Technical University of Berlin

Fraunhofer Institute for Open Communication Systems FOKUS

Faculty IV

Projekt Vertiefung / Vernetztes und Automatisiertes Fahren
Winter term 2022/2023

Cloud-based optimization of position estimation for bicycles and micro-mobility

Team members:

Qi Li (0464993)

Jiyuan Zhang (0461375)

Supervised by:

Bernd Schäufele

February 27, 2023

Contents

1 Introduction

With the ubiquity of smartphones, many modern transportation systems have been developed to simplify everyday life. However, for cyclists and micro-mobility enthusiasts, most services lack online assistance with live information during the ride. The complexity of terrain transformations in urban environments further complicates accurate GPS location. This is due to the significantly poorer GPS performance in urban areas, where signals can be blocked by buildings and trees, resulting in lower signal quality and delayed real-time transmission. Additionally, GPS signals can be blocked and influenced by high buildings, human bodies, and mountains, causing errors between 1 to 10 meters, which is unacceptable for certain applications. As a result, the assistance of other devices such as IMUs is necessary for achieving accurate positioning.

In this project, a fused positioning algorithm will be developed that utilizes data from multiple smartphones, particularly sensors such as gyroscopes and accelerometers. The aim is to implement a more accurate and smooth bicycle trajectory tracking algorithm and deploy it on a cloud server.

2 Objectives

The main objective was to use a fusion positioning algorithm to fuse IMU data like acceleration and rotation vector values in order to obtain a more accurate and smooth trajectory of the bicycle.
There were also secondary objectives: set up a backend solution running on a cloud server that allows people to use it anytime, anywhere with their mobile phones.

3 Structure

The entire implementation process is shown in figure 1

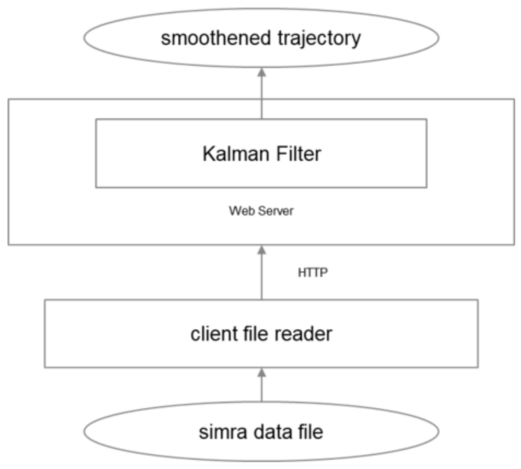


Figure 1: Implementation Process

First, we read the necessary data from the SimRa dataset and process it into the required format. Then, we use HTTP to transfer the data to the web server, and apply a kalman filter to process the data. Finally, we obtain an accurate and smooth estimate.

4 Dataset

we use the SimRa dataset which consists of the main routes of bicycle traffic in Berlin. It is done via a smartphone app that uses GPS information to track routes of bicyclists and uses built-in sensors to record many other datas like imu data. The dataset contains a lot of relevant information but what we need is ride information.2

Data type:

- GPS data: location – latitude and longitudes
- IMU data: accelerometer sensor data – X/Y/Z
- IMU data: gyroscope sensor data – a/b/c
- IMU data: linear accelerometer values – XL/YL/ZL(without gravity)
- IMU data: rotation vector values – RX/RY/RZ/RC
- GPS/IMU data: timestamp(the frequency of IMU data is higher than the GPS data.)

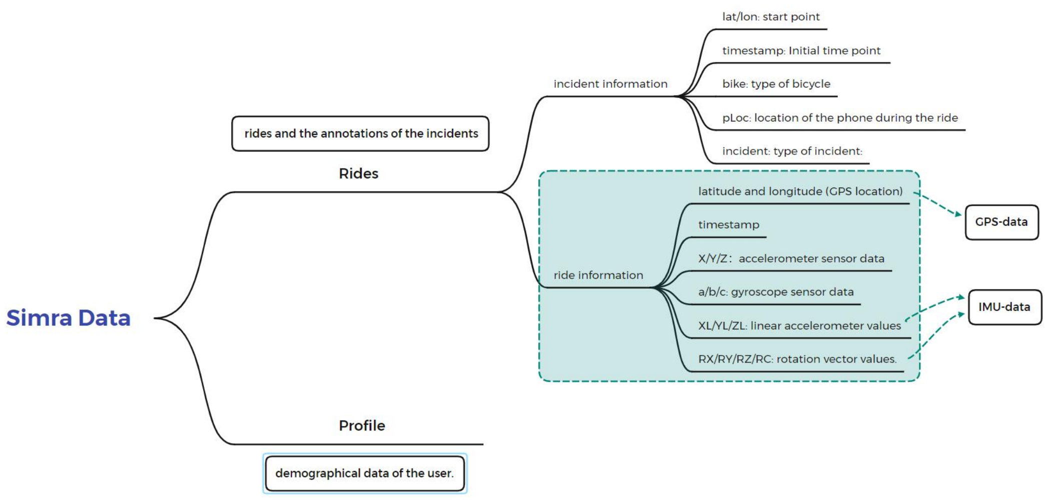

Simra Data
Rides
rides and the annotations of the incidents
incident information
lat/lon: start point
timestamp: Initial time point
bike: type of bicycle
pLoc: location of the phone during the ride
incident: type of incident.
ride information
latitude and longitude (GPS location)
timestamp
X/Y/Z: accelerometer sensor data
a/b/c: gyroscope sensor data
XL/YL/ZL: linear accelerometer values
RX/RY/RZ/RC: rotation vector values.
GPS-data
IMU-data
Profile
demographical data of the user.

Figure 2: SimRa Dataset

5 Kalman filter

5.1 Selection of Kalman filter

There are many different types of Kalman filters. In order to better solve our problem, after comparing several different types of Kalman filters, we have chosen the Error State Extended Kalman Filter(ESEKF).

Types	**Features**
Common Kalman Filter(KF)	linear estimator
	direct method of filtering(full state)
Extended Kalman Filter(EKF)	unlinear estimator
	direct method of filtering(full state)
Error State Extended Kalman Filter(ESEKF)	unlinear estimator
	indirect method of filtering(error state)

Table 1: Comparison of Kalman filters

From the table1, we can see that the common Kalman filter is a best linear unbiased estimator. However,it cannot be used directly to estimate states that are non-linear functions. But in our project, the pose of the bicycle includes it's orientation which is not a linear quantity. so we can not use common kalman filter.
So there is the extended Kalman filter(EKF). It is also a direct method of filtering like common Kalman filter. The EKF uses linearization to adapt the Kalman filter to nonlinear systems. It works by linearizing the nonlinear motion and measurement models to update the mean and covariance of the state.
But finally we choose the Error State Extended Kalman Filter(ESEKF). It is an improved version of EKF and it has better performance, because it is a indirect method of filtering and it focus on error state.[2]

5.2 Error State Extended Kalman Filter(ESEKF)

5.2.1 Principle of ESEKF

the state of the ESEKF is being composed of two parts: the large part called the nominal state x hat, and a small part called the error state, Delta x.

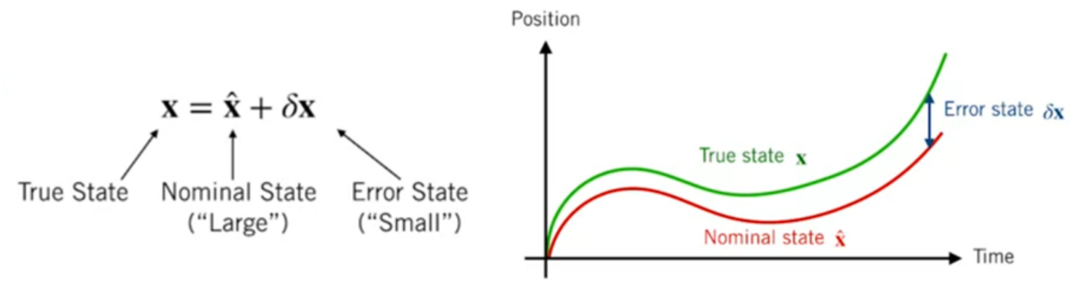

$\mathbf{x} = \hat{\mathbf{x}} + \delta\mathbf{x}$
True State
Nominal State ("Large")
Error State ("Small")
Position
Error state $\delta\mathbf{x}$
True state $\mathbf{x}$
Nominal state $\hat{\mathbf{x}}$
Time

Figure 3: State of ESEKF

We can see from the figure3 that the green line represents the true position of the bicycle, which is the quantity we are trying to estimate, while the red line represents the nominal state. The nominal state is our best estimate of what the true state could be based on our model and inputs. However, since our model is never perfect and there is always some random process noise, the error state is simply the difference between the nominal state and the true state at any given time. We can use this error state as a correction to the nominal state to bring us closer to the true state. Therefore, in the error state Kalman filter, instead of performing Kalman filtering on the full state, which might have lots of complicated non-linear behaviors, we will use the EKF to estimate the error state instead.

5.2.2 Loop of the ESEKF

Below is the simple prediction and update loop of the ESEKF:[1]
1. update nominal state with motion model:

$$\check{x}_k = f_{k-1}(x_{k-1}, u_{k-1}, 0) \tag{1}$$

2. propagate uncertainty(state covariance)

$$\check{P}_k = F_{k-1}P_{k-1}F_{k-1}^T + L_{k-1}Q_{k-1}L_{k-1}^T \tag{2}$$

3. if a measurement is available:

   1. computer kalman gain:

   $$K_k = \check{P}_k H_k^T (H_k \check{P}_k H_k^T + R)^{-1} \tag{3}$$

   2. computer error state

   $$\delta\hat{x}_k = K_k(y_k - h_k(\check{x}_k, 0)) \tag{4}$$

   3. correct nominal state

   $$\hat{x}_k = \check{x}_k + \delta\hat{x}_k \tag{5}$$

   4. correct state covariance

   $$\hat{P}_k = (1 - K_k H_k)\check{P}_k \tag{6}$$

By comparing the equations above with those of the common Kalman filter, we can see that computing the error state and correcting the nominal state is the core of the Error State Extended Kalman Filter. By processing the error state instead of the nominal state, we can often achieve better performance.

5.2.3 Advantages of the ESEKF

There are two good reasons to use the ESEKF.

- First, it often works better than the vanilla EKF because the small error state is more amenable to linear filtering than the large nominal state, which we can integrate non-linearly.
- Second, the error state formulation makes it much easier to work with constrained quantities like 3D rotations.

6 Result

The EKF algorithm for IMU-GPS fusion involves the following steps:

Initialization: The EKF is initialized with an initial estimate of the state, which typically comes from GPS data. The EKF also requires an initial estimate of the covariance matrix, which describes the uncertainty in the initial estimate.

Prediction: The EKF predicts the state of the system at the next time step based on the current state estimate and the IMU data. The prediction includes a prediction of the state mean and covariance.

Measurement update: When GPS data is available, the EKF updates the state estimate using the GPS measurement. This update includes a correction of the state mean and covariance based on the difference between the predicted state and the GPS measurement.

Looping: The prediction and measurement update steps are repeated for each time step, allowing the EKF to track the object's position and orientation over time.

In summary, IMU-GPS fusion algorithms like the EKF combine high-frequency, low-accuracy IMU data with low-frequency, high-accuracy GPS data to estimate the position and orientation of a moving object. The EKF is a common algorithm used for this purpose, and it involves initializing the filter, predicting the state of the system, updating the state with GPS measurements, and repeating the process for each time step.

First of all, as we mentioned earlier, the algorithm and the details. We will show the results4 of the algorithm. We can see the form of the bike's trajectory in the 3D map. The blue line in the picture is the bicycle's trajectory. Our subsequent simulations are based on this actual route.

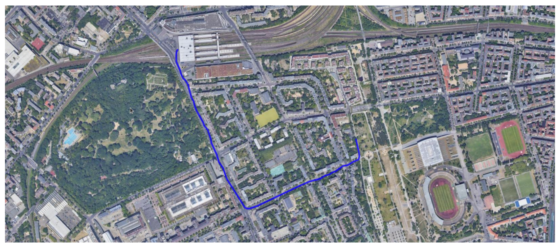


Figure 4: Trajectory in the 3D map

The result of our improved fusion is shown in the figure5. The blue line shows the conversion of the GPS data from the original data to the ENU coordinate system. When compared to the shape in the 3D map, they are the same. The red line is the curve generated by our fusion algorithm. We can clearly see that the orientation of the two curves is very different, even at the back where the orientation deviates completely from the actual position.

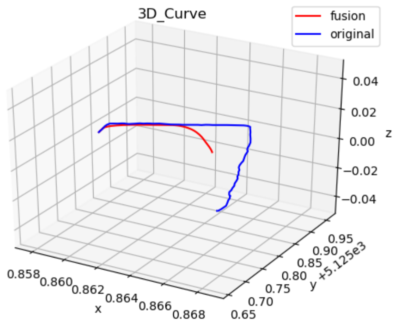


Figure 5: Fusion result in ENU coordinate system

We have still printed the changes in latitude and longitude separately, and we can see from the graph that over time. The difference between the latitude and longitude results before and after fusion is so large that the algorithm cannot be considered successful and we have to improve it.

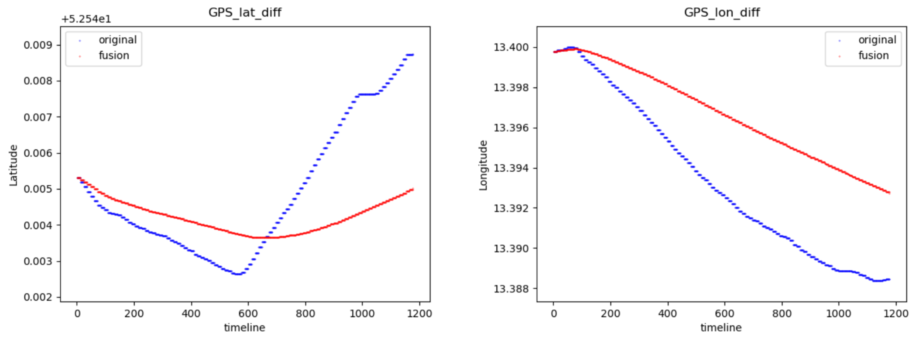


Figure 6: Differences in latitude and longitude

The details of the fusion algorithm and the raw data have been checked and experimented with. According to a survey of the data, in industry the IMU in sensors is updated at a frequency of around 200Hz. In the field of autonomous driving, the IMU update frequency is around 400Hz. The high update frequency of the IMU means that it can capture and record the movement of the device very quickly. For example, if a bicycle or a vehicle deflects slightly or accelerates or decelerates, the sensor's IMU will actually record it.

Types	**Frequency**
IMU sensors in industry	round 200 Hz
IMU sensors in Autonomous Driving	round 400 Hz
IMU sensors in our dataset	round 5 Hz

In our project data, we tested the update frequency of the IMU. This means that the IMU is updated every 0.2s and that if the bike or vehicle deflects slightly or accelerates or decelerates during the process, the IMU sensor located in the phone will not record it in time. With the ESKF algorithm, our prediction is partially off. This means that during the update phase of the fusion algorithm, the actual GPS position differs too much from the predicted position, causing the fused route to be shifted.

In order to solve this situation, and since we cannot re-record the data. We have to improve on the existing data to make our data more informative. We propose, among other things, to interpolate the raw data. We artificially add more frames to the middle of the two frames to create a higher frequency of IMU. We smooth and complete the IMU data. After our interpolation process, we then re-visualize the data fusion results.

From the adjusted algorithm 7, we can see that the red line represents the fusion result, and the blue current represents the position represented by the GPS in the source data. We can find the gap between the red line and the blue line It is not big and the red line shows more details as a result of fusion. We also display some information of longitude and latitude separately. You can see from the figure that after the algorithm is improved, our longitude and latitude are fused The gap between before and after fusion is not particularly large. But it will also show some gaps. This also proves the necessity of our fusion algorithm.

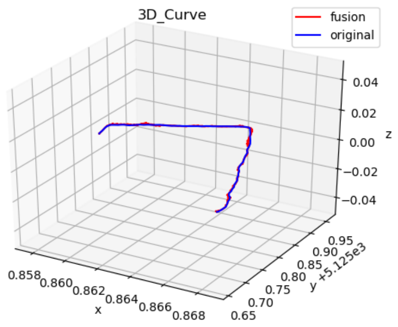


Figure 7: Fusion result in ENU coordinate system

In the presented results8 we try to analyze this data with different dimensions. First of all, this data is the original data and its duration is 296 seconds. The largest deviation in longitude and latitude is 0.0005, and the smallest deviation is -0.0003. There are a total of 17500 frames of data in our original data. In other words, it means that our IM U sensor has been updated nearly 20,000 times during the 296 seconds of cycling. Of course, as we said earlier, such an update is still very slow for the frequency of the IMU. We can't get the correct pose and where he is based on the transformation of the IMU. So after our interpolation process our frame count increased from 17,500 to 300,000 frames. We artificially speed up the frequency of data updates. It also proves that our algorithm becomes more accurate in the prediction stage. And the result is indeed like this.

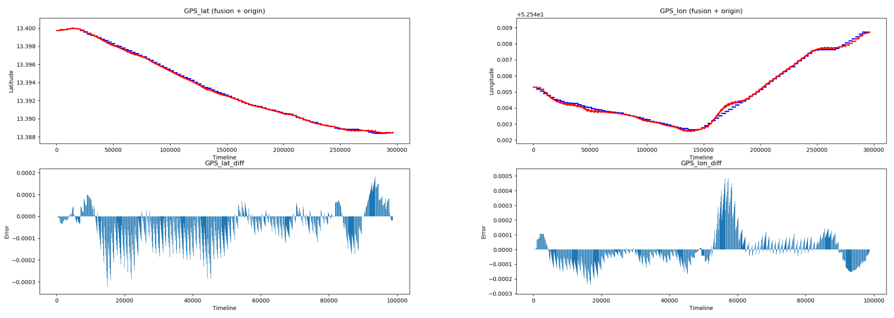
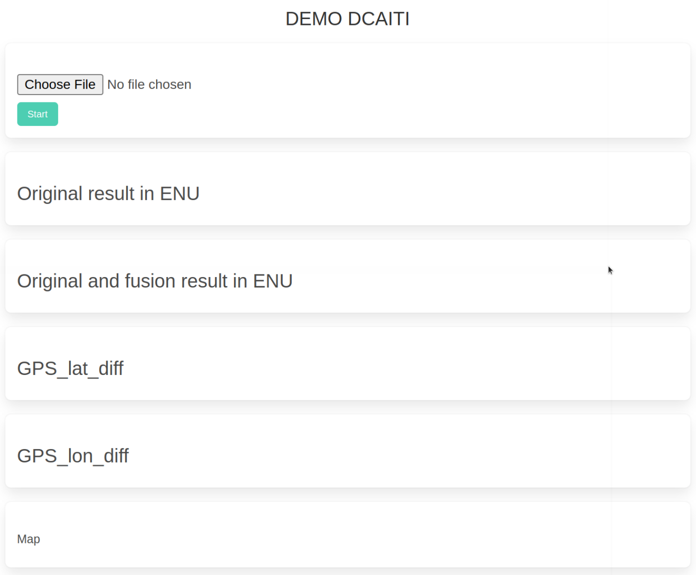


Figure 8: Differences in latitude and longitude

7 Web server

For the purpose of this project we are not only developing a fusion algorithm for GPS and imu. We also need to use the web page for data visualization. Because our fusion algorithm will be integrated into mobile phone software in future applications. In this way, we can run the operation algorithm and visualize the final results on the mobile phone. In this project we designed the frontend and backend of the webpage 9. We use javascript for the front end. We use Python's Django backend library for the backend. We use the fusion algorithm of gps and imu to calculate our original data in the process of building the frontend and backend of the webpage. That is to say, the input on the webpage is our Raw driving record data.


DEMO DCAITI

Original result in ENU

Original and fusion result in ENU

GPS_lat_diff

GPS_lon_diff

Map

Figure 9: Web server demo

When the system starts to run, our raw data enters the process of GPS and IMU fusion calculation and gets corresponding results. Such as longitude, latitude and the changes between them. Also includes their mapping in 3D maps. Other than that you can still see the trajectory formed by their coordinate system after converting to ENU coordinate system. This is convenient for us to compare with the driving trajectory in the 3D map to judge whether the algorithm is correct or not.

8 Future research and Conclusion

For this project, we have realized the development of the fusion positioning algorithm of imu and GPS, and we have also built the visualization on the web page. This includes front-end design and back-end operation. Of course, we still have several aspects that can be improved in the future. In the recording stage of raw data, we hope to use higher frequency IMU sensors to obtain data with faster update frequency to suit our algorithm Let the algorithm integrate the results that are better in line with reality and more accurate. Secondly, our ultimate goal is to integrate our algorithm into mobile applications. We still need a lot of tools such as Android Studio for integration into mobile applications. This is for It is of great significance in terms of automatic driving and vehicle navigation direction.

References

[1] Igor Brigadnov, Aleksandr Lutonin, and Kseniia Bogdanova. “Error State Extended Kalman Filter Localization for Underground Mining Environments”. In: *Symmetry* 15.2 (2023). ISSN: 2073-8994. DOI: `10.3390/sym15020344`. URL: `https://www.mdpi.com/2073-8994/15/2/344`.

[2] Venkatesh Madyastha et al. “Extended Kalman filter vs. error state Kalman filter for aircraft attitude estimation”. In: *AIAA Guidance, Navigation, and Control Conference*. 2011, p. 6615.