TYPE Original Research
PUBLISHED 07 January 2025
DOI 10.3389/fnbot.2024.1460255

OPEN ACCESS

EDITED BY
Long Jin,
Lanzhou University, China

REVIEWED BY
Xu Zhang,
Boston Children's Hospital and Harvard Medical School, United States
Paolo Mercorelli,
Leuphana University Lüneburg, Germany

*CORRESPONDENCE
Zhengjun Zhai
zhaizjun@nwpu.edu.cn

RECEIVED 05 July 2024
ACCEPTED 10 December 2024
PUBLISHED 07 January 2025

CITATION
Shen S, Chen J, Yu G, Zhai Z and Han P (2025) KalmanFormer: using transformer to model the Kalman Gain in Kalman Filters. *Front. Neurorobot.* 18:1460255. doi: 10.3389/fnbot.2024.1460255

KalmanFormer: using transformer to model the Kalman Gain in Kalman Filters

Siyuan Shen[1], Jichen Chen[2], Guanfeng Yu[3], Zhengjun Zhai[1]* and Pujie Han[4]

[1]School of Computer Science, Northwestern Polytechnical University, Xi'an, China, [2]Fourth Technical Department, Xi'an Microelectronics Technology Institute, Xi'an, China, [3]Research Office 16, AVIC Xi'an Aeronautics Computing Technique Research Institute, Xi'an, China, [4]Software Engineering College, Zhengzhou University of Light Industry, Zhengzhou, China

Introduction: Tracking the hidden states of dynamic systems is a fundamental task in signal processing. Recursive Kalman Filters (KF) are widely regarded as an efficient solution for linear and Gaussian systems, offering low computational complexity. However, real-world applications often involve non-linear dynamics, making it challenging for traditional Kalman Filters to achieve accurate state estimation. Additionally, the accurate modeling of system dynamics and noise in practical scenarios is often difficult. To address these limitations, we propose the KalmanFormer, a hybrid model-driven and data-driven state estimator. By leveraging data, the KalmanFormer promotes the performance of state estimation under non-linear conditions and partial information scenarios.

Methods: The proposed KalmanFormer integrates classical Kalman Filter with a Transformer framework. Specifically, it utilizes the Transformer to learn the Kalman Gain directly from data without requiring prior knowledge of noise parameters. The learned Kalman Gain is then incorporated into the standard Kalman Filter workflow, enabling the system to better handle non-linearities and model mismatches. The hybrid approach combines the strengths of data-driven learning and model-driven methodologies to achieve robust state estimation.

Results and discussion: To evaluate the effectiveness of KalmanFormer, we conducted numerical experiments in both synthetic and real-world dataset. The results demonstrate that KalmanFormer outperforms the classical Extended Kalman Filter (EKF) in the same settings. It achieves superior accuracy in tracking hidden states, demonstrating resilience to non-linearities and imprecise system models.

KEYWORDS

Kalman Filter, deep learning, transformer, Kalman Gain, supervised paradigm

1 Introduction

It is the most fundamental task to track the hidden state of a dynamical system by using the noisy measurements in real-time in many fileds, including singal processing (Yadav et al., 2023), navigation (Hu et al., 2003), information fusion (Xu et al., 2004), and automation control (Menner et al., 2023; Mercorelli, 2012a). A large number of algorithms were proposed to stress this issue, such as Bayesian estimation (Coué et al., 2003) and particle filter (Hue et al., 2002).

Kalman Filter (KF) (Kalman, 1960) is also an efficient recursive filter that can track the state of dynamic systems from a series of incomplete measurements with additive white Gaussian noise (AWGN). Low complexity implementation of KF, combined with theoretical foundation, resulted in it quickly becoming the popular method for state estimation problems.

The original Kalman Filters perform well in linear and Gaussian systems. The reality is that many nonlinear phenomena are encountered in real-world multi-sensor systems. Therefore, several variants of Kalman Filters are available to meet the requirements of nonlinear dynamic systems, including Extended Kalman Filters (Maybeck, 1982) (EKF) and Unscented Kalman Filters (UKF) (Wan and Van Der Merwe, 2001).

There are still limitations associated with the application of EKF and UKF in practical applications. Specifically, the Kalman Filter is a model-based method, and the performance of state estimation heavily depends on model accuracy. Furthermore, the noise covariance matrix is determined by prior process noise and measurement noise, which are assumed to be Additive White Gaussian Noise (AWGN). Additionally, there is no guarantee that the AWGN will accurately reflect the actual performance of the information fusion.

Several variants of Kalman Filters were proposed to overcome the above issue. For example, Huang et al. (2020) introduced a sliding window variational adaptive Kalman filter to simultaneously modify the state estimation and covariance matrix. Yu and Li (2021) presented an adaptive Kalman Filter that concentrated on unknown covariances of both dynamic multiplicative noise and additive noises. Xiong et al. (2020) employed a parallel adaptive Kalman Filter to estimate the attitude of the vehicle based on the Inertial Measurement Unit (IMU). Paolo Mercorelli introduced (Mercorelli, 2012b) a combination of the augmented EKF and EKF for sensorless Valve Control which avoids complicated observation.

Recent years have seen the application of deep learning techniques to multiple real-world applications such as computer vision (Voulodimos et al., 2018) and natural language processing (Otter et al., 2020). Particularly, some Deep Neural Networks (DNNs), such as the Recurrent Neural Network (RNN) (Elman, 1990), Long Short-Term Memory Network (LSTM) (Hochreiter and Schmidhuber, 1997), Gated Recurrent Unit (GRU) (Chung et al., 2014), and Transformer (Vaswani et al., 2017), have demonstrated excellent performance when it comes to processing time series data. For example, Xia et al. (2021) presented staked GRU and RNN to predict the payload of electricity. Zhang et al. (2020) applied LSTM to estimate the battery's state of health.

Furthermore, deep learning techniques have been utilized by some researchers to enhance the effects of the Kalman Filters. For example, Rangapuram et al. (2018) introduced RNN to forecast the state space parameters of linear systems. Coskun et al. (2017) utilized LSTM to learn the noisy parameters and motion model of the Kalman filters. EKFNet (Xu and Niu, 2021) used BPTT (Ruder, 2016) to learn the process and measurement noise from the measurement. Bence Zsombor Hadlaczky applied neural networks and EKF to estimate the wing shape (Hadlaczky et al., 2023). Dahal et al. (2024) introduced RobustStateNet, which applied RNN and Kalman Filters to perform ego vehicle state estimation. Zhang et al. (2023) adopted the Transformer to pre-estimate the vehicle mass, thus acting as an observation for EKF. Luttmann and Mercorelli (2021) employed EKF to accelerate the convergence of the learning system.

In this work, we present KalmanFormer, a hybrid data-driven and model state estimator that can be used to perform information fusion in multi-sensor systems. Our KalmanFormer uses a Transformer framework to track the Kalman Gain instead of computing it from the statistic moments.

The structure of this paper is organized as follows: Section 2 introduces the Kalman Filters and Transformer architecture. Section 3 details the methodology of the proposed KalmanFormer. Experiments will be discussed in Section 4. Section 5 concludes the whole paper.

2 Preliminary knowledge

2.1 Kalman Filter

The Kalman Filter algorithm (KF) is a classic algorithm of information fusion technology and is widely used to solve various optimal estimation problems. The classic Kalman Filters are composed of a state transition model and an observation model, which are expressed as follows:

$$\begin{cases} x_k = \mathbf{F}_k x_{k-1} + \mathbf{B_k} u_{k-1} + w_{k-1} \\ z_k = \mathbf{H_k} x_k + v_k \\ w_{k-1} \sim \mathcal{N}(0, \mathbf{Q}_k) \\ v_k \sim N(0, \mathbf{R}_k) \end{cases} \tag{1}$$

where x_k is the state vector of the system, $\mathbf{F}_k$ represents the state transition matrix, $\mathbf{B}_k$ is the control-input model which is applied to the control vector u_{k-1}, and $\mathbf{H_k}$ represents the observation function, which maps the true state space into the observed space. w_{k-1} and v_k are process noise and observation noises respectively. Process noise is assumed to be drawn from a zero multivariate normal distribution $\mathcal{N}$ with covariance $\mathbf{Q_k}$. Observation noise is assumed to be zero mean Gaussian white noise with covariance $\mathbf{R_k}$.

In general, Recursive Kalman Filter can be divided into two steps: Prediction and Updation. The information flow of the Kalman Filter is shown in Figure 1. As shown in Figure 1, the predict step uses the state estimate from the previous timestep to produce an *priori* estimate of the state at the current timestep, which is expressed as follows:

$$\begin{aligned} \hat{x}_{k|k-1} &= \mathbf{F}_k \hat{x}_{k-1|k-1} + \mathbf{B} u_{k-1} \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q_{k-1}} \end{aligned} \tag{2}$$

In the update phase, the innovation between the current a *priori* estimation and the current observation information, is multiplied by the optimal Kalman gain and combined with the previous state estimate to optimal the state estimate. This improved estimate based on the current observation is termed a *posteriori* state estimate, which is summarized as follows:

$$\begin{aligned} \mathbf{K}_k &= \frac{\mathbf{P}_{k|k-1}\mathbf{H}_k^T}{\mathbf{H}_k \mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + \mathbf{K}_k (z_k - \mathbf{H}_k \hat{x}_{k|k-1}) \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \end{aligned} \tag{3}$$

Noise has a significant impact on the performance of a Kalman Filter system, as it directly affects the accuracy of the estimation. A Kalman Filter is designed to optimally combine observations and predictions in the presence of noise, which controls how the Kalman filter weights the model predictions versus the actual observations.

The process noise covariance $\mathbf{Q}$ represents the uncertainty in the model of the system dynamics. Higher values of $\mathbf{Q}$ means we have less reliability on the prediction and place more trust in the observations.

The observation noise covariance $\mathbf{R}$ means the uncertainty in the observations. Higher values of $\mathbf{R}$ suggest more noise in

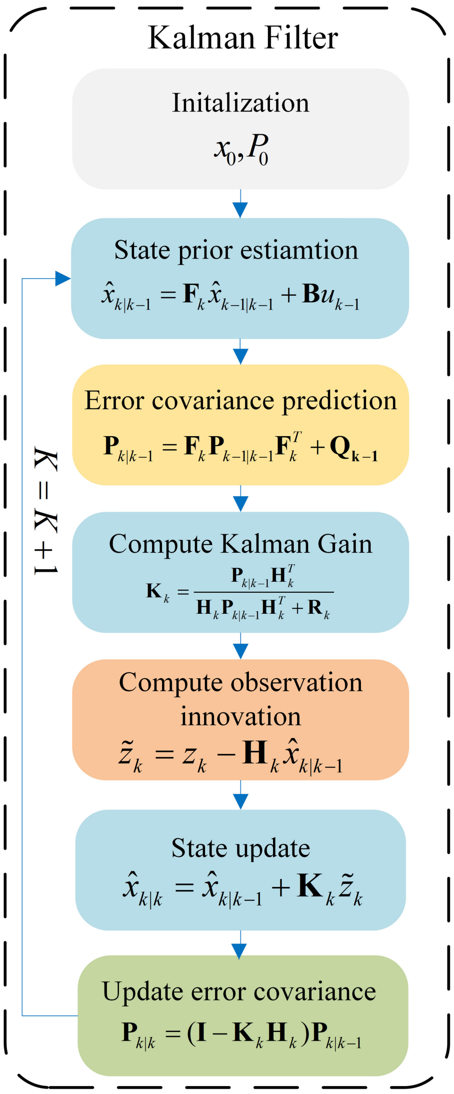
Kalman Filter
Initalization
x_0, P_0
State prior estiamtion
$\hat{x}_{k|k-1} = \mathbf{F}_k \hat{x}_{k-1|k-1} + \mathbf{B}u_{k-1}$
Error covariance prediction
$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_{k-1}$
Compute Kalman Gain
$\mathbf{K}_k = \frac{\mathbf{P}_{k|k-1}\mathbf{H}_k^T}{\mathbf{H}_k \mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k}$
Compute observation innovation
$\tilde{z}_k = z_k - \mathbf{H}_k \hat{x}_{k|k-1}$
State update
$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \mathbf{K}_k \tilde{z}_k$
Update error covariance
$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_{k|k-1}$
$K = K+1$

FIGURE 1
Information flow of the Kalman Filter.

the observation, so the Kalman Filter will pay more attention to its predictions.

The following tuning steps are necessary before using the Kalman Filters:

- Set initial state vector $\hat{x}_0$.
- Set initial noise values for **Q** and **R**.
- Tuning the Process Noise Covariance **Q**.
- Tuning the Observation Noise Covariance **R**.
- Test the performance and adjust **Q** and **R**.

Although the noise parameters are tuned before using the Kalman Filters. It is difficult to obtain an accurate state transition model and observation model, especially in the nonlinear occasion, which results in a significant degradation of Kalman Filter performance.

2.2 Extended Kalman Filters

Differentiable nonlinear functions may be used in place of the state transition and observation models in the extended Kalman Filter:

$$\begin{cases} \hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1}) + w_{k-1} \\ z_k = h(x_k) + v_k \\ w_{k-1} \sim \mathcal{N}(0, \mathbf{Q}_k) \\ v_{k-1} \sim \mathcal{N}(0, \mathbf{R}_k) \end{cases} \tag{4}$$

Similar to the linear Kalman Filter, x_k is the state vector of the system, w_{k-1} and v_k are process noise and observation noises respectively. Process noise is assumed to be drawn from a zero multivariate normal distribution $\mathcal{N}$ with covariance $\mathbf{Q_k}$. Observation noise is assumed to be zero mean Gaussian white noise with covariance $\mathbf{R_k}$.

Function f is used to predict the state from the previous estimation and function h is applied to produce the predicted measurement form the predicted state. Different from the linear Kalman Filter, the Jacobian of f and h are used to compute the covariance matrix in extended Kalman Filters.

At timestamp k, the Jacobian is evaluated with the current predicted states, thus it can be used in Kalman equations. The prediction procedure of EKF is presented as follows:

$$\begin{aligned} \hat{x}_{k|k-1} &= f(\hat{x}_{k-1|k-1}, u_{k-1}) \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q_{k-1}} \end{aligned} \tag{5}$$

The update procedure of EKF is calculated as follows:

$$\begin{aligned} \mathbf{K}_k &= \frac{\mathbf{P}_{k|k-1}\mathbf{H}_k^T}{\mathbf{H}_k \mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + \mathbf{K}_k (z_k - h(\hat{x}_{k|k-1})) \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_{k|k-1} \end{aligned} \tag{6}$$

where the state transition function and observation model are defined as the following Jacobians:

$$\begin{aligned} \mathbf{F}_k &= \frac{\partial f}{\partial x}\Big|_{\hat{x}_{k-1|k-1}} \\ \mathbf{H}_k &= \frac{\partial h}{\partial x}\Big|_{\hat{x}_{k|k-1}} \end{aligned} \tag{7}$$

2.3 Transformer

2.3.1 Transformer architecture

Transformer (Vaswani et al., 2017) was originally proposed in natural language processing and it has been applied in various sequence-to-sequence tasks. As shown in Figure 2, the Transformer is mainly composed of encoders and decoders with several basic transformer blocks. Transformer blocks inside the encoders and decoders remain in the same structure.

Encoders produce encodings for the input sequence, while the decoders take all the encodings from encoders and use contextual information to generate the prediction results. Each transformer block is composed of a multi-head attention layer, a feed-forward neural network, a skip connection connection, and a layer normalization operation.

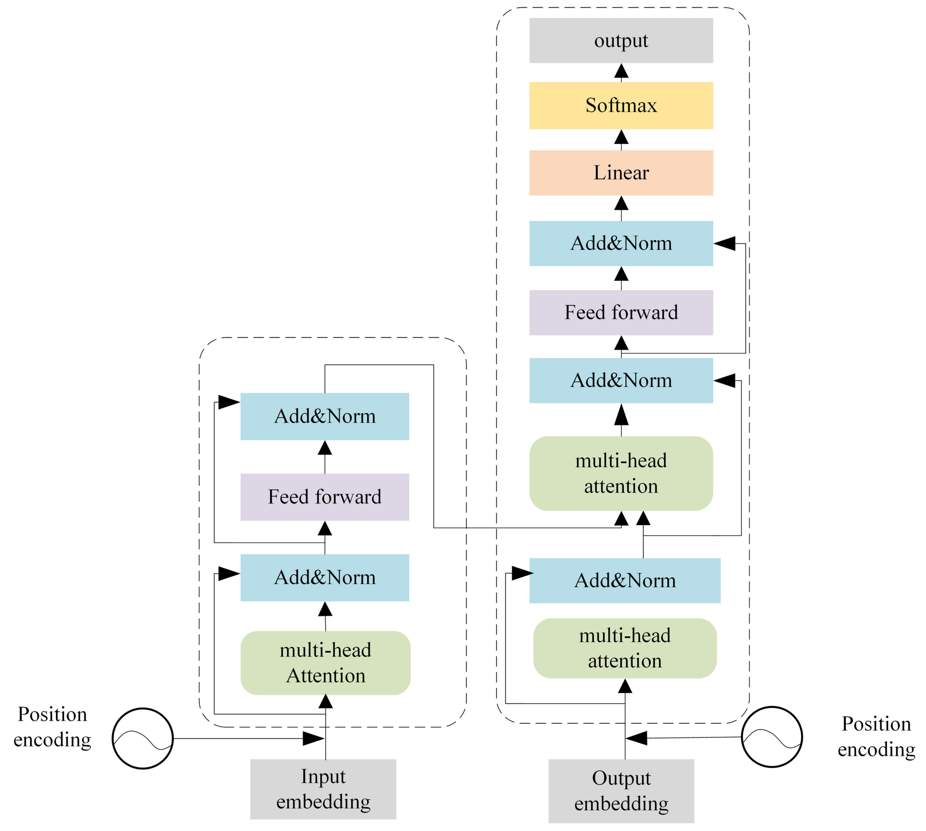

FIGURE 2
Architecture of the Transformer.

2.3.2 Self-attention mechanism

The Self-Attention Mechanism (SAM) is a core component of Transformer architecture, which seeks to emphasize the correlation between the input vector spaces.

As a first step, the input features are transformed into three different vectors using matrix multiplication, which is expressed as follows:

$$\begin{cases} \mathbf{Q} = F_{in}\mathbf{W}_Q \\ \mathbf{K} = F_{in}\mathbf{W}_K \\ \mathbf{V} = F_{in}\mathbf{W}_V \end{cases} \tag{8}$$

where **Q**, **K**, **V** are Query matrix, Key matrix, and Value matrix respectively. $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$ are used to generate the above-mentioned matrices. After that, attention map between different input vectors is calculated as follows:

- Compute scores between different input vectors with: $\mathbf{QK}^{\mathrm{T}}$.
- Normalize the scores to improve the stability with: $\frac{\mathbf{QK}^{\mathrm{T}}}{\sqrt{d_k}}$.
- Transform the scores into probabilities with softmax function: $\mathrm{softmax}(\frac{\mathbf{QK}^{T}}{\sqrt{d_k}})$.
- Generate the weighted value matrix with: $\mathrm{softmax}(\frac{\mathbf{QK}^{T}}{\sqrt{d_k}}) \cdot \mathbf{V}$.

The above process can be describe with a single function:

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{QK}^{\mathrm{T}}}{\sqrt{d_k}})\mathbf{V} \tag{9}$$

where d_k means the dimension of the input. This procedure is shown in Figure 3.

2.3.3 Position encoding

Transformer architecture can't guarantee the order of objects inside the sequence. Therefore, positional encoding is employed to assign a unique representation to each position inside the sequence. Cosine and sine functions are used to produce position encoding for varying frequencies, which is calculated as follows:

$$\begin{aligned} P(k,2i) &= \sin(\tfrac{k}{n^{2i/d}}) \\ P(k,2i+1) &= \cos(\tfrac{k}{n^{2i/d}}) \end{aligned} \tag{10}$$

where k is the position of an object inside the sequence, d means dimensions of the output embedding space, $P(k,j)$ is position function, n is a predefine scalar, i is used to map column indices.

Using the position encoding, even positions correspond to a sine function and odd positions correspond to cosine functions.

3 Methodology

In this section, we present our KalmanFormer: a hybrid model and data-driven Kalman Filter for estimating the state of dynamic systems. Our KalmanFormer combines the model-based Kalman Filters with Transformer (Vaswani et al., 2017) to tackle model

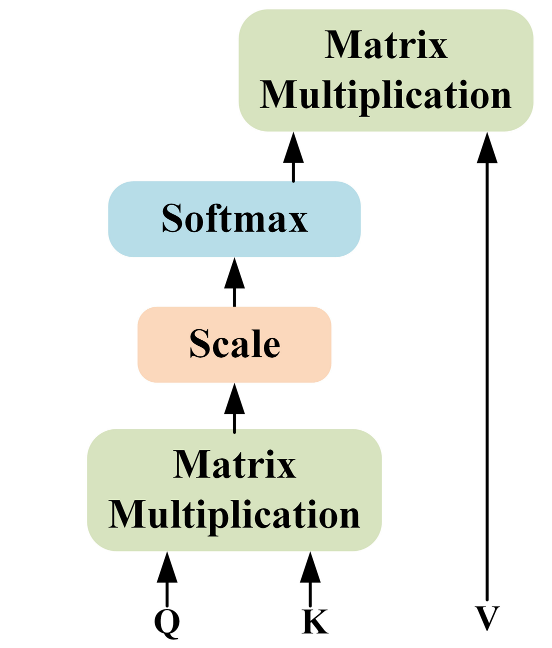

FIGURE 3
Illustration of the self-attention mechanism.

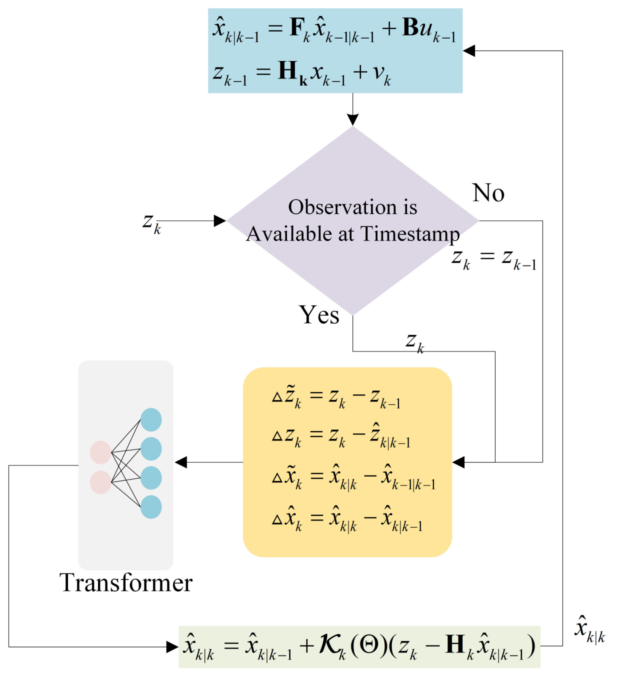
$\hat{x}_{k|k-1} = \mathbf{F}_k \hat{x}_{k-1|k-1} + \mathbf{B}u_{k-1}$
$z_{k-1} = \mathbf{H}_\mathbf{k} x_{k-1} + v_k$
z_k
Observation is Available at Timestamp
No
$z_k = z_{k-1}$
Yes
z_k
$\Delta\tilde{z}_k = z_k - z_{k-1}$
$\Delta z_k = z_k - \hat{z}_{k|k-1}$
$\Delta\tilde{x}_k = \hat{x}_{k|k} - \hat{x}_{k-1|k-1}$
$\Delta\hat{x}_k = \hat{x}_{k|k} - \hat{x}_{k|k-1}$
Transformer
$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \mathcal{K}_k(\Theta)(z_k - \mathbf{H}_k \hat{x}_{k|k-1})$
$\hat{x}_{k|k}$

FIGURE 4
Information flow of the KalmanFormer.

mismatch and non-linearities. As a first step, the information flow of our KalmanFormer will be presented. Subsequently, details information about the inputs for our KalmanFormer will be discussed. Following that, the architecture of the KalmanFormer and the training strategy will be introduced at the end of this section.

3.1 Information flow of KalmanFormer

In order to formulate our KalmanFormer, we identify the specific computation process of linear Kalman Filters that are based on unavailable knowledge. To be specific, the state transition model $\mathbf{F}_\mathbf{k}$ and observation model $\mathbf{H}_\mathbf{k}$ are available (although inaccurate), while the process noise $\mathbf{Q}_k$ and observation noise $\mathbf{R}_k$ are unavailable. As shown in Figure 1, unknown process noise and observation noise are used in Kalman Filters only for the purpose of calculating the Kalman Gain. To this end, we develop the KalmanFormer that tracks the Kalman Gain from the data and combines the learned Kalman Gain into the data flow of the Kalman Filter. The architecture of our KalmanFormer is provided in Figure 4. In the same manner as the model-based Kalman Filters, our KalmanFormer outputs the state estimate through two procedures: Prediction and Update.

1. In the prediction procedure, a *prior* state estimate of the current moment $\hat{x}^{-}_{k|k-1}$ is obtained from the previous *posterior* estimate $\hat{x}_{k-1|k-1}$.
2. In the update procedure, KalmanFormer uses the new observation z_k to compute the current state *posterior* $\hat{x}_{k|k}$ from the previous prior estimation $\hat{x}_{k|k-1}$, which is calculated in Equation 11. Instead of using the Kalman Gain matrix for the observation-update in the traditional Kalman Filters, KalmanFormer produces the Kalman Gain in a learned manner, denoted by $\mathcal{K}_\mathcal{K}(\Theta)$, with the trainable parameters Θ:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \mathcal{K}_\mathcal{K}(\Theta)(z_k - \mathbf{H}_k \hat{x}_{k|k-1}) \tag{11}$$

3.2 Input features

The model-based Kalman Filters, including EKF, UKF, and CKF compute the Kalman Gain from the known statistical information. We use a Transformer to model the Kalman Gain in a learned fashion in this paper. To calculate the Kalman Gain, we have to provide the information to a deep neural network to use the information to calculate the Kalman Gain. Inspired by KalmanNet (Revach et al., 2022), we devise the following quantities, which can be used for the input of the KalmanFormer:

- *The observation difference*:$\tilde{z}_k = z_k - z_{k-1}$
- *The innovation difference*: $z_k = z_k - \hat{z}_{k|k-1}$
- *The state evolution difference*: $\tilde{x}_k = \hat{x}_{k|k} - \hat{x}_{k-1|k-1}$, which represents the difference between two consecutive posterior state estimate.
- *The state update difference*: $\hat{x}_k = \hat{x}_{k|k} - \hat{x}_{k|k-1}$, which indicates the difference between the posterior state estimate and the prior state estimate.

Featurs $\tilde{x}_k$ and z_k indicate the uncertainty of the state estimates, while features z_k and $\hat{x}_k$ characterize the state transition and observation update process. Features z_k and $\tilde{z}_k$ contains the

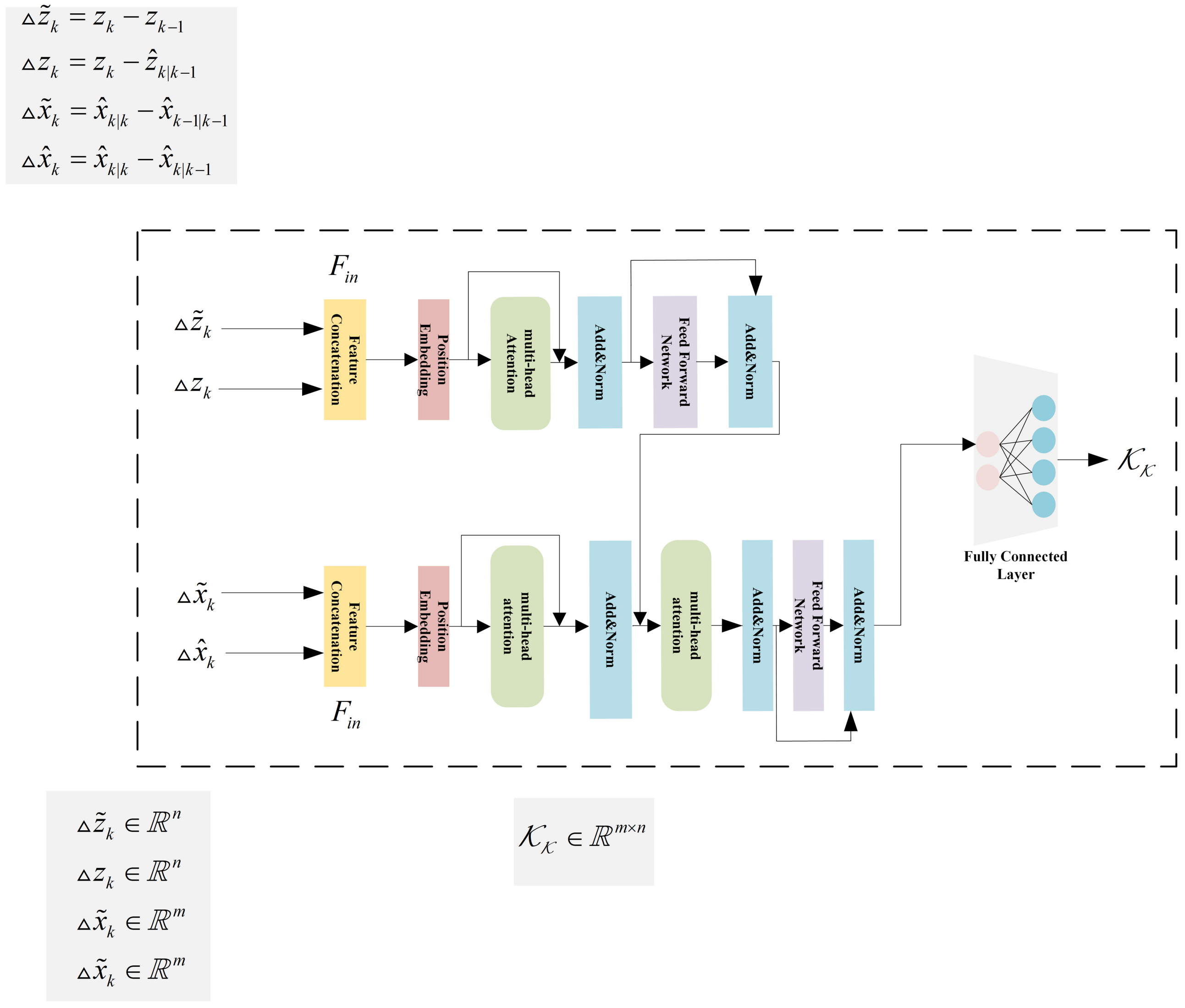
$\Delta\tilde{z}_k = z_k - z_{k-1}$
$\Delta z_k = z_k - \hat{z}_{k|k-1}$
$\Delta\tilde{x}_k = \hat{x}_{k|k} - \hat{x}_{k-1|k-1}$
$\Delta\hat{x}_k = \hat{x}_{k|k} - \hat{x}_{k|k-1}$
F_{in}
$\Delta\tilde{z}_k$
Δz_k
Feature Concatenation
Position Embedding
multi-head Attention
Add&Norm
Feed Forward Network
Add&Norm
$\Delta\tilde{x}_k$
$\Delta\hat{x}_k$
Feature Concatenation
Position Embedding
multi-head attention
Add&Norm
multi-head attention
Add&Norm
Feed Forward Network
Add&Norm
F_{in}
Fully Connected Layer
$\mathcal{K}_{\mathcal{K}}$
$\Delta\tilde{z}_k \in \mathbb{R}^n$
$\Delta z_k \in \mathbb{R}^n$
$\Delta\tilde{x}_k \in \mathbb{R}^m$
$\Delta\tilde{x}_k \in \mathbb{R}^m$
$\mathcal{K}_{\mathcal{K}} \in \mathbb{R}^{m\times n}$

FIGURE 5
The details of the KalmanFormer.

observation information, while features x_k and $\hat{x}_k$ characterize the states information of the system.

3.3 Details of the KalmanFormer

The internal of KalmanFormer uses the features discussed in the previous section to compute the Kalman Gain. As a first step, we will introduce the input features of the Transformer. To be specific, $\tilde{z}_k$, z_k, $\tilde{x}_k$, and $\hat{x}_k$ are used to construct our KalmanFormer. The data flow of the input features inside our KalmanFormer is shown in Figure 5.

Subsequently, the related observation features $\Delta\tilde{z}_k \in \mathbb{R}^n$ and $z_k \in \mathbb{R}^n$ are concatenate together to the input $F_{in} \in \mathbb{R}^{2n}$ of the Transformer encoders. And also, the related state features $\Delta\tilde{x}_k \in \mathbb{R}^m$ and $\Delta\hat{x}_k \in \mathbb{R}^m$ are concatenated together to the input for the Transformer decoders.

We devise three initial matrices $\mathbf{W_Q} \in \mathbb{R}^{2m\times 2m}$, $\mathbf{W_K} \in \mathbb{R}^{2m\times 2m}$, and $\mathbf{W_V} \in \mathbb{R}^{2m\times 2m}$ to generate the **Q**, **K**, and **V** matrices, which is used to produce self-attention score described in Section 2.3.2.

Following is the Add and Norm operation. To be specific, Layer normalization is used to perform Add and Norm operation, which is expressed as follows:

$$LayerNorm(X + attention) \tag{12}$$

Then the feed forward neural network is used to generate output, which is presented as:

$$FFN = \mathrm{ReLU}(\mathrm{X}\mathrm{W}_1 + b_1)\mathrm{W}_2 + b_2 \tag{13}$$

The feed-forward neural network is composed of two layers of the fully connected network. The W_1 and W_2 are the weights for the two layers of network. b_1 and b_2 are the bias. ReLU is the Rectified Linear Unit activate function.

Then the output of the Transformer encoder and the concatenated state input features to produce the learned Kalman Gain.

In our implementation, the input dimension is set to 4, the feed-forward dimension is set to 64, and 2 heads are employed in the Multi-head Self Attention Mechanism (MHSA). Furthermore, we stack the encoder and decoder 2 times to produce the learned Kalman Gain.

The information flow of the KalmanFormer is illustrated in Figure 5.

3.4 Training algorithm

A supervised learning paradigm is used to train the KalmanFormer using the available labeled data. Instead of producing the *posterior* estimate state, our KalmanFormer produces the Kalman Gain. Consequently, we define (Equation 21) to backpropagate the loss of Kalman Gain to train our KalmanFormer:

$$\frac{\partial L}{\partial K_k} = \frac{\partial ||K_k \Delta z_k - \Delta x_k||^2}{\partial K_k} = 2 \cdot (K_k \Delta z_k - \Delta x_k) \cdot \Delta z_k^T \tag{14}$$

where $\Delta x_k = x_k - \hat{x}_{k|k-1}$. The Equation 14 indicates that we can learn the computation of the Kalman Gain by training KalmanFormer end-to-end using the squared-error loss.

In general, the dataset used for training the KalmanFormer consists of N length Ttrajectories. Let T denote the length of i-th training trajectory inside the dataset. The dataset can be expressed by $\mathcal{D} = \{(\mathbf{Z_i}, \mathbf{X_i})\}_1^N$, where

$$Z_i = [z_1^{(i)}, z_2^{(i)} ..., z_T^{(i)}], X_i = [x_0^{(i)}, x_1^{(i)} ..., x_T^{(i)}] \tag{15}$$

The empirical loss function for the i-th trajectory training inside the dataset is defined as follows:

$$l_i(\Theta) = \frac{1}{T_i} \sum_{k=1}^{T_i} ||\Psi_\Theta(\hat{x}_{k-1}^i, z_k^{(i)}) - x_k^{(i)}||^2 + \xi \cdot ||\Theta||^2 \tag{16}$$

where Ψ_Θ represents the output of our KalmanFormer, Θ is the trainable parameters inside the KalmanFormer, and ξ is regularization coefficient. Let $\Delta \mathrm{x}_k^{(k)} = \mathrm{x}_k^{(k)} - \hat{\mathrm{x}}_{k|k-1}^{(k)}$ and $\Delta z_k^{(k)} = z_k^{(k)} - \hat{z}_{k|k-1}^{(k)}$ be the state prediction error and the measurement innovation at timestamp k. The partial derivative of the loss function respective to the Kalman gain matrix is:

$$\frac{\partial l(\Theta)}{\partial K_k(\Theta)} = \frac{1}{LT_k} \sum_{k=1}^{L} \sum_{k=1}^{T_k} \frac{\partial ||\Delta x_k^{(k)} - K_k(\Theta) \Delta z_k^{(k)}||_2^2}{\partial K_k(\Theta)} \tag{17}$$

By plugging into the chain rule:

$$\frac{\partial l(\Theta)}{\partial (\Theta)} = \frac{\partial l(\Theta)}{\partial K_k(\Theta)} \frac{\partial K_k(\Theta)}{\partial (\Theta)} \tag{18}$$

We can adopt a stochastic gradient descent algorithm to optimize Θ by using $\frac{\partial l(\Theta^*)}{\partial (\Theta^*)} = 0$.

4 Numerical experiments

In this section, we design a series of experiments to evaluate the performance of our proposed KalmanFormer and compare it to some other benchmarks. As a first step, we make a brief description of the training setup of our KalmanFormer. Following that, we conduct the simulation experiments including nonlinear cases to evaluate the performance of our proposed method. At the end of this section, IMU and GPS information are employed to investigate the effectiveness of our proposed method.

4.1 Implement details

To be specific, the dimensions of concatenated observation difference and innovation difference are 8, which is the input to the encoder for the transformer. Also, the dimensions of state evolution difference and state update difference are 4, which is the input to the decoders of Transformer. The feed-forward dimension inside the encoder and decoder is 64, and 2 heads are employed in the multi-head attention mechanism. Furthermore, we stack the encoder and decoder 2 times to produce the output. After the output is obtained, a fully connected layer is used to generate the learned Kalman Gain.

Furthermore, we conduct all of our training and validation experiments on the Pytorch (Paszke et al., 2019) platform using a single RTX 3090 GPU card, CUDA11.6, and cuDNN version 8. Furthermore, the Cosine Annealing Schedule is employed to adjust the learning rate in the training procedure, which can be expressed as follows:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + \cos\left(\frac{T_{cur}}{T_{max}}\pi\right)) \tag{19}$$

where η_t represents the learning rate of the current iteration, $\eta_{\min}$ and $\eta_{\max}$ mean the predefined minimum and maximum learning rate respectively. T_{cur} and T_{max} are the current iteration and maximum iterations respectively.

Adam (Kingma and Ba, 2014) optimizer is used to train the KalmanFormer. Different hyper parameters are performed on the simulation and multi-sensor fusion experiments. The specific information about the hyperparameters is shown in Table 1.

4.2 Simulation experiments

In this section, a series of simulation experiments are designed to demonstrate the effort of our proposed KalmanFormer. We make a comparison with EKF and KalmanNet (Revach et al., 2021).

4.2.1 Test metric

Mean Square Error (MSE) is used to evaluate the effect of our proposed KalmanFormer, which is computed as follows:

$$MSE = \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{T} |(x_{est} - x_{true})_i|^2 \tag{20}$$

TABLE 1 Details information about the hyperparameters.

Experiment type	Epochs	Batch size	Learning rate	Weight decay
Simulation	200	30	1e-3	1e-3
Multi-sensor fusion	100	10	1e-3	1e-4

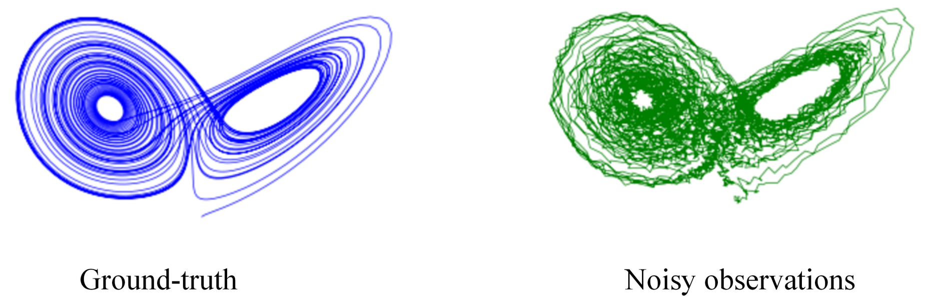

FIGURE 6
Illustration of trajectories.

where x_{est} means the output from our KalmanFormer, x_{true} represents corresponding ground-truth. N means the number of the testing trajectories. T is the length of current trajectory.

4.2.2 Non-linear Lorenz attractors

The Lorenz attractor (Tucker, 1999) describes a non-linear chaotic system used for atmospheric convection. The Lorenz system is expressed by following three differential equations that define the convection rate, the horizontal temperature variation, and the vertical temperature variation of a fluid:

$$\frac{\partial z_1}{\partial t} = 10(z_2 - z_1), \frac{\partial z_2}{\partial t} = z_1(28 - z_3) - z_2, \frac{\partial z_3}{\partial t} = z_1 z_2 - \frac{8}{3} z_3, \quad (21)$$

In order to generate the simulated trajectories, we run the Lorenz equations described at Equation 21 with a time step of $\Delta t = 0.05$ and add Gaussian noise of standard deviation $\sigma = 0.05$ to the results. The noisy data is considered as the measurements while the decimated data is regarded as the ground truth trajectory for our experiments. The trajectories of ground truth and noisy observations are shown in Figure 6.

Assuming a three-dimensional vector $x = [z_1, z_2, z_3]^{\mathrm{T}} \in \mathbf{R}$, the dynamic matrix $\mathbf{A}(x)$ of the system from Equation 21 is expressed as follows:

$$\mathbf{A}(x) = \begin{bmatrix} -10 & 10 & 0 \\ 28 - z_3 & -1 & 0 \\ z_2 & 0 & -\frac{8}{3} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \quad (22)$$

After that, Taylor expansion is used to obtain the state transition function:

$$\mathbf{F_k}(x_k) = \mathbf{I} + \sum_{j=1}^{J} \frac{(A(x_k)k)^j}{j!} \quad (23)$$

where $\mathbf{I}$ represents the identity matrix and $\mathbf{J}$ means the number of Taylor expansion. We set J = 5 in our experiments. For the measurement model, we set $\mathbf{H} = \mathbf{I}$. For the noise parameters, we set $\mathbf{Q} = \mathrm{q}^2\mathbf{I}, \mathbf{R} = r^2\mathbf{I}$, where q=0.8, r=1.

TABLE 2 Origin point information for NED frame.

Latitude origin	42.29322deg
Longitude origin	−83.709657 deg
Altitude origin	270 m

4.3 Multi-sensor information fusion

We further evaluate the effectiveness of the proposed KalmanFormer in multi-sensor fusion. We employ the Michigan NCLT dataset (Carlevaris-Bianco et al., 2016) with different types of sensors to perform our experiments.

The NCLT dataset was obtained from a mobile robot platform equipped with various sensors, including Real Time Kinematic GPS, IMU, Consumer-grade GPS, etc. In our experiments, IMU is employed to provide angular speed information and acceleration information, which is used to design the state transition function. The consumer-grade GPS is applied to provide the observation of the displacement. The Real-Time Kinematic GPS is used to generate a more accurate state of the system, which is used to evaluate the effectiveness of the proposed method.

4.3.1 Coordinates definition

A North-East-Down (NED) frame is employed to describe the robot's pose and position. Furthermore, the fixed origin point of the NED frame is shown in Table 2.

The angular and acceleration information from the IMU is measured in the IMU's reference frame, which closely aligns with the robot's reference coordinate. It is necessary to transform the IMU reading from IMU's frame into a global frame.

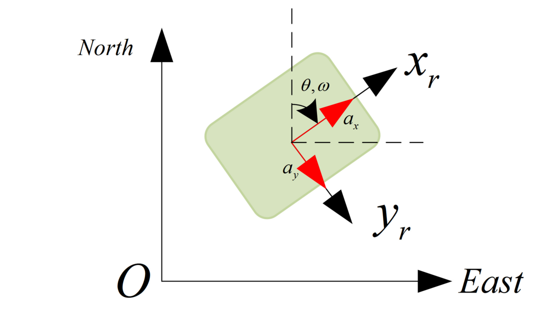

FIGURE 7
Transformation between IMU frame and Global Frame (NED).

As shown in Figure 7, we can obtain the transformation between the IMU frame and the global frame, which is calculated as:

$$\begin{cases} a_{gx} = a_x \cos(-\theta) - a_y \sin(-\theta) \\ a_{gy} = a_x \sin(-\theta) - a_y + \cos(-\theta) \end{cases} \tag{24}$$

4.3.2 State transition model

The state vector of the system in the global coordinate is defined as:

$$x_k{=}[x, y, v_x, v_y, \theta, \omega] \tag{25}$$

where x_k, y_k represent the position of the robot in the global frame. v_k and v_y represent the velocities. θ and ω mean the heading angle and angular velocities respectively.

In the global coordinate system, the state transition model takes the IMU's readings, including heading θ, angular velocity ω, and the accelerations as the control input. The state transition model (in the NED frame) is then:

$$\hat{x}_{k|k-1} = \mathbf{F_k}(x_{k-1}, u_{k-1}) = \begin{bmatrix} x_{k-1} + v_x\Delta k + \frac{1}{2}a_{gx}\Delta k^2 \\ y_{k-1} + v_y\Delta k + \frac{1}{2}a_{gy}\Delta k^2 \\ v_{x-1} + a_{gx}\Delta k \\ v_{y-1} + a_{gy}\Delta k \\ \theta_k \\ \omega_K \end{bmatrix} \tag{26}$$

4.3.3 Observation model

The GPS observation model produces a prediction of the expected GPS observation based on the predicted state. Here we use the consumer-grade GPS to produce the observation of the displacement. The observation model is expressed as follows:

$$z_{k|k-1} = \mathbf{H_k}\hat{x}_{k|k-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \hat{x}_{k|k-1} \tag{27}$$

4.3.4 Noise setting

The initial process noise $\mathbf{Q}_k$ and measurement noise $\mathbf{R}_k$ matrices of the EKF are expressed in Equations 2, 3. These $\mathbf{Q}_k$ and $\mathbf{R}_k$ matrices are determined using empirical data as well as completing experimental tuning. The initial $\mathbf{Q}_k$ and $\mathbf{R}_k$ matrices used in our experiments are developed as follows:

$$\mathbf{Q}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix} \tag{28}$$

$$\mathbf{R}_k = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \tag{29}$$

4.4 Model mismatch

4.4.1 State transition model mismatch

We devise experiments to investigate the robustness of the KalmanFormer when the state transition model is mismatched. This is achieved by using three 3-dimensional rotation matrices:

$$RZ = \begin{bmatrix} \cos(yaw) & -\sin(yaw) & 0 \\ \sin(yaw) & \cos(yaw) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{30}$$

$$RY = \begin{bmatrix} \cos(pitch) & 0 & \sin(pitch) \\ 0 & 1 & 0 \\ -\sin(pitch) & 0 & \cos(pitch) \end{bmatrix} \tag{31}$$

$$RX = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(roll) & -\sin(roll) \\ 0 & \sin(roll) & \cos(roll) \end{bmatrix} \tag{32}$$

$$yaw = roll = pitch = 1^\circ, 5^\circ \tag{33}$$

We evaluate the performance in the condition of model mismatch real-word NCLT datasets. The mismatched state transition real-world is expressed as follows:

$$\mathbf{F}_{real}^{\text{rotated}} = RX \bullet RY \bullet RZ \bullet \begin{bmatrix} 1 & \Delta k & \frac{1}{2}\Delta k^2 \\ 0 & 1 & \Delta k \\ 0 & 0 & 1 \end{bmatrix} \tag{34}$$

where $\mathbf{F}_{real}^{\text{rotated}}$ is the mismatched state transition model for the real-world experiments. In our experiments, the rotation angle is set to 1° and $5^\circ$to verify the model performance.

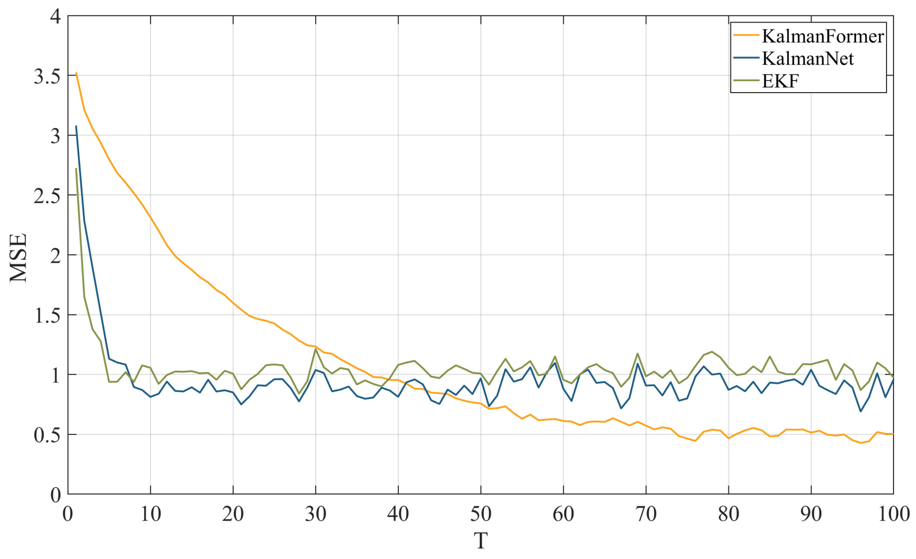

FIGURE 8
MSE comparison results with KalmanNet and EKF on synthetic dataset.

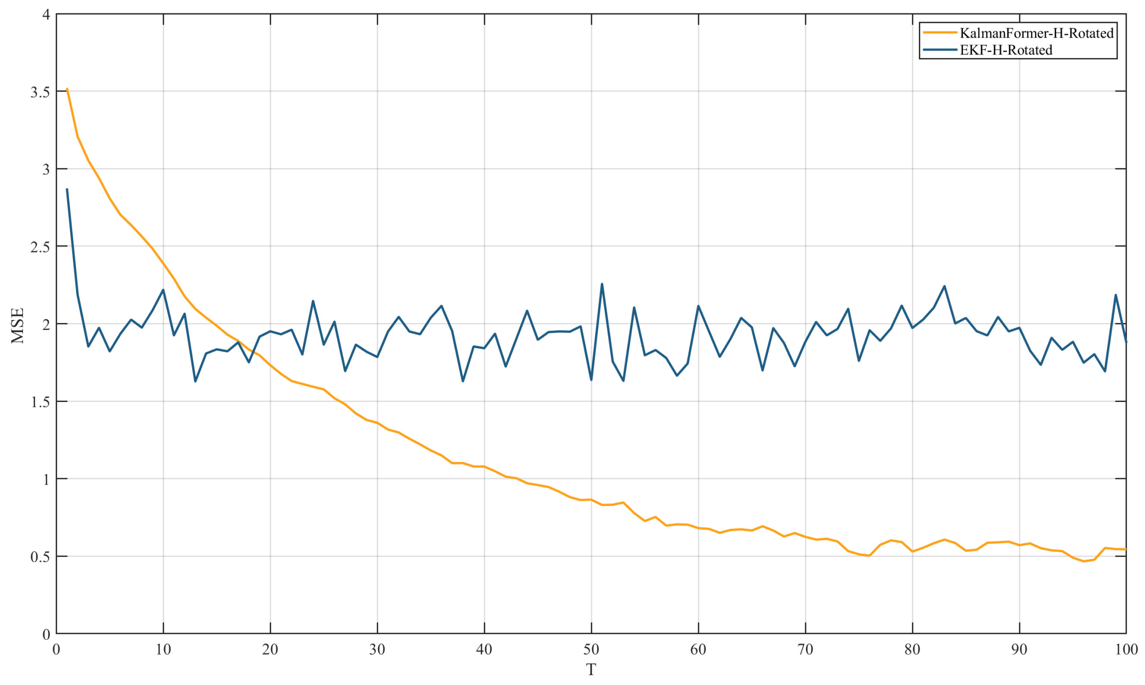

FIGURE 9
MSE comparison results with EKF when the observation is mismatched.

4.4.2 Observation model mismatch

Additionally, we investigate the performance of our proposed KalmanFormer with EKF when the observation function is mismatched. The mismatched observation function is expressed as follows:

$$\mathbf{H}_{rotated} = \mathbf{H} \bullet RX \bullet RY \bullet RZ \tag{35}$$

We set the rotation angle to 10° to validate the effectiveness on the simulation dataset.

Furthermore, we transform the observation in Cartesian coordinates into Spherical coordinates using the equation and compare the performance.

$$\begin{cases} r = \sqrt{z_1^2 + z_2^2 + z_3^2} \\ \theta = \cos^{-1}(\frac{z_3}{r}) \\ \phi = \tan^{-1}(\frac{z_2}{z_1}) \end{cases} \tag{36}$$

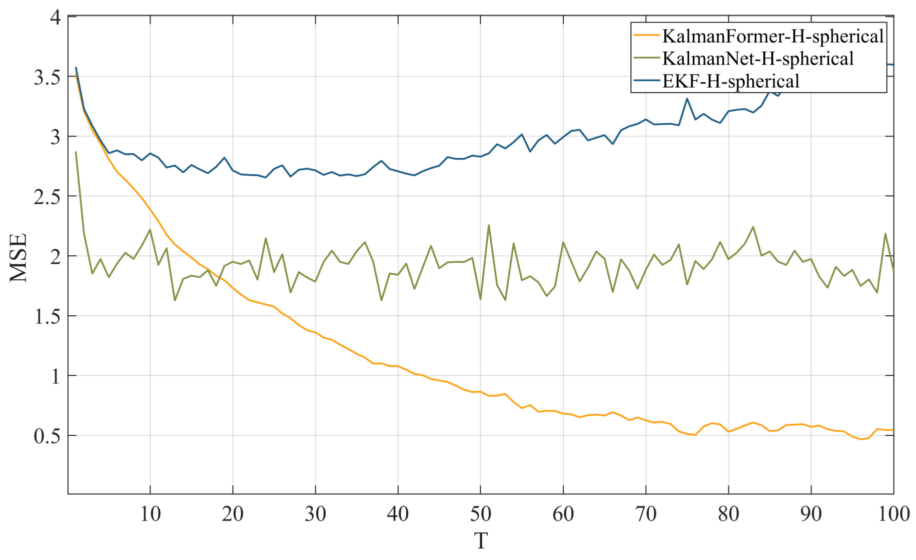

FIGURE 10
MSE comparison results with EKF when the observation is in spherical coordinate.

4.5 Evaluation results

In this section, we will discuss the performance of our proposed KalmanFormer with EKF and KalmanNet for both linear and non-linear systems. Furthermore, we will investigate the performance of the proposed KalmanFormer using the NCLT dataset.

4.5.1 Simulation results

MSE metric is used to demonstrate the effectiveness of the proposed KalmanFormer in non-linear Lorenz attractors. As shown in Figure 8, our KalmanFormer achieves a higher MSE result in the first 30 points of the Test sequence when compared to KalmanNet and EKF. However, after the 40 points, the MSE of our KalmanFormer is much lower than EKF and KalmanNet.

Besides that, Euclidean Distance is used to evaluate the effectiveness of our methodology over the whole test trajectories. Euclidean Distance is expressed as follows:

$$\text{distance} = \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{T} (x_{est}^{j} - x_{true}^{j})^2} \qquad (37)$$

where $x_{true} \in [\, z_1 \;\; z_2 \;\; z_3]^T \in \mathbb{R}$ means the ground truth of the state vector. $x_{est} \in [\, z_1 \;\; z_2 \;\; z_3]^T \in \mathbb{R}$ represents the estimation from our KalmanFormer. N is the number of the trajectories. T is the length for each trajectory.

Using Equation 37, the distance of our proposed method is 136. While the distance of KalmanNet is 209, which demonstrates the superiority of our KalmanFormer. In conclusion, KalmanFormer achieves more accurate performance on the Simulation Test set.

Additionally, Figure 9 reports the experiment results when the observation model is mismatched.

TABLE 3 The complexity comparison results on simulation experiments.

Method	Parameters	Storage (KB)	Inference time (s)
KalmanFormer	8,081	66	21
KalmanNet	23,928	46	19
EKF	\	\	20

We can observe that the proposed KalmanFormer achieves lower MSE performance than EKF in the same experiment setup when the observation model is disturbed by the rotation matrix.

Figure 10 reports the results when the observation in transformed into spherical coordinates. We can see that our proposed KalmanFormer achieves the best performance compared to KalmanNet and EKF in the condition of the mismatched observation.

Finally, we compare the time complexity of the KalmanFormer compared to EKF and KalmanNet through simulation experiments. Parameters, storage space, and inference time are adopted to verify the computational complexity of the KalmanFormer, KalmanNet, and EKF. The inference time is computed on the simulation experiments. The comparison results are shown in Table 3.

As shown in Table 3, the KalmanNet and the KalmanFormer have similar space demand and the similar running speeds on the simulation experiments. To be specific, the KalmanNet needs 44 KB harddisk space to store while the KalmanFormer 66KB needs disk space. Furthermore, we compare the inference time on the whole dataset. The inference time of the EKF is about 20s while the KalmanFormer runs about 21s. We can conclude that the proposed KalmanFormer has a similar time complexity with the EKF and KalmanNet and it can be further used in real-world applications.

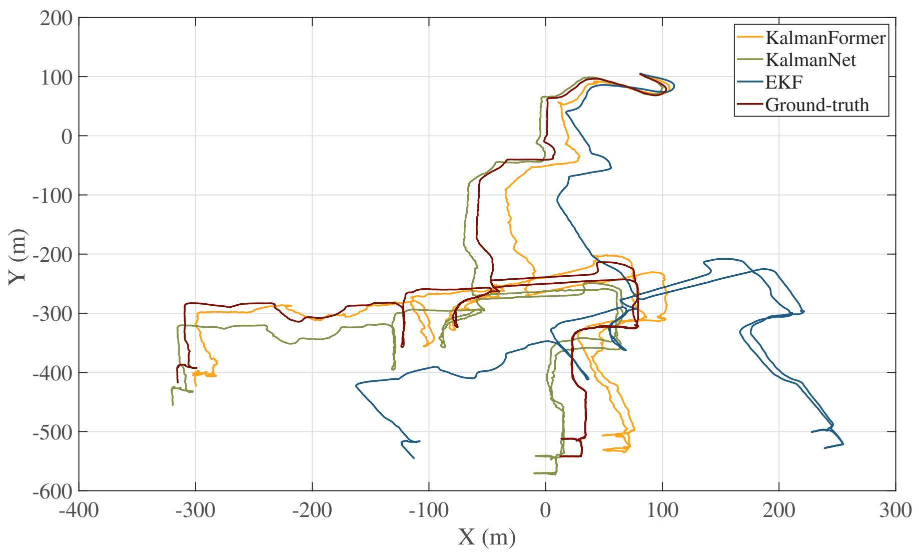

FIGURE 11
Trajectory comparison results with KalmanNet and EKF.

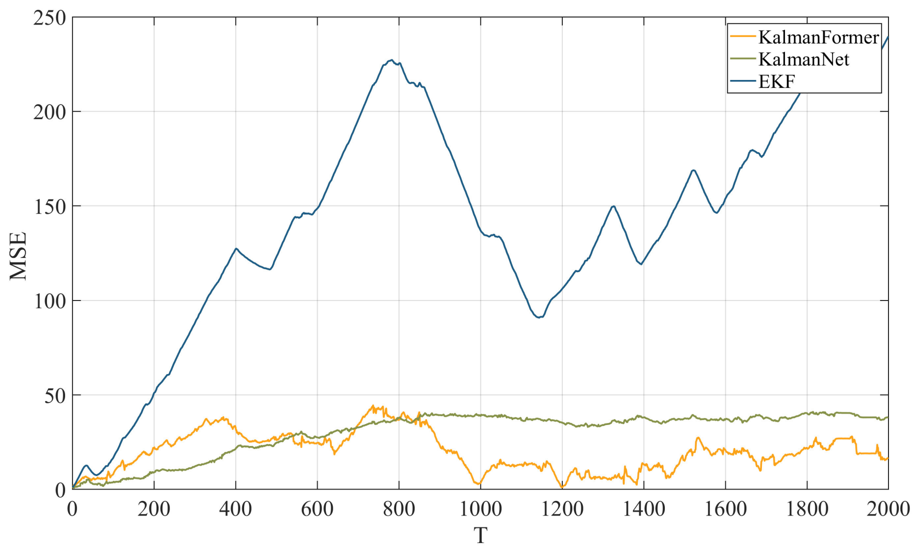

FIGURE 12
MSE performance with KalmanNet and EKF on NCLT dataset.

4.5.2 Multi-sensor information fusion

The trajectory we used for training and validating KalmanFormer and KalmanNet are obtained from the date of 2012-01-22 within the NCLT datasets. Furthermore, a date of 2012-04-29 trajectory is used to test the performance. The sample rate of the training, validation, and test is 1 *HZ*. The trajectory comparison result is shown in Figure 11.

As shown in Figure 11, our KalmanFormer performs better than EKF and KalmanNet. In order to evaluate the property of our KalmanFormer, we make a comparison with EKF and KalmanNet in terms of MSE using the same data in Figure 11. The result is shown in Figure 12. According to Figure 12, our KalmanFormer achieves similar accuracy at the first 500 points of the testing set. However, in the last 1,500 points, our method achieves better performance in MSE compared to KalmanNet.

Additionally, Equation 37 is used to evaluate the validity over the whole test trajectory quantitatively. The distance of KalmanFormer is 19 m, the distance of KalmanNet is 30 m, and the distance of EKF is 316 m, which proves the superiority of the KalmanFormer.

Finally, we investigate the results using the mismatched state transition function with the rotation angles of 1° and 5°. Figure 13 reports the results when the state transition function is mismatched.

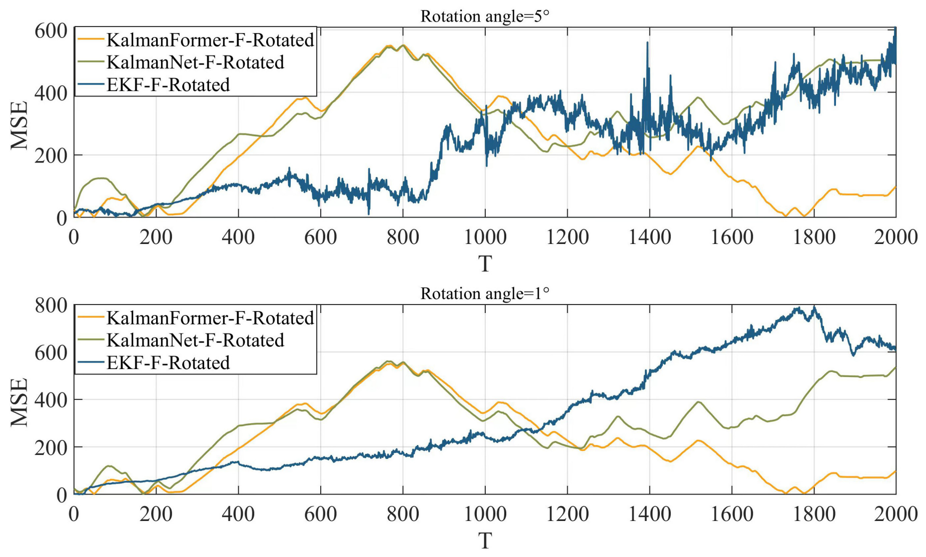

FIGURE 13
MSE performance when the state transition model is mismatched.

We can observe that when the state transition model is disturbed by the rotation matrix with a rotation angle of 1°, our KalmanFormer has a similar performance to the KalmanNet and outperforms the EKF. When the rotation angle is set to 5°, the performance of EKF degrades significantly. And the KalmanFormer outperforms the KalamNet and EKF. Even our KalmanFormer achieves lower MSE than KalmanNet.

5 Conclusion

In this paper, we proposed KalmanFormer, which is a hybrid of data-driven and model-driven implementation of the Kalman Filters. KalmanFormer incorporates a Transformer architecture within the learning process of computing the Kalman Gain (KG) and combines the learned KG into a traditional Kalman Filter. The proposed KalmanFormer uses the Kalman Filter without requiring any prior knowledge of process statistics or measurement noise statistics, even if the system model is mismatched. It has been demonstrated through numerical experiments that KalmanFormer is capable of achieving the minimum MSE when properly trained. It has also been proven that KalmanFormer is more robust to inaccurate knowledge of state space parameters in multi-sensor information fusion.

Data availability statement

The public NCLT dataset is used in this study. Researchers can get them from website of https://robots.engin.umich.edu/nclt/.

Author contributions

SS: Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing. JC: Conceptualization, Validation, Writing – review & editing. GY: Software, Writing – original draft. ZZ: Writing – original draft. PH: Data curation, Methodology, Writing – review & editing.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Carlevaris-Bianco, N., Ushani, A. K., and Eustice, R. M. (2016). University of Michigan north campus long-term vision and Lidar dataset. *Int. J. Rob. Res.* 35, 1023–1035. doi: 10.1177/0278364915614638

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. doi: 10.48550/arXiv.1412.3555

Coskun, H., Achilles, F., DiPietro, R., Navab, N., and Tombari, F. (2017). "Long short-term memory kalman filters: recurrent neural estimators for pose regularization,"9D in *IEEE International Conference on Computer Vision*, 5524–5532. doi: 10.1109/ICCV.2017.589

Coué, C., Fraichard, T., Bessiere, P., and Mazer, E. (2003). "Using bayesian programming for multi-sensor multi-target tracking in automotive applications,"9D in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)* (IEEE), 2104–2109. doi: 10.1109/ROBOT.2003.1241904

Dahal, P., Mentasti, S., Paparusso, L., Arrigoni, S., and Braghin, F. (2024). Robuststatenet: robust ego vehicle state estimation for autonomous driving. *Rob. Auton. Syst.* 172:104585. doi: 10.1016/j.robot.2023.104585

Elman, J. L. (1990). Finding structure in time. *Cogn. Sci.* 14, 179–211. doi: 10.1207/s15516709cog1402_1

Hadlaczky, B. Z., Friedman, N., Takarics, B., and Vanek, B. (2023). "Wing shape estimation with extended kalman filtering and kalmannet neural network of a flexible wing aircraft," in *Learning for Dynamics and Control Conference* (PMLR), 1429–1440.

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.* 9, 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Hu, C., Chen, W., Chen, Y., Liu, D., et al. (2003). Adaptive kalman filtering for vehicle navigation. *J. Global Posit. Syst.* 2, 42–47. doi: 10.5081/jgps.2.1.42

Huang, Y., Zhu, F., Jia, G., and Zhang, Y. (2020). A slide window variational adaptive kalman filter. *IEEE Trans. Circ. Syst.* 67, 3552–3556. doi: 10.1109/TCSII.2020.2995714

Hue, C., Le Cadre, J.-P., and Pérez, P. (2002). Sequential monte carlo methods for multiple target tracking and data fusion. *IEEE Trans. Signal Proc.* 50, 309–325. doi: 10.1109/78.978386

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *J. Basic Eng.* 82, 35–45. doi: 10.1115/1.3662552

Kingma, D. P., and Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. doi: 10.48550/arXiv.1412.6980

Luttmann, L., and Mercorelli, P. (2021). "Comparison of backpropagation and kalman filter-based training for neural networks," in *2021 25th International Conference on System Theory, Control and Computing (ICSTCC)* (IEEE), 234–241. doi: 10.1109/ICSTCC52150.2021.9607274

Maybeck, P. S. (1982). *Stochastic Models, Estimation, and Control.* London: Academic Press.

Menner, M., Berntorp, K., and Di Cairano, S. (2023). Automated controller calibration by kalman filtering. *IEEE Trans. Control Syst. Technol.* 31, 2350–2364. doi: 10.1109/TCST.2023.3254213

Mercorelli, P. (2012a). "A kalman filter for sensorless control of a hybrid hydraulic piezo actuator using MPC for camless internal combustion engines," in *2012 IEEE International Conference on Control Applications* (IEEE), 980–985. doi: 10.1109/CCA.2012.6402717

Mercorelli, P. (2012b). A two-stage augmented extended kalman filter as an observer for sensorless valve control in camless internal combustion engines. *IEEE Trans. Industr. Electr.* 59, 4236–4247. doi: 10.1109/TIE.2012.2192892

Otter, D. W., Medina, J. R., and Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. *IEEE Trans. Neural Netw. Learn. Syst.* 32, 604–624. doi: 10.1109/TNNLS.2020.2979670

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). "Pytorch: an imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 32.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). "Deep state space models for time series forecasting," in *Advances in Neural Information Processing Systems*, 31.

Revach, G., Shlezinger, N., Ni, X., Escoriza, A. L., Van Sloun, R. J., and Eldar, Y. C. (2022). Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Trans. Signal Proc.* 70, 1532–1547. doi: 10.1109/TSP.2022.3158588

Revach, G., Shlezinger, N., Van Sloun, R. J., and Eldar, Y. C. (2021). "Kalmannet: data-driven kalman filtering," in *ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE), 3905–3909. doi: 10.1109/ICASSP39728.2021.9413750

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*. doi: 10.48550/arXiv.1609.04747

Tucker, W. (1999). The lorenz attractor exists. *Compt. Rendus l'Acad. Sci. Mathem.* 328, 1197–1202. doi: 10.1016/S0764-4442(99)80439-X

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). "Attention is all you need," in *Advances in Neural Information Processing Systems*, 30.

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., et al. (2018). Deep learning for computer vision: a brief review. *Comput. Intell. Neurosci.* 2018:7068349. doi: 10.1155/2018/7068349

Wan, E. A., and Van Der Merwe, R. (2001). "The unscented kalman filter," in *Kalman Filtering and Neural Networks*, 221–280. doi: 10.1002/0471221546.ch7

Xia, M., Shao, H., Ma, X., and de Silva, C. W. (2021). A stacked GRU-RNN-based approach for predicting renewable energy and electricity load for smart grid operation. *IEEE Trans. Industr. Inform.* 17, 7050–7059. doi: 10.1109/TII.2021.3056867

Xiong, L., Xia, X., Lu, Y., Liu, W., Gao, L., Song, S., et al. (2020). IMU-based automated vehicle body sideslip angle and attitude estimation aided by gnss using parallel adaptive kalman filters. *IEEE Trans. Vehic. Technol.* 69, 10668–10680. doi: 10.1109/TVT.2020.2983738

Xu, L., and Niu, R. (2021). "Ekfnet: learning system noise statistics from measurement data," in *ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE), 4560–4564. doi: 10.1109/ICASSP39728.2021.9415083

Xu, L., Zhang, J. Q., and Yan, Y. (2004). A wavelet-based multisensor data fusion algorithm. *IEEE Trans. Instrum. Meas.* 53, 1539–1545. doi: 10.1109/TIM.2004.834066

Yadav, S., Saha, S. K., and Kar, R. (2023). An application of the kalman filter for EEG/ERP signal enhancement with the autoregressive realisation. *Biomed. Signal Process. Control* 86:105213. doi: 10.1016/j.bspc.2023.105213

Yu, X., and Li, J. (2021). Adaptive kalman filtering for recursive both additive noise and multiplicative noise. *IEEE Trans. Aerosp. Electron. Syst.* 58, 1634–1649. doi: 10.1109/TAES.2021.3117896

Zhang, H., Yang, Z., Xiong, H., Zhu, T., Long, Z., and Wu, W. (2023). Transformer aided adaptive extended kalman filter for autonomous vehicle mass estimation. *Processes* 11:887. doi: 10.3390/pr11030887

Zhang, W., Li, X., and Li, X. (2020). Deep learning-based prognostic approach for lithium-ion batteries with adaptive time-series prediction and on-line validation. *Measurement* 164:108052. doi: 10.1016/j.measurement.2020.108052