Uncertainty aware inertial navigation and indoor position fusion

Xiangyu Wen

4th Year Project Report
Artificial Intelligence
School of Informatics
University of Edinburgh

2023

Abstract

Inertial navigation systems estimate the position and velocity of a moving object by measuring the accelerations and angular velocities it experiences using sensors called IMU. However inertial-based method suffers from accumulated errors over time and is unreliable for long-term navigation. Therefore in order to obtain reliable navigation for a long time, a feasible way is to fuse the inertial information with other sources of position information, an example is GPS and IMU fusion. This is a common technique used for tracking, navigation, and mapping applications. However, GPS signals can be affected by various factors, especially in indoor environments, resulting inaccurate position and velocity estimation. This project aims to develop a pedestrian inertial navigation system with uncertainty awareness in indoor environments and use the predicted displacement and uncertainty integrating with the position obtained from indoor WiFi measurement to obtain a reliable IPS (Indoor Positioning System). Mobile phones are a versatile platform for this project as most modern smartphones are equipped with a variety of sensors, including WiFi and IMU. Therefore the data can be collected using a single device. Another advantage is that most people already have a mobile phone. This means that the IPS can be easily deployed without requiring additional hardware.

Acknowledgements

First of all, I would like to thank my supervisor Professor Chris LU for his guidance, affirmation, encouragement and assistance since last summer. I would also like to express my gratitude to the other members of the MAPS LAB for providing a good research and learning environment, especially for the collaborations with Peize Li and Qiyue Xia in this project. Most importantly, I am grateful to my parents for their long-term material and spiritual support. Furthermore, I would like to thank the useful tool ChatGPT provided by OpenAI for assisting me in polishing sentences in this thesis. Finally, I am grateful that the weather in Edinburgh for the past four years hasn't been worse.

Contents

Chapter 1

Intorduction

Indoor navigation refers to the process of using various technical means to determine and track the location of people in an indoor environment. The demand for indoor location-based services in indoor environments has been rapidly increasing in recent years, for example, in public places such as hospitals, museums and shopping malls, an accurate and reliable navigation system can provide location-sensitive guidance, information, or marketing.

While outdoor navigation systems can heavily depend on GPS (Global Positioning System), designing a reliable Indoor Positioning System (IPS) remains a challenging problem. With GPS signals often blocked or weakened by solid materials such as walls and roofs, IPS utilizes onboard sensors such as inertial measurement units (IMU), and building-dependent infrastructures such as WiFi, Bluetooth or magnetic fingerprints. The noisy nature of these measurements often means leveraging multiple sources of sensor measurements is a must.

Among these sensor measurements, inertial navigation methods such as pedestrian dead reckoning (PDR) have the benefit of high frequency and short-term accuracy while suffering from long-term drift due to the accumulation of sensor errors. On the contrary, WiFi localization has the benefit of uniqueness across large areas, while suffering from low location resolution in short distances. The fusion of inertial navigation and WiFi localization will logically create an IPS that is simultaneously accurate in the short term and reliable in the long term. The IPS project of the MAPS Lab is an ongoing project that focuses on the research of (1) uncertainty-aware inertial navigation, (2) uncertainty-aware WiFi localization, and (3) the effective use of uncertainty estimation in IPS sensor fusion.

As a further context of the MAPS Lab IPS project, we aim to build a novel IPS without the need for custom infrastructure in public buildings. Normally in a WiFi-based localization system, we either need to place custom WiFi Access Points (APs) or send surveyors to survey the WiFi signals in each building of the IPS. This will significantly limit the IPS's wide deployment in the countless indoor public spaces in the world. To solve this problem, we use crowdsourced WiFi data from everyday smartphone users visiting these public spaces and use the anonymised data to build a WiFi radio map for

each building. While this solves the wide deployment problem, the WiFi localization will have larger errors than traditional methods, and the crowdsourced radio map will often not have full coverage of the building. We aim to solve these challenges with our novel uncertainty estimation and fusion techniques.

This thesis tackles two of the three research branches of the MAPS Lab IPS project. The first task is to develop an uncertainty-aware inertial navigation system based on deep learning for indoor positioning using a mobile phone. The data-driven based model will estimate the user's position and uncertainty in real time using the IMU sensor data and deep learning techniques. The second task is to construct a sensors fusion IPS that can effectively use the estimated displacement and uncertainty from the inertial model for further positioning fusion with a WiFi location system developed by partners in the IPAB MAPS Lab.

In this thesis, the following research questions are aimed to address:

- How to implement the uncertainty-aware inertial navigation system?
- How to fuse the motion uncertainty estimation from the uncertainty-aware inertial navigation system with the localization from the WiFi system?
- Whether the estimated uncertainty is useful for the fusion?

Chapter 2

Background

2.1 Inertial navigation

Pedestrian Dead Reckoning (PDR) is a method of tracking the movement of pedestrians using an inertial sensor (IMU). IMU typically contains accelerometers, gyroscopes, and magnetometers. These sensors measure the linear acceleration, angular acceleration and magnetic field of the device. Relative motion is estimated using these measurements and thus construct a trajectory from the starting point.

Kalman filter [Kalman, 1960] is a recursive method to update the state estimate with the corresponding covariance matrix over time. In the context of PDR, the Kalman filter can be used to fuse inertial sensors, such as accelerometers and gyroscopes to estimate the displacement. However traditional Kalman filter assumes linearity and the PDR problem is highly non-linear. To address this problem, a variant of KF – the Extended Kalman Filter (EKF) is commonly used in PDR. The EKF works in two stages. First, the prediction stage uses the motion model to predict the state and its associated covariance matrix. Second, the correction stage incorporates the measurements from the sensors to correct the state estimate and corresponding covariance matrix. The accuracy and reliability of EKF methods depend on the accuracy of the motion model and the measurement model, thus developing accurate models for PDR is essential in order to obtain high-quality PDR using EKF.

ZUPT (Zero Velocity Update) is a technique used in inertial navigation systems, especially the foot-mounted IMU. In Foxlin's seminal paper [Foxlin, 2005], the sensor data is processed using a Kalman filter to estimate the position and velocity of the pedestrian, and zero-velocity update is used to reduce accumulated velocity errors by detecting and correcting the drift in the velocity estimates when the pedestrian is stationary using the fact that the pedestrian is stationary while the foot in on the ground. When the foot is stationary, an zero acceleration measurement show be observed and non-zero measurement can be attributed to sensor noise. In short, ZUPT detects the stationary period of the pedestrian using a threshold-based approach and set the velocity state to zero for these periods.

Besides traditional methods, recently deep learning has shown its power in the inertial

navigation area. IONet [Chen et al., 2018] first proposed to regress the velocity and direction using LSTM (Long Short-Term Memory) [Hochreiter and Schmidhuber, 1997]. IONet breaks the cycle of continuous integration by segmenting inertial data into independent windows, and the authors show this can be formulated as an optimization problem and demonstrate how deep recurrent neural networks can produce highly accurate trajectories. Their experiments show that IONet can even predict the odometry for non-periodic movements, such as shopping trolleys. RoNIN [Yan et al., 2019] short for Robust Neural Inertial Navigation, is another deep learning-based PDR system. It explores residual networks, temporal CNNs and RNNs to regress the footpaths in 2D. It is trained on a large dataset of sensor data and corresponding ground truth positions, which enables it to learn the complex relationship and achieve high robustness whatever the phone pose is.

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that is most commonly used in deep learning-based inertial data processing problems. LSTM is designed to handle the vanishing gradient problem, which occurs when training very deep recurrent neural networks. The vanishing gradient problem arises when the gradients used to update the weights in the network become very small, causing the network to converge very slowly or not at all. The principle of LSTMs is using a memory cell that can store information over long periods of time, allowing the network to learn long-term dependencies in the series data [Hochreiter, 1998] [Hochreiter and Schmidhuber, 1997]

It is crucial to note that RIDI [Yan et al., 2017] is a significant data-driven approach to pedestrian dead reckoning, which does not rely on a deep learning network. It proposed an important technique called *stabilized-IMU frame*. This preprocessing technique rotates the measurements from the imu's frame to the world frame, it plays a vital role in the proposed system presented in this thesis.

2.2 Uncertainty estimation

Uncertainty is a fundamental concept in many areas of science, engineering and decision-making. It is often expressed as a range of possible outcomes or a probability distribution that describes the likelihood of different outcomes and can arise from many sources, including imperfect models or predictions, and inherent randomness or variability in the system being studied. In the context of neural networks, the quantification and estimation of uncertainty are essential for understanding the limits of model performance, generalisation and robustness. This chapter aims to provide an overview of the different types of uncertainty and methods for estimating uncertainty in neural networks.

Uncertainty estimation involves determining the range of possible outcomes or the level of confidence in a given prediction or measurement, taking into account noise, error and variability in the data or model. Uncertainty in neural networks can be broadly classified into two types: aleatoric uncertainties and epistemic uncertainties [Kendall and Gal, 2017].

Aleatoric uncertainty arises from the inherent noise or variability in the data. Therefore it is also called data uncertainty. Aleatoric uncertainties exist even in an ideal model, and they can be further divided into two classes:

Homoscedastic uncertainty: constant across all input data, representing the uniform noise.

Heteroscedastic uncertainty: varies across different input data, representing the varying noise.

Epistemic uncertainty exists because of the limitations of the neural network model in capturing the true underlying data-generating process. Therefore it is also called model uncertainty. This uncertainty could theoretically be minimized by using a better training approach.

Since the importance of uncertainty in neural networks was realized, a great number of uncertainty prediction methods have been proposed. Bayesian neural networks consider the weights of the neural network as random variables with a prior probability distribution and use the posterior distribution to make predictions and output uncertainties [Blundell et al., 2015]. In general, BNNs are more complex and need more computational resources to train compared to traditional neural networks.

MC dropout [Gal and Ghahramani, 2016] extends the dropout technique to inference by using the same dropout layers used during training to generate multiple predictions with different dropped-out neurons to estimate the model uncertainty, in their work, the author proved that dropout can be used as a Bayesian approximation. However, one disadvantage of using MC dropout is that an accurate uncertainty estimation required a large number of forward inferences, therefore not very suitable for real-time application.

To overcome these problems [Kendall and Gal, 2017] firstly proposed a method to combine the aleatoric and epistemic uncertainty in a novel Bayesian deep learning framework.However, Bayesian neural networks often face a number of limitations, such as the difficulty of directly inferring the posterior distribution of weights given the data

and the large computation resources required to sample the distribution during inference. [Amini et al., 2020] proposed Deep Evidential Regression. This method learns both epistemic and aleatoric uncertainty on continuous regression problems without sampling during inference or training.

2.3 Positioning Fusion

Positioning fusion refers to the process of integrating data from multiple sources to provide accurate and reliable position, and motion information for various applications, such as navigation, robotics, and mapping. For example, GPS-Inertial fusion is a common and important application. GPS-Inertial Fusion combines data from Global Positioning System (GPS) receivers and Inertial Measurement Units (IMUs) to improve the accuracy and reliability of positioning information. Fusion algorithm can combine the advantages of sensors to overcome the problem when using a single sensor, such as lack of credibility of GPS signal in some cases and the drift of IMUs [Caron et al., 2006].

The extended Kalman filter(EKF) and the particle filter(PF) are widely used for recursive state estimation. These recursive filter algorithms estimate the state at time k using all data from time 1 to time k. Both extended Kalman filter and particle filter are employed to estimate the state of nonlinear and non-Gaussian systems in the presence of noisy measurements, therefore are very suitable for positioning fusion tasks.

The Extended Kalman Filter is an extension of the Kalman filter for nonlinear systems. It linearizes the nonlinear system equations at each time step using first-order Taylor series expansions, thereby allowing the use of the classic Kalman filter framework. The EKF consists of two main steps: the prediction step and the correction step (update step).

EKF propagates the state estimate and its covariance forward in time using the nonlinear motion model and its Jacobian in the prediction step. In a fusion positioning problem, the motion model usually comes from sensors that bring high-frequency position updates, such as visual/wheel/inertial odometry. The model typically includes equations that describe how the state variables change with respect to time, as well as any known inputs or disturbances. In the correction step, EKF incorporates the measurements into the state estimate, such as the GPS measurement, lidar measurement or WiFi measurement. The Kalman gain is computed based on the measurement model using the predicted covariance and the Jacobian. Then Kalman gain is then used to update the state estimate and its covariance by using the difference between the actual measurements and the predicted measurements obtained from the measurement model. It is a widely used algorithm in fusion with different sensors such as GPS and IMUs in different areas like autonomous vehicles, robotics, and aviation. However, one limitation of EKF is it assumes that the motion and measurement model noise is Gaussian.

Unlike EKF, the particle filter (PF) is a recursive Bayesian filtering technique for estimating the state of nonlinear and non-Gaussian systems. It uses a set of particles to represent the state's posterior probability density function, where each particle

represents a possible state. The PF algorithm consists of three main stages: predicting, importance weighting, and resampling.

In the predicting stage, particles are propagated forward in time using the process model (as the motion model in EKF) The updates depend on previous states and the process model noise. During the importance weighting stage, the same measurements are used for every particle to compute the particle likelihood using the difference between the predicted observations and the actual measurements. The resampling stage resamples the particles to maintain a set of particles with an effective representation of the posterior distribution to avoid particle degeneracy, which means only a few particles have significant weights.

Chapter 3

Uncertainty aware inertial navigation

In this chapter, a comprehensive approach is proposed to achieve accurate displacement and uncertainty estimation through the development of two distinct parts: the inertial navigation component and the uncertainty estimation component. The first component is responsible for providing precise relative position while the latter deals with the estimation of the confidence level associated with the location estimation. These two components make up a deep neural network that can be trained end-to-end using a self-collected dataset using the Vicon tracking system.

3.1 Inertial navigation

Since IONet [Chen et al., 2018] explored using the LSTM to regress the velocity and heading of IMUs, the field of inertial navigation has experienced a significant shift towards deep learning methods because of the superior performance.

This thesis seeks to leverage the power of deep learning in pedestrian location tracking using smartphones. Unlike foot-mounted IMUs, the traditional approach of using zero-velocity updates can not be easily adapted. Deep learning-based methods require vast data to train, consequently, this chapter is dedicated to explaining the extensive effort put into collecting and preprocessing the data for the neural network model.

3.1.1 Data collection

The data required to train a deep neural network for inertial navigation consists of two parts: the raw IMUs measurement and the ground truth location.

The raw IMU data can be recorded using a high-quality smartphone IMU sensor at a specific frequency. The frequency of data recording is a crucial parameter, it determines the accuracy and performance of the resulting prediction. While a higher frequency can indeed bring better accuracy, it also comes with several trade-offs. For example, higher IMU frequencies can increase power consumption a lot, particularly on limited battery life devices such as mobile phones. In this thesis, the raw IMU data was collected at

100Hz using a HUAWEI Mate 40 Pro phone for a better user experience. The HUAWEI Mate 40 Pro use an ICM-20690 IMU from InvenSense.

Accurate ground truth data is essential for creating a high-quality dataset that can be used to train the deep neural network. In this thesis, the ground truth data was recorded using the Vicon tracking system in the Informatics Forum G.17. The Vicon tracking system is a high-precision motion capture system. It is commonly used in research and industry. The system uses multiples high-speed cameras to capture the motion of reflective markers placed on the object or person being tracked. Then the captured data is then processed using specialized software to output an accurate and precise position and orientation of the desired tracking object within the tracking area.

4 markers are used for the data collection, these markers are placed asymmetrically around the phone to make sure the recorded Vicon ground truth is correct. Because the tracking will lose when markers are absent from the tracking cameras when the phone pose changes a lot, such as the phone is placed in the trouser pocket. An assumption is made for this dataset, which is the user also holds the phone.

6 sequences of walking data were recorded for the training, each sequence contains about 8 min of data.

3.1.2 Data prepocessing

3.1.2.1 Time synchronisation

Time synchronisation is an essential process for the ground truth recorded and the raw IMU data recorded to obtain a high-quality training dataset. To achieve this, three quick squats were done at the beginning of every sequence. Figure 3.1 shows the three quick squats resulting in three clear peaks in the z-axis acceleration of the phone IMU. Similarly, 3 valleys in the z-axis position of Vicon recorded ground truth is shown in Figure 3.2.

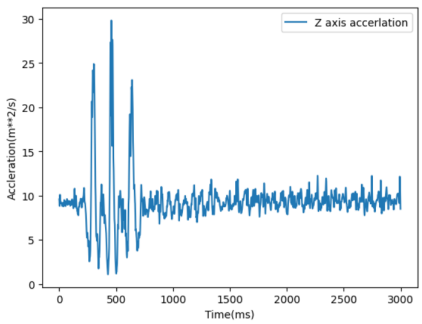

Figure 3.1: Phone z-axis acceleration

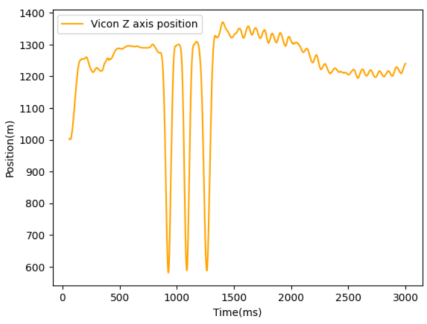

Figure 3.2: Vicon z-axis position

Time synchronisation is done by aligning the peaks and valleys, as shown in Figure 3.3.

3.1.2.2 IMU frame stabilization

[Yan et al., 2017] proposed stabilized IMU frame for the inertial navigation problem. Unlike inertial navigation in an autonomous car, the orientation of IMU is not consistent,

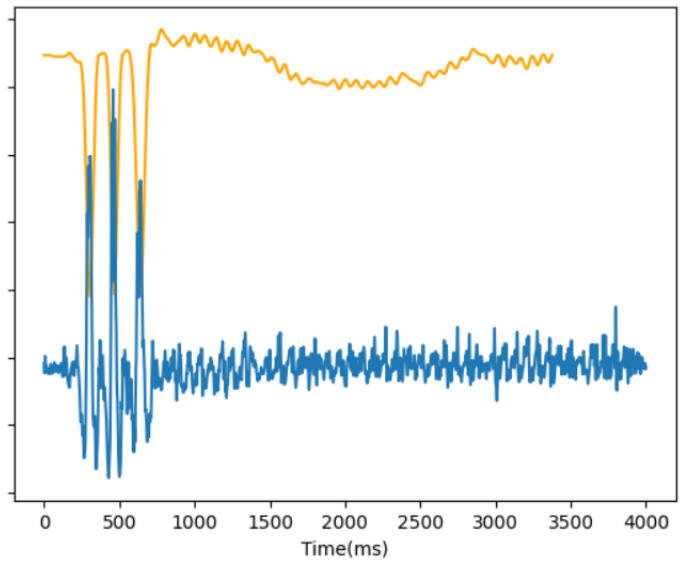

Figure 3.3: Time synchronisation

it varies with time. This technique rotates the IMU measurements from the IMU frame to the World frame using the IMU orientation. This allows the learning algorithm to work with a consistent and coherent set of measurements. An ablation study of this would be shown in the later chapter. There are different ways to obtain the orientation to perform the frame stabilization, and this significantly affects the training.

[Yan et al., 2019] and [Yan et al., 2017] use the orientation from phone device orientation (e.g., via Kalman filter on IMU signals) In an android device, this is called the game rotation vector sensor, in their implementation, the geomagnetic field is not used as the authors claim that the magnetic field would damages rotation estimations due to its instability. The determination of the roll and pitch angle is not hard as the information on the direction of gravity can be obtained from the accelerometer. However, the yaw angle depends on gyroscope readings and this integration process is subject to drift, it is the accumulation of small errors over time.

[Liu et al., 2020] use an external Visual-inertial ground-truth rotation to perform the frame stabilization in their work. In our dataset, high-quality external ground-truth rotation is also available from the Vicon tracking system. This can provide a better orientation than the device orientation obtained from the phone.

Another orientation that can be used for IMU frame stabilization is the phone device orientation that uses the geomagnetic field-generated yaw. This is useful when the Vicon orientation is not available. This approach can outperform the first stabilization method for the training set.

When orientation is obtained, the stabilized measurement can be simply computed by incorporating the unstabilized measurements with the orientation rotation matrix using matrix multiplication. However, some data points might be Nan among the recorded dataset due to the sensor error/tracking loss in the Vicon, orientation interpolation is necessary to prevent runtime error.

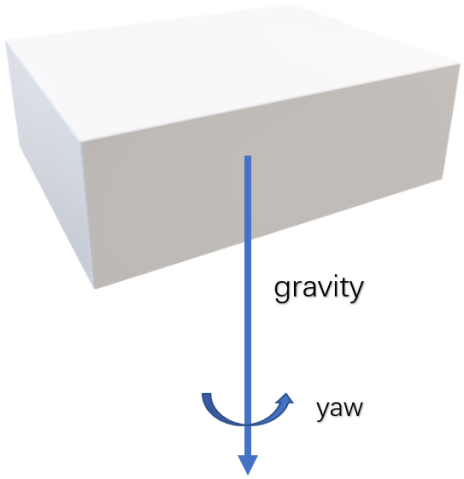

Figure 3.4: The yaw of an object

SLERP (Spherical Linear Interpolation) [Shoemake, 1985] is a method for interpolating between two orientations represented as unit quaternions. Iterpolating is done along the shortest path between the two quaternions on a unit sphere, instead of interpolating linearly between the quaternion components.

$$q(t) = \frac{\sin((1-t)\theta)}{\sin(\theta)} q_0 + \frac{\sin(t\theta)}{\sin(\theta)} q_1$$

where q_0 and q_1 are two unit quaternions representing the two orientations being interpolated, t is the interpolation parameter (a value between 0 and 1), θ is the angle between the two quaternions.

The stabilized IMU measurements are shown in Figure 3.5 and 3.6. The x-axis acceleration would centre in 0 meters per second square, and the z-axis acceleration would centre in around 9.8 meters per second square (gravitational acceleration).

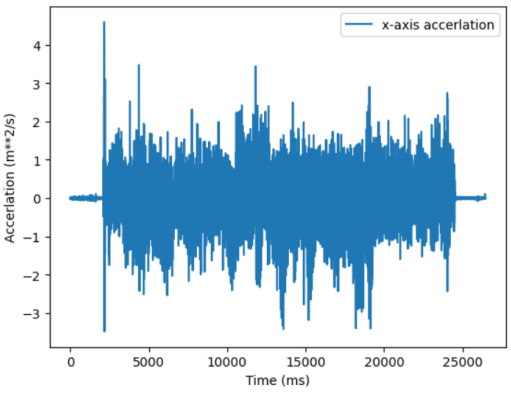

Figure 3.5: stablized x-axis acceleration

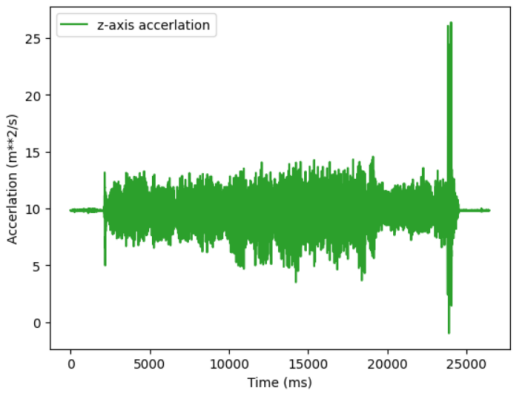

Figure 3.6: stablized z-axis position

3.1.2.3 Data augmentation

Data augmentation is a technique commonly used in machine learning to artificially increase the size of a dataset by creating new training data from existing examples. A larger dataset can improve the model performance by obtaining a better generalization ability to unseen data.

The network aims to regress the 2D displacement given the stabilized IMU measurement. It is important to note that the heading of the displacement should be influenced by the yaw information obtained from the orientation used for stabilization. Therefore the dataset can be augmented by giving different horizontal rotations to sequences following [Yan et al., 2019]. Therefore the overall data preprocessing pipeline is:

Algorithm 1 Data prepocessing

$quaternion \leftarrow$ orientation obtained from sensors
$accerlation \leftarrow$ measurements obtained from accelerator
$gyroscope \leftarrow$ measurements obtained from gyroscope
$prepocessed \leftarrow$
$interpolated\_quaternion = SLERP(quaternion)$
for $rotation = [r_1, r_2, ..., r_n]$
  $rotated\_quaternion = interpolated\_quaternion.yaw + rotation$
  $stablized\_accerlation = rotated\_quaternion@accerlation$
  $stablized\_gyroscope = rotated\_quaternion@gyroscope$
  $prepocessed$ APPEND $(stablized\_accerlation, stablized\_gyroscope)$
RETURN $prepocessed$

3.1.3 Deep learning network and architecture design

The network is designed to regress the 2D displacement from a window of IMU measurements (3-axis stabilized acceleration from the accelerator and 3-axis angular velocity from the gyroscope). This network inherits the approach proposed by IONet [Chen et al., 2018] to extract features from IMU data using LSTM. However, unlike IONet, which regresses relative rotation and translation from raw IMU measurements, in this task, the orientation is known. Only 2D displacement in the world frame needs to be regressed just as Ronin [Yan et al., 2019]. Apart from these, a self-attention module is inserted into the network to increase the capacity and representational power.

Figure 3.7 shows the visualization of network architecture. In this application, the IMU windows have a length of 100, and because the IMU records data at 100Hz, the network estimates one prediction per second. The LSTM is a great tool to handle the time sequence data [Hochreiter and Schmidhuber, 1997], the two LSTM layers used in the network have [128,256] units correspondingly. FCs in the Figure represent the fully connective layers. 3 FC layers [128,32,2] are stacked to regress the desired 2D displacement.

The self-attention mechanism was initially designed for NLP tasks with the purpose of addressing the issue of long-term dependencies. [Vaswani et al., 2017] [Cheng et al., 2016]

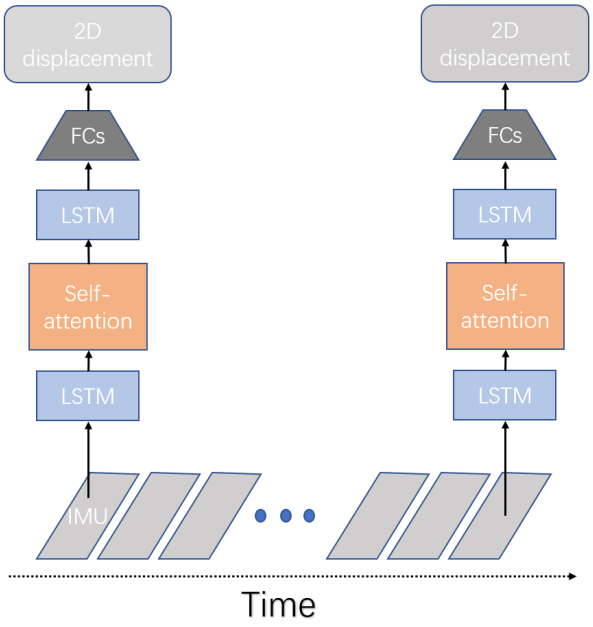

Figure 3.7: Network architecture

milliEgo [Lu et al., 2020] propose to use self-attention to capture the long-range dependencies and global correlations of the IMU features, and prove that it is good for the odometry regression task. Therefore the self-attention module used here aims to realize ego-motion self-regulation by using the channel-wise IMU feature attention. Firstly an attention map is generated using the similarity between embedding channels.

$$a_M = \sigma[W_M Z_M]$$

where W_M is a learnable matrix that projects the IMU feature Z_M to the same embedding space. σ represents a non-linear activation such as sigmoid and softmax.

After this attention mask is generated, the attended feature $\hat{Z_M}$ is calculated by:

$$\hat{Z_M} = a_M \odot Z_M$$

By doing so, the information from different channels in the features is reweighted and fused together. High-dimensional features might be noisy, and they may contain irrelevant or redundant information. Self-attention can help the model learn informative feature representations by allowing it to attend to the most relevant features at each time step, for example, attend more to the linear acceleration during straight walking. As we aim to regress the displacement from acceleration, and the displacement is the double integral of acceleration over time, the network can also benefit from the fact that the self-attention mechanism takes into account the long-term dependencies in IMU data, as time series data may contain complex temporal dependencies that are challenging to capture.

3.2 Uncertainty estimation and loss function

3.2.1 Theory of Deep Evidential Regression

Compared to MC dropout[Gal and Ghahramani, 2016], which only considers the epistemic uncertainties and is slow during inference, the method proposed by deep evidential regression [Amini et al., 2020] is more lightweight and can simultaneously consider both the aleatoric uncertainties and the epistemic uncertainties without a large modification on the original network.

The goal of uncertainty estimation is to estimate the posterior distribution $q(\mu,\sigma^2) = p(\mu,\sigma^2|y_1,...,y_N)$ where (μ,σ^2) are the unknown mean and variance; $y_1,...,y_N$ are the observed targets. Assuming that the estimated distribution can be factorized such that $q(\mu,\sigma^2) = q(\mu)q(\sigma^2)$, the distribution is a Normal Inverse-Gamma (NIG) distribution:

$$p(\mu,\sigma^2|\gamma,\nu,\alpha,\beta) = \frac{\beta^\alpha\sqrt{\nu}}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}}\left(\frac{1}{\sigma^2}\right)^{\alpha+1}\exp\left\{-\frac{2\beta+\nu(\gamma-\mu)^2}{2\sigma^2}\right\}$$

The Normal Inverse-Gamma (NIG) distribution is a four-parameter family of probability distributions that are commonly used in Bayesian statistics and finance. It is a conjugate prior to the normal distribution with an unknown mean and variance. (such as the x-axis displacement and y-axis displacement in the inertial navigation network). Virtual observations are a technique used in Bayesian statistics to incorporate prior information into the likelihood function. The idea of a virtual observation is to add a 'virtual' set of data points to the actual data to represent prior knowledge about the parameters. Virtual observations are usually generated from a prior distribution associated with the parameter being estimated. For example, if the parameters are the mean and variance of a normal distribution, then the prior distribution might be a normal inverse gamma (NIG) distribution. The effect of adding virtual observations is to move the posterior distribution towards the prior distribution. In other words, the prior has a greater effect on the posterior distribution as more virtual observations are added. [Jordan, 2009]

In the formula above, the mean μ can be estimated from the virtual observation ν with sample mean γ and the variance σ^2 can be estimated from the virtual observation α with sample mean γ and sum of squared deviations 2ν. Therefore, instead of predicting a single mean μ value using the neural network, the uncertainty estimation network should also estimate γ,ν,α,β for each value that is desired to be regressed.

3.2.2 Loss function of Deep Evidential Regression

The loss function of this task inherits the loss function from the loss function proposed in [Amini et al., 2020]. There are 2 optimization goals for a neural network with uncertainty estimation. The first one is maximizing the model evidence.

$$L_{NLL}(w) = \frac{1}{2}\log\frac{\pi}{\nu} - \alpha\log(\Omega) + (\alpha+\frac{1}{2})\log\left((y_i-\gamma)^2\nu+\Omega\right) + \log\left(\frac{\Gamma(\alpha)}{\Gamma(\alpha+\frac{1}{2})}\right)$$

where $\Omega = 2\beta(1+\nu)$, this loss fitting the data to the evidential model.

The second goal is to enforce a prior to inflating uncertainty while removing incorrect evidence.

$$L_R(w) = |y - \mathrm{E}[\mu_i]| \cdot \Phi = |y - \gamma| \cdot (2\nu + \alpha)$$

where y is the ground truth label. A hyperparameter coefficient, λ, is added to balance these two goals.

In summary, for each target (x-axis displacement and y-axis displacement), 4 parameters correspond to the NIG hyperparameters, $(\gamma, \nu, \alpha, \beta)$. These four values determine the location and also the uncertainty of the prediction. Then the loss function is modified as mentioned above.

$$L_{evidence}(w) += L_{NLL}(w) + \lambda L_R(w)$$

As what [Liu et al., 2020] mentioned, training an inertial navigation network with uncertainty-aware loss can be hard to converge. Therefore MSE (mean squared error) is used for the first several epochs to stabilize the network.

$$L_{mse}(w) = \frac{1}{2}\sum_{i=1}^{2}(y_i - \gamma_i)^2$$

where the y_1 represents the x-axis displacement and y_2 represents the y-axis displacement and γ_1 and γ_2 are the corresponding NIG mean prediction.

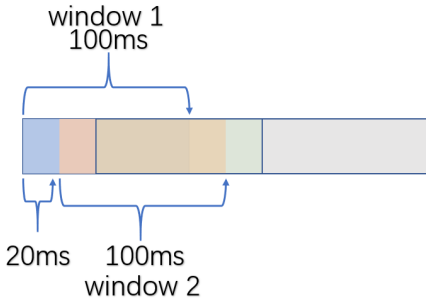

Figure 3.8: Data rolling

3.3 Experiment and Evaluation

3.3.1 Experiment implementation

The uncertainty-aware inertial navigation model is implemented using TensorFlow [Abadi et al., 2015]. 2D-IONet is also implemented for a baseline comparison. The 2D-IONet estimate the translation and relative rotation using the raw 6-axis IMU data. Therefore data preprocessing is skipped for the IONet training. For both models, the rolling window technique is used for the training set with a step size of 20 to further augment the training set as shown in Figure 3.8.

Both models are trained on an evidential version as described above. The 2D-IONet converges in around 10 epochs, while the proposed network converges in around 30 epochs.

3.3.2 Inertial navigation performance evaluation

3.3.2.1 Evaluation metrics

Two metrics are used for the evaluation:

Average Trajectory Error (ATE): defined as the Root Mean Squared Error (RMSE) between estimated location and ground truth location. ATE measure the accuracy of the estimated trajectory and the ground truth trajectory. ATE is a standard metric that was first proposed in [Sturm et al., 2012].

Average Relative Displacement Error (ARDE): defined as the Root Mean Squared Error (RMSE) between estimated displacement and ground truth displacement. Compared with ATE, ARDE measures the motion in the local frame and ignores the location error accumulation.

3.3.2.2 Vicon validation sequences

The models are first evaluated on the Vicon recorded validation sequences. Figure 3.9 and 3.10 show the trajectories estimated from the models, the proposed method has a better performance on both metrics. Notice that the 2D-IONet has a poor ATE performance as it suffers from heading error accumulation. The proposed method

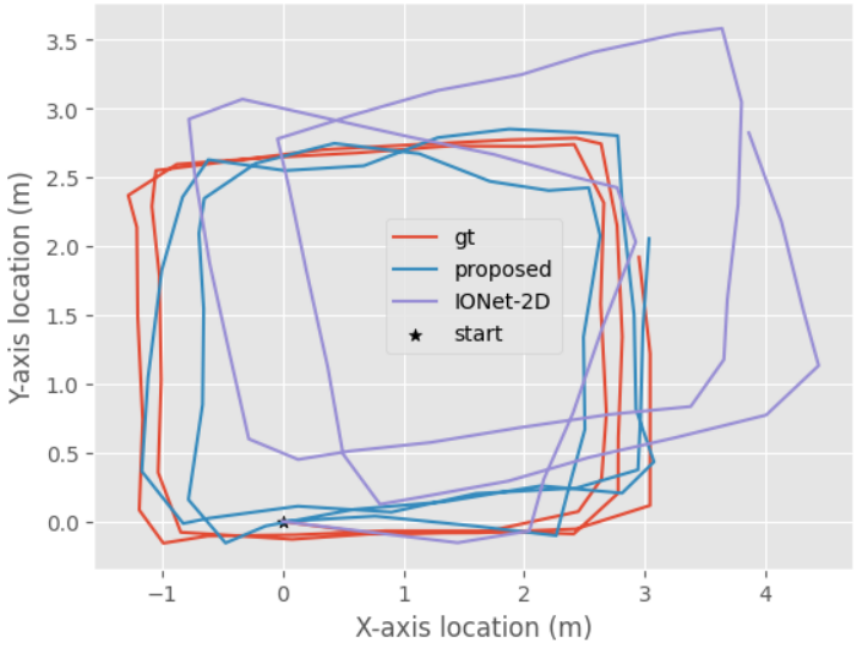

Figure 3.9: Vicon validation sequence 1. The proposed ATE = 0.3076, ARDE = 0.1186. The IONet-2D ATE = 1.135, ARDE = 0.1493.

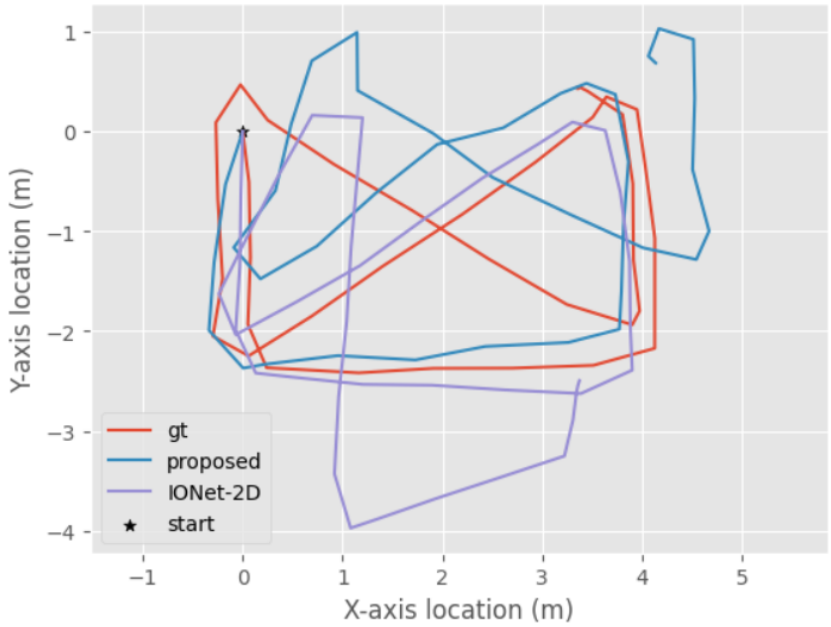

Figure 3.10: Vicon validation sequence 2. The proposed ATE = 0.6566, ARDE = 0.1460. The IONet-2D ATE = 1.135, ARDE = 0.2871.

estimates the 2D-displacements given orientation, therefore can have a much better ATE performance.

3.3.2.3 Informatics Forum test sequences

The test sequences are recorded in the Informatics Forums building, the IMU-Lidar odometry is used as the pseudo-ground truth. The initial heading is ambiguous for both IONet-2D and the proposed method, thus heading is aligned using the first 10 seconds of the test sequence.

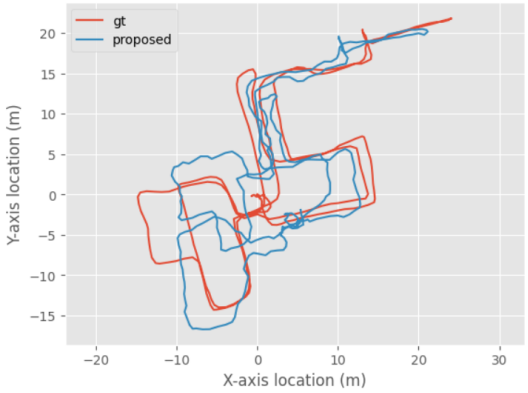

Figure 3.11: Proposed; ATE = 4.017, ARDE = 0.2986

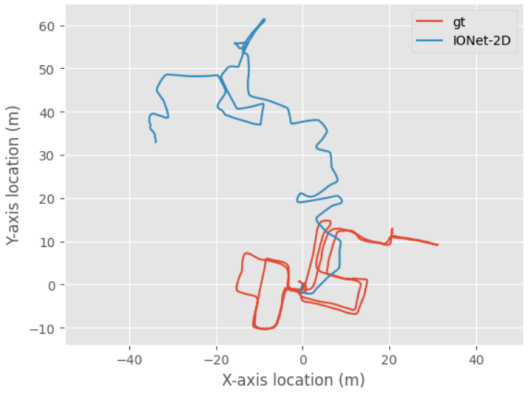

Figure 3.12: IONet-2D; ATE = 39.48, ARDE = 0.8437

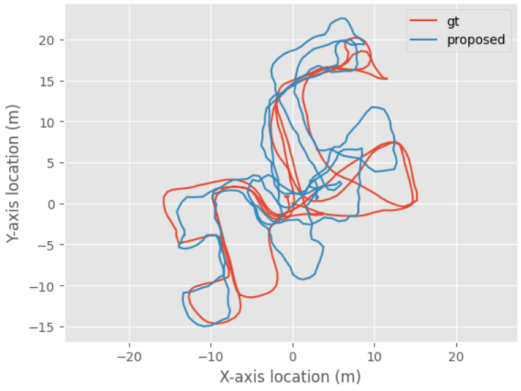

Figure 3.13: Proposed; ATE = 3.033, ARDE = 0.2579

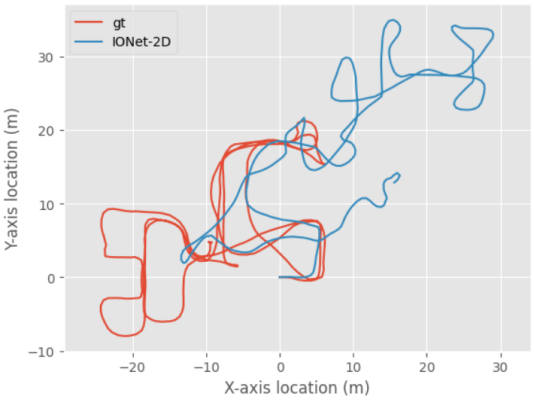

Figure 3.14: IONet-2D; ATE =20.93, ARDE = 0.9344

Figures 3.11 to 3.16 show the displacement estimation from the proposed model in the test sequences recorded in the Forums F0. In these long ($>$ 6 mins) test data, the proposed method shows a great improvement from IONet-2D, especially in ATE. IONet-2D fail to return a reasonable motion estimation most of the time in test data.

3.3.3 Error correlated uncertainty estimation

Uncertainty refers to our level of confidence in the prediction while error refers to the discrepancy between the predicted value and the ground truth.

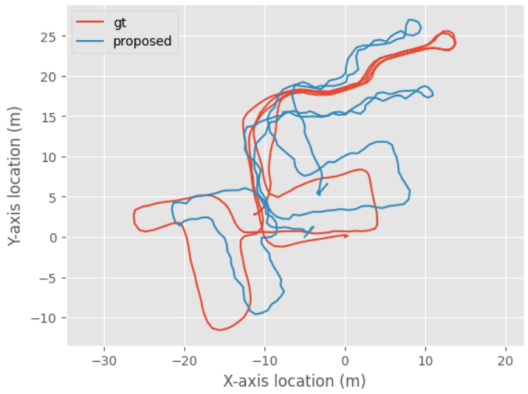

Figure 3.15: Proposed; ATE = 13.57, ARDE = 0.6671

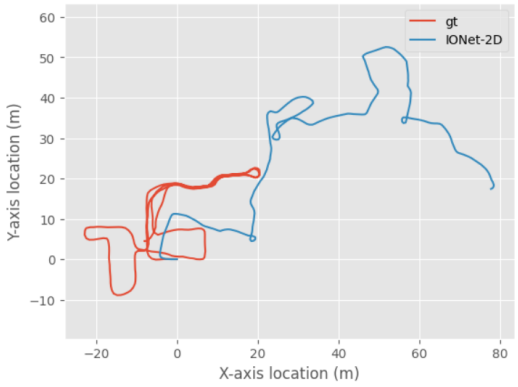

Figure 3.16: IONet-2D; ATE =47.34, ARDE = 0.8694

The uncertainty estimation is used to inform the accuracy of the prediction of the filter algorithm model in the fusion. The Kalman gain in EKF is designed to minimize the impact of the motion uncertainty on the estimation process, therefore a bad uncertainty would result in a less optimal Kalman gain. In PF, the motion noise affects the propagation of these particles, and if not properly accounted for, it can lead to particle degeneracy or divergence, which degrades the accuracy of the estimation.

In other words, we hope that a data point with high error has greater estimated uncertainty. Therefore, we expect the predicted uncertainty to positively correlate with the error.

Figure 3.17 shows the scatter plot of estimated uncertainties and corresponding ARDE error in all 3 Forums F0 test sequences (about 1200 data points). A positive correlation between uncertainty and the ARDE error can be observed, and thus the estimated uncertainty is reasonable and can provide information in the fusion.

3.3.4 Stabilization methods analyzing

Experiments using different stabilization methods when the Vicon orientation is not available. The dataset used for this experiment is recorded in the Bayes Center with an external Lidar-IMU pseudo ground truth [Shan et al., 2020] [Shan and Englot, 2018]. Approach 1: the geomagnetic field-free orientation and approach 3: the geomagnetic field-aided orientation are compared in this experiment. For a relatively fair comparison, the orientation of approach 1 is generated using the Madgwick filter [Madgwick et al., 2010] from the raw IMU reading as it generates a better yaw estimation than the EKF during test time.

Figures 3.18 to 3.21 show the estimated trajectories during the training process. Where after the first 10 epochs, the data stabilized using approach 3 starts to show a converge signal while approach 1 fails to learn even the straight line. Approach 3 converges to a reasonable and acceptable point where the test ATE = 4.449m in epoch 30 and approach 1 still doesn't converge at all.

I argue that this is because, in this regression problem, the goal is to estimate a global

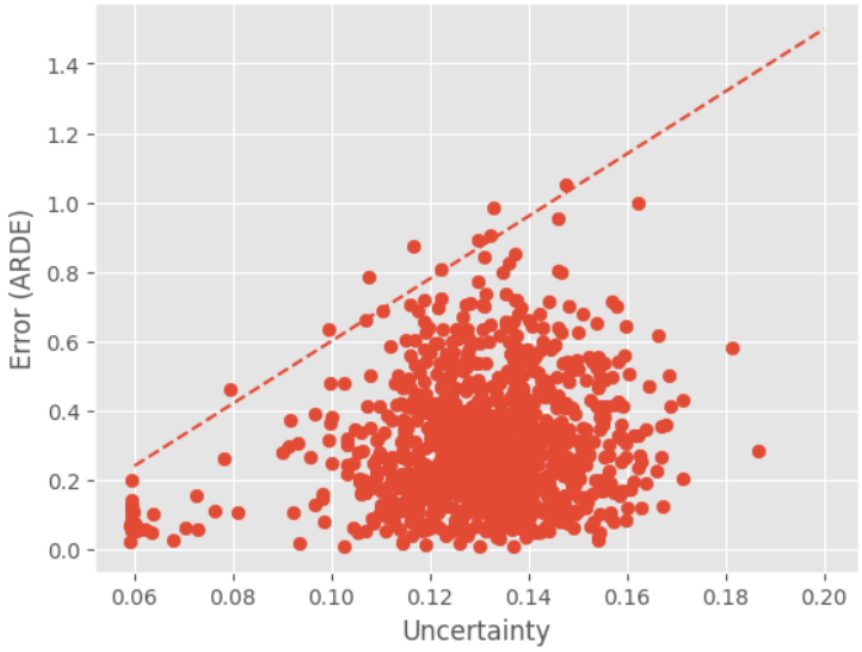

Figure 3.17: Error Uncertainty Correlation

frame displacement instead of the local frame translation and rotation, therefore even though the magnetic field might damage the relative rotation estimations due to its instability, it can provide a better global consistent yaw angle throughout the training sequence, which is more important in the training process. The gyroscope measurements are instantaneous and local therefore they are prone to drift. The accelerator can only aid with the estimation of roll and pitch using the gravity direction. Using an external information source to provide the reference of yaw can be helpful to the whole system. Therefore in the situation when the Vicon data is not available, using the yaw angle generated aided by the magnetometer can benefit the training of global frame displacement estimation.

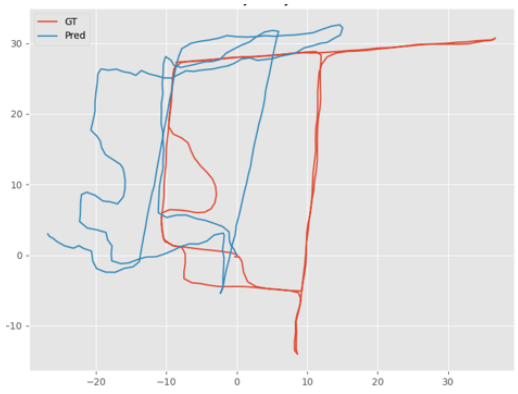

Figure 3.18: Epoch 10 Apporche 3: geo-magnetic field-aided

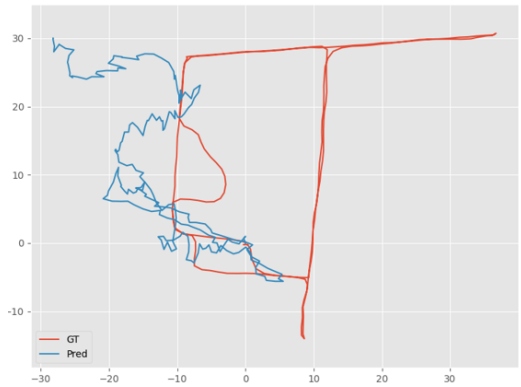

Figure 3.19: Epoch 10 Approach 1: geo-magnetic field-free

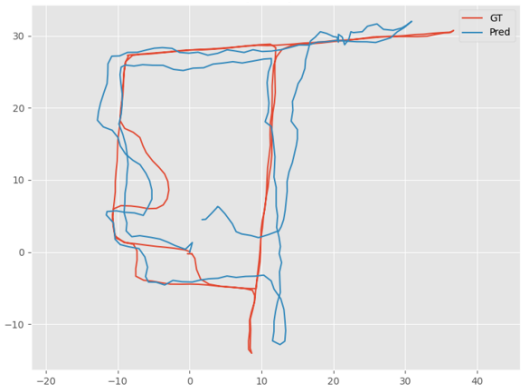

Figure 3.20: Epoch 30 Apporche 3: geo-magnetic field-aided

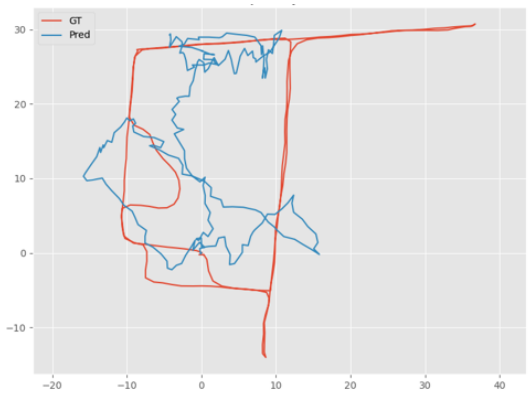

Figure 3.21: Epoch 30 Approach 1: geo-magnetic field-free

3.4 Limitations and future works

As mentioned in the dataset section, the current dataset is a small dataset that makes several strong assumptions. Especially assuming that the pedestrian is holding the phone while using the system.

Another problem observed from the trained model lacks of generalization ability, i.e performance is somewhat degraded when testing with other people's recorded test data.

Thirdly the model tends to underestimate the user velocity in the Forums and the Bayes, I argue that this is because of the large difference in average speed between the training set and the test set. The Vicon system works within a small area (about 5m * 3m), this results in more deceleration due to turning and stopping in the training data, leading to a smaller average speed.

In conclusion, most of the limitations of the current model are due to insufficient training data. Therefore future works include finding suitable methods for recording Ground Truth data for multi-pose phone IMU. Secondly, to enrich the diversity of the training data and enhance the model's generalization ability, it is better to have different users record the training data instead of just one person. To address the speed ratio problem, different normalization approaches can be explored and considered.

Chapter 4

Indoor fusion positioning

4.1 Introduction to WiFi localization

WiFi fingerprinting, also known as WiFi-based indoor localization, is the process of determining the location of a person or object within a building using WiFi signals. WiFi signals are generated by routers or access points (AP) to create a wireless network that devices can connect to. [Crow et al., 1997] When a smartphone or a laptop is connected, it receives WiFi signals from the access point or router and uses them to transmit and receive data.

WIFI indoor localization using fingerprint can be broadly divided into two phases: the offline sampling stage and the online positioning stage. In the offline sampling stage, a WiFi radio map is created using the AP's MAC address, received signal strength indication (RSSI) value and position information by collecting WIFI data in the desired region as shown in 4.1. In the positioning stage, user location is estimated by incorporating current WIFI data (RSSI values and corresponding MAC addresses) and the WiFi radio map. [Crow et al., 1997][BASRI and El Khadimi, 2016][Ge and Qu, 2016]

The WiFi localization system used in the fusion section of this thesis is the work of other members of the MAPS lab, so the implementation and algorithm details will not be described here. But unlike other approaches, this WiFi localization system relies on crowdsourced data and is infrastructure-free. It means that this is a low-cost, scalable solution that can be easily deployed to different indoor environments. This localization system estimates the current position and corresponding uncertainty of the phone holder and takes WiFi raw signal data as input. The average localization error varies from 4 to 7 meters depending on the radio map quality. Due to hardware limitations and user experience, the experimental smartphone collected raw WiFi data approximately once every 3 to 10 seconds.

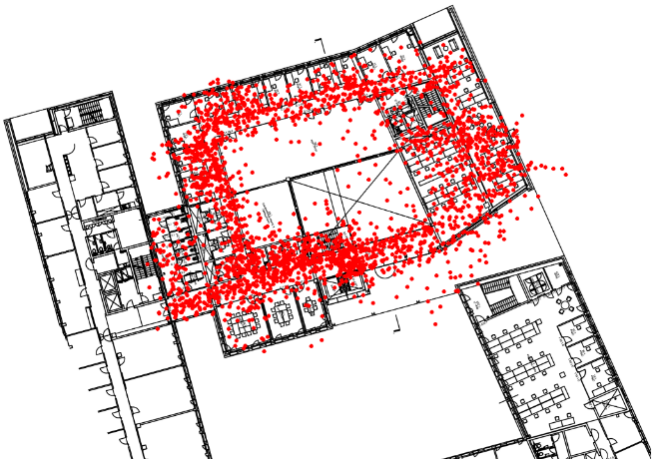

Figure 4.1: Bayes F1 radio map

4.2 Extended Kalman Filter

The Extended Kalman Filter (EKF) is a mathematical algorithm that is used to estimate the state of a system in the presence of uncertain or noisy sensor measurements[Kalman, 1960]. It can be used for inertial WiFi positional fusion as follows:

4.2.1 The motion model

The motion model for the EKF is used to predict the next state of the system based on the current state and control inputs. In the proposed EKF system, the motion model receives dx (x-axis displacement) dy (x-axis displacement) as inputs, and calculates the states:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \begin{bmatrix} v\cos\theta_{k-1} & -v\sin\theta_{k-1} \\ v\sin\theta_{k-1} & v\cos\theta_{k-1} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} dx \\ dy \end{bmatrix} + \mathbf{w}_k \right)$$

where $x_k = [x, y, \theta, v]^T$

x and y are the world (map) frame 2D position. θ represents the inertial frame to the world (map) frame rotation. v is the ratio of the estimated velocity from IMU to the EKF velocity. w_k is the motion model noise.

4.2.2 The measurement model

The measurement model for EKF is a mathematical function that describes the relationship between the state of a system and the sensor measurements. In the proposed system, the measurement model relates the current state 2D coordinates to the WiFi localization $y_k = [wifi_x, wifi_y]^T$ in the world frame:

$$y_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathrm{x}_k + \mathbf{n}_k$$

4.2.3 The prediction step

The prediction step of the Extended Kalman Filter is the first step of the algorithm. It is used to predict the state of a system at the next time step based on the current state and control (inertial) input. The prediction step involves two main stages: the state prediction and the covariance prediction.

$$\check{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, 0)$$

where $\tilde{\mathbf{x}}_k$ is the predicted state at time k from the corrected state at time k-1 using the control u_k. f represents the defined motion model. The 0 represents noise is ignored when calculating the $\check{\mathbf{x}}_k$.

$$\check{\mathbf{P}}_k = \mathbf{F}_{k-1}\check{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{L}_{k-1}\mathbf{Q}_{k-1}\mathbf{L}_{k-1}^T$$

where $\check{\mathbf{P}}_k$ is the state covariance at time k, Q_{K-1} is the noise covariance at time k.

$$\mathbf{F}_{k-1} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}_{k-1}}, \mathbf{L}_{k-1} = \frac{\partial \mathbf{f}}{\partial \mathbf{w}_k}$$

F_{k-1} is the motion model jacobian with respect to the last state while L_{k-1} is the motion model jacobian with respect to the noise of the last state. In the defined model:

$$F = \begin{bmatrix} 1 & 0 & -v*dx*sin(\theta) - v*dy*cos(\theta) & dx*cos(\theta) - dy*sin(\theta) \\ 0 & 1 & v*dx*cos(\theta) - v*dy*sin(\theta) & dx*sin(\theta) + dy*cos(\theta) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$L = \begin{bmatrix} v*cos(\theta) & -v*sin(\theta) \\ v*sin(\theta) & v*cos(\theta) \end{bmatrix}$$

$$Q = \begin{bmatrix} \textbf{estimated x axis uncertainty} & 0 \\ 0 & \textbf{estimated y axis uncertainty} \end{bmatrix}$$

4.2.4 The correction step

The correction step is the second step of the algorithm. It is used to update the state estimate based on the sensor measurements. Therefore the correction step is skipped execute when WiFi measurements are not available. The measurement correction is done using the current state estimate and the measurement model.

Firstly, the measurement model Jacobians at time k are computed:

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k)$$

$$\mathbf{H}_k = \frac{\partial \mathbf{h}}{\partial \mathbf{x}_k}, \mathbf{M}_k = \frac{\partial \mathbf{h}}{\partial \mathbf{n}_k}$$

where h is the measurement model.

Then incorporate covariances to compute the Kalman Gain:

$$\mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{H}_k^T \left(\mathbf{H}_k \check{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{M}_k \mathbf{R}_k \mathbf{M}_k^T \right)^{-1}$$

where R_k is the measurement noise covariance. The Kalman Gain K_k is used to correct the predicted state:

$$\check{\mathbf{y}}_k = \mathbf{h}\left(\check{\mathbf{x}}_k, \mathbf{0}\right)$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k \left(\mathbf{y}_k - \check{\mathbf{y}}_k \right)$$

Covariance correction is done through:

$$\hat{\mathbf{P}}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \check{\mathbf{P}}_k$$

The measurement model is linear in the proposed system, therefore the

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} M_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The measurement noise covariance R_k is constructed using the estimated uncertainty from the WiFi localization model.

4.2.5 Fusion results

Figure 4.3 shows the trajectory error of the EKF fusion result with respect to the timestamp in a test sequence recorded in the Informatics Forum. The filter needs a number of measurements at the beginning of the sequence to initialize. During the initialization phase, the error is high as the filter the v and θ in states are not converged.

Figure 4.2 shows the EKF fusion result in the same test sequence after the initialization phase (120s). The ATE (average trajectory error) of the original IMU result from the inertial navigation model is 6.875m. The average position error of the WiFi measurement of this sequence is 3.929m. And the EKF result trajectory has an ATE of 3.272. The filter algorithm combines the motion estimates from IMU represented in the IMU frame with the map frame localization from WiFi, resulting in a more accurate trajectory compared to using only IMU with manually set frame transformations. Additionally, this fusion method provides localization results once per second, and these results have smaller errors compared to using WiFi measurements alone which only appears once per 9 seconds in this sequence.

Figure 4.4 shows the result of another test sequence, the ATE of the EKF result is 3.167m while the IMU alone is 13.10m and the WiFi measurements have an average location error of 4.021m.

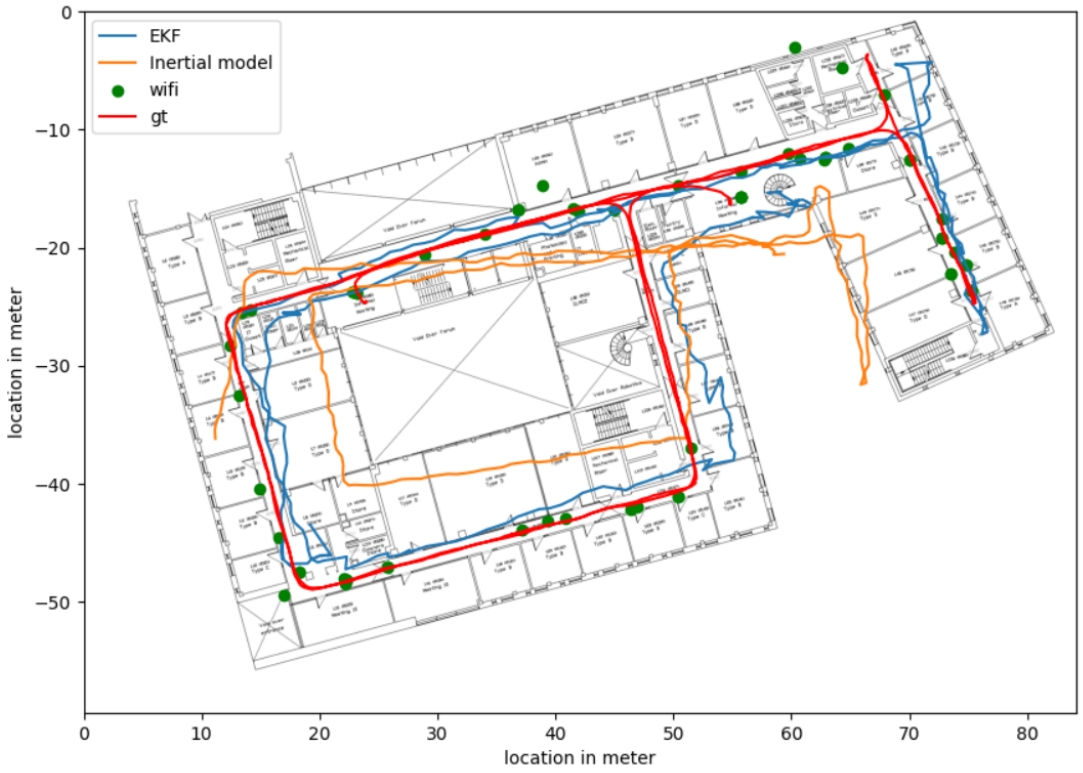

Figure 4.2: EKF fusion result 1 IF F1

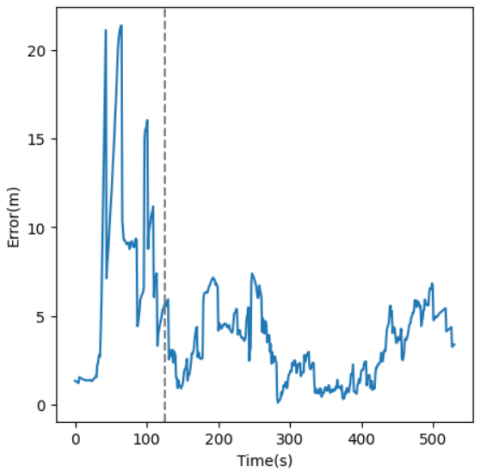

Figure 4.3: EKF fusion error 1 IF F1

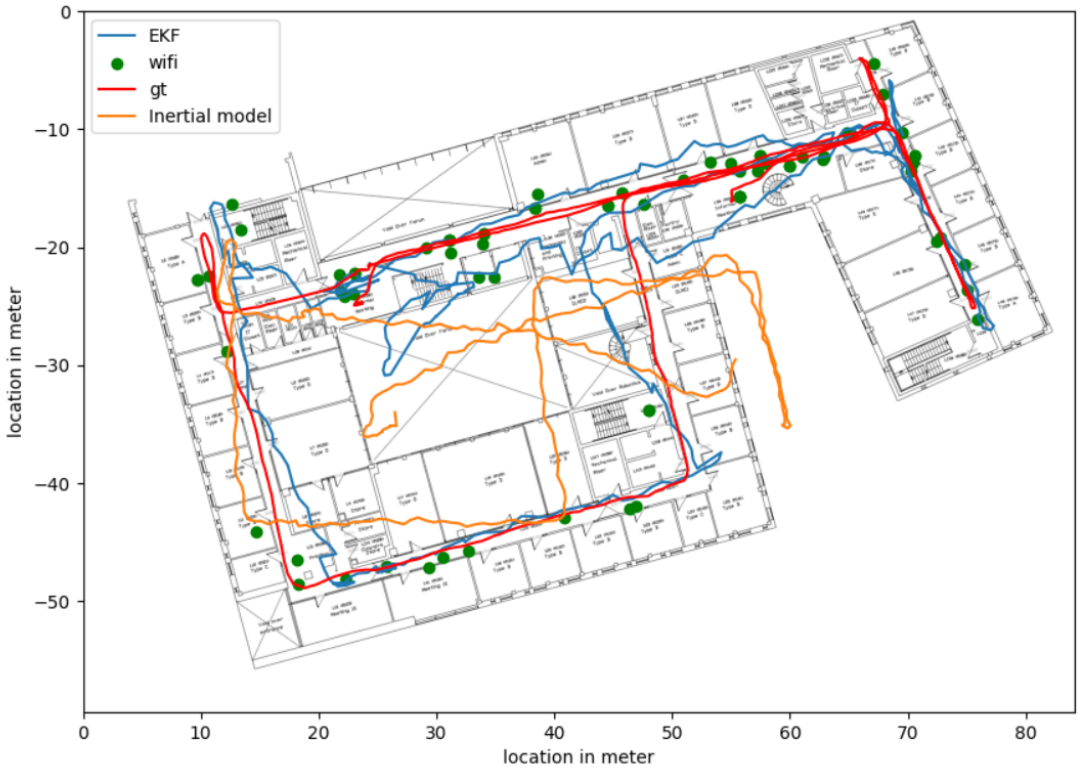

Figure 4.4: EKF fusion result 2 IF F1

4.2.5.1 Uncertainty ablation study

The above experiments use the estimated uncertainties from the proposed IMU navigation system and the WiFi localization system. An ablation study is necessary to evaluate whether the estimated uncertainties are useful for filter fusion4.1. Figure 4.5 shows an experiment using constant covariance (mean of estimated uncertainty) and the estimated uncertainty in another test sequence. Figure 4.6 shows the cumulative distribution function (CDF) of the ATE of the EKF fusion results. This result shows that using the estimated uncertainty from the uncertainty-aware inertial navigation model can benefit the EKF fusion for this test sequence. It holds for most of the cases, for example in the test sequence 1 (Figure 4.2) the ATE of using constant motion model covariance is 3.348m, which is worse than the 3.272m from the estimated uncertainty, the CDF is in 4.7. However, on some occasions, the results can be pretty close. For example, in the test sequence 2 (Figure 4.4). Using the mean uncertainty gives an ATE of 3.170m, which have only a 0.1% improvement from 3.167m.

Therefore, the conclusion is that the estimated uncertainty from the uncertainty-aware inertial navigation model can usually benefit the EKF fusion, but this improvement is very small, usually around 1%, occasionally even smaller, and sometimes does not improve at all.

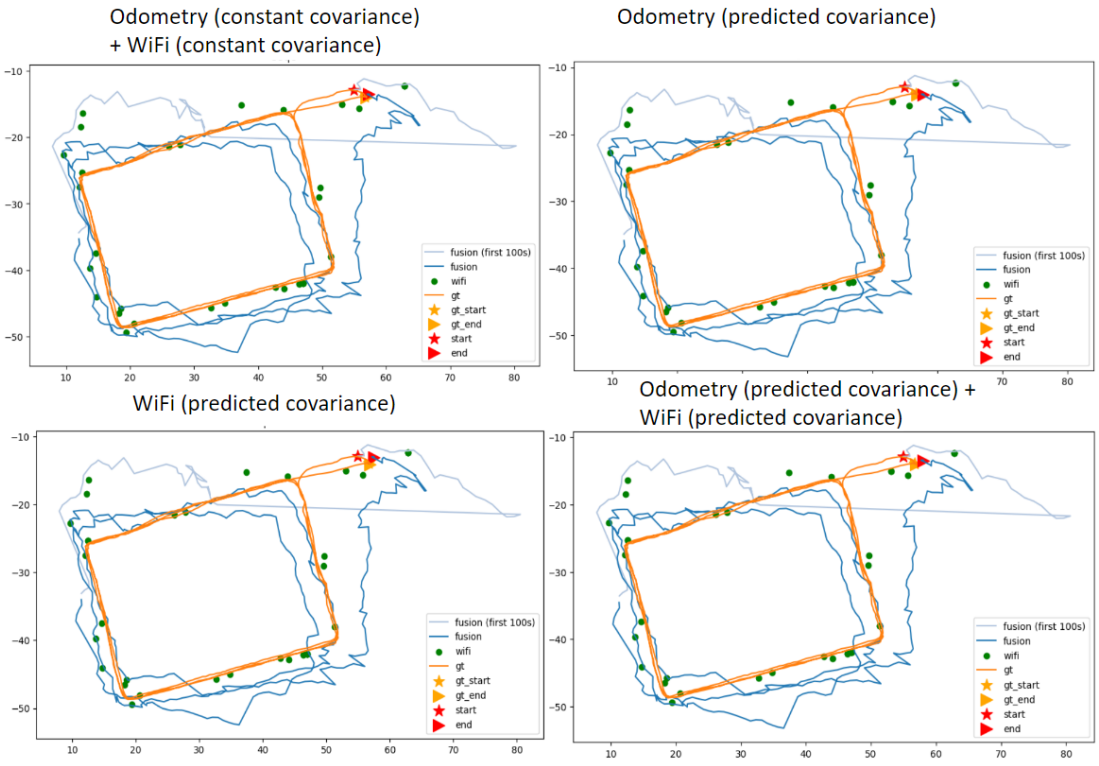

Figure 4.5: Uncertainty ablation study

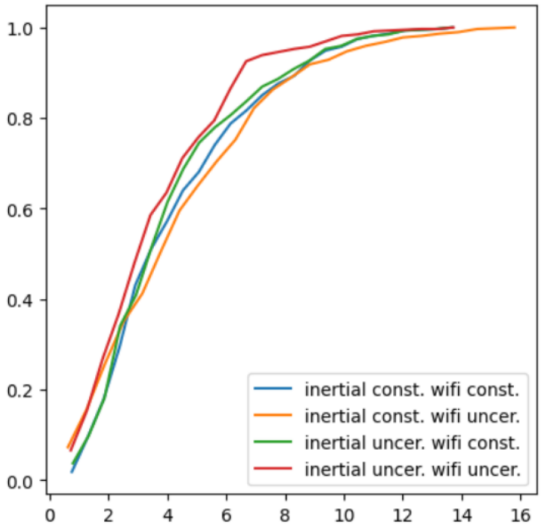

Figure 4.6: CDF of uncertainty ablation study

Sequence	Estimated Uncertainty	Constant Uncertainty	Improvement
1	3.272	3.348	1.02%
2	3.167	3.170	0.10%
3	3.189	3.562	1.12%

Table 4.1: ATE of ablation study

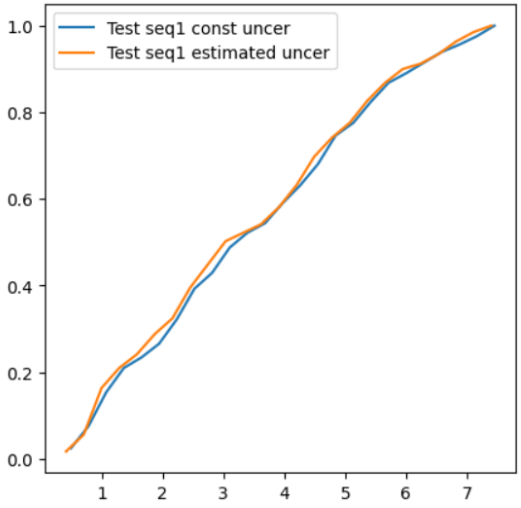

Figure 4.7: CDF of test seq 1

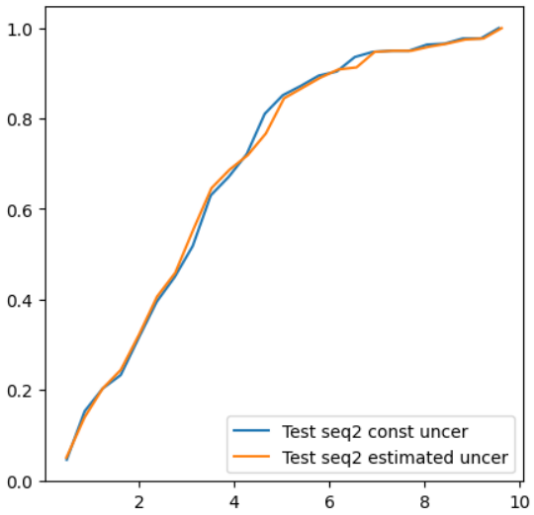

Figure 4.8: CDF of test seq 2

4.2.5.2 Limitation

The EKF have multiple limitations that hurt the fusion. Firstly, the EKF assume the process noise and the measurement noise are Gaussian and have zero means, and the linearization introduces errors if the model is highly nonlinear. Secondly, it lacks robustness and is sensitive to noise measurements. If the initial estimates are poor or the WiFi measurements are too noisy, the EKF may converge to a suboptimal solution or diverge altogether for example in the test data in the Bayes where our radio map has a poor quality. Figure 4.9 shows the diverged EKF result.

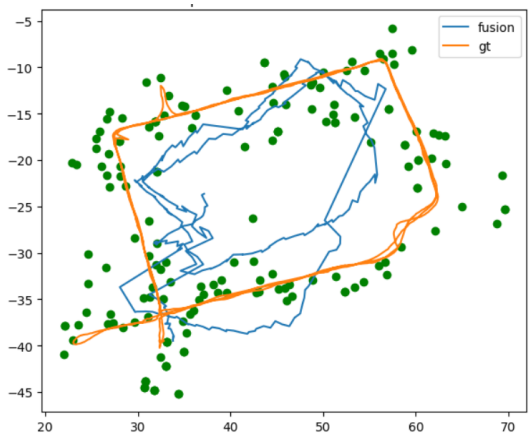

Figure 4.9: EKF diverged

4.3 The Particle Filter for positional fusion

The Particle Filter is a powerful algorithm for state estimation in nonlinear systems and can be a good alternative to the EKF when nonlinear models and non-Gaussian noise distributions are present. The first step of the PF is initialization.

4.3.1 Initialization

For each particle i, the initial state $x(0)_i$ is drawn from the initial state distribution $p(x(0))$. The state x contains 4 parameters the 2D position, the heading and the velocity ratio. From the initial state distribution of this fusion problem, the 2D position is initialized using the first WiFi measurement in the sequence, with a noise depending on the estimated measurement uncertainty. The heading is randomly initialized from 0 to 2π as no heading information is available from both sources. The velocity ratio is set to 1, it can be set to a different number if any prior information about the estimated velocity and the real velocity is known. The weight of the particle w is initialized to 1.

4.3.2 The prediction step

During the prediction stage, for each particle, the state is propagated forward using the motion model.

$$x(k) = f(x(k-1), u(k), w(k))$$

where f is the model, u(k) is the control input at time k, and w(k) is the motion uncertainty estimated from the model.

$$x(k)_x = x(k-1)_v * u(k)_{distance} * cos(x(k-1)_{heading}) + noise(w(k))$$

$$x(k)_y = x(k-1)_v * u(k)_{distance} * sin(x(k-1)_{heading}) + noise(w(k))$$

$$x(k)_{heading} = x(k-1)_{heading} + u(k)_{heading}) + noise(w(k))$$

where $x(k-1)_v$, $x(k-1)_{heading}$, $x(k-1)_x$, $x(k-1)_y$ are the previous state velocity ratio, heading and the 2D position correspondingly. The *noise* is a function that generates random Gaussian noise using the estimated uncertainty.

4.3.3 The important weighting step

The part weight w is updated during the correction phase. Given WiFi measurements and corresponding uncertainty at time k $wifi_k$ and $ucertainty_k$:

$$w_k = w_{k-1} * likelihood(x_k, wifi_k, uncertainty_k)$$

where the likelihood is a function that calculates the likelihood of the particle state matching with the measurements.

$$likelihood(x_k, wifi, uncertainty) = e^{a*L2Norm(wifi,x_k)^2*uncertainty}$$

where a is a negative number, this is a hyperparameter to control the weight update rate. Its principle is to assign higher probability weights to particles that are closer to the WiFi measurements and to decrease the probability if the WiFi measurement value has greater uncertainty. This function can extend to the case when multiple WiFi measurements are available at time k, i.e using top M estimated locations from the WiFi localization system.

$$likelihood(x_k, wifi, uncertainty) = e^{\frac{a}{M}\sum_{i=1}^{M} L2Norm(wifi_i,x_k)^2*uncertainty_i}$$

4.3.4 The Resampling step

The ratio of effective particles in the Filter can be calculated by:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w_i)^2}$$

where N is the total number of particles in the filter. Resampling takes place when $N_{eff} < r$, r is a hyperparameter between 0 and 1 to control the frequency of resampling.

During the resampling, the standard systematic method [Liu and Chen, 1998] is performed. Calculate the cumulative sum of the particle weights:

$$w_{cum}(i) = w_{cum}(i-1) + w_i$$

where $w_{cum}(0) = 0$ and $w(i)$ is the weight of the ith particle. Then generate a random number u_1 between 0 and $\frac{1}{N}$:

$$u_1 = \frac{1}{N} \cdot Uniform(0,1)$$

After u_1 is obtained, select the first particle with weight w(1) that satisfies: $u_1 \leq w_{cum}(1)$ For each i = 2 to n, generate the next random number u_i:

$$u_i = u_1 + (i-1)/n$$

and select the particle with weight w(i) that satisfies:

$$u_i \leq w_{cum}(i) \leq u_{i+1}$$

This process is repeated until n particles have been selected, and then the particle weights w_i to w_N are reset to 1.

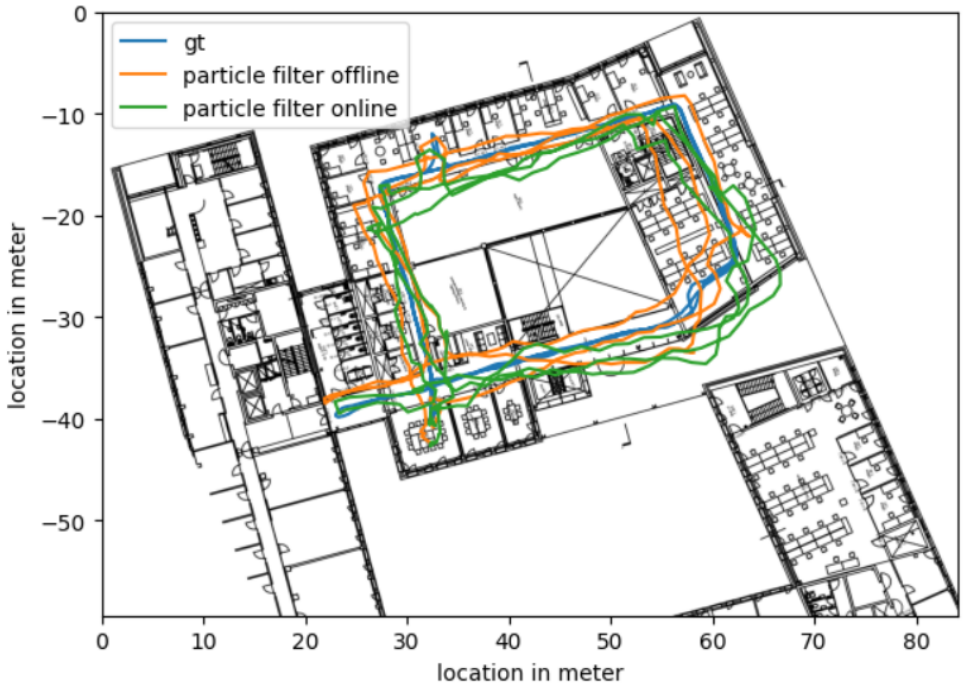

Figure 4.10: Sequence 4 PF fusion result Bayes F1

4.3.5 Fusion results

Unlike the EKF, tracing back history is possible in the PF, which means in offline mode, a better result can be obtained compared to the online mode. Therefore both online particle filter (current best estimation during the predictions and corrections) results and offline particle filter (mean history position of particles after all predictions and corrections) results will be shown in this section. In these experiments, the 1000 particles are sufficient for the filter prediction, the hyperparameter a in the likelihood function is set to be -0.002 and Neff ratio r is set to be 0.9. The top 3 WiFi measurements are used for the likelihood function as it generates the best result during test time. With an Intel i5 10400f CPU, this setup of PF takes about 100 times longer than the EKF fusion.

The figures and table below show the PF fusion on test sequences 4, 5 and 6. Compared with the EKF, PF has the ability to handle noisy measurement data, for example in sequence 4, where EKF fail to converge (Figure 4.9), a significant performance improvement can be observed from the table 4.2. Especially the offline PF fusion can even benefit the localization results when the WiFi measurements are poor as in sequence 6, the ATE have a 131% improvement from using the IMU model alone with aid of WiFi measurements with an average error larger than 9.3m.

4.3.6 Uncertainty ablation study

Unlike EKF, the PF algorithm is highly stochastic, and in multiple experiments, different initialization and propagation can lead to different results as shown in Figures 4.13 and 4.8. However, the differences between using estimated uncertainty and constant

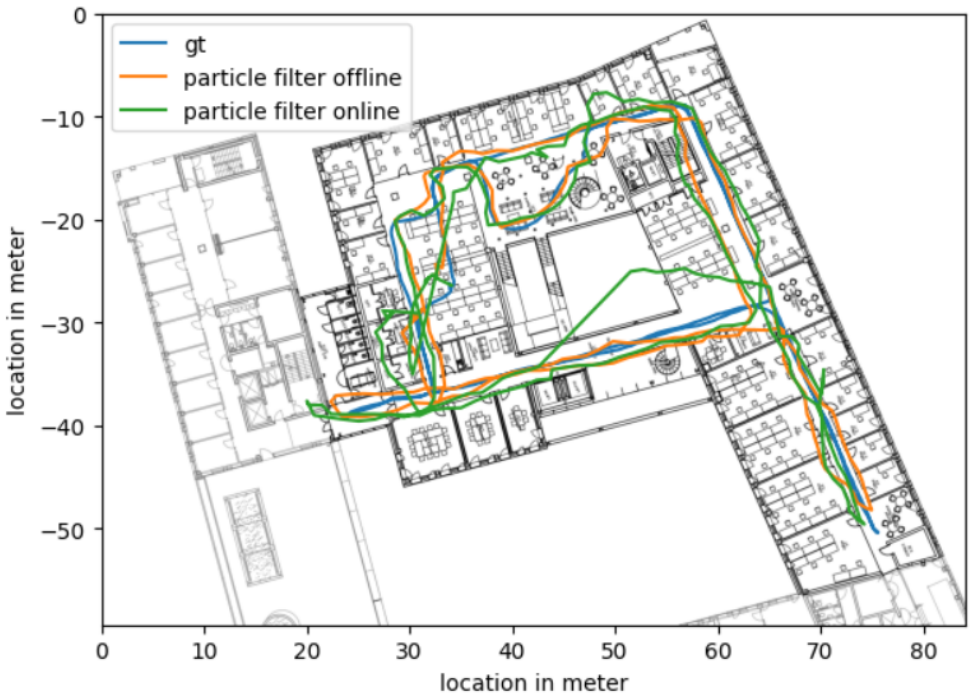

Figure 4.11: Sequence 5 PF fusion result Bayes F3

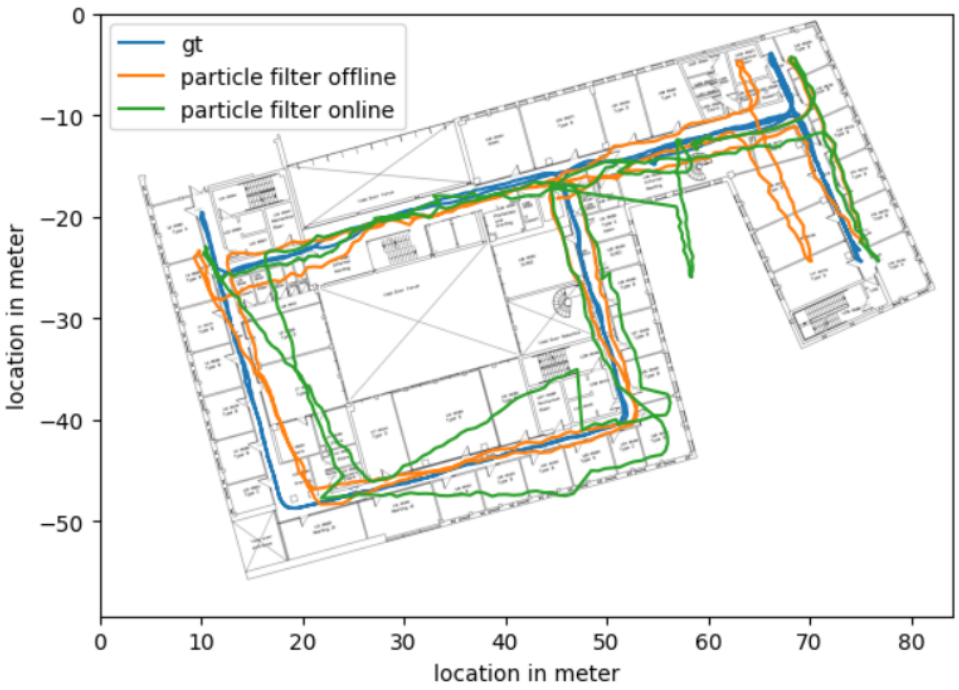

Figure 4.12: Sequence 6 PF fusion result Forum F1

Sequence	Online PF ATE	Offline PF ATE	WiFi Average error	IMU ATE
4	2.560	1.993	5.887	8.996
5	2.404	1.857	6.550	4.499
6	6.005	3.110	9.307	7.174

Table 4.2: Particle filter fusion results

uncertainty are often smaller than those caused by randomness. Therefore, sometimes better performance is observed with estimated uncertainty and sometimes the opposite is true. This means that it is difficult for PF to fully utilize the information provided by uncertainty estimation.

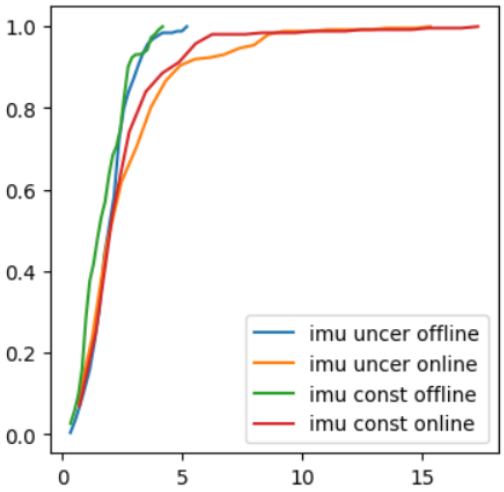

Figure 4.13: CDF 1 of test seq 4

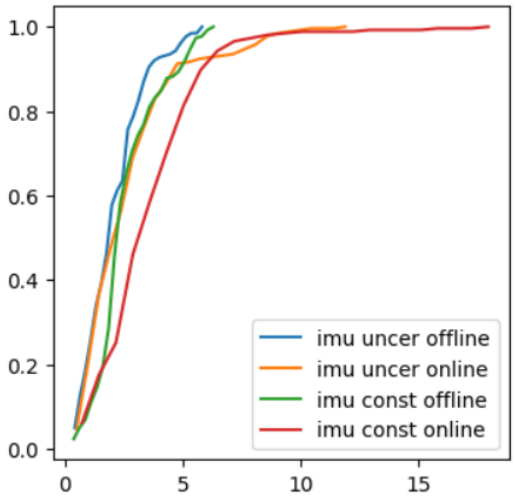

Figure 4.14: CDF 2 of test seq 4

4.3.7 Partial radio map fusion experiment

Unlike traditional single-sensor IPS, the proposed system uses a multi-sensor fusion algorithm and therefore brings the benefit of being able to rely on another sensor for position prediction when one sensor is unreliable. In this section, a partial radio map fusion experiment is done by using an incomplete radio map as shown in Figure 4.16, which makes the WiFi measurements unreliable in the south corridor. The uncertainty estimation technique can act as a indicator to reflect these situations to the filter.

Table 4.3 and Figure 4.3 show the result of this experiment. It can be observed that the PF especially the offline PF can return a reasonable trajectory even if the radio map is incomplete (i.e when the user walks outside the service-provided region). The table 4.3 also shows that, the performance is better when using the estimated uncertainty from the WiFi localization model instead of using a constant uncertainty in the fusion.

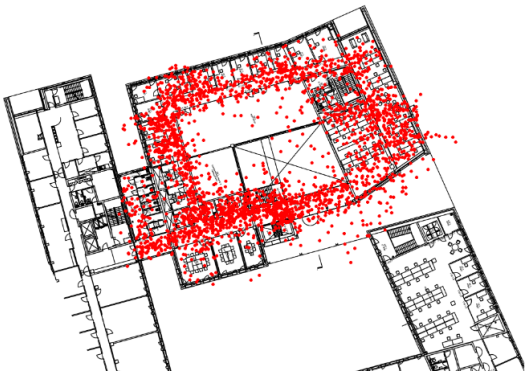

Figure 4.15: original radio map

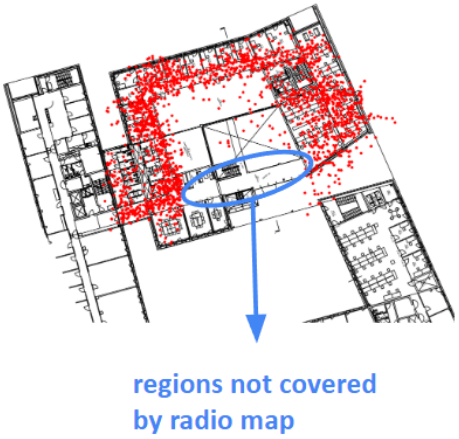

Figure 4.16: partial radio map

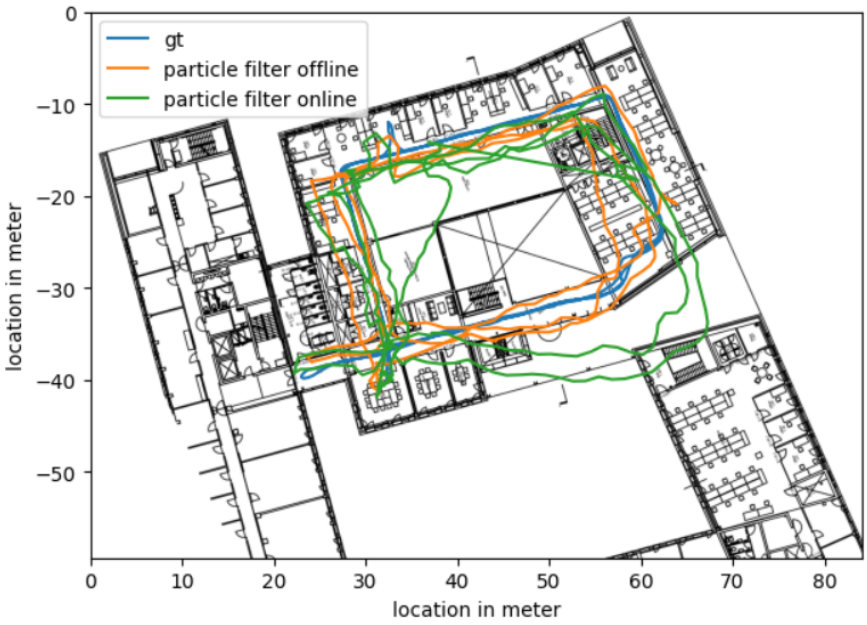

Figure 4.17: Partial radio map fusion experiment (estimated WiFi uncertainty)

Experiment	Online PF ATE	Offline PF ATE	WiFi error
original baseline	2.560	1.993	5.887
partial map estimated uncertainty	4.525	2.372	7.259
partial map constant uncertainty	4.832	2.589	7.259

Table 4.3: Particle filter partial map fusion experiment

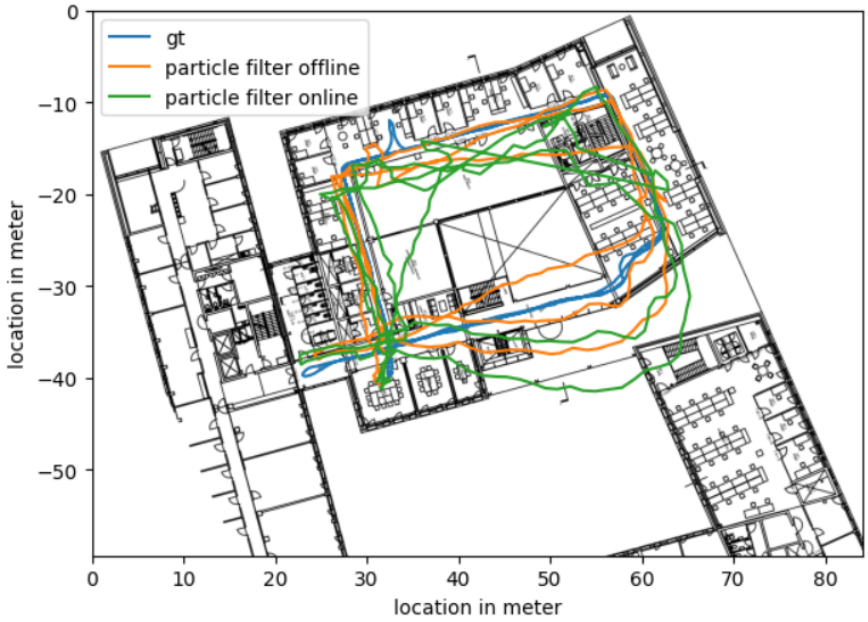

Figure 4.18: Partial radio map fusion experiment (constant WiFi uncertainty)

4.3.8 Limitations

Even though the PF is a powerful algorithm that can successfully fuse noisy motion and measurement data, it has a high computational complexity. The resampling is computationally expensive, particularly when the effective sample size is small. Additionally, determining the particle sizes and other hyperparameters can be a difficult and time-consuming process especially when during the inference, these can only be set empirically.

Chapter 5

Discussion and Conclusions

5.1 Conclusions

This thesis develops an uncertainty-aware inertial navigation system and uses it for indoor positional fusion in the Informatics Forums and Bayes Center. With the help of external WiFi positioning measurements with an average error of 4 to 10 meters, the proposed system can provide more accurate localization results once per second, and in the best case, has an improvement of over 140%.

The inertial navigation component itself is a deep learning-based regression network that predicts the 2D displacements from the IMU readings. Compared to its ancestor IONet [Chen et al., 2018], the proposed method uses a better data preprocessing pipeline and converts the predicted state from the translation and rotation in the IMU self-frame to 2D displacement in the world coordinate system, resulting in a significant performance improvement.

Two methods are used for the fusion, the PF and the EKF. While both methods are effective, the PF is more resilient and can handle situations with noisier measurement data. However, the PF introduces randomness and necessitates more computational resources.

According to Figure 3.17 and the EKF fusion results, the estimated uncertainties from the inertial navigation component show a desired error-correlated tendency and it can slightly improve the EKF fusion performance. The experiment in 4.3.7 illustrates that the uncertainty-aware technique can help the PF fusion in extreme cases, i.e. when a sensor is unreliable.

5.2 Novelty

The novelty of the proposed system lies mainly in the concept of using uncertainty to help the multi-sensor fusion IPS. When the results of a sensor are unreliable, e.g. if the user is out of range of the WiFi radio map, or if the IMU is accidentally damaged, etc. The estimated uncertainty can expose this information to the fusion algorithm and the

algorithm can rely more on the other sensor. The experiment in section 4.3.7 is a great example.

5.3 Limitations and future works

The improvement of using uncertainties estimated from the inertial navigation model in fusion is small in normal cases, even the performance impact of the randomness of PF outweighs this improvement. Even though the uncertainty in the model predictions is meaningful, neither fusion method can make good use of this information. Thus a question worth continuing to explore is adding the uncertainty estimation goal to the network turning the problem into a multi-objectives regression problem, such a network is more difficult to converge and suffers some performance losses. Then whether the uncertainty estimation is meaningful for the EKF and the PF positional fusion should be reconsidered in future works.

For the inertial navigation component, future works on this project should focus on building a better and more generalized dataset as described in section 3.4.

For the indoor positioning fusion, it is worth exploring better algorithms that can take advantage of this estimated uncertainty information for fusion, potentially the learning-based method, such as the differentiable particle filter and differentiable extended Kalman filter [Jonschkowski et al., 2018] [Kloss et al., 2021].

The Conditional Random Field (CRF) [Lafferty et al., 2001] can be a way the addressing randomness in the PF by discretizing the state representations, it can also potentially benefit the computational complexity.

Bibliography

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Amini et al., 2020] Amini, A., Schwarting, W., Soleimany, A., and Rus, D. (2020). Deep evidential regression.

[BASRI and El Khadimi, 2016] BASRI, C. and El Khadimi, A. (2016). Survey on indoor localization system and recent advances of wifi fingerprinting technique. In *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*, pages 253–259.

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks.

[Caron et al., 2006] Caron, F., Duflos, E., Pomorski, D., and Vanheeghe, P. (2006). Gps/imu data fusion using multisensor kalman filtering: Introduction of contextual aspects. *Information Fusion*, 7:221–230.

[Chen et al., 2018] Chen, C., Lu, X., Markham, A., and Trigoni, N. (2018). Ionet: Learning to cure the curse of drift in inertial odometry.

[Cheng et al., 2016] Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas. Association for Computational Linguistics.

[Crow et al., 1997] Crow, B., Widjaja, I., Kim, J., and Sakai, P. (1997). Ieee 802.11 wireless local area networks. *IEEE Communications Magazine*, 35(9):116–126.

[Foxlin, 2005] Foxlin, E. (2005). Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46.

[Gal and Ghahramani, 2016] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning.

[Ge and Qu, 2016] Ge, X. and Qu, Z. (2016). Optimization wifi indoor positioning knn algorithm location-based fingerprint. In *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 135–137.

[Hochreiter, 1998] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Jonschkowski et al., 2018] Jonschkowski, R., Rastogi, D., and Brock, O. (2018). Differentiable particle filters: End-to-end learning with algorithmic priors.

[Jordan, 2009] Jordan, M. (2009). Posteriors, conjugacy, and exponential families for completely random measures.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45.

[Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision?

[Kloss et al., 2021] Kloss, A., Martius, G., and Bohg, J. (2021). How to train your differentiable filter. *Autonomous Robots*.

[Lafferty et al., 2001] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Liu and Chen, 1998] Liu, J. S. and Chen, R. (1998). Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044.

[Liu et al., 2020] Liu, W., Caruso, D., Ilg, E., Dong, J., Mourikis, A. I., Daniilidis, K., Kumar, V., and Engel, J. (2020). TLIO: Tight learned inertial odometry. *IEEE Robotics and Automation Letters*, 5(4):5653–5660.

[Lu et al., 2020] Lu, C. X., Saputra, M. R. U., Zhao, P., Almalioglu, Y., de Gusmao, P. P. B., Chen, C., Sun, K., Trigoni, N., and Markham, A. (2020). milliego: Single-chip mmwave radar aided egomotion estimation via deep sensor fusion.

[Madgwick et al., 2010] Madgwick, S., Vaidyanathan, R., and Harrison, A. (2010). An efficient orientation filter for inertial measurement units (imus) and magnetic angular rate and gravity (marg) sensor arrays. Technical report, Department of Mechanical Engineering.

[Shan and Englot, 2018] Shan, T. and Englot, B. (2018). Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE.

[Shan et al., 2020] Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., and Daniela, R. (2020). Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE.

[Shoemake, 1985] Shoemake, K. (1985). Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, page 245–254, New York, NY, USA. Association for Computing Machinery.

[Sturm et al., 2012] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

[Yan et al., 2019] Yan, H., Herath, S., and Furukawa, Y. (2019). Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods.

[Yan et al., 2017] Yan, H., Shan, Q., and Furukawa, Y. (2017). Ridi: Robust imu double integration.