Diff-Kalman: Difference-Driven Learning for Structure-Preserving Kalman Filtering

Jia Gao Xianglei Xing* Yang Qing
Tianshuo Zhang Wenzhe Zhai

College of Intelligent Systems Science and Engineering
Harbin Engineering University
Harbin 150001, China

Abstract

State estimation is a critical challenge in modeling dynamic systems. Classical Kalman filters can struggle with nonlinear dynamics, model mismatch, and inaccurate noise statistics. Although learning-aided methods improve adaptability, many focus on the update step or learn high-dimensional filtering quantities, complicating optimization and leaving prediction modeling underexplored. To address this, we propose Diff-Kalman, which leverages state differences to predict state increments and scaling matrices for correcting both the prediction and update phases of the Kalman filter. Diff-Kalman combines traditional dynamic models with neural networks to adaptively correct system behavior. In the prediction phase, Diff-Kalman utilizes a Transformer-based architecture with self-attention to extract localized patterns from historical states, followed by cross-attention to predict future state increments. These increments are then fused with increments predicted by the dynamic model, while a GRU-based Scaling Predictor (GSP) dynamically adjusts the prior covariance. In the update phase, another GSP dynamically adjusts the Kalman gain, improving estimation accuracy and robustness. Experimental results on linear MOT trajectories and nonlinear Lorenz dynamics show that Diff-Kalman improves estimation accuracy compared with representative baselines. The code is publicly available at https://github.com/heu-gj/Diff-Kalman.

1 Introduction

State estimation is a key task in dynamic system modeling and control. It aims to reconstruct the internal state of the system by using prior knowledge and measurement sequences. Accurate estimation is crucial for improving system performance and stability. In 1960, Rudolf E. Kalman [1] proposed the Kalman filter (KF). Under linear-Gaussian assumptions, it provides a recursive minimum mean-square error estimate and has become a standard method for state estimation in linear systems. It is widely used in fields such as multiple object tracking [2–4], navigation systems [5–8], and robotics [9, 10].

Although the KF has demonstrated significant performance in state estimation for linear systems, it exhibits considerable limitations when applied to real-world scenarios involving nonlinear systems. To overcome these limitations, researchers have developed several variants of the KF specifically designed for nonlinear systems. One such method is the Extended Kalman Filter (EKF) [11], which linearizes the state-space model of a nonlinear system through a first-order Taylor expansion. In addition, other classical variants, including the Unscented Kalman Filter (UKF) [12] and Cubature

*Corresponding author: Xianglei Xing (xingxl@hrbeu.edu.cn).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Kalman Filter (CKF) [13], have been further introduced to enhance the adaptability and estimation accuracy of nonlinear systems.

Due to their reliance on accurate dynamic models and noise statistics, EKF and similar methods can still struggle under model mismatch and complex nonlinear dynamics. In recent years, researchers have combined data-driven methods with KF to improve state estimation in nonlinear systems. Revach *et al.* [14] proposed KalmanNet (KNet), a hybrid model combining KF and a recurrent neural network (RNN), addressing model mismatch and nonlinearities through Kalman gain learning. Cohen *et al.* [15] introduced A-KIT, an adaptive algorithm that integrates EKF with Transformer-based deep learning to dynamically adjust the process noise covariance matrix.

Although integrating neural networks with Kalman filtering has shown promising potential in state estimation tasks, existing approaches still face several key challenges. Many methods [14, 16–18] rely on neural networks to learn Kalman filtering quantities, such as the Kalman gain or covariance matrices. Learning these quantities, particularly in high-dimensional systems, can increase the optimization burden. Moreover, these methods often focus solely on modeling the update phase, while the prediction phase relies entirely on linear dynamics or Taylor expansion. Such approximations may fail to capture complex system behavior, and they cause prior errors to accumulate and degrade overall estimation performance. To address these issues, we propose Diff-Kalman, a hybrid Kalman-style filtering framework that retains the recursive predict-update structure of the Kalman filter. Unlike existing methods, Diff-Kalman learns structured, difference-driven corrections to the model-based filtering process. This design avoids directly regressing high-dimensional matrices by learning structured correction terms instead. Additionally, Diff-Kalman incorporates neural network corrections in both the prediction and update phases: during the prediction phase, it estimates state increments to refine the prior state estimate, and during the update phase, it predicts a scaling matrix to adaptively adjust the Kalman gain.

Our main contributions are summarized as follows.

- We propose Diff-Kalman, a hybrid Kalman-style filtering framework that integrates model-based priors with data-driven corrections through state increments and structured scaling.
- We build a Transformer-based state difference modeling module that captures temporal dynamic patterns and forecasts state increments using attention mechanisms. These increments are fused with model-predicted increments to refine prior state estimates.
- We propose a GRU-based Scaling Predictor (GSP) that generates scaling matrices to dynamically adjust the prior covariance and Kalman gain within the KF/EKF. This design improves posterior state estimation while preserving the recursive predict-update structure and the positive-semidefinite covariance property.
- We present a flexible and general framework based on KF/EKF, which applies to both linear and nonlinear systems. Experiments on linear MOT trajectories and nonlinear Lorenz dynamics demonstrate improved prior estimation accuracy and competitive posterior estimation performance compared with representative baselines.

2 Related Work

State estimation is a fundamental problem in dynamic system modeling, and data-driven Kalman filtering (KF) has recently gained increasing attention. Significant progress has been made by integrating KF with neural networks (NNs) in a synergistic manner [19–22]. These hybrid approaches can be broadly classified into two categories based on their integration strategy: External combination and Internal integration [23, 24].

External Combination. In this category, NNs operate outside the KF's recursive filtering process and provide auxiliary estimates or corrections while retaining the core filtering procedure. Tian et al. [25] and Liu et al. [26] respectively leveraged LSTMs or hybrid CNN-GRU models to process input sequences and compensate for system inaccuracies prior to the KF step. Zhao et al. [27] used a CNN-GRU framework to provide robust measurements during signal interruptions. Other approaches focus on parameter estimation. RL-AKF [28] employed reinforcement learning to estimate the process noise covariance matrix. More complex frameworks, such as the encoder-decoder by Shi et al. [29] and the cascaded architecture by Dido [30], have been proposed to integrate diverse information sources for enhanced state estimation.

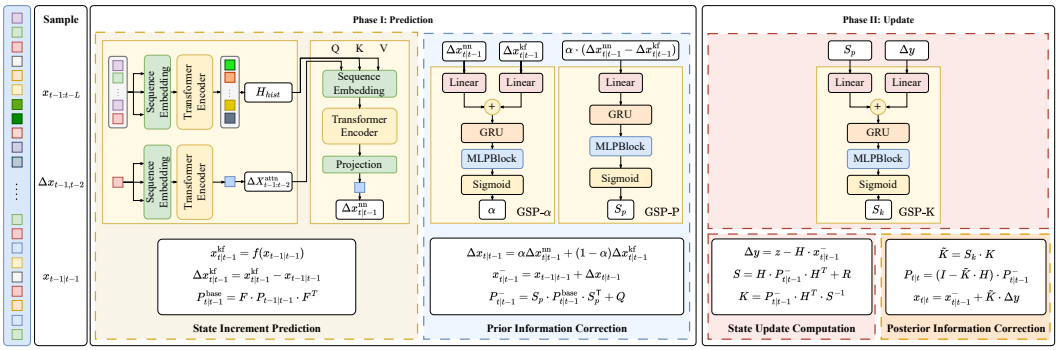


Figure 1: The overall architecture of **Diff-Kalman**, which consists of two main phases: the **Prediction Phase** and the **Update Phase**.

Internal Integration. This approach involves a tighter coupling, where NNs are embedded within the KF's iterative loop to replace or enhance core components. A key development in this area is KalmanNet (KNet) [14], which used a GRU-based architecture to directly learn the Kalman gain, addressing model mismatch and nonlinearities. Subsequent research has extended this concept from various perspectives: Split-KalmanNet (SKNet) [17] improved robustness to heterogeneous mismatches with a parallel network structure. Cholesky-KalmanNet (CKNet) [31] introduced a Cholesky decomposition to keep the covariance matrix positive definite. KalmanFormer [16] modeled Kalman gain dynamics using a Transformer. MAML-KalmanNet [32] incorporated a meta-learning paradigm to reduce the dependency on large datasets. Bayesian KalmanNet [18] introduced uncertainty quantification through Bayesian deep learning without needing extra domain knowledge. Other works have unified structured state-space models with dedicated neural modules [33–36] or incorporated the KF into the learning process to accelerate convergence [37].

3 Methodology

3.1 Problem Formulation

We consider a general stochastic state-space model:

$$x_t = f(x_{t-1}) + w_t, \quad z_t = h(x_t) + v_t, \tag{1}$$

where x_t and z_t denote the latent state and observation, $f(\cdot)$ and $h(\cdot)$ are the transition and observation functions, and w_t and v_t are the process and measurement noises with covariance matrices Q and R, respectively.

State estimation aims to infer the latent state from historical observations. The KF performs recursive prediction and update for linear systems, while the EKF extends it to nonlinear systems through local linearization. However, their performance may degrade under model mismatch and inaccurate noise assumptions.

3.2 Overview of Diff-Kalman

The overall architecture of Diff-Kalman is illustrated in Fig. 1. The framework consists of two distinct phases: the prediction phase and the update phase. In the prediction phase, Diff-Kalman fuses model-based priors with neural corrections to refine the prior state estimate and adjust the model-propagated covariance. In the update phase, it incorporates observations and uses the GRU-based Scaling Predictor (GSP) to adaptively correct the Kalman gain, improving the posterior state estimate.

To capture temporal dynamics, we define a historical state sequence in Equation 2 and its corresponding first-order difference sequence in Equation 3:

$$X_{t-1:t-L} = [x_{t-1}, x_{t-2}, \ldots, x_{t-L}]^\top \tag{2}$$

$$\Delta X_{t-j:t-k} = [(x_{t-j} - x_{t-j-1}), \ldots, (x_{t-k+1} - x_{t-k})]^\top \tag{3}$$

where $X_{t-1:t-L} \in \mathbb{R}^{L\times m}$ represents the historical state sequence within a fixed window, m is the state dimension, and L is the history sequence length (HSL). $\Delta X_{t-j:t-k} \in \mathbb{R}^{(k-j)\times m}$ represents the corresponding sequence of state differences from time step $t-j$ to $t-k$ and $j < k$.

3.3 Stage I: Prediction Phase

3.3.1 State Increment Prediction

The state increment prediction is performed through two parallel branches: a dynamic model-based branch and a neural network-based branch.

Dynamic Model-Based Branch In this branch, given the previous posterior state estimate $x_{t-1|t-1}$ and covariance $P_{t-1|t-1}$, the transition model $f(\cdot)$ predicts the prior state and propagates the covariance:

$$x^{\text{kf}}_{t|t-1} = f(x_{t-1|t-1}) \tag{4}$$

$$P^{\text{base}}_{t|t-1} = FP_{t-1|t-1}F^{\top} \tag{5}$$

where $x^{\text{kf}}_{t|t-1} \in \mathbb{R}^{m\times 1}$ denotes the prior state estimate at time t, and $P^{\text{base}}_{t|t-1} \in \mathbb{R}^{m\times m}$ denotes the model-propagated covariance component before adding the process noise covariance Q. For linear systems, the transition function simplifies to $f(x) = Fx$, where F is a constant state transition matrix. For nonlinear systems, the local transition matrix is obtained from the Jacobian of f evaluated at the previous state estimate: $F = J_f(x_{t-1|t-1})$.

The dynamic model-based state increment is calculated as:

$$\Delta x^{\text{kf}}_{t|t-1} = x^{\text{kf}}_{t|t-1} - x_{t-1|t-1} \tag{6}$$

where $\Delta x^{\text{kf}}_{t|t-1} \in \mathbb{R}^{m\times 1}$ represents the state increment predicted by the dynamic model. This branch offers a stable and interpretable prior estimate and a model-propagated covariance, serving as the foundation for subsequent neural correction.

Neural Network-Based Branch In this branch, we adopt a Transformer architecture to predict state increments. The model takes as input a fixed-length sequence of historical states $X_{t-1:t-L}$ and their corresponding state difference sequence $\Delta X_{t-j:t-k}$.

Leveraging the Self-Attention mechanism, the Transformer learns localized evolution patterns from the historical state sequence, enabling implicit modeling of the system's spatiotemporal dynamics:

$$H_{\text{hist}} = \text{Self-Attention}(X_{t-1:t-L}) \tag{7}$$

$$\Delta X^{\text{attn}}_{t-j:t-k} = \text{Self-Attention}(\Delta X_{t-j:t-k}) \tag{8}$$

where $H_{\text{hist}} \in \mathbb{R}^{L\times m}$ represents the temporal context of historical states and reflects potential local trends in system evolution. Meanwhile, $\Delta X^{\text{attn}}_{t-j:t-k}$ encodes the difference sequence and reflects short-term variations. To further capture the immediate dynamic trend, we set $j = 1$ and $k = 2$, such that $\Delta X^{\text{attn}}_{t-j:t-k}$ contains only the most recent difference token.

To predict the current state increment, we introduce a Cross-Attention mechanism. $\Delta X^{\text{attn}}_{t-j:t-k}$ is used as the query, while H_{hist} serves as both the key and value:

$$\Delta x^{\text{nn}}_{t|t-1} = \text{Cross-Attention}(\Delta X^{\text{attn}}_{t-j:t-k}, H_{\text{hist}}, H_{\text{hist}}) \tag{9}$$

where $\Delta x^{\text{nn}}_{t|t-1} \in \mathbb{R}^{m\times 1}$ represents the neural network-based prediction of the current state increment, integrating contextual information from historical states. Since the query contains a single token, the Cross-Attention output corresponds to a single time step.

3.3.2 Prior Information Correction

The prior information correction includes two components: the prior state correction and the prior covariance correction.

Prior State Correction To refine the estimate of the prior state, we introduce a GRU-based Scaling Predictor, denoted as GSP-α. This module dynamically generates a weighting vector to adaptively fuse the dynamic model-based and neural network-based state increments:

$$\boldsymbol{\alpha} = \text{GSP-}\alpha(\Delta x^{\text{nn}}_{t|t-1}, \Delta x^{\text{kf}}_{t|t-1}) \tag{10}$$

$$\Delta x_{t|t-1} = \boldsymbol{\alpha} \odot \Delta x^{\text{nn}}_{t|t-1} + (\mathbf{1} - \boldsymbol{\alpha}) \odot \Delta x^{\text{kf}}_{t|t-1} \tag{11}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{m\times 1}$ is a weighting vector generated through a sigmoid activation, $\mathbf{1} \in \mathbb{R}^{m\times 1}$ is an all-ones vector, and m denotes the state dimension. The fused state increment $\Delta x_{t|t-1} \in \mathbb{R}^{m\times 1}$ is computed in a component-wise manner from both sources, which allows a fine-grained correction of the prior state.

The fused state increment is added to the previous posterior estimate to obtain the final prior state estimate:

$$x^{-}_{t|t-1} = x_{t-1|t-1} + \Delta x_{t|t-1} \tag{12}$$

where $x^{-}_{t|t-1} \in \mathbb{R}^{m\times 1}$ represents the prior estimate of the current state after correction, and $x_{t-1|t-1} \in \mathbb{R}^{m\times 1}$ denotes the posterior estimate of the state at the previous time step.

Prior Covariance Correction To refine the prior covariance in the prediction phase, we introduce GSP-P, which outputs a diagonal scaling matrix to adjust the model-propagated covariance $P^{\text{base}}_{t|t-1}$, thereby obtaining the adjusted prior covariance.

GSP-P takes as input the discrepancy between the state increment predicted by the neural network $\Delta x^{\text{nn}}_{t|t-1}$ and that predicted by the dynamic model $\Delta x^{\text{kf}}_{t|t-1}$, weighted by the weighting vector $\boldsymbol{\alpha}$:

$$S_p = \text{GSP-P}\left(\boldsymbol{\alpha} \odot \left(\Delta x^{\text{nn}}_{t|t-1} - \Delta x^{\text{kf}}_{t|t-1}\right)\right) \tag{13}$$

where $S_p \in \mathbb{R}^{m\times m}$ denotes the diagonal scaling matrix, whose diagonal entries lie in $(0, 1.5)$, obtained by scaling the sigmoid outputs of GSP-P by 1.5.

The adjusted prior covariance matrix is obtained as follows:

$$P^{-}_{t|t-1} = S_p P^{\text{base}}_{t|t-1} S_p^{\mathsf{T}} + Q \tag{14}$$

where $P^{\text{base}}_{t|t-1}$ denotes the model-propagated covariance component as defined in Eq. (5), $P^{-}_{t|t-1} \in \mathbb{R}^{m\times m}$ denotes the resulting prior uncertainty matrix, and $Q \in \mathbb{R}^{m\times m}$ denotes the process noise covariance. Hereafter, we call $P^{-}_{t|t-1}$ the adjusted prior covariance, which serves as a PSD-preserving representation of prior uncertainty within the Kalman-style recursion, rather than a direct estimate of the classical estimation error covariance.

3.4 Stage II: Update Phase

3.4.1 State Update Computation

In the update phase, the corrected prior state estimate $x^{-}_{t|t-1}$ is combined with the current observation z_t to compute the innovation and its covariance estimate:

$$\Delta y_t = z_t - H x^{-}_{t|t-1} \tag{15}$$

$$S_t = H P^{-}_{t|t-1} H^{\mathsf{T}} + R \tag{16}$$

where $\Delta y_t \in \mathbb{R}^{n\times 1}$ denotes the observation residual and $S_t \in \mathbb{R}^{n\times n}$ denotes the innovation covariance matrix, computed from the adjusted prior covariance and measurement noise covariance. Here, n denotes the dimension of the observation states.

To further adapt the update process, we employ GSP-K to generate a scaling matrix:

$$S_k = \text{GSP-K}(S_p, \Delta y_t) \tag{17}$$

where $S_k \in \mathbb{R}^{m\times m}$ denotes the diagonal scaling matrix. Its diagonal entries lie in $(0, 1.5)$, obtained by scaling the sigmoid outputs of GSP-K by 1.5. This matrix adjusts the Kalman gain adaptively, enabling dimension-wise control of measurement influence during the update.

Since the Kalman gain inherently depends on the prior covariance and the observation noise, and S_p adjusts the model-propagated covariance component used to construct the prior covariance, it indirectly influences the Kalman gain. To achieve context-aware adjustment, GSP-K conditions on both S_p and Δy_t to adaptively amplify or attenuate the Kalman gain based on the prior scaling and current observation residual.

3.4.2 Posterior Information Correction

After obtaining the scaling matrix S_k, the Kalman gain is adaptively modulated as follows:

$$\tilde{K}_t = S_k P_{t|t-1}^{-} H^{\mathsf{T}} S_t^{-1} \tag{18}$$

where $\tilde{K}_t \in \mathbb{R}^{m \times n}$ denotes the scaled Kalman gain, and S_k enables dimension-wise modulation of the observation influence.

The posterior state estimate and its uncertainty matrix are then updated using the scaled gain:

$$x_{t|t} = x_{t|t-1}^{-} + \tilde{K}_t \Delta y_t \tag{19}$$

$$P_{t|t} = (I - \tilde{K}_t H) P_{t|t-1}^{-} (I - \tilde{K}_t H)^{\top} + \tilde{K}_t R \tilde{K}_t^{\top}. \tag{20}$$

where $x_{t|t} \in \mathbb{R}^{m \times 1}$ denotes the posterior state estimate after correction, and $P_{t|t} \in \mathbb{R}^{m \times m}$ denotes the corresponding posterior uncertainty matrix. This formulation preserves the recursive Kalman-style predict-update structure and the PSD property of the posterior uncertainty matrix. The PSD-preserving property of the prior and posterior uncertainty matrices is formally established in Lemma 1 in Appendix C.

3.5 Loss Function

3.5.1 State Loss

The state loss includes two parts: the prior state error and the posterior state error. The prior error supervises the refined prior estimate produced by the GSP-α module, which fuses dynamic state increments based on models and neural networks. The posterior error guides the posterior estimate adjusted by the GSP-K module, which adaptively scales the Kalman gain for observation integration.

$$\mathcal{L}_{\text{prior}}^{\text{state}} = \frac{1}{N} \sum_{i=1}^{N} \left\| x_{t|t}^{\text{gt},(i)} - x_{t|t-1}^{-,(i)} \right\|^2 \tag{21}$$

where $\mathcal{L}_{\text{prior}}^{\text{state}}$ represents the prior state loss, $x_{t|t}^{\text{gt},(i)} \in \mathbb{R}^{m \times 1}$ represents the ground truth state at time t, and $x_{t|t-1}^{-,(i)} \in \mathbb{R}^{m \times 1}$ represents the prior estimate of the current state after correction.

$$\mathcal{L}_{\text{post}}^{\text{state}} = \frac{1}{N} \sum_{i=1}^{N} \left\| x_{t|t}^{\text{gt},(i)} - x_{t|t}^{(i)} \right\|^2 \tag{22}$$

where $\mathcal{L}_{\text{post}}^{\text{state}}$ represents the posterior state loss, $x_{t|t}^{\text{gt},(i)} \in \mathbb{R}^{m \times 1}$ represents the ground truth state at time t, and $x_{t|t}^{(i)} \in \mathbb{R}^{m \times 1}$ represents the posterior estimate of the current state after corrections.

Together, these terms constrain both the learned prior correction and the observation-updated posterior estimate.

3.5.2 Covariance Loss

To improve the model capability to capture state uncertainty, Diff-Kalman introduces a covariance loss to regularize the prior uncertainty representation. Since the true prior error covariance is generally unavailable, we approximate it using the outer product of a single-sample state estimation error as a pseudo-label.

$$P_{t|t-1}^{\text{gt},(i)} = \left(x_{t|t}^{\text{gt},(i)} - x_{t|t-1}^{-,(i)} \right) \left(x_{t|t}^{\text{gt},(i)} - x_{t|t-1}^{-,(i)} \right)^{\top} \tag{23}$$

where $P_{t|t-1}^{\mathrm{gt},(i)} \in \mathbb{R}^{m\times m}$ represents a sample-wise pseudo-label constructed from the outer product of the difference between the ground truth state at time t and the corrected prior state estimate. It is not the true prior error covariance, which would require an expectation over prediction-error realizations. Instead, it serves as an auxiliary training target.

The covariance loss is then defined as:

$$\mathcal{L}_{\text{prior}}^{\text{cov}} = \frac{1}{N}\sum_{i=1}^{N}\left\| P_{t|t-1}^{\mathrm{gt},(i)} - P_{t|t-1}^{-,(i)} \right\|_F^2 \tag{24}$$

where $\mathcal{L}_{\text{prior}}^{\text{cov}}$ represents the prior covariance loss, and $P_{t|t-1}^{-,(i)} \in \mathbb{R}^{m\times m}$ denotes the adjusted prior uncertainty matrix for the i-th sample. In implementation, both the pseudo-label and the adjusted prior uncertainty matrix are Frobenius-normalized before computing this loss; for notational simplicity, the normalization is omitted from the equation.

This term acts as an auxiliary regularizer for the prior uncertainty representation rather than directly enforcing its absolute scale. We supervise only the prior uncertainty matrix, since the posterior uncertainty matrix depends jointly on the Kalman gain and observation noise and is not directly supervised. Diff-Kalman instead adjusts it indirectly through learned gain scaling.

3.5.3 Total Loss

The overall training objective combines the above losses into a weighted sum:

$$\mathcal{L}_{\text{total}} = \lambda_1 \cdot \mathcal{L}_{\text{prior}}^{\text{state}} + \lambda_2 \cdot \mathcal{L}_{\text{post}}^{\text{state}} + \lambda_3 \cdot \mathcal{L}_{\text{prior}}^{\text{cov}} \tag{25}$$

where λ_1, λ_2, and λ_3 are hyperparameters balancing the relative importance of each loss term. In our implementation, we set $\lambda_1 = \lambda_2 = \lambda_3 = 1$ for simplicity.

Combined, these losses guide the model to achieve accurate and robust state estimation while regularizing the prior uncertainty representation.

4 Experiments

In this section, the noise covariances Q and R are set to $\sigma_q^2\mathbf{I}$ and $\sigma_r^2\mathbf{I}$, respectively. The trajectory sequence length is denoted by T, and the History Sequence Length (HSL) by L. The Taylor expansion order is set to $J = 5$.

4.1 Experimental Setup

We conducted our experiments on an Ubuntu 20.04 system equipped with an Intel(R) Xeon(R) Gold 6326 CPU and an NVIDIA GeForce RTX 4090 D GPU. The software environment includes Python 3.10, PyTorch 2.5.1, and CUDA 12.1. For optimization, we utilize the Adam optimizer with a learning rate of 1×10^{-5} and a weight decay coefficient of 1×10^{-4}. For learning-based methods, results are averaged over five random seeds $\{10, 20, 30, 40, 50\}$, with the second row in the corresponding result tables reporting the standard deviation.

4.2 Datasets

MOT Trajectory Dataset. We extract target motion trajectories from the MOT17 dataset [38] as the evaluation data, referred to as MOT Trajectory. The MOT17 dataset comprises seven real-world video sequences captured in diverse indoor and outdoor public scenes, focusing primarily on pedestrian motion. We evaluate the performance of Diff-Kalman on this dataset under a linear dynamic system.

Lorenz Attractor Dataset. We use the motion trajectories of the Lorenz attractor as the evaluation dataset. The Lorenz attractor is a classic three-dimensional nonlinear dynamical system, widely used for validating nonlinear estimation and prediction algorithms due to its complex chaotic behavior. The highly nonlinear characteristics of this system provide an ideal environment for evaluating the performance and robustness of Diff-Kalman in handling complex nonlinear dynamic systems.

4.3 Main Results

Table 2: Test performance on the Lorenz Attractor dataset under different observation noise levels ($\sigma_r^2 \in \{1, 0.1, 0.01, 0.001\}$, corresponding to SNR levels of $\{0, 10, 20, 30\}$ dB, $\sigma_q^2 = 0.1$). Means and standard deviations over five random seeds are reported in separate rows for learning-based methods. Lower is better. **Bold** and underlined values indicate the best and second-best results, respectively.

Method	SNR = 0 [dB]		SNR = 10 [dB]		SNR = 20 [dB]		SNR = 30 [dB]	
	Prior	Posterior	Prior	Posterior	Prior	Posterior	Prior	Posterior
EKF [11]	-4.84	-6.26	-10.76	-12.67	-15.53	-20.96	-17.21	-30.45
UKF [12]	-4.92	-6.53	-10.80	-12.76	-15.53	-20.45	-17.20	-30.00
CKF [13]	-5.01	-6.85	-11.34	-13.00	-15.83	-20.98	-18.14	-30.85
KNet [14]	-9.31	-11.30	-12.73	-16.04	-16.98	-23.36	-17.40	-30.31
	±0.18	±0.16	±0.20	±0.17	±0.24	±0.21	±0.30	±0.26
SKNet [17]	-9.70	-11.42	-13.82	-17.28	-17.46	**-25.92**	-17.46	-31.75
	±0.13	±0.12	±0.14	±0.12	±0.17	±0.15	±0.20	±0.18
CKNet [31]	-10.23	-11.73	-15.34	-18.01	-16.53	-23.82	-17.46	-31.32
	±0.10	±0.09	±0.11	±0.10	±0.13	±0.11	±0.15	±0.13
Diff-Kalman	**-11.48**	**-11.97**	**-17.69**	**-18.54**	**-18.89**	-24.20	**-23.97**	**-32.21**
	±0.07	±0.06	±0.08	±0.05	±0.09	±0.07	±0.11	±0.09

Benchmarking on the MOT Trajectory As shown in Table 1, Diff-Kalman outperforms the classical Kalman filter (KF) [1], the adaptive KF baselines ML-KF [39] and IAE-KF [40], and the neural baselines KNet [14], CKNet [31], and SKNet [17] across all sequence lengths. While KNet improves posterior estimates by learning the Kalman gain, its priors suffer due to the lack of explicit dynamics. In contrast, Diff-Kalman augments prediction with GSP-α-based state-increment fusion, adjusts the propagated covariance via GSP-P, and refines updates via GSP-K gain scaling, achieving the lowest prior and posterior MSEs across all evaluated sequence lengths while maintaining stable performance over long sequences.

Table 1: MOT results under different trajectory sequence lengths. Results are reported as prior and posterior MSE [dB], where lower is better. Standard deviations are reported in separate rows. **Bold** and underlined values indicate the best and second-best results, respectively.

Method	$T = 20$		$T = 50$		$T = 100$	
	Prior	Posterior	Prior	Posterior	Prior	Posterior
KF [1]	20.03	17.43	19.93	17.32	19.07	16.35
ML-KF [39]	19.38	17.12	18.81	16.40	18.65	16.27
IAE-KF [40]	19.41	17.10	18.13	16.03	17.43	15.63
KNet [14]	21.54	16.60	21.50	16.04	20.74	14.84
	±0.12	±0.09	±0.14	±0.10	±0.18	±0.13
CKNet [31]	19.77	17.53	17.94	15.97	17.15	15.95
	±0.11	±0.06	±0.06	±0.03	±0.04	±0.07
SKNet [17]	19.03	17.16	17.52	16.21	16.94	15.63
	±0.04	±0.01	±0.01	±0.01	±0.01	±0.01
Diff-Kalman	**18.11**	**16.46**	**17.12**	**15.79**	**15.98**	**14.70**
	±0.05	±0.05	±0.08	±0.05	±0.25	±0.11

Benchmarking on the Lorenz Attractor As shown in Table 2, Diff-Kalman achieves the lowest prior MSE under all observation noise levels and the lowest posterior MSE in most settings. At SNR = 20 dB, SKNet achieves a lower posterior MSE than Diff-Kalman, while Diff-Kalman remains competitive. Its differential fusion and adaptive gain mechanisms contribute to accurate and robust state estimation in highly nonlinear and chaotic systems.

Ablation Study. Table 3 shows that each component contributes to Diff-Kalman. The Transformer increment predictor and GSP-α improve the prior estimate on MOT, but state correction alone does not converge on the Lorenz system. Adding GSP-P restores convergence and improves posterior accuracy, highlighting the importance of covariance correction after modifying the prior state. The full model with GSP-K achieves the best results on both datasets, confirming that coordinated correction of the prior state, prior covariance, and Kalman gain is necessary for stable and accurate filtering.

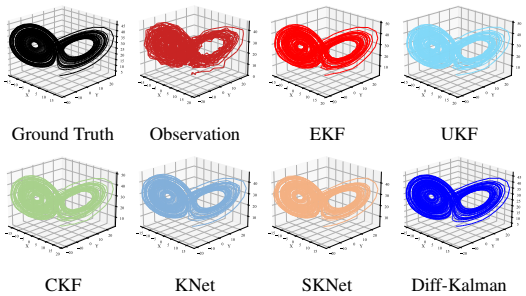


Figure 2: State estimation results on the Lorenz test set with $T = 3000$, $\sigma_r^2 = 1$, and $\sigma_q^2 = 0.1$.

Lorenz Qualitative Analysis. Figure 2 visualizes state estimates on the Lorenz test set. Classical filters such as EKF, UKF, and CKF exhibit

Table 3: Ablation study on model components across MOT Trajectory ($T = 50$) and Lorenz Attractor ($T = 3000$) datasets. ✓ indicates the component is used. Results are prior and posterior MSE [dB]. **Bold** numbers indicate the best results in each setting. "–" indicates that the corresponding model configuration failed to converge.

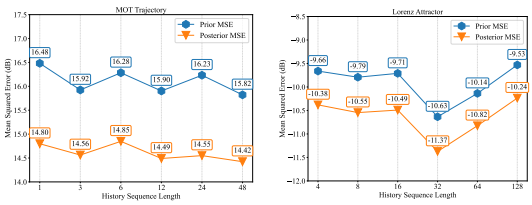
Method	Transformer	GSP-α	GSP-P	GSP-K	MOT Trajectory		Lorenz Attractor	
					Prior	Posterior	Prior	Posterior
A					19.93	17.31	-4.85	-6.27
B	✓				17.81	17.16	–	–
C	✓	✓			17.55	17.12	–	–
D	✓	✓	✓		17.31	15.96	-10.39	-10.56
E	✓	✓	✓	✓	**17.12**	**15.79**	**-11.48**	**-11.97**

Table 4: Robustness evaluation on the Lorenz dataset ($L = 32$, hidden $= 128$, Epochs=300). We test Q/R mismatch ($Q^* = \alpha_Q Q$, $R^* = \beta_R R$), F/H rotation mismatch, observation dropout, and their combined stress setting. The rotation angle is reported in degrees (Rot. deg.). Results are posterior MSE [dB], where lower is better. **Bold** values indicate the best results in each setting, "–" denotes no perturbation; "N/A" indicates methods that do not use Q/R inputs.

Setting	**F mis.**	**H mis.**	α_Q	β_R	**Dropout**	**Rot. deg.**	**KNet**	**SKNet**	**CKNet**	**Diff-Kalman**
Baseline	–	–	1.0	1.00	0.0	0	-11.29	-11.43	-11.67	**-11.82**
Q/R mis.	–	–	2.0	0.50	0.0	0	**-10.97**	N/A	N/A	-9.69
Severe Q/R mis.	–	–	4.0	0.25	0.0	0	**-10.41**	N/A	N/A	-9.00
F mis.	rotate	–	1.0	1.00	0.0	3	20.36	20.98	20.68	**15.52**
H mis.	–	rotate	1.0	1.00	0.0	3	-2.77	0.74	-5.13	**-5.35**
Dropout	–	–	1.0	1.00	0.2	0	-6.72	-8.61	-9.24	**-10.91**
Full Stress	rotate	rotate	4.0	0.25	0.2	3	23.88	23.42	20.29	**8.01**

large deviations under high observation noise, reflecting the difficulty of estimating chaotic nonlinear dynamics with local model approximations. Learning-aided baselines, including KNet and SKNet, better recover the overall attractor shape but still show local distortions and phase shifts near turning regions. Diff-Kalman produces trajectories that more closely follow the ground truth across the X, Y, and Z dimensions. This suggests that combining learned state-increment correction with covariance and gain scaling improves the ability of the recursive estimator to track nonlinear chaotic dynamics.

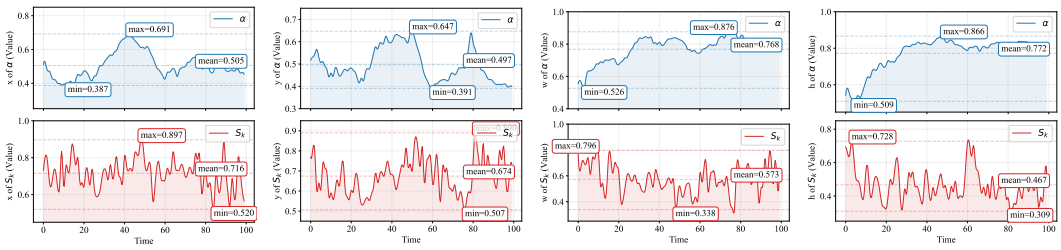
System Robustness Analysis. Table 4 evaluates robustness to unseen mismatch and dropout, with models trained on $T = 256$ windows under the Baseline setting and evaluated with frozen weights on $T = 3000$ sequences. For Q/R mismatch, Diff-Kalman is more sensitive because it explicitly uses the supplied Q and R in covariance and gain computation, whereas KNet mainly uses them for initialization rather than step-wise covariance propagation. SKNet and CKNet do not take Q/R as inference inputs in their evaluated protocols. However, under structural and observation-side shifts, Diff-Kalman exhibits better robustness than KNet, SKNet, and CKNet. In particular, under full stress, Diff-Kalman achieves an MSE of 8.01 dB, substantially lower than the competing methods. This result is consistent with the complementary roles of its correction mechanisms: state-increment fusion refines prior prediction, while GSP-P and GSP-K adjust the prior covariance and modulate the Kalman gain, respectively.



(a) MOT Trajectory Dataset (b) Lorenz Attractor Dataset

Figure 3: Effect of history sequence length on prior and posterior MSE.

Analysis of History Sequence Length. We vary the History Sequence Length L, which defines the input window of the self-attention module, and report the results in Figure 3. On MOT Trajectory, both prior and posterior MSEs decrease with larger L and saturate after $L = 3$, suggesting that a short history is sufficient to capture the relatively smooth target motion. On Lorenz, performance varies non-monotonically, with the best results achieved at $L = 32$, which approximately covers one oscillation cycle and captures key temporal dependencies. Longer histories may introduce outdated



(a) α, S_k for x (b) α, S_k for y (c) α, S_k for w (d) α, S_k for h

Figure 4: Temporal variation of α and S_k for MOT state components $\{x, y, w, h\}$.

or inconsistent information in chaotic dynamics, degrading attention-based estimation. Overall, Diff-Kalman benefits from historical context, but the optimal HSL is system-dependent.

Analysis of α and S_k on MOT. Figure 4 shows the temporal evolution of α and S_k for the MOT state variables $\{x, y, w, h\}$. The learned fusion weight α exhibits different behaviors for position and size components. For position states (x and y), α varies within a wider range, approximately 0.3–0.7, indicating that the model dynamically balances the motion prior and learned increment according to target movement. For size states (w and h), α becomes more stable after an initial adaptation period and stays around 0.77 with smaller variance, indicating a relatively stable preference for the learned increment in size estimation. In contrast, the diagonal entries of S_k change more frequently, mostly within 0.3–0.9, suggesting that the update step adaptively modulates the Kalman gain in response to measurement residuals. These trends show that Diff-Kalman learns dimension-wise adaptive fusion and gain scaling rather than applying a fixed correction across all dimensions.

5 Conclusion

In this paper, we propose Diff-Kalman, a hybrid Kalman-style filtering framework. Unlike existing data-driven methods that directly predict Kalman parameters, Diff-Kalman learns structured corrections in the form of state increments and scaling matrices. It uses a Transformer to predict state increments and GSP-based scaling to adjust prior uncertainty and adaptively adjust the Kalman gain, improving state estimation accuracy in both prediction and update phases. Experiments on linear and nonlinear systems demonstrate improved prior estimation and competitive posterior estimation performance across various settings, although robustness varies with the type of model and noise mismatch.

Acknowledgments and Disclosure of Funding

The work is supported by Natural Science Foundation of China No. 62676111 and Polar Intelligent Shipping Project No. D040308, and the HEU CISSE Decanal Innovation Fund for Ph.D. Students.

References

[1] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. doi: 10.1115/1.3662552.

[2] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[3] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision*, pages 1–21. Springer, 2022.

[4] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022.

[5] Oliviero Vouch, Andrea Nardin, Alex Minetto, Simone Zocca, Matteo Valvano, and Fabio Dovis. Aided Kalman filter models for gnss-based space navigation. *IEEE Journal of Radio Frequency Identification*, 2024.

[6] Chungeng Tian, Ning Hao, and Fenghua He. T-ESKF: Transformed Error-State Kalman Filter for Consistent Visual-Inertial Navigation. *IEEE Robotics and Automation Letters*, 2025.

[7] Hebin Wang, Quanbo Ge, Huaping Liu, and Xiaohu Yuan. Multi-robot active slam for observation fusion and attractor. *CAAI Transactions on Intelligent Systems*, 16(2):371–377, 2021.

[8] Zhiguang Feng and Sibo Yao. Dynamic event-triggered active disturbance rejection formation control for constrained underactuated auvs. *IEEE/CAA Journal of Automatica Sinica*, 12(2): 460–462, 2025.

[9] Ying Liu and Yuwen Li. Positioning accuracy improvement for target point tracking of robots based on Extended Kalman Filter with an optical tracking system. *Robotics and Autonomous Systems*, 179:104751, 2024.

[10] Tianyi Zhang, Wenhan Cao, Chang Liu, Tao Zhang, Jiangtao Li, and Shengbo Eben Li. Robust State Estimation for Legged Robots with Dual Beta Kalman Filter. *arXiv preprint arXiv:2411.11483*, 2024.

[11] Yuyang Zhou, Qichun Zhang, Hong Wang, Ping Zhou, and Tianyou Chai. Ekf-based enhanced performance controller design for nonlinear stochastic systems. *IEEE Transactions on Automatic Control*, 63(4):1155–1162, 2017.

[12] Eric A Wan and Rudolph Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 adaptive systems for signal processing, communications, and control symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[13] Ienkaran Arasaratnam and Simon Haykin. Cubature Kalman Filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, 2009. doi: 10.1109/TAC.2009.2019800.

[14] Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG Van Sloun, and Yonina C Eldar. KalmanNet: Neural network aided Kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2022.

[15] Nadav Cohen and Itzik Klein. Adaptive kalman-informed transformer. *Engineering Applications of Artificial Intelligence*, 146:110221, 2025.

[16] Siyuan Shen, Jichen Chen, Guanfeng Yu, Zhengjun Zhai, and Pujie Han. KalmanFormer: using transformer to model the Kalman Gain in Kalman Filters. *Frontiers in Neurorobotics*, 18: 1460255, 2025.

[17] Geon Choi, Jeonghun Park, Nir Shlezinger, Yonina C Eldar, and Namyoon Lee. Split-KalmanNet: A robust model-based deep learning approach for state estimation. *IEEE transactions on vehicular technology*, 72(9):12326–12331, 2023.

[18] Yehonatan Dahan, Guy Revach, Jindrich Dunik, and Nir Shlezinger. Bayesian KalmanNet: Quantifying uncertainty in deep learning augmented Kalman filter. *IEEE Transactions on Signal Processing*, 2025.

[19] Zhenhua Cui, Jiyong Dai, Jianrui Sun, Dezhi Li, Licheng Wang, and Kai Wang. Hybrid methods using neural network and Kalman filter for the state of charge estimation of lithium-ion battery. *Mathematical Problems in Engineering*, 2022(1):9616124, 2022.

[20] Xianglei Xing, Tianfu Wu, Song-Chun Zhu, and Ying Nian Wu. Inducing hierarchical compositional model by sparsifying generator network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14296–14305, 2020.

[21] Xianglei Xing, Kejun Wang, Tao Yan, and Zhuowen Lv. Complete canonical correlation analysis with application to multi-view gait recognition. *Pattern Recognition*, 50:107–117, 2016.

[22] Xianglei Xing, Ruiqi Gao, Tian Han, Song-Chun Zhu, and Ying Nian Wu. Deformable generator networks: unsupervised disentanglement of appearance and geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1162–1179, 2020.

[23] Yuting Bai, Bin Yan, Chenguang Zhou, Tingli Su, and Xuebo Jin. State of art on state estimation: Kalman filter driven by machine learning. *Annual Reviews in Control*, 56:100909, 2023.

[24] Wenzhe Zhai, Haomiao Li, Chaoqun Zheng, and Xianglei Xing. Multi-view gait recognition with joint local multi-scale and global contextual spatio-temporal features. *IEEE Transactions on Circuits and Systems for Video Technology*, 2024.

[25] Yalin Tian, Zengzeng Lian, Penghui Wang, Mengqi Wang, Zhe Yue, and Huabin Chai. Application of a long short-term memory neural network algorithm fused with Kalman filter in UWB indoor positioning. *Scientific reports*, 14(1):1925, 2024.

[26] Yaohua Liu, Qingsong Luo, and Yimin Zhou. Deep learning-enabled fusion to bridge gps outages for ins/gps integrated navigation. *IEEE Sensors Journal*, 22(9):8974–8985, 2022.

[27] Shuai Zhao, Yilan Zhou, and Tengchao Huang. A novel method for ai-assisted ins/gnss navigation system based on cnn-gru and ckf during gnss outage. *Remote Sensing*, 14(18):4494, 2022.

[28] Xile Gao, Haiyong Luo, Bokun Ning, Fang Zhao, Linfeng Bao, Yilin Gong, Yimin Xiao, and Jinguang Jiang. RL-AKF: An adaptive Kalman filter navigation algorithm based on reinforcement learning for ground vehicles. *Remote Sensing*, 12(11):1704, 2020.

[29] Zhuangwei Shi. Incorporating Transformer and LSTM to Kalman Filter with EM algorithm for state estimation. *arXiv preprint arXiv:2105.00250*, 2021.

[30] Kunyi Zhang, Chenxing Jiang, Jinghang Li, Sheng Yang, Teng Ma, Chao Xu, and Fei Gao. Dido: Deep inertial quadrotor dynamical odometry. *IEEE Robotics and Automation Letters*, 7 (4):9083–9090, 2022.

[31] Minhyeok Ko and Abdollah Shafieezadeh. Cholesky-KalmanNet: Model-Based Deep Learning with Positive Definite Error Covariance Structure. *IEEE Signal Processing Letters*, 2024.

[32] Shanli Chen, Yunfei Zheng, Dongyuan Lin, Peng Cai, Yingying Xiao, and Shiyuan Wang. MAML-KalmanNet: A Neural Network-Assisted Kalman Filter Based on Model-Agnostic Meta-Learning. *IEEE Transactions on Signal Processing*, 2025.

[33] Dong Yang, Fei Jiang, Wei Wu, Xuefei Fang, and Muyong Cao. Low-complexity acoustic echo cancellation with neural Kalman filtering. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[34] Lincan Li, Hanchen Wang, Wenjie Zhang, and Adelle Coster. Stg-mamba: Spatial-temporal graph learning via selective state space model. *arXiv preprint arXiv:2403.12418*, 2024.

[35] Wenzhe Zhai, Xianglei Xing, Mingliang Gao, and Qilei Li. Zero-shot object counting with vision-language prior guidance network. *IEEE Transactions on Circuits and Systems for Video Technology*, 2024.

[36] Wenzhe Zhai, Xianglei Xing, Jing-an Cheng, and Gwanggil Jeon. Privacy-preserving object counting in lightweight intelligent transportation system. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[37] Peter G Chang, Gerardo Durán-Martín, Alexander Y Shestopaloff, Matt Jones, and Kevin Murphy. Low-rank extended Kalman filtering for online learning of neural networks from streaming data. *arXiv preprint arXiv:2305.19535*, 2023.

[38] Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, Stefan Roth, and Laura Leal-Taixé. MOTChallenge: A benchmark for single-camera multiple target tracking. *International Journal of Computer Vision*, 129(4):845–881, 2021.

[39] Zili Wang, Jianhua Cheng, Bing Qi, Sixiang Cheng, and Sicheng Chen. An adaptive Kalman filtering algorithm based on maximum likelihood estimation. *Measurement Science and Technology*, 34(11):115114, 2023.

[40] Raman Mehra. On the identification of variances and adaptive Kalman filtering. *IEEE Transactions on automatic control*, 15(2):175–184, 1970.

A Additional Architectural Details

A.1 GRU-Based Scaling Predictor

To enable adaptive fusion and scaling in Diff-Kalman, we introduce a GRU-based Scaling Predictor (GSP). In this work, GSP refers to a family of GRU-based modules, including GSP-α for state-increment fusion, GSP-P for prior covariance correction, and GSP-K for Kalman gain modulation. These modules share the same basic architecture, but differ in their output ranges.

Given a set of input variables X_i, each input is first projected by a linear layer and then summed:

$$F = \sum_{i=1}^{N_{\text{in}}} \text{Linear}(X_i), \tag{26}$$

where N_{in} is the number of input variables. The fused feature is then processed by a GRU and an MLP block:

$$F_{\text{gru}}, h_t = \text{GRU}(F, h_{t-1}), \tag{27}$$

$$z = \text{MLPBlock}(F_{\text{gru}}). \tag{28}$$

For the three modules, the corresponding MLP outputs are denoted by z_α, z_p, and z_k, respectively.

For GSP-α, the sigmoid output is directly used as the component-wise fusion weight:

$$\boldsymbol{\alpha} = \sigma(z_\alpha), \qquad \boldsymbol{\alpha} \in (0,1)^m. \tag{29}$$

where m denotes the state dimension, and $\boldsymbol{\alpha}$ is an m-dimensional weighting vector that assigns an individual fusion weight to each state component. Thus, the fused state increment is computed as

$$\Delta x_{t|t-1} = \boldsymbol{\alpha} \odot \Delta x^{\text{nn}}_{t|t-1} + (\mathbf{1} - \boldsymbol{\alpha}) \odot \Delta x^{\text{kf}}_{t|t-1} \tag{30}$$

This guarantees a component-wise convex combination of the neural and model-based increments.

For GSP-P and GSP-K, the sigmoid outputs are scaled by a fixed bound w_s and converted into diagonal matrices:

$$S_p = \text{diag}(w_s \sigma(z_p)), \qquad S_k = \text{diag}(w_s \sigma(z_k)). \tag{31}$$

We set $w_s = 1.5$, so the diagonal entries of S_p and S_k lie in $(0, 1.5)$.

The diagonal design avoids unconstrained full-matrix regression and reduces the number of predicted scaling coefficients from $O(m^2)$ to $O(m)$. This constrained parameterization can help mitigate overfitting and improve training stability under limited data, while preserving dimension-wise adaptive correction. The GRU is adopted as a lightweight sequential module for generating scaling factors, complementing the Transformer-based state-increment predictor.

A.2 Rationale for Correction Information

In this section, we explain the rationale behind selecting state increments and scaling matrices as the correction information.

State Increments We predict state increments rather than full future state values so that the model can focus on local temporal changes instead of learning the entire state transition from scratch. By using first-order differences, the model learns local state changes and combines them with the prior information from the dynamic model. This design improves prior state estimation while retaining an explicit contribution from the underlying dynamic model.

Scaling Matrices We predict scaling matrices to adjust the prior uncertainty matrix and Kalman gain instead of directly predicting unconstrained full-matrix corrections. This design reduces the number of learnable parameters and avoids unconstrained matrix prediction. In particular, the diagonal parameterization enables dimension-wise adaptive scaling, while the positive-semidefinite (PSD) property of the prior uncertainty matrix is preserved by the congruence-form update, as established in the structure preservation analysis.

In contrast, using an unconstrained additive form such as $P_{\text{new}} = P_{\text{old}} + \Delta P$ does not guarantee that the corrected matrix remains positive semidefinite. Since an unconstrained correction matrix ΔP may introduce sufficiently negative directions, the resulting matrix P_{new} can become indefinite, thereby violating the PSD requirement.

B Experimental Settings

B.1 Metrics

Mean Squared Error (MSE) To evaluate the overall estimation performance, we follow the evaluation metrics used in [14, 17, 31] and compute the Mean Squared Error (MSE) in decibels (dB). Specifically, we define:

$$\text{MSE[dB]} = 10 \log_{10} \left(\frac{1}{NTm} \left\| \mathbf{X} - \hat{\mathbf{X}} \right\|_F^2 \right) \tag{32}$$

where $\mathbf{X}, \hat{\mathbf{X}} \in \mathbb{R}^{N \times T \times m}$ represent the ground-truth and estimated state sequences, respectively, and $\| \cdot \|_F$ denotes the Frobenius norm. This metric measures the average squared error over all samples, time steps, and state dimensions, where lower values indicate higher estimation accuracy.

C Theoretical Analysis

C.1 Kalman Structure Preservation Analysis

In this section, we demonstrate that the proposed Diff-Kalman framework retains the recursive predict-update structure and PSD covariance form, while enhancing it with learnable correction terms.

Lemma 1 (Structure-Preserving Property of Diff-Kalman)**.** *Assume that $P_{t|t-1}^{base} \succeq 0$, $Q \succeq 0$, and $R \succ 0$. Let $S_p, S_k \in \mathbb{R}^{m \times m}$ be diagonal scaling matrices with strictly positive entries. Diff-Kalman computes the adjusted prior covariance as*

$$P_{t|t-1}^{-} = S_p P_{t|t-1}^{base} S_p^\top + Q, \tag{33}$$

and the innovation covariance and scaled Kalman gain as

$$\mathcal{S}_t = H P_{t|t-1}^{-} H^\top + R, \tag{34}$$

$$\tilde{K}_t = S_k P_{t|t-1}^{-} H^\top \mathcal{S}_t^{-1}. \tag{35}$$

If the posterior covariance is updated by the Joseph form

$$P_{t|t} = (I - \tilde{K}_t H) P_{t|t-1}^{-} (I - \tilde{K}_t H)^\top + \tilde{K}_t R \tilde{K}_t^\top, \tag{36}$$

then $P_{t|t-1}^{-} \succeq 0$ and $P_{t|t} \succeq 0$. Therefore, Diff-Kalman preserves the recursive predict-update form and the positive-semidefinite covariance structure of Kalman filtering.

Proof. First, consider the corrected prior covariance. For any vector $v \in \mathbb{R}^m$,

$$v^\top S_p P_{t|t-1}^{\text{base}} S_p^\top v = (S_p^\top v)^\top P_{t|t-1}^{\text{base}} (S_p^\top v) \geq 0, \tag{37}$$

because $P_{t|t-1}^{\text{base}} \succeq 0$. Since $Q \succeq 0$, we have

$$v^\top P_{t|t-1}^{-} v = v^\top S_p P_{t|t-1}^{\text{base}} S_p^\top v + v^\top Q v \geq 0. \tag{38}$$

Thus, $P_{t|t-1}^{-} \succeq 0$.

Next, consider the posterior covariance. For any vector $v \in \mathbb{R}^m$, using the Joseph-form update gives

$$v^\top P_{t|t} v = v^\top (I - \tilde{K}_t H) P_{t|t-1}^{-} (I - \tilde{K}_t H)^\top v + v^\top \tilde{K}_t R \tilde{K}_t^\top v \tag{39}$$

$$= \big((I - \tilde{K}_t H)^\top v\big)^\top P_{t|t-1}^{-} \big((I - \tilde{K}_t H)^\top v\big) + (\tilde{K}_t^\top v)^\top R (\tilde{K}_t^\top v) \geq 0, \tag{40}$$

because $P_{t|t-1}^{-} \succeq 0$ and $R \succ 0$. Hence, $P_{t|t} \succeq 0$.

The diagonal positivity of S_p and S_k ensures that the learned modules act as coordinate-wise rescaling operators rather than unconstrained matrix replacements. Consequently, the prediction and update steps retain the standard recursive filtering form while introducing structured learned corrections. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions and the evaluated settings. The claims are limited to MOT trajectory data and Lorenz dynamics, and additional limitations are discussed in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The supplementary material discusses limitations, including the limited evaluation domains, diagonal scaling assumption, lack of classical MMSE optimality after learned gain scaling, and additional computational cost.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The structure-preserving property and PSD covariance preservation are formally stated with assumptions and proved in the supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper and supplementary material provide the model architecture, datasets, noise settings, optimizer, hyperparameters, seeds, hardware, and evaluation protocol needed to reproduce the main results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The implementation is publicly available on GitHub, together with configuration files and instructions for reproducing the main experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the training and test settings, including datasets, sequence lengths, noise settings, optimizer, learning rate, weight decay, random seeds, model dimensions, and compute environment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report mean performance and standard deviations over five random seeds for the learning-based methods. These standard deviations reflect variability due to random initialization and training stochasticity.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The supplementary material reports the hardware and software environment used for experiments, including CPU, GPU, Python, PyTorch, and CUDA versions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and believe the research conforms to it. The work uses public or synthetic trajectory data and does not involve human-subject studies.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The work may have positive impacts for robust state estimation in tracking, navigation, and sensor fusion. Potential negative impacts include misuse in surveillance-oriented tracking systems; these risks are application-dependent and should be considered before deployment.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk models, scraped datasets, generative models, or other assets requiring special safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the datasets and prior methods used in the experiments. The MOT trajectory data are derived from MOTChallenge/MOT17, and the Lorenz data are synthetically generated from the specified dynamical system.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper releases an anonymized code package with configuration files and documentation for reproducing the experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The research does not involve human subjects, so IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development does not involve LLMs as an important, original, or non-standard component. Any use of language tools, if any, was limited to writing or editing support and did not affect the scientific methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.