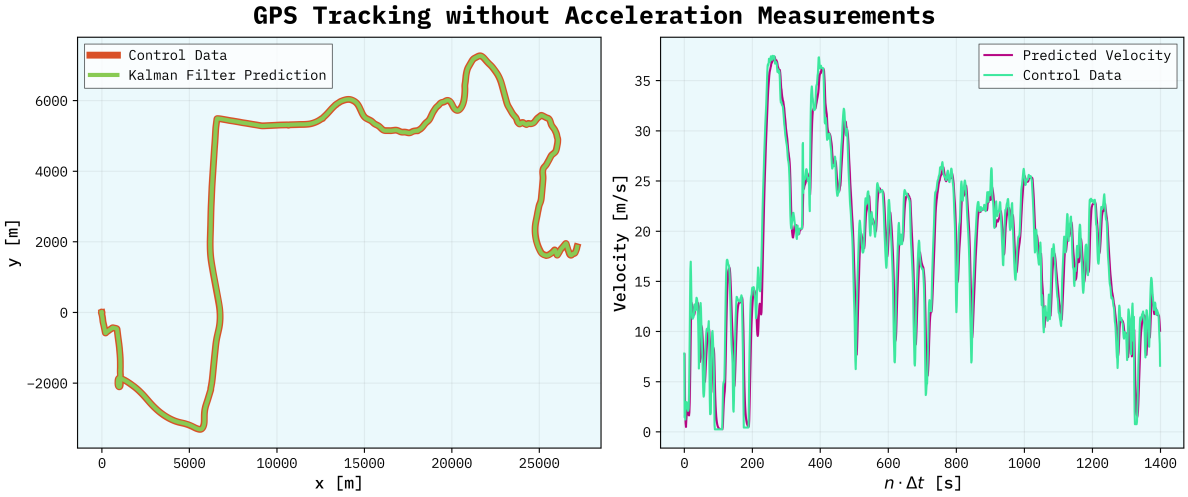

Figure 12: Filtered Cartesian GPS data without acceleration measurements.

The filter does quite a remarkable job of tracking the path even without the acceleration data. The velocity predictions are also very good. Makes me question why even bother gathering the acceleration data in the first place. I've plotted my argument below in Figure 13.

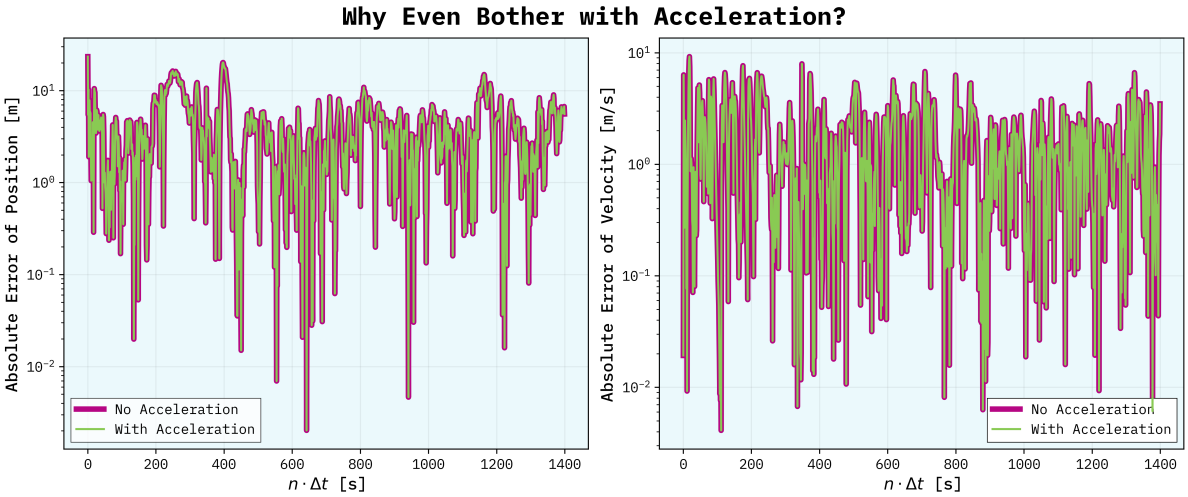

Figure 13: Position and Velocity absolute error with and without acceleration measurements.

It is also likely that I made a mistake in the code. I did not have time to check this. But hey, it seems to work just fine.

Since we're already playing around with missing data we could imagine that Mr. Čopar goes through a tunnel where he looses GPS signal. I simulated this by skipping the measurements for the tunnel. It

turns out that the filter still does a good job of tracking the path but it's performance is highly dependent on the location of the tunnel with respect to how difficult the path is to predict during the tunnel. Taking a simple example of a tunnel on a straight piece of road with medium velocity, the filter does a good job of predicting the path. This is shown in Figure 14.

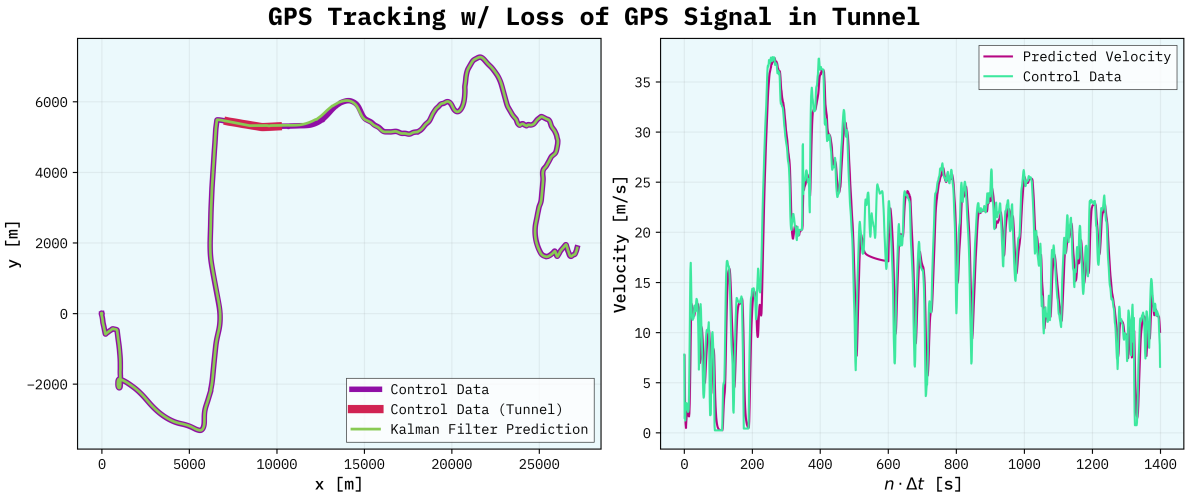

Figure 14: Filtered Cartesian GPS data with a tunnel.

Velocity predictions are slightly stunted during the tunnel but other than that it seems to work just fine. Now taking a more difficult location for the tunnel, on a gentle turn we see that the filter still manages to predict the path but it's performance is noticeably worse. The filter does quickly catch up however. This can be seen in Figure 15.

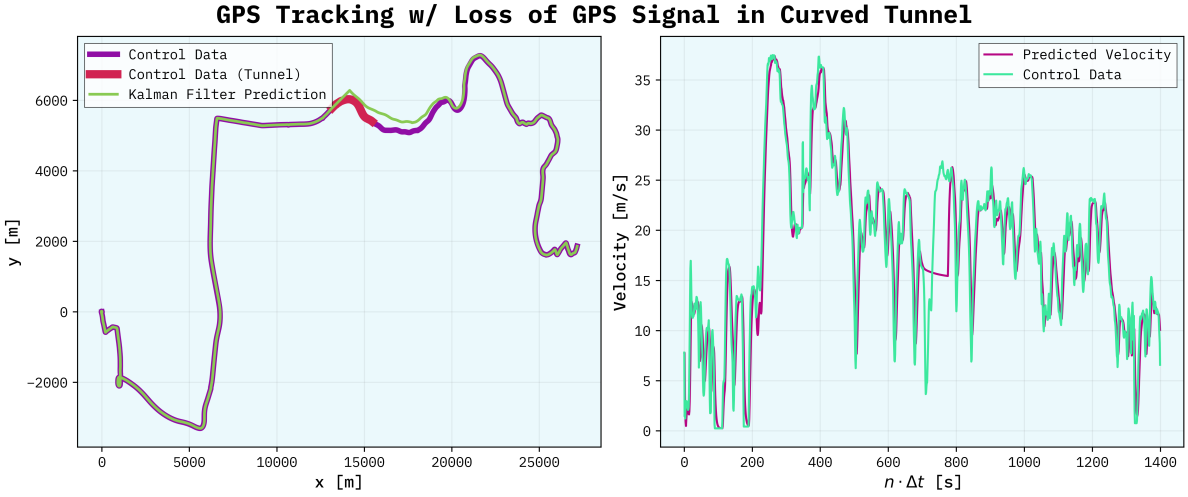

Figure 15: Filtered Cartesian GPS data with a tunnel on a turn.

So it seems that the filter has no problems with straight sections of missing data but struggles with more difficult path predictions. We can think of a pathological case which would cost our Ministry of Transport a lot of money. Let's say most of the straight highway section heading north from Ljubljana is inside a tunnel. This means that the filter has quite a large chunk of data missing. The results of this are shown in Figure 16.

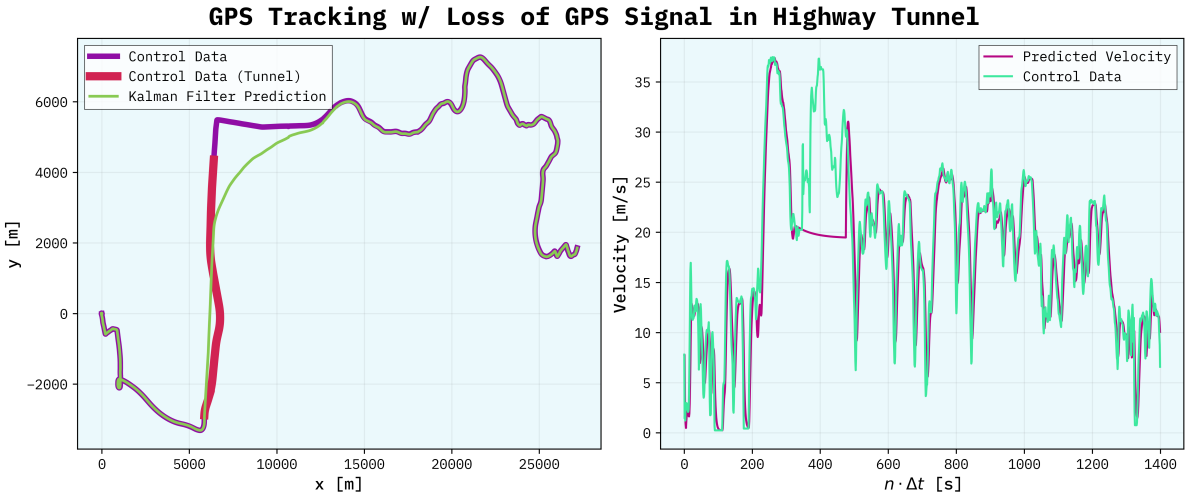

Figure 16: Filtered Cartesian GPS data with a tunnel on a straight highway.

We can see that despite the section being more or less straight and at high velocity the filter has a lot of trouble predicting the path. It practically skips the sharp turn at the end of the tunnel. I assume this is to an accumulation of errors for the period we have no data. Once data is available again the filter uses it with additional reserve due to the large uncertainty in the state estimate which causes it to lag behind and take quite some time to catch up. We can see this effect minimized if we make a gap in the tunnel where GPS data can once again be acquired for a short while before dipping back into the tunnel. The results of this are shown in Figure 17.

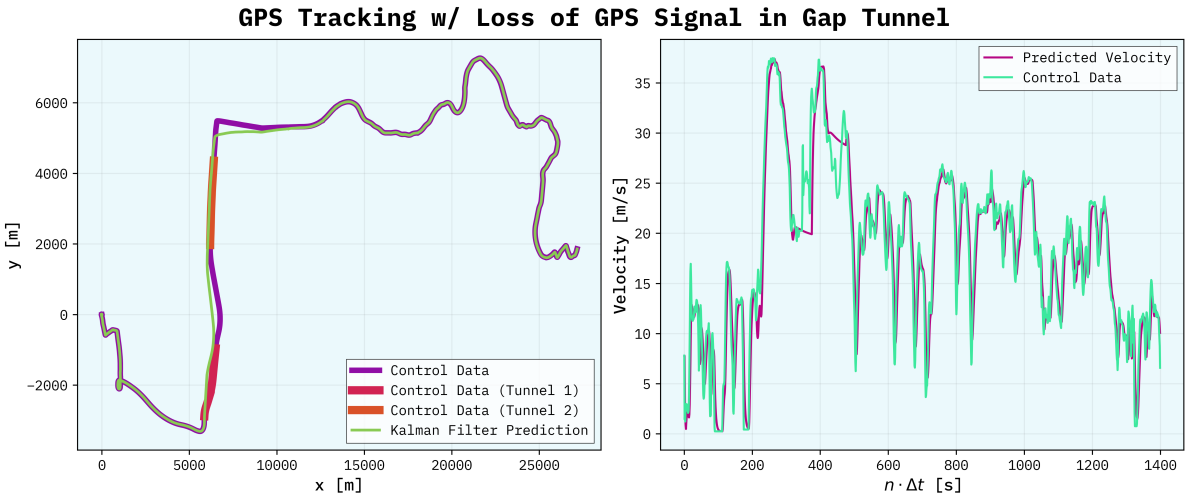

Figure 17: Filtered Cartesian GPS data with a tunnel on a straight highway with a gap.

We can see that the filter catches up considerably faster than before. This is a good argument for having GPS data available at regular intervals. The effects of the first tunnel are non-existent in this case, which we can prove by taking just the second tunnel. We see that we get a near identical result to the case with the gap. This is shown in Figure 18.

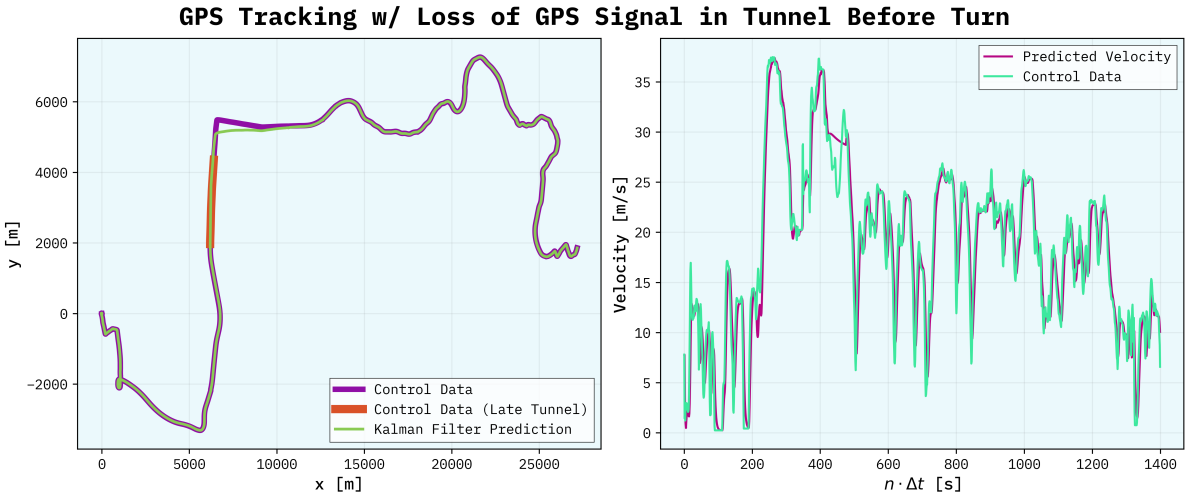

Figure 18: Filtered Cartesian GPS data with a tunnel at the end of the highway.

I tried my hand at actually mapping Mr. Čopar's path to a real map. I used the OpenStreetMap API in combination with `tilemapbase` to do this. I unfortunately failed quite miserably. I had a lot of problems with converting the distance from meters to changes in latitude and longitude and then changes in latitude and longitude to something called the **Web Mercator** projection. I suspect that the data itself could also be off? I'm not sure. The first street from the Faculty of Mathematics and Physics seems to be scaled correctly but then errors start to accumulate. In the end we do not even get halfway to Litija. I've included the plot in Figure 19.

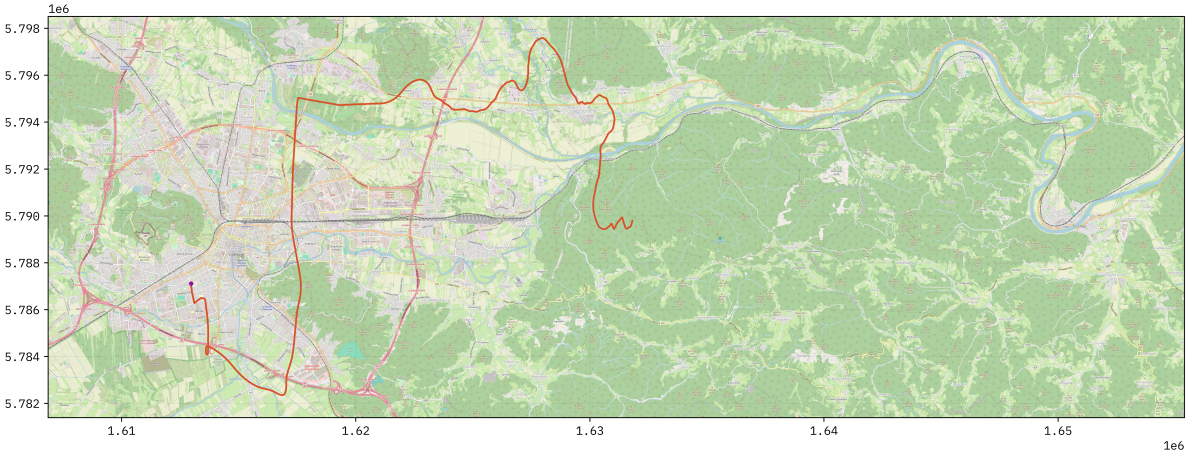

Figure 19: Mr. Čopar's path on a map.

4.3 Filtering of Relative GPS Data

Lastly lets have a look at the scary reality of relative GPS data. This took a bit of effort to turn into code but with the `KalmanFilter` template class it was still reasonable. The results of filtering the relative GPS data are shown in Figure 20.

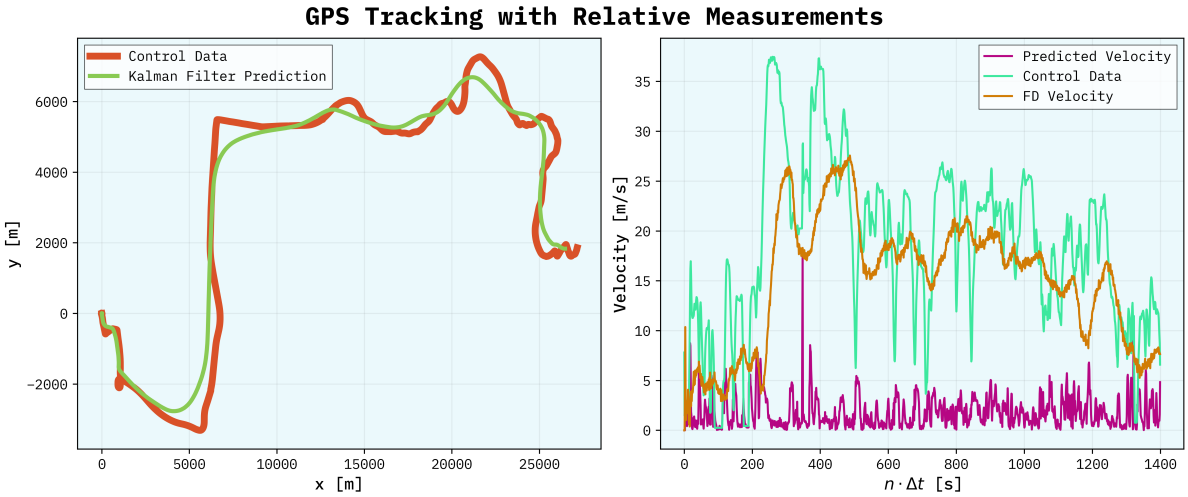

Figure 20: Filtered Relative GPS data with $\sigma_a = 0.05$.

Once again the result we get is abysmal compared to the Cartesian GPS data, however there exists a simple solution. Assuming these measurements were indeed made with a mobile phone we can be certain that the error in the acceleration measurements is much larger than 0.05 m/s^2. Figure 21 shows the results of filtering the relative GPS data with $\sigma_a = 1$.

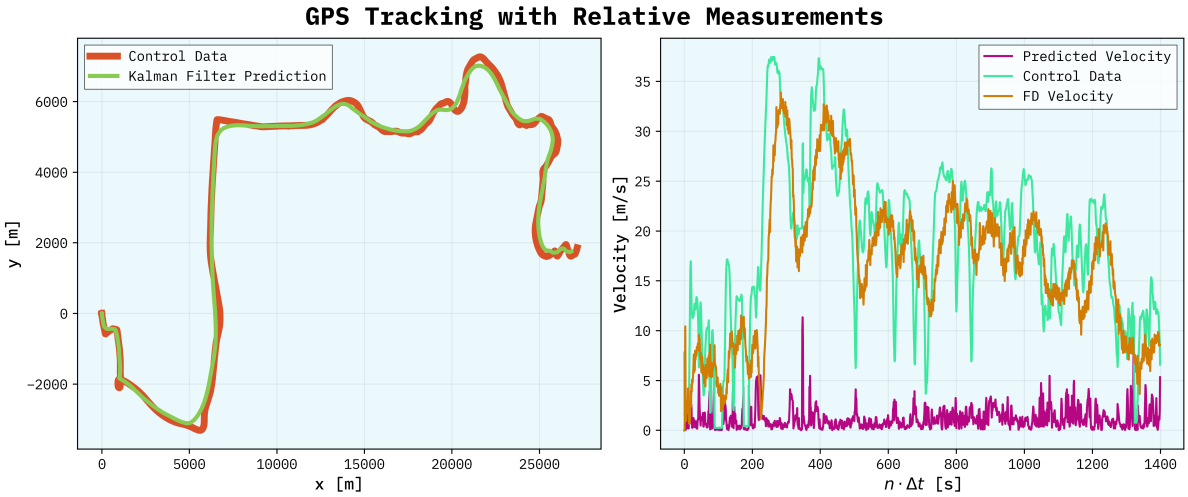

Figure 21: Filtered Relative GPS data with $\sigma_a = 1$.

The path is now quite a bit better but still not as good as the Cartesian GPS data. The velocity predictions are complete trash to be honest. I tried to mitigate this by taking a sort of *finite difference* approach to estimate the velocity from the position data. This is shown on the right-hand side in orange. These estimates are considerably better but will completely fail in the case of missing data. Thus the later plots will only show the filter's velocity predictions.

As we've done before we can take a look at what happens if we only take every 5th measurement into account. The results are shown in Figure 22.

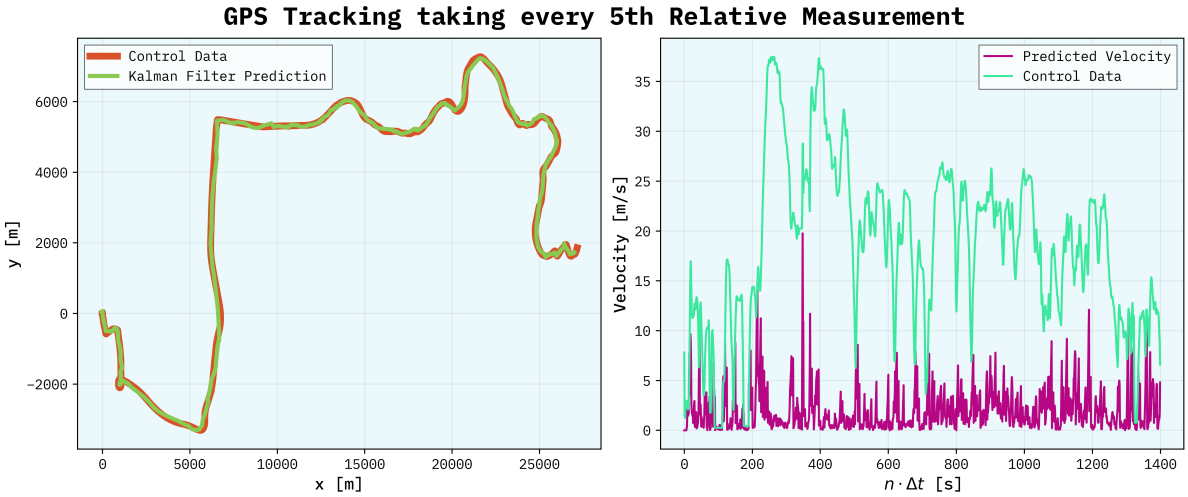

Figure 22: Filtered Relative GPS data with every 5th measurement.

To my surprise the filter **does better** with less data. I don't know if this caused by a fluke in my programming or if it's a real effect. I can't think of a possible explanation. The velocity predictions are still quite bad but lets call them a tad better than before. We can again take this further by taking only every 10th measurement into account. The results are shown in Figure 23.

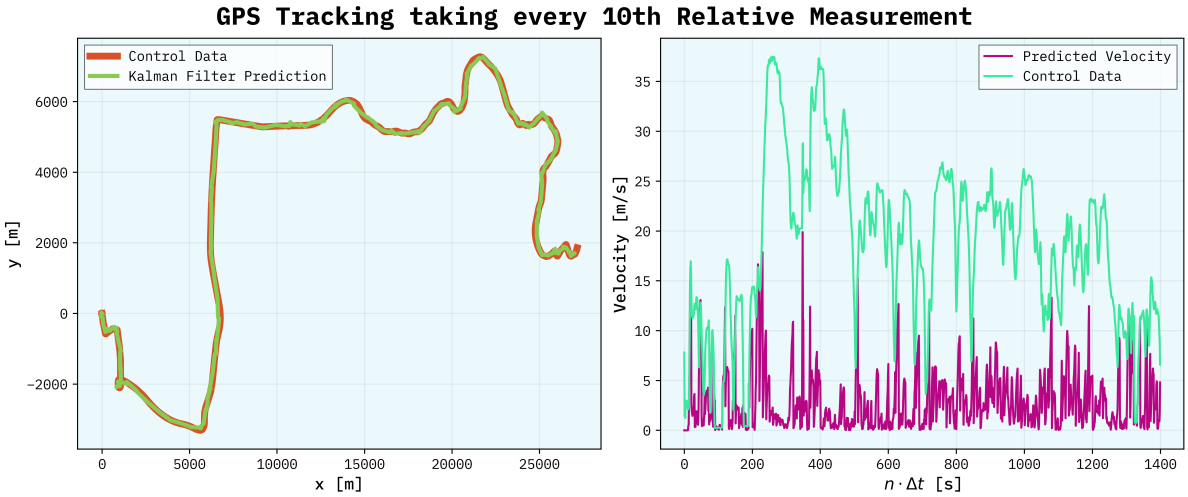

Figure 23: Filtered Relative GPS data with every 10th measurement.

Shockingly the filter still tracks well, better than with the full dataset in fact. The velocity predictions might be slightly better but the predicted path is no doubt more jagged than before. From this it seems to make sense to conclude that taking every 5th measurement is the best option. Again I'd like to stress that I don't know if this is a real effect or a nasty bug in my code. Does not mean that we can't explore this a little further. Since this task has a serious lack of much beloved parameter scans I decided to do exactly that. I scanned for different values of σ_a and intervals of missing data represented by N. $N = 1$ means we take all data, $N = 5$ means we take every 5th data point and so on. The results are shown in Figure 24.

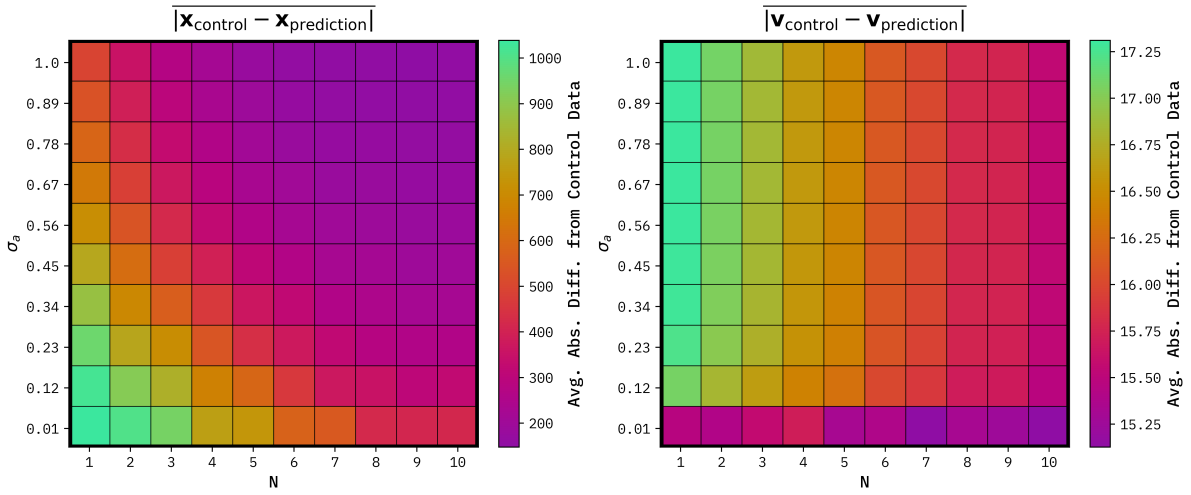

Figure 24: Average absolute difference in position and velocity for different values of σ_a and N.

Agh man this will turn out to be quite embarrassing if it turns out to be a bug but the parameter scan results are consistent with what we've seen a moment ago. It seems that while velocity predictions are better at low values of σ_a and N, the position predictions are better at higher values of σ_a and N. Worth noting that the velocity heatmap does look a bit strange with the lowest row yielding low errors while the rest above yield high errors. This might be due to possible funny business if we take $\sigma_a = 0.01$. To thouroghly investigate this strange phenomenon I've also plotted corresponding heatmaps for residuals of all the variables. The results are shown in Figure 25.

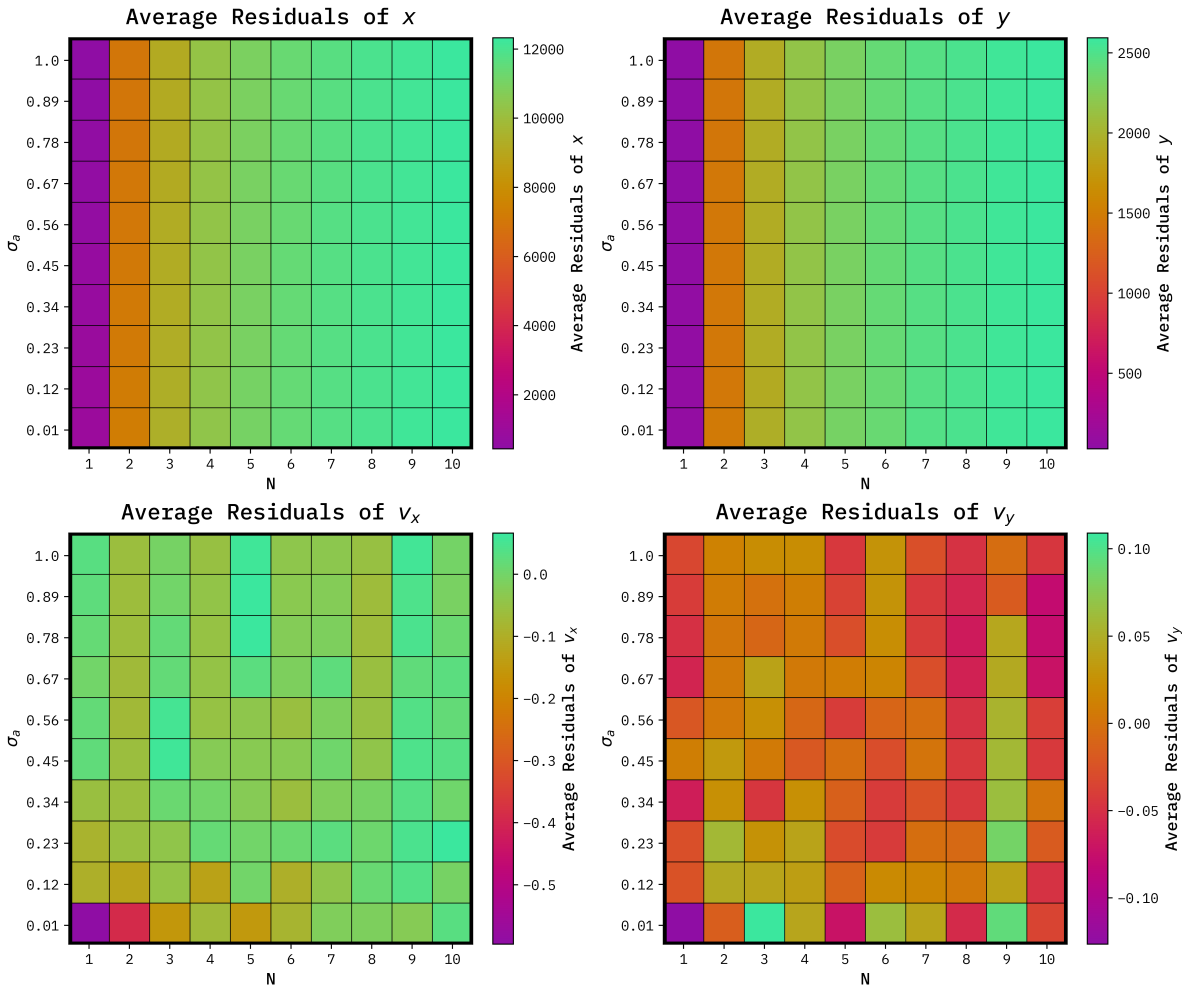

Figure 25: Heatmaps of the average residual for different values of σ_a and N.

The results are consistent with our findings so far. The residuals for the position are lowest at higher values of σ_a and entirely independent of N and the residuals for the velocity are lowest at lower values of σ_a and N. Lastly I've plotted the roots of the diagonal elements of the covariance matrix in Figure 26.

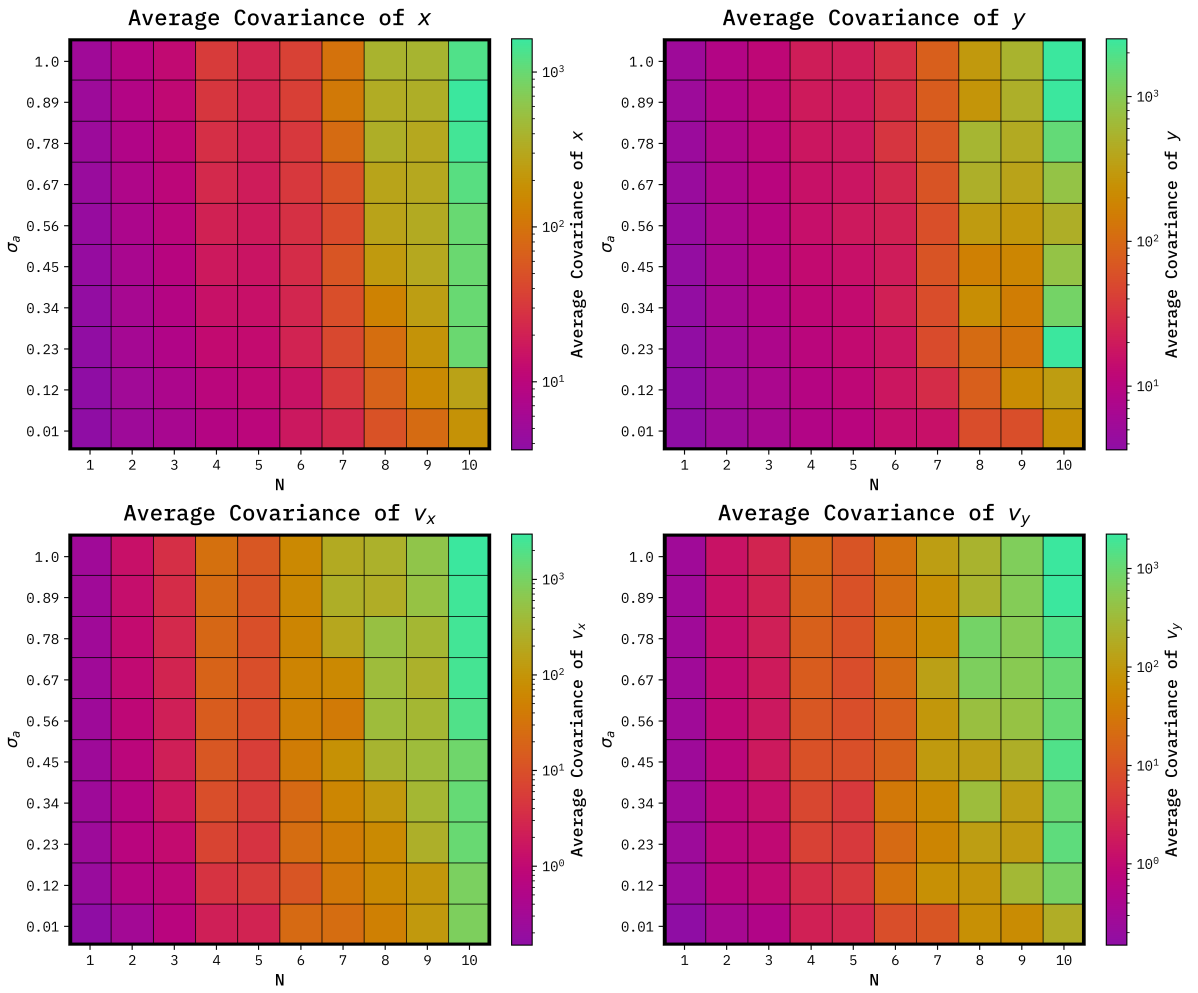

Figure 26: Average roots of the diagonal elements of the covariance matrix for different values of σ_a and N.

Oddly enough the the covariance for velocities is lowest at the lowest values of σ_a and N. This is not what I expected. The covariance for the position is lowest at the highest values of σ_a and N, which is consistent with what we've now discovered about the position predictions. I don't really have an explanation for this. For this entire effect. But I'm too short on time to investigate this further.

5 Conclusion and Comments

This task was a pain to start due to previous negative experiences with the Kalman filter. I was pleasantly surprised how well the professor explained its working and what makes it so powerful and cool. Something I was missing before. Due to my speedrun of the remaining tasks I didn't have enough time to explore more possible uses of the Kalman filter but I hope I plotted enough diverse data to make up for that. The parameter scan at the end is still a mystery to me. At least the colors look pretty I guess...

References

[1] Matthew Reed. An interview with jack crenshaw. http://www.trs-80.org/interview-jack-crenshaw/, Accessed 16th August 2024.

[2] Oliver Woodman and Robert Harle. Pedestrian localisation for indoor environments. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08, page 114–123, New York, NY, USA, 2008. Association for Computing Machinery.