sensors

MDPI

Article

Deep Kalman Filter: Simultaneous Multi-Sensor Integration and Modelling; A GNSS/IMU Case Study

Siavash Hosseinyalamdary

Department of Earth Observation Science (EOS), Faculty of Geo-information Science and Earth Observation (ITC), University of Twente, Enschede 7514AE, The Netherlands; s.hosseinyalamdary@utwente.nl; Tel.: +31-(0)53-489-3921

Received: 15 March 2018; Accepted: 16 April 2018; Published: 24 April 2018

Abstract: Bayes filters, such as the Kalman and particle filters, have been used in sensor fusion to integrate two sources of information and obtain the best estimate of unknowns. The efficient integration of multiple sensors requires deep knowledge of their error sources. Some sensors, such as Inertial Measurement Unit (IMU), have complicated error sources. Therefore, IMU error modelling and the efficient integration of IMU and Global Navigation Satellite System (GNSS) observations has remained a challenge. In this paper, we developed deep Kalman filter to model and remove IMU errors and, consequently, improve the accuracy of IMU positioning. To achieve this, we added a modelling step to the prediction and update steps of the Kalman filter, so that the IMU error model is learned during integration. The results showed our deep Kalman filter outperformed the conventional Kalman filter and reached a higher level of accuracy.

Keywords: deep Kalman filter; Simultaneous Sensor Integration and Modelling (SSIM); GNSS/IMU integration; recurrent neural network (RNN); deep learning; Long-Short Term Memory (LSTM)

1. Problem Statement

Global Navigation Satellite Systems (GNSS) enable us to locate ourselves within a few centimeters all over the world. This system consists of a Global Positioning System (GPS), Galileo, GLobal Orbiting NAvigation Satellite System (GLONASS), and Beidu, and it is integrated into our daily lives, from car navigators to airplanes. Unfortunately, GNSS positioning requires a clear sky view; therefore, it is not available in urban canyons where GNSS signals are blocked by high-rise buildings. Other alternative navigation solutions have been applied to overcome this shortcoming of GNSS positioning and bridge its gaps in urban canyons.

Among alternative navigation solutions, inertial navigation and visual odometry are cost-effective and do not require any infrastructure. The Inertial Measurement Unit (IMU) is a composition of accelerometers and gyroscopes and it estimates the position, velocity, and orientation of a platform from measured accelerations and angular rates.

The error characteristics of IMU sensors are complicated. They differ significantly from one technology, one manufacturer, and even one sensor, to another. The cold atom is the most accurate IMU [1], but it is very expensive and not applicable for commercial applications, such as mobile mapping. Other IMU technologies, such as mechanical IMU and Micro-Electro-Mechanical System (MEMS), suffer from common error sources, such as bias and scale factor errors and some technology-specific error sources, such as dead zone errors in Ring Laser Gyros (RLGs).

IMU sensors have systematic, random, and computational error sources. IMU manufacturers use controlled environments, such as turntable, to estimate and remove systematic errors. The calibration procedures associated with controlled environment are costly and it cannot fully remove systematic errors. The systematic and random errors of IMU can be estimated if the relationship between

these errors and the observations are known. When GNSS positioning is available, these errors are approximated using pre-existing IMU error models and this process is called IMU calibration. When IMU errors are estimated, they are removed from IMU observations and IMU positioning becomes more accurate. Unfortunately, the pre-existing models of IMU errors are not accurate and they are based on some statistical assumptions that may not hold.

In addition, Bayes filters, such as the Kalman and particle filters, have some limitations that may introduce computational error. For instance, the conventional Kalman filter is linear and cannot handle non-linear error sources. Therefore, if the conventional Kalman filter is applied for non-linear IMU error modelling, it leads to computational error. Variants of the Kalman filter and particle filter have been developed to overcome the shortcomings of the conventional Kalman filter, but they are also limited to statistical assumptions.

If the error characteristics of IMU sensors can be accurately modelled, the accuracy of IMU positioning can be significantly improved. Currently, scientists suggest modelling error sources of IMU separately. For instance, Reference [2] suggested modelling accelerometer and gyroscope bias as a random constant, and [3] applied the first-order Gauss–Markov stochastic process to model these biases.

In this paper, we introduce the deep Kalman filter to simultaneously integrate GNSS and IMU sensors and model IMU errors. In contrast to previously proposed approaches, our approach does not have any pre-defined IMU error model and it is learned from observations. Therefore, we do not need to assume any stochastic or deterministic behavior of IMU errors. In contrast to previously proposed approaches, we can accurately model non-linear, time-variant, highly correlated IMU error sources.

1.1. Literature Review

There are a few scientific endeavors to harness IMU errors and provide accurate alternative positioning systems in the absence of GNSS positioning. As we have already discussed, accelerometer and gyro bias were modelled as random constants in [2], and they were modelled as first-order Gauss–Markov stochastic processes in [3]. Shin and El-Sheimy [4] used gravity and the earth rotation rate to calibrate IMU errors.

Wang et al. [5] treated IMU error sources as a time series and applied the AutoRegressive and Moving Average (ARMA) method to model IMU error sources. In addition to the statistical estimators, shallow Multi-Layer Perceptron (MLP) networks have been utilized to model IMU error sources in the presence of GNSS positioning [6,7]. Adaptive Neural Fuzzy Information Systems (ANFIS) have also been applied to capture IMU uncertainties [8,9]. Toth and colleagues applied neural networks and fuzzy logic [10] to model IMU error sources. Navidi et al. [11] used hybrid Fuzzy Inference Systems (FIS) and the second-order extended Kalman filter to integrate GNSS and IMU sensors. Xing and colleagues showed that chaotic particle swarm optimization significantly reduced gyroscope random drift [12]. Applied neural networks are shallow and they do not accurately calibrate IMU since they do not consider the IMU's correlation over time. In other words, IMU error sources should be studied as a time series and they should be modelled over time.

It has been shown that shallow neural networks can only model simple phenomena and that complicated systems should be modelled using deep neural networks [13]. Therefore, we applied deep neural network for sensor integration. To our knowledge, we are the first to apply deep neural networks for GNSS and IMU integration and IMU error modelling.

Mirowski and Lecun [14] introduced dynamic factor graphs and reformulated Bayes filters as recurrent neural networks. In their proposed approach, the observation and system models of the Kalman filter are learned from observations. Gu et al. [15] reformulated the Kalman filter and recurrent neural network to model face landmark localization in videos. Krishnan et al. [16] integrated the Kalman filter and variational methods for learning famous handwritten digit datasets, Modified National Institute of Standards and Technology (MNIST).

1.2. Kalman Filter

The unknown vector, which is estimated in the Kalman filter, is called a state vector and it is represented by $\boldsymbol{x} \in \mathbb{R}^n$, where t indicates the state vector at time t. It also depends on the observation vectors, $z_{1:t}$, where $\boldsymbol{z} \in \mathbb{R}^m$, and the initial state of the system $\boldsymbol{x}_0$. The probability of the state vector at the current time is $\Pr(\boldsymbol{x}_t|z_{1:t}, \boldsymbol{x}_0)$. Therefore, the current state vector is estimated using Maximum Likelihood (ML) estimation, such that:

$$\hat{\boldsymbol{x}}_t = \underset{\boldsymbol{x}_t}{\operatorname{argmax}} \Pr(\boldsymbol{x}_t|z_{1:t}, \boldsymbol{x}_0) \tag{1}$$

The probability of the current state vector depends on the previous state vectors, such that:

$$\hat{\boldsymbol{x}}_t = \underset{\boldsymbol{x}_t}{\operatorname{argmax}} \frac{\Pr(z_t|\boldsymbol{x}_t)\Pr(\boldsymbol{x}_t|z_{1:t-1}, \boldsymbol{x}_0)}{\Pr(z_{1:t})} \tag{2}$$

The denominator in Equation (2) is a normalization constant and it does not contribute to the maximization of likelihood in Equation (2), such that:

$$\hat{\boldsymbol{x}}_t = \underset{\boldsymbol{x}_t}{\operatorname{argmax}} \Pr(z_t|\boldsymbol{x}_t)\Pr(\boldsymbol{x}_t|z_{1:t-1}, \boldsymbol{x}_0) \tag{3}$$

The state vector at the current time directly depends on the previous state vectors. Therefore, the marginalization of previous state vectors is applied and leads to state vector estimation based on previous state vectors, such that:

$$\hat{\boldsymbol{x}}_t = \underset{\boldsymbol{x}_t}{\operatorname{argmax}} \Pr(z_t|\boldsymbol{x}_t) \int \Pr(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})\Pr(\boldsymbol{x}_{1:t-1}|z_{1:t-1}, \boldsymbol{x}_0) d\boldsymbol{x}_{1:t-1} \tag{4}$$

Based on the Markovian assumption, the state vector at the current time only depends on the state vector at previous times, such that:

$$\hat{\boldsymbol{x}}_t = \underset{\boldsymbol{x}_t}{\operatorname{argmax}} \Pr(z_t|\boldsymbol{x}_t) \int \Pr(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\Pr(\boldsymbol{x}_{t-1}|z_{1:t-1}, \boldsymbol{x}_0) d\boldsymbol{x}_{t-1} \tag{5}$$

where $\Pr(z_t|\boldsymbol{x}_t)$ is the posterior probability estimation of the current state vector. The state vector best estimate in the previous time is $\hat{\boldsymbol{x}}_{t-1} = \underset{\boldsymbol{x}_{t-1}}{\operatorname{argmax}} \Pr(\boldsymbol{x}_{t-1}|z_{1:t-1}, \boldsymbol{x}_0)$. Therefore, Equation (5) is reformulated as:

$$\hat{\boldsymbol{x}}_t = \underset{\boldsymbol{x}_t}{\operatorname{argmax}} \Pr(z_t|\boldsymbol{x}_t) \int \Pr(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\hat{\boldsymbol{x}}_{t-1} d\boldsymbol{x}_{t-1} \tag{6}$$

where $\int \Pr(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\hat{\boldsymbol{x}}_{t-1} d\boldsymbol{x}_{t-1}$ is the prior probability estimation. This predicts the current state vector based on the system model and the posterior estimation of the state vector at previous times.

In the Kalman filter, the state vector is related to the state vector at previous times, $t-1$, using the system model, f, such that:

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}) + \boldsymbol{\epsilon}_t \tag{7}$$

where $\boldsymbol{\epsilon}_t$, is the noise of the system model. The state vector relates to the observation vector, $\boldsymbol{z}_t \in \mathbb{R}^m$, with the observation model, g, such that:

$$\boldsymbol{z}_t = g(\boldsymbol{x}_t) + \boldsymbol{\omega}_t \tag{8}$$

where ω_t, is the noise of the observation model. The state and observation models are assumed to be linear in the Kalman filter. Therefore, these functions can be replaced by F and G matrices, respectively. The system model can be rewritten as:

$$\boldsymbol{x}_t = F\boldsymbol{x}_{t-1} + \boldsymbol{\epsilon}_t \tag{9}$$

and similarly, the observation model can be rewritten as:

$$\boldsymbol{z}_t = G\boldsymbol{x}_t + \boldsymbol{\omega}_t \tag{10}$$

The noise of the system and observation models are also assumed to have a Normal distribution in the Kalman filter, such that:

$$\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, Q_t) \tag{11}$$

$$\boldsymbol{\omega}_t \sim \mathcal{N}(0, R_t) \tag{12}$$

where Q_t and R_t are the covariance matrices of the system and observation models.

The Kalman filter is composed of two-stage optimization. In the first stage, the current state vector is predicted based on the state vector in the previous time, such that:

$$\boldsymbol{x}_t^- = F\boldsymbol{x}_{t-1}^+ \tag{13}$$

The predicted values are represented by the superscript '$-$' and the updated values are represented by the superscript '+'. Error propagation is applied to estimate the covariance matrix of the current state vector based on the covariance matrix of the state vector in the previous time, such that:

$$P_t^- = FP_{t-1}^+F + Q_t \tag{14}$$

where P_t^- is the predicted covariance matrix of the state vector.

In the update stage of the Kalman filter, the current state vector is updated by using the observation vector, such that:

$$\boldsymbol{x}_t^+ = \boldsymbol{x}_t^- + K_t\left(\boldsymbol{z}_t - G_t\boldsymbol{x}_t^-\right) \tag{15}$$

and the covariance matrix of the updated current state vector is calculated, such that:

$$P_t^+ = (I - K_tG_t)^T P_t^- (I - K_tG_t) + {K_t}^T R_t K_t \tag{16}$$

where K_t is the Kalman gain matrix, calculated as:

$$K_t = P_t^- {G_t}^T (G_t P_t^- {G_t}^T + R_t)^{-1} \tag{17}$$

For derivation of the Kalman filter equations, the reader is referred to [2]. Figure 1 shows the Kalman filter.

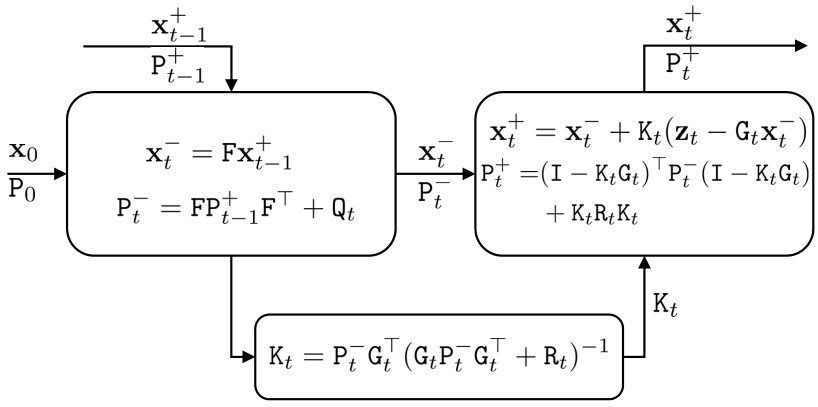

Figure 1. The Kalman filter procedure, which consists of prediction (**left-up box**) and update steps (**right-up box**).

Shortcomings of the Kalman Filter

There are a number of shortcomings of the Kalman filter. Namely, F and G are linear models with Gaussian noise. Therefore, they cannot model non-linear functions or linear functions with non-Gaussian noise. In addition, these functions are time invariant, meaning they do not change over time. A number of variations in the Kalman filter, such as extended and unscented Kalman filters, can handle non-linear observation and system models. In addition, the particle filter works on observation and system models with other distributions than Gaussian noise.

Nonetheless, the observation and system models of Bayes filters should be known beforehand. In other words, scientists should find models to relate state vectors to previous state vectors and observations. As an example, we discussed that accelerometer and gyro bias can be modelled in different ways. Unfortunately, the observation and system models cannot be determined beforehand in many applications.

Another shortcoming of Bayes filters is their Markovian assumption. In Bayes filters, it is assumed that the current state vector only depends on the previous state vector and it is independent from older state vectors. Although this property significantly simplifies the system model and makes these filters very efficient, it makes the Bayes filter insensitive to system behavior with longer correlation times. In other words, complicated error models with long correlation times cannot be modelled using Bayes filters.

Here, we provide our discussion with an example of a case study on IMU error modelling. IMU models have different error sources depending on the applied technology in the accelerometer and gyroscope of the IMU. One IMU manufacturer can calibrate IMU error sources different to another manufacturer. Moreover, the error sources of MEMS sensors can significantly differ from one MEMS sensor to another, even in one MEMS family. Pre-defined IMU error models cannot handle the high variation of error sources in MEMS sensors. In addition, the high correlation between different MEMS error sources makes IMU error modelling very complicated. Often, it is not possible to discriminate error sources in MEMS sensors.

2. Methodology

In this paper, we add system modelling to the Kalman filter and refer to it as the deep Kalman filter. In other words, the deep Kalman filter is able to estimate the system model and it is useful in many applications, such as GNSS/IMU integration, where the system model is complicated. In order to estimate the system model of the Kalman filter, we add latent variables to the Kalman filter. Latent variables are not observed and they are invisible in the state vector, but the state vector depends on the latent variables. As an example, IMU errors depend on temperature, but this is not observed and cannot be estimated. These variables are represented by the latent vector, $\boldsymbol{h}_t$, where the subscription t stands for the latent vector at the current time. The current latent vector depends on the previous latent vectors, $\boldsymbol{h}_{t-1:t-T}$, where T is the number of previous latent vectors utilized for the current latent vector estimation.

We assume the current state vector, $\boldsymbol{x}_t$, depends on the current latent vectors, $\boldsymbol{h}_t$. Therefore, the current state vector indirectly depends on previous state vectors, $\boldsymbol{x}_{t-1:t-T}$, and the Markovian assumption does not hold anymore. We design our modelling step in the way that the current state vector depends on the current latent vector and the current latent vector depends on previous state vectors and previous latent vectors. This is one of many different architectures of possible networks, but it significantly simplifies our network.

Let us assume there is a function ϕ that relates the current latent vector to the previous latent vectors and previous state vectors, such that:

$$\boldsymbol{h}_t = \phi\left(\boldsymbol{x}^{+}_{t-1:t-T}, \boldsymbol{h}_{t-1:t-T}\right) \tag{18}$$

and the current state vector is directly related to the current latent vector by a function, λ, such that:

$$\boldsymbol{x}_t^{+-} = \lambda(\boldsymbol{h}_t) + \boldsymbol{\mu}_t \tag{19}$$

we name the posterior estimation based on our model as $\boldsymbol{x}_t^{+-}$. In other words, $\boldsymbol{x}_t^{+-}$ is the predicted posterior estimation of the state vector. We can approximate ϕ and λ with a combination of linear and non-linear functions. Linear functions can be represented by matrix multiplication, where coefficients of linear functions are represented by coefficient matrices, W. The non-linear function, σ, has no coefficient. Therefore, our network is designed, such that:

$$\boldsymbol{h}_t = \sigma\left(W_{xh}\boldsymbol{x}_{t-1:t-T}^{+}, W_{hh}\boldsymbol{h}_{t-1:t-T}\right) \tag{20}$$

$$\boldsymbol{x}_t^{+-} = \sigma(W_{xx}\boldsymbol{h}_t) + \boldsymbol{\mu}_t \tag{21}$$

Let us name the coefficients of the latent vector as W_h, where $W_h = [W_{xh}, W_{hh}]$. In order to model our system, it suffices to estimate W_h and W_{xx}. Figure 2 shows the probabilistic graphical model of the Kalman filter and deep Kalman filter. The upper part of the deep Kalman filter is the prediction and update steps and it is similar to the conventional Kalman filter. However, we added a modelling step to the Kalman filter, which is the lower part of the deep Kalman filter in Figure 2.

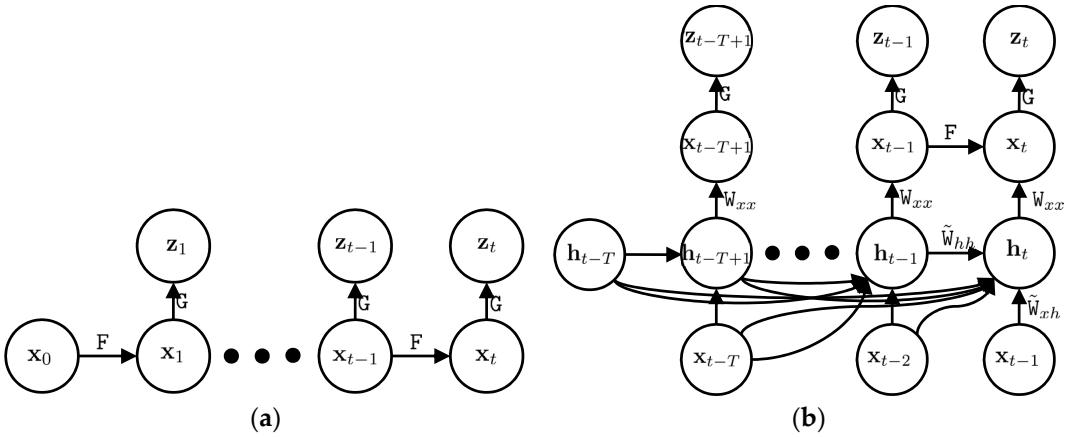

Figure 2. The probabilistic graphical model of the Kalman filter (**a**) and deep Kalman filter (**b**); x, z, and h are the state vector, observation vector, and latent vector, respectively. The matrices F and G are the system model and observation model of the Kalman filter and W is the coefficient matrix of our proposed IMU model. The two upper layers of the deep Kalman filter are similar to the Kalman filter and the added two lower layers enable the deep Kalman filter to estimate our system model.

2.1. Expectation Maximization

When the parameters of the system model are unknown, in addition to the state vector, the system model and state vector cannot be directly estimated. In other words, the state vector depends on the latent variables of the system model and therefore, it is not estimable. In order to find the state vector and latent vector, an expectation maximization approach is utilized [17]. Expected Maximization is an iterative approach to estimate the latent vectors, as well as the state vector. In the first step, the latent vector is guessed, represented by $\boldsymbol{h}_t^{(0)}$, where superscript 0 shows that this is the initial guess of the latent vector. The state vector is estimated based on the guessed latent vector, such that:

$$\hat{\boldsymbol{x}}_t^{(1)} = \underset{\boldsymbol{x}_t}{\text{argmax}} \Pr(\boldsymbol{x}_t | \boldsymbol{h}_t^{(0)}, \boldsymbol{z}_{1:t}, \boldsymbol{x}_0) \tag{22}$$

In contrast to Equation (1), where the state vector is estimated based on a pre-defined system model, Equation (22) obtains the state vector based on the guessed latent vector. In the next step, the latent vector is estimated based on the calculated state vector in Equation (22), such that:

$$\hat{\boldsymbol{h}}_t^{(1)} = \underset{\boldsymbol{h}_t}{\text{argmax}} \Pr(\hat{\boldsymbol{x}}_t^{(1)} | \boldsymbol{h}_t, \boldsymbol{z}_{1:t}, \boldsymbol{x}_0) \tag{23}$$

Using Equation (21), it suffices to estimate W_{xx} in order to calculate the state vector, $\boldsymbol{x}_t$. Similarly, it suffices to estimate W_h in order to find the latent vector, $\boldsymbol{h}_t$, and model our system. Replacing the coefficient matrices from Equations (20) and (21) into Equation (22), derives the coefficient matrices, such that:

$$\hat{W}_{xx}^{(1)} = \underset{W_{xx}}{\text{argmax}} \Pr(\boldsymbol{x}_t | \boldsymbol{h}_t^{(0)}, \boldsymbol{z}_{1:t}, \boldsymbol{x}_0) \tag{24}$$

When the coefficient matrix $\hat{W}_{xx}^{(1)}$ is estimated, the state vector, $\hat{\boldsymbol{x}}_t^{(1)}$, is calculated. The coefficient matrices of Equation (20) are estimated, such that:

$$\hat{W}_h^{(1)} = \underset{W_h}{\text{argmax}} \Pr(\hat{\boldsymbol{x}}_t^{(1)} | \boldsymbol{h}_t, \boldsymbol{z}_{1:t}, \boldsymbol{x}_0) \tag{25}$$

The iterative estimation of latent and state vectors continues until the algorithm converges to its solution, the system model is determined and the state vector is estimated. Mirowski and LeCun [14] showed that expectation maximization can lead to recurrent neural networks.

Expectation maximization will lead to a global maximum in convex functions. However, it is most likely it will converge to a local maximum, since modelling is not convex for complicated models. One of the challenges is finding an approach to maximize Equations (24) and (25).

Figure 3 shows the scheme of the proposed deep Kalman filter.

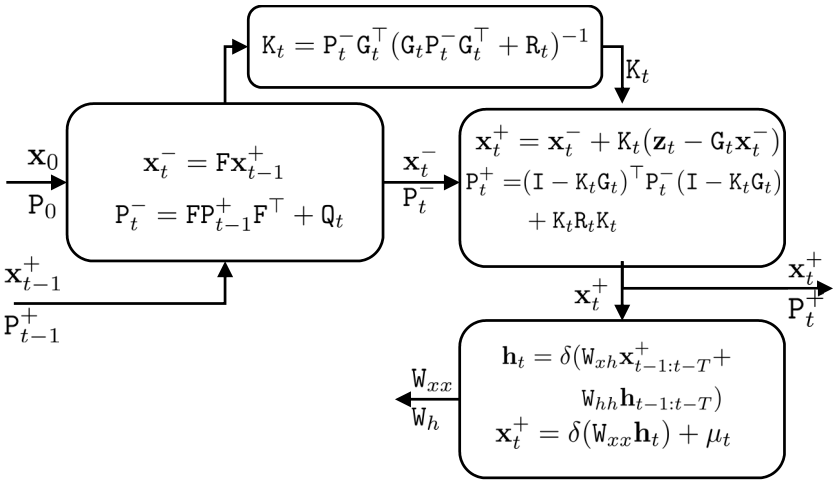

Figure 3. The deep Kalman filter procedure. The IMU modelling step (**right-bottom**) has been added to the Kalman filter. The modelling is accomplished in two steps: in the first step, the current latent vector is estimated based on previous latent and state vectors; and in the second step, the current state vector is estimated based on the current latent vector.

2.2. Recurrent Neural Network

We rewrite Equation (6) based on the latent vector estimation, such that:

$$\hat{\boldsymbol{x}}_t = \underset{\boldsymbol{x}_t}{\text{argmax}} \Pr(\boldsymbol{z}_t | \boldsymbol{x}_t) \int \Pr(\boldsymbol{x}_t | \boldsymbol{h}_t) \Pr(\boldsymbol{h} | \boldsymbol{h}_{t-1:t-T}, \boldsymbol{x}_{t-1:t-T}) \hat{\boldsymbol{x}}_{t-1:t-T} d\boldsymbol{x}_{t-1:t-T} \tag{26}$$

In the E step of the EM algorithm, we calculate $\hat{\boldsymbol{x}}_t$ based on the guessed model and the observations. In the case of GNSS and IMU integration, we can use the Kalman filter to estimate the state vector, $\boldsymbol{x}_t^+$. Since GNSS observations are accurate, the estimated state vector is accurate too. In the M step of the EM algorithm, we use the guessed model to calculate the state vector and we represent the approximated state vector based on the guess model by $\boldsymbol{x}_t^{+-}$. If our model is accurate, the approximate state vector, $\boldsymbol{x}_t^{+-}$, should be close to the estimated state vector, $\boldsymbol{x}_t^+$. Therefore, their difference is stated as an energy function, E, such that:

$$E(W_h, W_{xx}) = \frac{1}{2}(\boldsymbol{x}_t^{+-} - \boldsymbol{x}_t^+)^2 \tag{27}$$

We can model our system if we minimize this energy function. By replacing $\boldsymbol{x}_t^{+-}$ from Equation (21), Equation (26) is reformulated as:

$$E(W_h, W_{xx}) = \frac{1}{2}\left(\sigma(W_{xx}\boldsymbol{h}_t) - \boldsymbol{x}_t^+\right)^2 \tag{28}$$

Therefore, the gradient of the coefficient matrix, W_{xx}, is calculated as:

$$\frac{\partial E(W_h, W_{xx})}{\partial W_{xx}} = \frac{\partial \sigma(W_{xx}\boldsymbol{h}_t)}{\partial W_{xx}}\left(\sigma(W_{xx}\boldsymbol{h}_t) - \boldsymbol{x}_t^+\right) \tag{29}$$

The gradient of the latent vector coefficient matrix, W_h, is also estimated as:

$$\frac{\partial E(W_h, W_{xx})}{\partial W_h} = \frac{\partial \sigma\left(W_{xh}\boldsymbol{x}_{t-1:t-T}^+ + W_{hh}\boldsymbol{h}_{t-1:t-T}\right)}{\partial W_h}\frac{\partial \sigma(W_{xx}\boldsymbol{h}_t)}{\partial W_{xx}}\left(\sigma(W_{xx}\boldsymbol{h}_t) - \boldsymbol{x}_t^+\right) \tag{30}$$

In order to minimize the energy function, we use an iterative gradient descent. Therefore, the coefficient matrices of the modelled system, W_{xx} and W_h, are estimated, such that:

$$W_{xx}^{(m+1)} = -\frac{\partial E(W_h, W_{xx})}{\partial W_{xx}}\mu + W_{xx}^{(m)} \tag{31}$$

$$W_h^{(m+1)} = -\frac{\partial E(W_h, W_{xx})}{\partial W_h}\mu + W_h^{(m)} \tag{32}$$

where μ is the learning rate. When the coefficient matrices are determined and system model is learned, $\boldsymbol{x}_t^{+-}$ can be estimated based on the previous state vectors, $\boldsymbol{x}_{t-1:t-T}^+$, in Equations (20) and (21).

2.3. Long Short-Term Memory

Recurrent neural networks have a drawback, known as the exploding-vanishing gradient problem [18]. When T is large and we model the system for a long time span, the gradients are multiplied in several layers. If the gradients are large, their multiplication becomes humongous and their gradient explodes. On the contrary, if the gradients are small, their multiplication becomes insignificant, known as a vanishing gradient. In order to prevent such an effect in recurrent neural networks, the Long Short-Term Memory (LSTM) method involves the introduction of gated memories [19]. In LSTM, the network consists of cells and each cell can memorize the previous state vectors using an input gate, remember them using an output gate, and forget them using a forget gate. Let us represent the input gates as $\boldsymbol{i}_t$, output gates as $\boldsymbol{o}_t$, and forget gates as $\boldsymbol{f}_t$. They are represented by a combination of linear and non-linear functions, such that:

$$\boldsymbol{f}_t = \sigma\left(W_f\boldsymbol{x}_t + U_f\boldsymbol{h}_{t-1}\right) \tag{33}$$

$$\boldsymbol{i}_t = \sigma(W_i\boldsymbol{x}_t + U_i\boldsymbol{h}_{t-1}) \tag{34}$$

$$\boldsymbol{o}_t = \sigma(W_o \boldsymbol{x}_t + U_o \boldsymbol{h}_{t-1}) \tag{35}$$

where σ is the non-linear function and the linear functions are represented by the coefficient matrices, W_f, W_i, and W_o. The cell state, $\boldsymbol{c}_t$, and latent layer, $\boldsymbol{h}_t$, are estimated as:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \sigma(W_c \boldsymbol{x}_t + U_c \boldsymbol{h}_{t-1}) \tag{36}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \sigma(\boldsymbol{c}_t) \tag{37}$$

where $\circ$ is a Hadamard product. For long term correlation, the input gate can keep information from previous state vectors and the gradients of previous state vectors are accessible. Therefore, the gradients do not explode or vanish in the back-propagation process. The forget gate controls the complexity of the model and it removes the uncorrelated previous state vectors.

3. Implementation

In this section, we explain the details of our model implementation. We first explain the implementation of our Kalman filter and then we explain the implementation of our proposed deep Kalman filter.

It has been shown that the Kalman filter works better on IMU errors than IMU output [2,7] Therefore, we utilized the IMU mechanization to estimate position, velocity, and orientation. However, IMU mechanization is not perfect and errors related to position, velocity, and orientation remain in the system. We defined the state vector of the Kalman filter in terms of positioning error, velocity error, orientation error, and the bias of accelerometers and gyroscopes. The system model utilized in the Kalman filter is similar to Reference [3] and described in the appendix. The GNSS observations are used as the observation vector and applied to the observation model to update the state vector.

We implemented an extended Kalman filter and applied the system and observation models to predict and update the state vector. When GNSS observations are available, the extended Kalman filter predicts and updates the state vector. If GNSS observations are not available, the extended Kalman filter only predicts the state vector.

For the implementation of our deep Kalman filter, we calculated the posterior estimation of the state vectors, $\boldsymbol{x}_t^+$, when GNSS observations were available. Since we used the Real-Time Kinematic (RTK) technique for GNSS positioning, our posterior estimation was anticipated to be very accurate and it could be used as a ground truth for our IMU error modelling. We used our proposed deep Kalman filter to predict the state vector and the predicted state vector using our model was $\boldsymbol{x}_t^{+-}$. We tried to predict $\boldsymbol{x}_t^{+-}$ as close as possible to $\boldsymbol{x}_t^+$ and, therefore, we got the best estimate of the modelled IMU errors. In other words, we found a model to calculate $\boldsymbol{x}_t^{+-}$ as an approximation of $\boldsymbol{x}_t^+$ in the presence of GNSS positioning and we replaced prior estimation of the Kalman filter, $\boldsymbol{x}_t^-$, to $\boldsymbol{x}_t^{+-}$ in the absence of GNSS positioning. If the system model is accurately estimated, the predicted state vector used has higher accuracy if the proposed deep Kalman filter is applied.

In order to implement the proposed approach, we applied a predictive network, where the state vector was predicted from previous state vectors and latent vectors. This predictive network is shown in Figure 4. In the predictive network, we predicted a sequence of state vectors from $t - T + 1$ to t, $\boldsymbol{x}_{t-T+1}^+, ..., \boldsymbol{x}_t^+$, based on a sequence of state vectors $t - T$ to $t - 1$, $\boldsymbol{x}_{t-T}^+, ..., \boldsymbol{x}_{t-1}^+$. In other words, we learned the model, in addition to the state vector prediction, in every step. The sequence $\boldsymbol{x}_{t-T+1}^+, ..., \boldsymbol{x}_t^+$ was used as a target layer in our network and the sequence $\boldsymbol{x}_{t-T}^+, ..., \boldsymbol{x}_{t-1}^+$ was applied as an input layer. The prediction of our model was a sequence of predicted state vectors, $\boldsymbol{x}_{t-T}^{+-}, ..., \boldsymbol{x}_{t-1}^{+-}$. The hidden layer had a sequence of $\boldsymbol{h}_{t-T}, ..., \boldsymbol{h}_t$. We calculated the coefficient matrices W_{xx} and W_h in order to model the IMU errors.

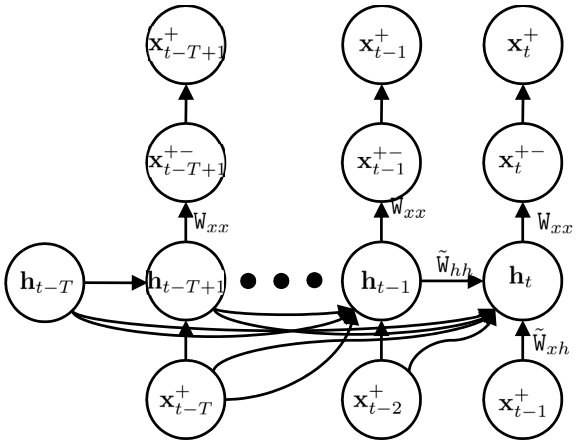

Figure 4. IMU error modelling reformulated as a time series prediction. The posterior estimation of the state vector is utilized in the target node (**top layer**), the output of our model is considered as the output node (**second layer**), the previous state vectors are in the input layer (**bottom layer**) and the latent vectors are in the hidden layer (**third layer**).

In order to implement an efficient predictive recurrent neural network, we designed a network with only one latent layer while every latent vector, $\boldsymbol{h}_t$, contained 3000 variables, known as nodes, in neural network terminology. We were benefited by mini-batch learning, where sequences are added together and processed once. The batch size of our networks was 800. This speeds up the learning process and improves learning convergence. Another important factor of training in recurrent neural networks is the learning rate. We used 0.05 as the initial learning rate, with 5×10^{-3} learning rate decay. We tried several sequence lengths and studied the impact of sequence length on accurate error modeling.

When GNSS observations are not available, the posterior estimation of the state vector cannot be calculated. Therefore, $\boldsymbol{x}_t^+$ is not estimable and we can only predict the state vector. There are two ways to predict the current state vector: using the system model of the Kalman filter or applying the modelled system model of IMU in the deep Kalman filter. In the first approach, the state vector is predicted based on a pre-defined system model, as in Equation (13). In the second approach, the predicted state vector is estimated based on our learned IMU model, as in Equations (20) and (21). We anticipated that the modelled system in the deep Kalman filter would perform better than the system model of the Kalman filter. In other words, the modelled IMU errors were anticipated to be more accurate than the predicted IMU errors of the Kalman filter.

4. Experiment

In order to evaluate our proposed approach, the KITTI benchmark [20] was utilized. This benchmark contains multi-sensor datasets and labeled information. Among various sensors, GPS and IMU information was used in this paper. We used the longest KITTI dataset, dataset No. 34, collected on 3 October 2011. This dataset contains 7:46 min of data collection, mostly driven in residential areas with a clear sky view. The trajectory of the platform exceeds 1.7 km, as shown in Figure 5. The OXTS RT 3003 navigation system was applied to collect the GPS and IMU information. The GPS and IMU information had already been synchronized and collected in 10 Hz. Manufacturers had performed calibration procedures and the calibration parameters, such as level-arm, were internally applied to transfer the GPS information into the IMU local frame. Dual-frequency RTK was used to estimate the accurate position of the IMU, with accuracy better than 10 cm.

We used the GPS observations as the ground truth to train our network and model IMU errors. When the weights of our network were learned and IMU errors were modelled, we estimated the IMU errors in two ways and compared them. First, we did not use GPS observations and used different

approaches to predict IMU errors and correct IMU positioning. Second, we used GPS observations and calculated IMU errors. The calculated IMU errors using GPS observations were utilized to correct IMU positioning and we used it as baseline to evaluate corrected IMU positioning without using GPS observations.

Figure 5. The trajectory of the KITTI dataset, #34. It is the longest KITTI dataset where the vehicle travels more than 1.7 km.

5. Results

In order to evaluate the results, we divided trajectory into two parts: one for training and calibration and one for testing and evaluation. In the training phase, the parameters of the system model were estimated in the extended Kalman filter, as described in the appendix. In the deep extended Kalman filter, IMU errors were modelled in addition to the prediction and update stages. In other words, we trained our network to predict the posterior estimation of IMU errors using a recurrent neural network (RNN) and consequently, model IMU errors. When the network was trained, we anticipated that we would be able to predict the posterior estimation of IMU errors and correct IMU positioning.

In the testing part, we estimated IMU errors using GNSS observations and corrected IMU positioning. We utilized the posterior estimation of the state vector as the ground truth. The extended Kalman and deep extended Kalman filters were applied to predict IMU errors without using GNSS observations and to correct IMU positioning. Corrected IMU positioning using these two approaches was compared with the ground truth and the IMU positioning error was calculated. The Root Mean Square Error (RMSE) of the IMU positioning was applied to evaluate the results of the extended Kalman filter and deep extended Kalman filter.

In the first experiment, we used a simple RNN to predict IMU errors and removed them from the IMU positioning. In this experiment, we studied the performance of the extended Kalman filter and deep extended Kalman filter using different sequence lengths. In other words, the sequences of the IMU errors, utilized to predict IMU errors in the RNN, had different lengths. We plotted the RNN for the sequence lengths of 10, 20, and 50, as shown in Figure 6.

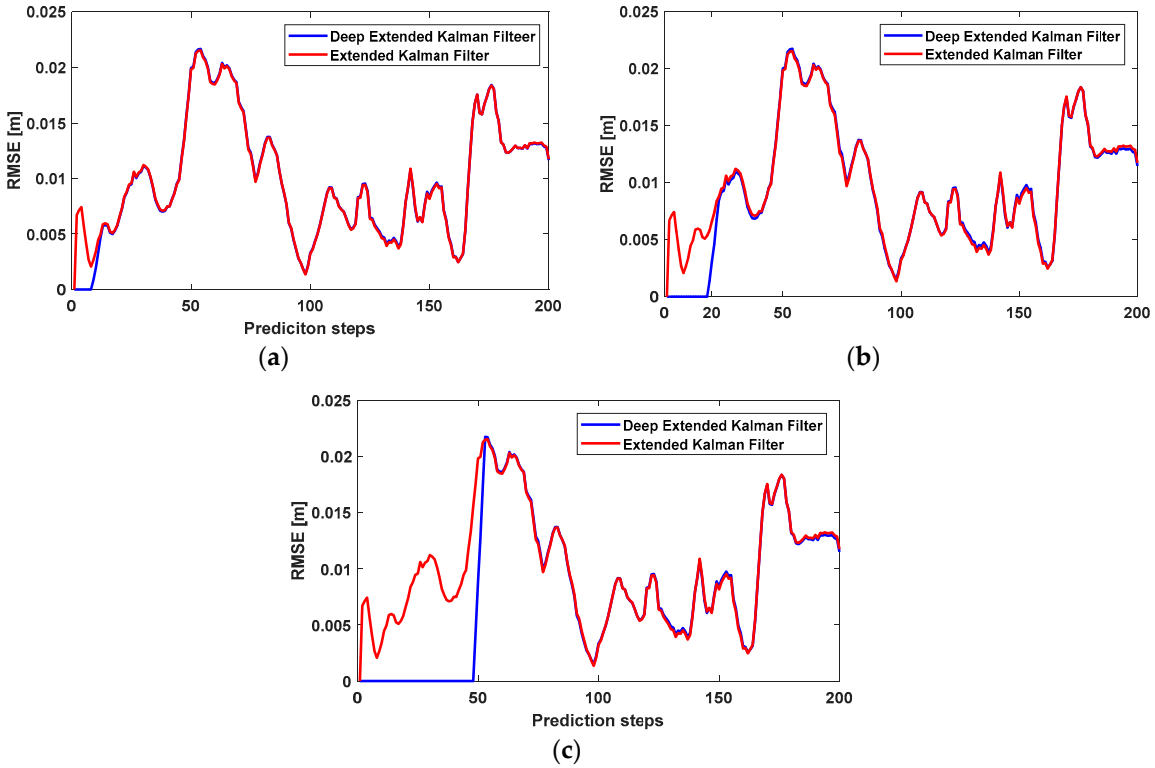

Figure 6. The RMSE of the deep extended Kalman filter and extended Kalman filter [21]. We used different sequence lengths of a simple recurrent neural network (RNN) for the IMU modelling of the deep extended Kalman filter. (**a**) Simple RNN with a sequence length of 10; (**b**) simple RNN with a sequence length of 20; (**c**) simple RNN with a sequence length of 50.

Figure 6 shows that the performance of RNN depends on the sequence length of previous state vectors. If the sequence length is large, the network uses more IMU errors in the previous time. In other words, the network applies more heuristics to predict IMU errors and remove them from IMU positioning. Therefore, the RMSE of the deep extended Kalman filter was less than the RMSE of the extended Kalman filter, at earlier times.

Despite the effectiveness of RNN in the prediction of IMU errors over short periods, it cannot predict IMU errors over a long period of time. The RMSE of the deep extended Kalman filter was lower than the RMSE of the extended Kalman filter at earlier times, but the deep extended Kalman filter lost its effectiveness and the two approaches had the same RMSE over a longer period of time. This effect was due to the vanishing-exploding gradient problem of RNN. This well-known problem of RNN has been overcome by the development of Long Short-Term Memory (LSTM). LSTM uses memory gates to remember previous IMU errors and prevents repetitive gradient multiplications.

Figure 7 shows IMU positioning using the LSTM network, with a sequence length of 10. In other words, instead of using RNN to predict IMU errors, we used LSTM to predict IMU errors and remove them from IMU positioning. The sequence length of this network was similar to Figure 6a. However, the effectiveness of this network did not disappear over a longer time period. As shown in Figure 7, the deep extended Kalman filter outperformed the extended Kalman filter the majority of the time. There were instances where the extended Kalman filter had better accuracy than the deep extended Kalman filter, but these were limited to a few short instances. The total RMSE of the deep extended Kalman filter was 0.0100 m and the total RMSE of the extended Kalman filter was 0.0114 m for 20 s. In summary, the deep extended Kalman filter had better accuracy compared to the extended Kalman filter.

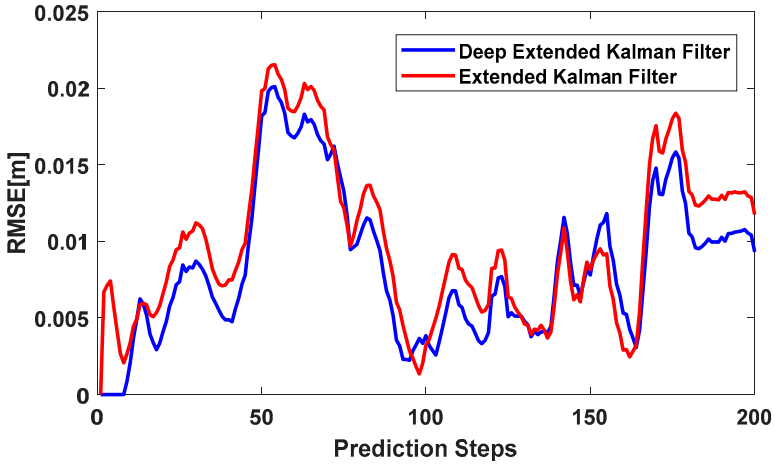

Figure 7. RMSE of the deep extended Kalman filter and extended Kalman filter. The deep extended Kalman filter IMU modelling was based on LSTM with a sequence length of 10.

Zhang et al. [22] showed that the computational complexity of simple RNN depends on the depth of the network and the sequence length of the heuristics. Our RNN model had one layer and, roughly speaking, the complexity of the network was linearly proportional to the sequence length of the RNN for the one-layer network. In other words, training the network with a sequence length of 50 was five times slower than training the same network with a sequence length of 10. We experienced a little overhead for longer networks. LSTM is slow, therefore there are a few variants of LSTM, aimed to speed up its training step. We utilized the fast LSTM algorithm provided by Torch library in this paper.

6. Conclusions

In this paper, we introduced the deep Kalman filter to learn the system model of the Kalman filter. In addition to the prediction and update steps of the Kalman filter, we added a modelling step to the deep Kalman filter. We applied the deep Kalman filter to model IMU errors and correct IMU positioning. In the deep Kalman filter, the model of IMU errors was learned using the Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) methods when GNSS observations were available. IMU errors could be predicted using the learned model in the absence of GNSS observations.

We compared the Kalman and deep Kalman filters using the KITTI dataset. The results showed that our approach outperformed the conventional Kalman filter and a higher level of accuracy was achieved using the deep Kalman filter. In addition, the deep Kalman filter using the RNN method predicted IMU errors over short time periods, but its effectiveness disappeared over longer time periods due to the vanishing-exploding gradient problem. Employing the deep Kalman filter based on the Long Short-Term Memory (LSTM) method predicted IMU errors over longer periods of time and outperformed the Kalman filter.

Conflicts of Interest: The author declared no conflict of interest.

Appendix A. System and Observation Models of Kalman Filter

The applied IMU model in the Kalman filter, described in this paper is as follows. The state vector is:

$$\boldsymbol{x}_t = \left[\delta \boldsymbol{p}^T \; \delta \boldsymbol{v}^T \; \delta \boldsymbol{\theta}^T \; \boldsymbol{b}_{acc}{}^T \; \boldsymbol{b}_{gyro}{}^T\right]_t^T \tag{A1}$$

where $\delta \boldsymbol{p}$ is the IMU positioning error, $\delta \boldsymbol{v}$ is the IMU velocity error, $\delta \boldsymbol{\theta}$ is the IMU orientation error, and $\boldsymbol{b}_{acc}$ and $\boldsymbol{b}_{gyro}$ are accelerometer and gyroscope bias. The system model is similar to [3], as follows:

$$F = \begin{bmatrix} \mathbf{0} & \boldsymbol{D} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{F} & \mathbf{0} & \boldsymbol{R} \\ \mathbf{0} & \boldsymbol{K} & \mathbf{0} & \boldsymbol{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\boldsymbol{\beta}_{acc} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\boldsymbol{\beta}_{gyro} \end{bmatrix} \tag{A2}$$

where $\mathbf{0}$ is a 3×3 zero matrix. $\boldsymbol{D} = \begin{bmatrix} 0 & \frac{1}{R_M+h} & 0 \\ \frac{1}{(R_N+h)cos\varphi} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is the transformation between a geographic coordinate system to a linear coordinate system, where h is the ellipsoidal height and R_M and R_N are the curvature radiuses of the meridian and prime vertical. $\boldsymbol{F} = \begin{bmatrix} 0 & f_u & -f_n \\ -f_u & 0 & f_e \\ f_n & -f_e & 0 \end{bmatrix}$ is a skew-symmetric matrix, where its components are acceleration in east, north, and up. The rotation matrix is shown by $\boldsymbol{R}$; $\boldsymbol{\beta}_{acc}$ and $\boldsymbol{\beta}_{gyro}$ are the coefficients of the first-order Gauss–Markov model for the accelerometer and gyroscope bias; and $\boldsymbol{D} = \begin{bmatrix} 0 & \frac{1}{R_M+h} & 0 \\ \frac{1}{R_N+h} & 0 & 0 \\ \frac{\tan\varphi}{R_N+h} & 0 & 0 \end{bmatrix}$, where φ is the latitude of the IMU. When GNSS positioning is available, the observation vector is as follows:

$$\delta \boldsymbol{p}_{obs} = \boldsymbol{p}_{GNSS} - \boldsymbol{p}_{IMU} \tag{A3}$$

where $\boldsymbol{p}_{GNSS}$ is the GNSS positioning and $\boldsymbol{p}_{IMU}$ is the IMU positioning, estimated by IMU mechanization. The observation model is calculated, such that:

$$G = \begin{bmatrix} \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{A4}$$

where $\boldsymbol{I}$ is 3×3 identity and $\mathbf{0}$ is 3×3 zero matrices.

References

1. Bochkati, M.; Schön, S.; Schlippert, D.; Schubert, C.; Rasel, E. Could cold atom interferometry sensors be the future inertial sensors?—First simulation results. In Proceedings of the 2017 DGON Inertial Sensors and Systems (ISS), Karlsruhe, Germany, 19–20 September 2017; pp. 1–20.
2. Jekeli, C. *Inertial Navigation Systems with Geodetic Applications*; Walter de Gruyter: Berlin, Germany, 2001.
3. Noureldin, A.; Karamat, T.B.; Georgy, J. *Fundamentals of Inertial Navigation, Satellite-Based Positioning and Their Integration*; Springer: Berlin/Heidelberg, Germany, 2013.
4. Shin, E.-H.; El-Sheimy, N. A new calibration method for strap down inertial navigation systems. *Z. Vermess* **2002**, *127*, 1–10.
5. Wang, S.; Deng, Z.; Yin, G. An Accurate GPS-IMU/DR Data Fusion Method for Driverless Car Based on a Set of Predictive Models and Grid Constraints. *Sensors* **2016**, *16*, 280. [CrossRef] [PubMed]
6. Chiang, K.W.; Noureldin, A.; El-Sheimy, N. Constructive Neural-Networks-Based MEMS/GPS Integration Scheme. *IEEE Trans. Aerosp. Electron. Syst.* **2008**, *44*, 582–594. [CrossRef]
7. Noureldin, A.; El-Shafie, A.; Bayoumi, M. GPS/INS Integration Utilizing Dynamic Neural Networks for Vehicular Navigation. *Inf. Fusion* **2011**, *12*, 48–57. [CrossRef]
8. Abdel-Hamid, W.; Noureldin, A.; El-Sheimy, N. Adaptive Fuzzy Prediction of Low-Cost Inertial-Based Positioning Errors. *IEEE Trans. Fuzzy Syst.* **2007**, *15*, 519–529. [CrossRef]
9. Zhang, L.; Liu, J.; Lai, J.; Xiong, Z. Performance Analysis of Adaptive Neuro Fuzzy Inference System Control for MEMS Navigation System. *Math. Probl. Eng.* **2014**, *2014*, 961067. [CrossRef]
10. Toth, C.; Grejner-Brzezinska, D.A.; Moafipoor, S. Pedestrian Tracking and Navigation Using Neural Networks and Fuzzy Logic. In Proceedings of the 2007 IEEE International Symposium on Intelligent Signal Processing, Alcala de Henares, Spain, 3–5 October 2007; pp. 1–6.

11. Navidi, N.; Landry, R., Jr.; Cheng, J.; Gingras, D. A New Technique for Integrating MEMS-Based Low-Cost IMU and GPS in Vehicular Navigation. *J. Sens.* **2016**, *2016*, 5365983. [CrossRef]
12. Xing, H.; Hou, B.; Lin, Z.; Guo, M. Modeling and Compensation of Random Drift of MEMS Gyroscopes Based on Least Squares Support Vector Machine Optimized by Chaotic Particle Swarm Optimization. *Sensors* **2017**, *17*, 2335. [CrossRef] [PubMed]
13. Minsky, M.L.; Papert, S.A. *Perceptrons: Expanded Edition*; MIT Press: Cambridge, MA, USA, 1988.
14. Mirowski, P.; Lecun, Y. Dynamic Factor Graphs for Time Series Modelling. In Proceedings of the Machine Learning and Knowledge Discovery in Databases—European Conference, Bled, Slovenia, 7–11 September 2009; Volume 5782, pp. 128–143.
15. Gu, J.; Yang, X.; De Mello, S.; Kautz, J. Dynamic Facial Analysis: From Bayesian Filtering to Recurrent Neural Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1531–1540.
16. Krishnan, R.G.; Shalit, U.; Sontag, D. Structured Inference Networks for Nonlinear State Space Models. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
17. Ghahramani, Z.; Roweis, S.T. Learning nonlinear dynamical systems using an EM algorithm. In *Advances in Neural Information Processing Systems 11*; MIT Press: Cambridge, MA, USA, 1999; pp. 431–437.
18. Goodfellow, I.; Bengio, I.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
19. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
20. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision Meets Robotics: The KITTI Dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
21. Hosseinyalamdary, S.; Balazadegan, Y. Error Modeling of Reduced IMU using Recurrent Neural Network. In Proceedings of the Indoor Positioning and Indoor Navigation (IPIN), Sapporo, Japan, 18–21 September 2017.
22. Zhang, S.; Wu, Y.; Che, T.; Lin, Z.; Memisevic, R.; Salakhutdinov, R.; Bengio, Y. Architectural complexity measures of recurrent neural networks. In *Advances in Neural Information Processing Systems 29*; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 1830–1838.

© 2018 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).