Practical Implementation of KalmanNet for Accurate Data Fusion in Integrated Navigation

Jian Song, Wei Mei, Yunfeng Xu, Qiang Fu, and Lina Bu

***Abstract*—The extended Kalman filter has been widely used in sensor fusion to achieve integrated navigation and localization. Efficiently integrating multiple sensors requires prior knowledge about their errors for setting the filter. The recently emerged KalmanNet managed to use recurrent neural networks to learn prior knowledge from data and carry out state estimation for problems under non-linear dynamics with partial information. In this letter, the KalmanNet is implemented for integrated navigation using data from GPS/Wheels and the Inertial Measurement Unit. Therein, a practical strategy for the training algorithm of truncated backpropagation through time is presented by taking advantage of the first-order Markov property of the system state of the Kalman filter, which improves the training robustness and performance of the existing KalmanNet. Experimental results on the Michigan NCLT dataset show that our fusion KalmanNet significantly outperforms the conventional EKF-based fusion algorithm with an improvement of 20% $\sim$ 40% in average RMSE.**

***Index Terms*—Integrated navigation and localization, Kalman filter, recurrent neural networks, sensor fusion.**

I. Introduction

NAVIGATION and localization systems are widely used in various fields, such as aerospace, smart agricultural, and vehicle navigation, and have attracted intensive research in recent years [1], [2], [3], [4], [5], [6]. For vehicle navigation, various measurement systems, e.g., Inertial Measurement Unit (IMU), Wheel Encoders (Wheels), and GPS are often utilized to achieve integrated navigation with enhanced performance. Each kind of sensor has its distinctive characteristics. GPS generally has an accurate performance but can degrade in circumstances with mask effects and multipath [7]; IMU and Wheels are a class of dead reckoning sensors that can provide relative position of a vehicle, but their positioning errors drift with time [8].

In order to achieve integrated navigation, the extended Kalman filter (EKF) is often used as the mainstream technology to fuse information from a group of sensors [1], [2], [9], [10], [11], [12]. The EKF-based fusion algorithm requires that the setting of the filter parameters should be accurate for the noise distribution of the sensors. In [13], the authors showed that an incorrect filter parameter could cause the navigation solution, including attitude, to diverge rapidly.

Recently, the technique of data-aided filtering, e.g., KalmanNet [14], [15], has emerged as a kind of efficient method for handling estimation problems under non-linear dynamics with partial information. The overall structure of the KalmanNet is similar to that of the Kalman filter (KF). The only difference is that KalmanNet exploits recurrent neural networks (RNNs) instead of analytic models to learn the Kalman Gain (KG) from input features, such as measurement residuals, thereby obtaining the capability to overcome model mismatch and nonlinearity. Nevertheless, it has been observed in our work that KalmanNet may diverge rapidly in training and lead to **Not-a-Number (NaN)** when model mismatch is severe.

In this letter, our main contributions include two points: (1) We successfully implement KalmanNet and its variant for sensor fusion with different data rates. (2) Based on the first-order Markov property of the system state of KF, we proposed a training method more applicable to KalmanNets to enhance their training stability. Experimental results on Michigan NCLT dataset [16] show that the implemented KalmanNet significantly outperforms the conventional EKF-based fusion algorithm with an improvement of 20% $\sim$ 40% in average RMSE, and the training method we employed can reduce the training difficulty of existing KalmanNets in model mismatch scenarios while enhancing their performance.

The rest of this letter is arranged as follows. Section II introduces the sensor fusion task and our proposed method. Section III gives the experimental results on the real-world dataset. Conclusions are presented in Section IV.

II. Methods

A. System Model

Consider a discrete-time non-linear state space model represented via

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{w}_t, \qquad \mathbf{x}_t \in \mathbb{R}^m, \tag{1a}$$

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{v}_t, \qquad \mathbf{y}_t \in \mathbb{R}^n. \tag{1b}$$

In (1a), $\mathbf{x}_t$ and $\mathbf{u}_t$ are the state vector and control input at time t. $\mathbf{w}_t$ is a sequence of zero-mean Gaussian of assumed known covariance matrix $\mathbf{Q}_t = \mathbb{E}[\mathbf{w}_t \mathbf{w}_t^\top]$. $\mathbf{f}(\cdot)$ is the transition function. In (1b), $\mathbf{y}_t$ is the measurement vector at time t, $\mathbf{v}_t$ is a sequence of zero-mean Gaussian of assumed known covariance matrix $\mathbf{R}_t = \mathbb{E}[\mathbf{v}_t \mathbf{v}_t^\top]$. $\mathbf{h}(\cdot)$ is the measurement function.

Sensor fusion can be viewed as a state estimation problem, which aims to recover the state $\mathbf{x}_t$ using the noisy measurements

Manuscript received 27 May 2024; accepted 11 July 2024. Date of publication 19 July 2024; date of current version 25 July 2024. The associate editor coordinating the review of this article and approving it for publication was Prof. Giuseppe Thadeu Freitas de Abreu. *(Corresponding author: Wei Mei.)*

The authors are with the Shijiazhuang Campus, Army Engineering University of PLA, Shijiazhuang 050003, China (e-mail: songj9507@aeu.edu.cn; meiwei@sina.com; hbkd_xyf@hebust.edu.cn; Fu_Qiang@aeu.edu.cn; bulina90@aeu.edu.cn).

The source code: https://github.com/SongJgit/KalmanNet4SensorFusion.

Digital Object Identifier 10.1109/LSP.2024.3431443

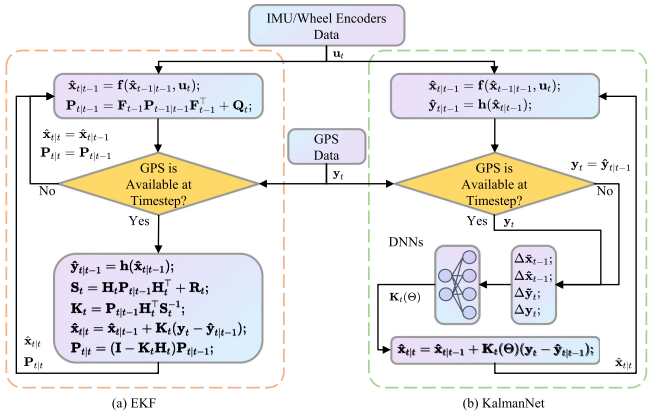

Fig. 1. Flow chart of the EKF/KalmanNet for sensor fusion.

from multiple sensors with different data rates up to time t. The EKF algorithm is a common solution to this problem [1], [2], [5], [13]. Its successful implementation, however, depends on accurate statistical information of sensors, such as $\mathbf{Q}$ and $\mathbf{R}$.

B. Extended Kalman Filter for Sensor Fusion

We consider an EKF-based, loosely coupled sensor fusion algorithm [1], [2], [12], [13], [17], which is used to fuse data from sensors with different data rates and then estimate the current position information. Given the model described by (1), this algorithm can be defined as two stages: *prediction* and *correction*. As shown in Fig. 1(a), the two stages correspond to different sensors. The *prediction stage* usually models sensors with high update frequency but low accuracy, such as IMU/Wheel Encoders, while the *correction stage* models sensors with lower update frequency but higher accuracy, such as GPS. This loosely coupled strategy ensures stable position estimates when the GPS signal is lost.

1) Prediction Stage: As shown in Fig. 1(a), the EKF predicts state vector $\hat{\mathbf{x}}_{t|t-1}$ and its covariance $\mathbf{P}_{t|t-1}$ using control input $\mathbf{u}_t$, previous state vector $\hat{\mathbf{x}}_{t-1|t-1}$ and its covariance $\mathbf{P}_{t-1|t-1}$, where the control input $\mathbf{u}_t$ comes from IMU/Wheel Encoders.

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{f}(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_t), \tag{2a}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_{t-1}\mathbf{P}_{t-1|t-1}\mathbf{F}_{t-1}^{\top} + \mathbf{Q}_t, \tag{2b}$$

where $\mathbf{F}_{t-1} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}|_{\mathbf{x}=\hat{\mathbf{x}}_{t-1|t-1}}$ is the Jacobian of $\mathbf{f}(\cdot)$ evaluated at $\hat{\mathbf{x}}_{t-1|t-1}$.

2) Correction Stage: If GPS measurement vector $\mathbf{y}_t$ is available, then the $\hat{\mathbf{x}}_{t|t}$ and $\mathbf{P}_{t|t}$ are computed via

$$\hat{\mathbf{y}}_{t|t-1} = \mathbf{h}(\hat{\mathbf{x}}_{t|t-1}), \tag{3a}$$

$$\mathbf{S}_t = \mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^{\top} + \mathbf{R}_t, \tag{3b}$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^{\top}\mathbf{S}_t^{-1}, \tag{3c}$$

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}), \tag{3d}$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_{t|t-1}, \tag{3e}$$

where $\mathbf{H}_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}|_{\mathbf{x}=\hat{\mathbf{x}}_{t|t-1}}$ is the Jacobian of $\mathbf{h}(\cdot)$ evaluated at $\hat{\mathbf{x}}_{t|t-1}$ and $\mathbf{S}_t$ is innovation covariance matrix, $\mathbf{K}_t$ is KG.

C. GPS-IMU/Wheels Data Fusion

1) Sensor Data: We unify the data of all sensors into a local coordinate frame aligned with cardinal directions, with px pointing north, py east, and pz down. We refer to [12], [16] for the detailed methods of coordinate transformation. Therefore, the state vector at time t can be defined as $\mathbf{x}_t = [px, py, vx, vy, \theta, \omega]^{\top}$, where $[px, py]^{\top}$ represents the position in the local coordinate frame, $[vx, vy]^{\top}$ are the corresponding velocity, $[\theta, \omega]^{\top}$ are the heading angle and angular velocity respectively. Here, we do not consider the z-axis for the sake of a simpler model.

2) State Model: We adopt the state model from [12], [16], which uses $[v_l, v_r, \theta, \omega]^{\top}$ as control input $\mathbf{u}$, where v_l and v_r represent the velocity of the left and right wheels, θ and ω are IMU's estimated heading and angular velocity. With the velocity of the left and right wheels, the velocity of the vehicle is calculated from

$$v_c = \frac{1}{2}(v_r + v_l). \tag{4}$$

Then the state model can be defined as

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{w}_t = \begin{bmatrix} px_{t-1} + v_c\cos\theta_t\Delta t \\ py_{t-1} + v_c\sin\theta_t\Delta t \\ v_c\cos\theta_t \\ v_c\sin\theta_t \\ \theta_t \\ \omega_t \end{bmatrix} + \mathbf{w}_t, \tag{5}$$

where Δt is the duration of the time step.

3) Measurement Model: Since the data from all sensors are harmonized into a local frame, measurements are expressed as a linear relation with respect to the state vectors:

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{v}_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}\mathbf{x}_t + \mathbf{v}_t. \tag{6}$$

D. KalmanNet for Sensor Fusion

We consider replacing EKF with KalmanNet [14] or its variant Split-KalmanNet [15] for sensor fusion applications. They are a class of hybrid methods that integrate deep neural networks (DNNs) into the KF while maintaining the core structure of the KF, enabling robust state estimation when there exists model mismatch.

The KalmanNet directly replaces the computation of $\mathbf{K}_t$ in (3) with a trainable DNN, and thus (2b) and (3b)-(3e) are replaced with

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\Theta)(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}), \tag{7}$$

where $\mathbf{K}_t(\Theta)$ is a DNN with trainable parameters Θ. The DNN, which consists of RNNs and fully connected (FC) layers, uses the following features as inputs at each time step to recursively compute $\mathbf{K}_t$ while implicitly tracking the covariance matrices $\mathbf{Q}$, $\mathbf{R}$ and $\mathbf{P}$.

$$F1: \Delta\tilde{\mathbf{x}}_t = \hat{\mathbf{x}}_{t|t} - \hat{\mathbf{x}}_{t-1|t-1}, \quad F2: \Delta\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t|t} - \hat{\mathbf{x}}_{t|t-1},$$
$$F3: \Delta\tilde{\mathbf{y}}_t = \mathbf{y}_t - \mathbf{y}_{t-1}, \quad F4: \Delta\mathbf{y}_t = \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}. \tag{8}$$

For $F1$ and $F2$, the available features at time t are $\Delta\tilde{\mathbf{x}}_{t-1}$ and $\Delta\hat{\mathbf{x}}_{t-1}$, respectively.

Algorithm 1: TBPTT for KalmanNet.

Training:
  while *not converged* **do**:
    Select randomly sequences to form batch;
    Init RNN hidden state;
    for t **from** 1 **to** D **do**:
      Run model for one step;
      if t **divides** k **then**:
        Detaches gradients;
      if t **divides** w **or** $t = D$ **then**:
        Run BPTT;

Split-KalmanNet is an alternative architecture that is more robust in model mismatch scenarios, where two RNNs were used to track the covariance matrices $\mathbf{P}_{t|t-1}$ and $\mathbf{S}_t$, respectively, which allows to compute the KG via (3c).

It should be noted that for both KalmanNet and Split-KalmanNet, measurement $\mathbf{y}_t$ is not available at every moment in sensor fusion tasks. To make KalmanNet work in this case, we use $\mathbf{y}_t = \hat{\mathbf{y}}_{t|t-1}$ to compute *F3* and *F4*. Fig. 1(b) shows the flow chart of the KalmanNet for sensor fusion.

E. Practical Training Strategy for KalmanNet

We use a practical training strategy for our implementation of KalmanNet, which effectively improves its training robustness and brings additional performance improvements. KalmanNet is typically trained using the truncated BPTT (TBPTT) [18], which splits a sequence of total length T into M sequences of length D and treats each sequence of length D as an independent training sample [19]. This algorithm is more like truncated "sequences" BPTT, which works well in most cases. However, this algorithm may put the KalmanNet into a dilemma: (1) With a large D, the potential problem of gradient explosion in KalmanNet, which is caused by severe model mismatch, may become much more serious and lead to **NaN**. (2) With a small D, it limits the temporal scale over which the network can learn.

In our implementation of KalmanNet, such a problem will be alleviated. Besides, our training strategy will also consider the first-order Markov property of the system state $\mathbf{x}_t$. As indicated by (1a), the state $\mathbf{x}_t$ only depends on the previous state $\mathbf{x}_{t-1}$. Therefore, KalmanNet needs only a short interaction range to be sufficient to capture the dependencies in the actual sequence. Based on this consideration, we suggest an alternative TBPTT algorithm, which can be denoted as *TBPTT*(k, w, D). As shown in Algorithm 1, this method looks like a sliding window algorithm with window size w and step size w. It performs BPTT every w timesteps on the sequence D and detaches the gradient relevant with the hidden state h of the RNN every k timesteps within each window. It limits the gradient of the parameter to k time steps. When $k = w = D$, it will be consistent with the truncated "sequences" BPTT mentioned above.

There are several advantages to doing this:

1) Updating the weights multiple times per sequence (i.e., $w < D$) alleviates KalmanNet's training problems caused by the model mismatch. It enables us to employ a comparatively larger D, thereby augmenting the temporal scale over which the network can learn.
2) Each update of the weights relies only on the gradient generated at the last few time steps(e.g., $k = 2$), which makes the model focus primarily on input information from the most recent time steps.
3) Since $D \geq k$ and $D \geq w$, the network's memory capacity is not limited by k and w. On the contrary, appropriate k and w can mitigate the gradient vanishing and utilize the input information more thoroughly.

TABLE I
RMSE IN METERS FOR DIFFERENT METHODS ON TEST DATASET

Method \ Traj. Date	2012 11-04	2012 11-16	2013 04-05	Avg. of RMSE
Traj. Length[sec]	4834	4917	4182	-
GPS-Only	46.14	19.50	34.53	33.39
Wheels-Only EKF	117.76	81.61	83.99	94.46
Wheels-with-GPS EKF	18.76	12.29	8.94	13.33
Wheels-with-GPS KalmanNet[a]	**12.46**	**5.60**	5.98	**8.01**
Wheels-with-GPS Split-KalmanNet[b]	13.84	7.12	**5.36**	8.77

[a] Trained on *TBPTT*$(2, 4, 50)$ setting.
[b] Trained on *TBPTT*$(2, 8, 100)$ setting.
Bold values represent the best significant results.

III. EXPERIMENTAL RESULTS

We use the Michigan NCLT dataset to evaluate the performance of KalmanNet and its variant in sensor fusion.

A. Experiment Setup

1) Dataset Description: The NCLT dataset consists of data collected by the Segway Robotics Platform. It contains 27 trajectories, each of which provides data from sensors such as Consumer-grade GPS (5 Hz), Wheel Encoders (37 Hz), and IMU (47 Hz), as well as Ground Truth (100 Hz) [12], [16].

2) Data Processing: The 27 trajectories were randomly divided into three sections: 22 for training, 2 for validation, and 3 for testing. After sampling at 1 Hz, the average length of the training, validation, and testing trajectories are 5040, 2917, and 4644, respectively.

3) Compared Models:

- EKF: Follow the setting in [12], which is derived from the sensor's parameters.
- KalmanNet and Split-KalmanNet: The model and training strategy are discussed in Sections II-D and II-E.

4) Hyper-Parameter Setting: In our experiments, KalmanNet and its variant were trained by the same hyperparameter, *epoch*, *batch size*, and *learning rate* were 50, 256, and 0.001, respectively. To verify the effectiveness of Algorithm 1 on KalmanNets, we trained it on multiple settings of $k = 2$, $w \in \{4, 8\}$, $D \in \{50, 100, 200\}$.

5) Evaluation Metrics: We use root mean square error (RMSE) for a certain trajectory and average RMSE for all trajectories of the testing dataset.

B. Experimental Results

Table I gives the comparison outcomes of EKF and KalmanNets on the test dataset, whereas Table II illustrates the performance of KalmanNet and Split-KalmanNet with different

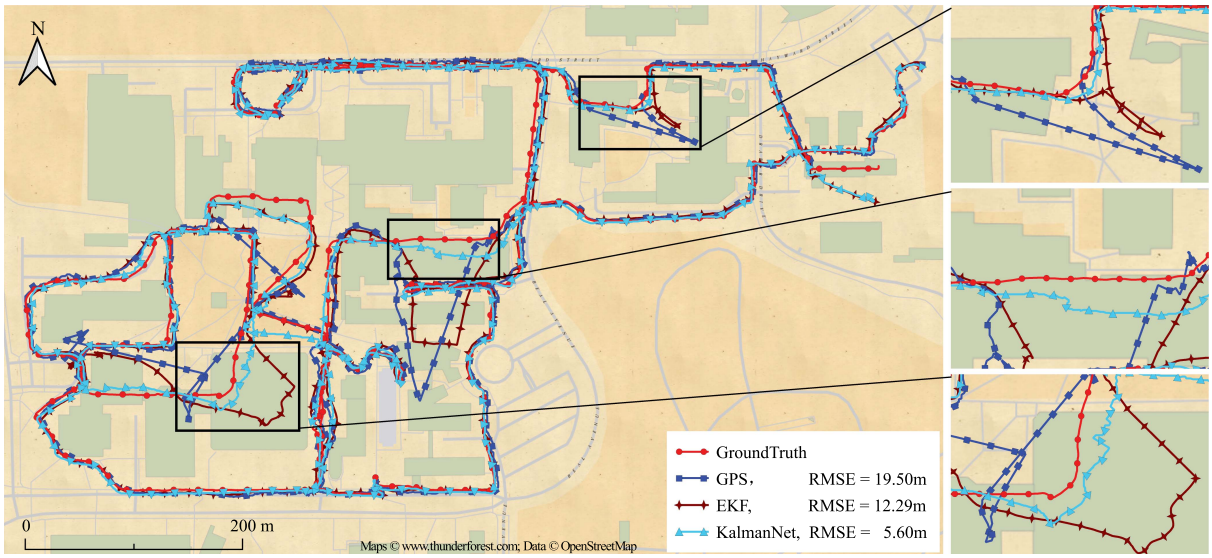

Fig. 2. This figure was generated by overlaying the estimates of all algorithms and ground truth positions onto a map. The trajectory in the test dataset was collected on 2012-11-16, and the length is 4917 after sampling at 1 Hz. Both EKF and KalmanNet used the "Wheels-with-GPS" mode. KalmanNet was trained on $TBPTT(2, 4, 50)$.

TABLE II
AVERAGE RMSE IN METERS FOR DIFFERENT TBPTT SETTINGS ON TEST DATASET

Method	D \ (k,w)	(2,4)	(2,8)	$(D, D)^*$
Wheels-with-GPS KalmanNet	50	**8.01**	9.00	11.12
	100	**8.62**	9.20	**NaN**
	200	10.59	**10.45**	**NaN**
Wheels-with-GPS Split-KalmanNet	50	**9.62**	9.97	17.91
	100	9.86	**8.77**	9.96
	200	10.08	**9.61**	10.05

* (D, D) means $k = w = D$, the original *TBPTT* in [14].
Bold values represent the best significant results.

settings of $TBPTT(k, w, D)$. "GPS-Only" means that the position is given by using the raw GPS measurements. "Wheels-Only" uses only the data of Wheel Encoders to perform the *prediction* without any *correction* from the GPS. "Wheels-with-GPS" represents the position estimated using the *prediction* and *correction*.

From Table I, we can observe that the KalmanNet possesses a lower average RMSE of 8.01 m, with improvements of 75% to "GPS-Only" and 39% to "Wheels-with-GPS EKF". Although Split-KalmanNet is slightly behind KalmanNet, it still has 73% and 34% improvements to "GPS-Only" and "Wheels-with-GPS EKF" in terms of average RMSE. Fig. 2 shows the results of KalmanNet, EKF, and GPS, as well as the ground truth of the 2012-11-16 test path. As we can see, for the integrated navigation task, the KalmanNet can reduce the estimation error of the EKF from 12.29 m to 5.60 m. The magnified view on the right of Fig. 2 shows that the advantage of the KalmanNet to EKF is particularly significant when the robot is navigating indoors and the GPS signal is generally not available. The experiments verified the necessity of sensor fusion and the outstanding performance of KalmanNet in sensor fusion.

As we can see from Table II, the KalmanNet and Split-KalmanNet trained with our strategy outperform the original versions $(k = w = D)$ for $D = 50$, 100, and 200. Specifically, our method is now competent at training a long trajectory. For example, with a large $D = 100$ or 200, KalmanNet has become difficult to train using the original setting of TBPTT due to **NaN** caused by the model mismatch. In contrast, KalmanNet still performs well when trained with $w \in \{4, 8\}$. Meanwhile, with a small $D = 50$, the average RMSE of KalmanNet and Split-KalmanNet descend to 8.01 m and 9.62 m, respectively, which improves 28% and 46% to $TBPTT(50, 50, 50)$. These experiments suggest the effectiveness of our training strategy in reducing the training difficulty of existing KalmanNets and improving their performance. Such improvements are facilitated by the sliding window mechanism and the k-step gradient detach, as elaborated in Section II-E. The adoption of a sliding window augments the temporal scale over which the network can learn. Successful applications on a smaller $k = 2$ show that KalmanNets require only short-term interactions to be sufficient to capture dependencies in sequences that are governed by the first-order Markov property of the system state of KF.

IV. CONCLUSION

In this letter, we proposed a training method more applicable to KalmanNets to enhance their training stability and performance. This method uses a sliding window to update RNN weights multiple times per sequence, which augments the temporal scale over which the network can learn. Considering the Markov property of the system state, each update of the weights relies only on the gradient generated at the last few time steps. Experimental results on the NCLT real dataset demonstrate the effectiveness of our method for the task of integrated navigation.

REFERENCES

[1] K. Saadeddin, M. F. Abdel-Hafez, and M. A. Jarrah, "Estimating vehicle state by GPS/IMU fusion with vehicle dynamics," *J. Intell. Robot. Syst.*, vol. 74, no. 1/2, pp. 147–172, Apr. 2014.

[2] B. Suwandi, T. Kitasuka, and M. Aritsugi, "Low-cost IMU and GPS fusion strategy for apron vehicle positioning," in *Proc. IEEE Region 10 Conf.*, Nov. 2017, pp. 449–454.

[3] N. Chebrolu, P. Lottes, A. Schaefer, W. Winterhalter, W. Burgard, and C. Stachniss, "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1045–1052, Sep. 2017.

[4] Y. Liu, X. Fan, C. Lv, J. Wu, L. Li, and D. Ding, "An innovative information fusion method with adaptive Kalman filter for integrated INS/GPS navigation of autonomous vehicles," *Mech. Syst. Signal Process.*, vol. 100, pp. 605–616, Feb. 2018.

[5] N. A. Anbu and D. Jayaprasanth, "Integration of inertial navigation system with global positioning system using extended Kalman filter," in *Proc. IEEE Int. Conf. Smart Syst. Inventive Technol.*, Nov. 2019, pp. 789–794.

[6] F. Wu, H. Luo, H. Jia, F. Zhao, Y. Xiao, and X. Gao, "Predicting the noise covariance with a multitask learning model for Kalman filter-based GNSS/INS integrated navigation," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, Art. no. 8500613.

[7] A. Poulose, O. S. Eyobu, and D. S. Han, "An indoor position-estimation algorithm using smartphone IMU sensor data," *IEEE Access*, vol. 7, pp. 11165–11177, 2019.

[8] S. Campbell et al., "Sensor technology in autonomous vehicles : A review," in *Proc. IEEE 29th Ir. Signals Syst. Conf.*, Jun. 2018, pp. 1–4.

[9] L. Bento, U. Nunes, F. Moita, and A. Surrecio, "Sensor fusion for precise autonomous vehicle navigation in outdoor semi-structured environments," in *Proc. IEEE Intell. Transp. Syst.*, Sep. 2005, pp. 245–250.

[10] A. Poulose, B. Senouci, and D. S. Han, "Performance analysis of sensor fusion techniques for heading estimation using smartphone sensors," *IEEE Sensors J.*, vol. 19, no. 24, pp. 12369–12380, Dec. 2019.

[11] A. D. Kulkarni, G. G. Narkhede, and S. N. Motade, "Sensor fusion: An advance inertial navigation system using GPS and IMU," in *Proc. IEEE Int. Conf. Comput., Commun., Control Automat.*, Aug. 2022, pp. 1–5.

[12] A. Agrahari, A. Barlow, S. Naik, and S. Skarica, "Localization of mobile robot using classical Bayesian filtering and sensor data," 2023. [Online]. Available: https://github.com/AbhinavA10/mte546-project

[13] K. Park, W. Kim, and J. Seo, "Effects of initial attitude estimation errors on loosely coupled smartphone GPS/IMU integration system," in *Proc. IEEE Int. Conf. Control, Automat. Syst.*, Oct. 2020, pp. 800–803.

[14] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. G. Van Sloun, and Y. C. Eldar, "KalmanNet: Neural network aided Kalman filtering for partially known dynamics," *IEEE Trans. Signal Process.*, vol. 70, pp. 1532–1547, 2022.

[15] G. Choi, J. Park, N. Shlezinger, Y. C. Eldar, and N. Lee, "Split-KalmanNet: A robust model-based deep learning approach for state estimation," *IEEE Trans. Veh. Technol.*, vol. 72, no. 9, pp. 12326–12331, Sep. 2023.

[16] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset," *Int. J. Robot. Res.*, vol. 35, no. 9, pp. 1023–1035, Aug. 2016.

[17] F. Caron, E. Duflos, D. Pomorski, and P. Vanheeghe, "GPS/IMU data fusion using multisensor Kalman filtering: Introduction of contextual aspects," *Inf. Fusion*, vol. 7, no. 2, pp. 221–230, Jun. 2006.

[18] I. Sutskever, "Training recurrent neural networks," Ph.D. dissertation, University of Toronto, Toronto, ON, Canada, 2013.

[19] R. J. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Comput.*, vol. 2, no. 4, pp. 490–501, Dec. 1990.