drones

Article

Mamba-2-Based Continuous-Discrete Extended Kalman Filter for Passive UAV Bearings-Only Tracking with Uncertain Measurement Noise

Hao Wu, Guoxu Zeng *, Ali Mehmood, Yijie Zhao, Chaoqi Li and Mingbo Yang

School of Information and Navigation, Air Force Engineering University, Xi'an 710077, China; wuhaostudy@163.com (H.W.); amehmood425@gmail.com (A.M.); zhaoyijie030426@163.com (Y.Z.); lichaoqistudy@126.com (C.L.); ymb_study@163.com (M.Y.)

* Correspondence: zgx1997416@163.com

Highlights

What are the main findings?

- The proposed Mamba-2 block utilizes the state-space model that captures long-term dependencies in innovation sequences, enabling direct and accurate prediction of time-varying measurement noise variance.
- The proposed MSCDEKF framework uniquely integrates a square-root continuous-discrete extended Kalman filter with the Mamba-2 model, simultaneously ensuring both accuracy and robustness in UAV-based bearings-only tracking, maintaining stable and precise results while countering time-varying measurement noise.

What are the implications of the main finding?

- The MSCDEKF eliminates the need for manual parameter tuning and avoids posterior approximation errors, offering a more practical and accurate solution for real-world UAV BOT under unknown and time-varying noise conditions.
- The integration of Mamba-2 with square-root continuous-discrete EKF provides a robust, numerically stable, and computationally efficient framework that can be extended to other UAV problems with low observability and complex noise environments.

Abstract

Using UAVs for bearings-only tracking (BOT) of moving targets is a key challenging issue due to weak observability and time-varying measurement noises. To overcome these limitations, we introduce a hybrid framework that integrates a square-root continuous-discrete extended Kalman filter (MSCDEKF) with the Mamba-2 neural network. The MSCDEKF improves estimation accuracy and numerical stability under weak observability conditions by adopting continuous-time state update while ensuring numerical stability through square-root covariance implementation. Meanwhile, the state-space modeling of the Mamba-2 network provides superior continuous measurement noise prediction compared to conventional recurrent neural network approaches by explicitly capturing signal evolution dynamics. Experimental validation in constant velocity (CV) and constant turn (CT) UAV-based BOT scenarios demonstrates that our framework achieves the following: (1) at least a 34.5% improvement in state estimation accuracy (measured by average root-mean-square error, ARMSE) compared with variational Bayesian filters; (2) significant improvements in noise variance estimation; and (3) parameter-free operation with minimal manual tuning. This work establishes a different paradigm for robust UAV-based BOT in complex environments characterized by unknown and time-varying noise conditions.

Check for updates

Academic Editor: Anastasios Dimou

Received: 27 May 2026
Revised: 10 July 2026
Accepted: 12 July 2026
Published: 14 July 2026

Keywords: UAV-borne passive sensing; bearings-only tracking; continuous–discrete filtering; Mamba-2 network; robust filtering

1. Introduction

Unmanned aerial vehicles (UAVs), also called drones, are widely used in the field of target tracking due to their mobility and flexibility, which allow UAVs to surveil large areas and ignore the terrain environment, significantly enhancing tracking performance [1]. Bearings-only tracking (BOT) from UAV platforms, which relies solely on passive bearing observations to track a target, also demonstrates certain advantages in terms of UAV's anti-interference capability, stealth, and survivability in hostile or electromagnetically contested environments [2–4]. However, the BOT problems on UAVs remain a highly challenging task due to its strong nonlinearity and potentially low observability [5–7]. The Kalman filter (KF) is commonly used for linear target tracking [8], while the extended Kalman filter (EKF) has been widely applied to nonlinear cases. However, the conventional EKF struggles with insufficient observability in UAV-based BOT scenarios. The EKF incorporating continuous-discrete system models (CD-EKF) improves state prediction accuracy by better aligning with real-world conditions, where target states evolve continuously while measurements occur discretely [9,10]. This enhancement significantly improves state estimation accuracy under the low observability conditions characteristic in BOT tasks from UAV platforms. Furthermore, square-root implementations of the EKF are widely adopted due to their numerical stability [11]. Applying a square-root formulation to the CD-EKF enhances the convergence properties of BOT tracking [12]. Therefore, our study primarily operates within the framework of square-root-based CD-EKF.

In real-world scenarios, the statistical properties of measurement noise are often unknown and time-varying, while accurate noise characterization is critical for filter performance. Many studies have explored methods to estimate the measurement noise covariance matrix. M-estimator-based filters, such as those employing the Huber function, correntropy, and statistical similarity measures, improve robustness by detecting outliers and adjusting noise covariance weights accordingly. However, these approaches rely on accurate initial noise covariance assumptions and struggle with time-varying noise. The Sage–Husa adaptive Kalman filter (SHAKF) is a real-time maximum a posteriori noise estimator capable of updating statistical noise characteristics adaptively [13]. However, SHAKF does not guarantee convergence to the true noise covariance, which may lead to filter divergence.

The variational Bayesian-based adaptive Kalman filter (VBAKF) is a Bayesian approach that estimates the error covariance matrix by selecting an appropriate conjugate prior distribution, enabling effective estimation of time-varying noise covariance [14,15]. Huang et al. [16] proposed a novel variational Bayesian adaptive Kalman filter (N-VBAKF) for linear systems with inaccurate and time-varying process and measurement noise covariance matrices, achieving good results. Despite its advantages, N-VBAKF requires multiple fixed-point iterations, which limits its practical applicability. To overcome this issue, Qiao et al. [17] introduced a modified strong tracking sliding window variational adaptive Kalman filter (MST-SWVAKF), which integrates a fading factor with a sliding window-based VBAKF. This approach eliminates the need for fixed-point iterations and adaptively adjusts the window length to balance computational efficiency and tracking performance. While these VBAKFs effectively estimate time-varying noise, they remain computationally inconvenient and rely on numerous empirically tuned prior parameters, reducing their practicality. Particularly, inappropriate parameter configurations can severely compromise the estimation performance. In addition, the VB method inherently introduces approximation errors in posterior distribution estimation, ultimately degrading measurement accuracy.

The rapid advancement of deep learning has inspired new approaches to this problem. Several neural network architectures designed for sequential data processing, such as vanilla recurrent neural networks (RNNs) [18], long short-term memory (LSTM) networks [19,20], gated recurrent units (GRUs) [21], and Transformers [22], have been explored for discrete observation processing and state estimation in an end-to-end, model-free manner. However, these approaches require extensive training data and a large number of model parameters due to the absence of explicit prior knowledge and models [23]. Consequently, numerous studies have investigated the integration of filtering techniques with deep learning methodologies. For instance, Xu et al. [24] proposed EKFNet, a framework capable of automatically estimating process and measurement noise covariance pairs from measurement data. But their approach involves intensive training processes and exhibits limited generalization capabilities. Xu et al. [25] later introduced a Transformer-based KF capable of adaptively adjusting measurement noise covariance. This approach leverages a residual denoising autoencoder (RDAE) to process raw ephemeris data, enhancing generalization ability. However, the RDAE and Transformer models employed in the method are specifically designed for high-dimensional data, such as ephemeris data, making their application to the BOT scenario challenging due to the inherently low-dimensional and continuously varying nature of measurement noise. Revach et al. [26] developed KalmanNet, an interpretable, and computationally efficient RNN-based state estimator. KalmanNet utilizes a deep neural network (DNN) to directly compute the Kalman gain, eliminating the need for explicit prior models. However, this design prevents it from leveraging existing noise statistical models, and the low observability inherent to BOT scenarios makes it difficult for RNNs to extract the necessary informative features. More recently, Hao et al. [27] introduced RKFNet, which combines robust Kalman filtering with deep learning techniques. While RKFNet effectively incorporates prior information such as process noise covariance and dynamical models, it is primarily tailored for non-Gaussian heavy-tailed noise rather than time-varying noise, limiting its suitability for BOT tracking.

In this work, we propose a novel square-root CD-EKF framework to address the UAV BOT problem under time-varying measurement noise. By integrating the square-root CD-EKF with the Mamba-2 model [28,29], our approach achieves robust and highly accurate tracking performance. The Transformer scales quadratically along the input length, and it is incapable of inference out of the input window. As a sub-quadratic model, Mamba-2 offers higher computational efficiency compared to Transformer and enables sequential input processing similar to RNNs without finite input windows. Inspired by control methods such as the KF, Mamba-2 employs structured state-space model (SSM) to efficiently capture long-range dependencies with linear-time complexity. Prior SSMs models are input-invariant so they cannot perform as well as Transformer. Mamba-2 constructs SSM parameters based on inputs, achieving an effect analogous to the attention mechanism of Transformer. Consequently, Mamba-2 outperforms traditional RNNs and prior SSM models in modeling continuous signal processes, facilitating more precise measurement noise estimation.

Furthermore, the square-root CD-EKF framework mitigates the challenges posed by low observability in UAV BOT scenarios. Our proposed hybrid approach leverages the strengths of both the Mamba-2 model and the square-root CD-EKF framework, offering significant advantages over existing methods. Compared with VB-based approaches, our method eliminates the need for extensive parameter tuning, making it more practical. Moreover, it avoids the estimation errors associated with posterior approximations in VB methods, resulting in improved accuracy. Compared with end-to-end DNN approaches, our method features lower model complexity, reduced computational and training costs, and greater interpretability.

The main contributions of this work are summarized as follows:

1. We developed a systematic approach for generating training datasets specifically designed for diverse BOT application involving drone adoption, ensuring robust performance across varying conditions.
2. We propose a dual-execution strategy for Mamba-2: during training, it utilizes batched processing with parallel observation sequences, similar to Transformer, to enhance training efficiency. During UAV tracking, it adopts an RNN-like recurrent processing scheme with single-step observations, ensuring computational efficiency in real-time filtering.
3. We designed a square-root CD-EKF framework integrated with Mamba-2 (MSCDEKF). The MSCDEKF achieves stable and accurate performance when applied to UAV BOT tasks under low observability and time-varying measurement noise.

To evaluate the effectiveness of the proposed method, we conduct experiments on BOT tasks from an UAV platform using constant velocity (CV) and constant turn (CT) models with time-varying measurement noise. The performance of the proposed approach is compared against various benchmark filtering methods.

The remainder of this paper is structured as follows: Section 2 introduces the system model and CD-EKF framework. Section 3 details the architecture of the Mamba-2 network and the dataset generation process. Section 4 presents the proposed algorithm. Section 5 provides experimental results and analysis. Finally, Section 6 concludes the study.

2. CD-EKF Framework for UAV Passive BOT

Consider a nonlinear continuous-time stochastic state model represented by an Itô-type stochastic differential equation (SDE):

$$\mathrm{d}\boldsymbol{x}(t) = \boldsymbol{F}(\boldsymbol{x}(t), t)\mathrm{d}t + \boldsymbol{G}(t)\mathrm{d}\boldsymbol{\omega}t \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ denotes an n-dimensional state vector, $\boldsymbol{F}(\boldsymbol{x}(t), t) = \boldsymbol{f}'(\boldsymbol{x}(t), t) : \mathbb{R}^n \to \mathbb{R}^n$ denotes the drift function governing the dynamic behavior of the target, $\boldsymbol{G}(t)$ is a diffusion matrix of size n, and $\boldsymbol{\omega}(t)$ is a Wiener process (Brownian motion) with zero mean and covariance matrix $\boldsymbol{Q}$.

The CD-EKF framework relies on the predicted values of the state mean and error covariance matrix, which satisfy the moment differential equations (MDEs) [30]:

$$\hat{\boldsymbol{x}}'(t) = \boldsymbol{F}(\hat{\boldsymbol{x}}(t), t) \tag{2}$$

$$\boldsymbol{P}'(t) = \boldsymbol{J}(\hat{\boldsymbol{x}}(t))\boldsymbol{P}(t) + \boldsymbol{P}'(t)\boldsymbol{J}^T(\hat{\boldsymbol{x}}(t)) + \boldsymbol{G}(t)\boldsymbol{Q}(t)\boldsymbol{G}^T(t) \tag{3}$$

$$\boldsymbol{J}(\hat{\boldsymbol{x}}(t)) := \frac{\partial \boldsymbol{F}(\hat{\boldsymbol{x}}(t), t)}{\partial \hat{\boldsymbol{x}}} \tag{4}$$

where $\boldsymbol{P}$ denotes the a priori error covariance and $\boldsymbol{J}(\hat{\boldsymbol{x}}(t))$ denotes the Jacobian of $\boldsymbol{F}(\hat{\boldsymbol{x}}(t), t)$.

The measurement $\boldsymbol{z}_k$ collected by the sensor and the state $\boldsymbol{x}(t_k)$ follow the discrete measurement model:

$$\boldsymbol{z}_k = \boldsymbol{h}_{ms}(\boldsymbol{x}(t_k)) + \boldsymbol{v}_k \tag{5}$$

where k represents the discrete-time index, $\boldsymbol{h}_{ms} : \mathbb{R}^n \to \mathbb{R}^{m_z}$ represents the measurement function, and $\boldsymbol{v}_k$ represents the measurement noise, which follows a Gaussian distribution $\boldsymbol{v}_k \sim N(0, \boldsymbol{R})$. Here, $\boldsymbol{R}$ is a diagonal covariance matrix.

To obtain an accurate numerical solution to the MDE (2) at the next sampling time t_{k+1}, the CD-EKF employs the fourth-order Runge–Kutta (RK) method, given by the following:

$$\begin{cases} \hat{\boldsymbol{x}}(t_{k+1|k}) = \hat{\boldsymbol{x}}(t_{k|k}) + \frac{h}{6}(\boldsymbol{k_1} + \boldsymbol{k_2} + \boldsymbol{k_3} + \boldsymbol{k_4}) \\ \boldsymbol{k_1} = \boldsymbol{F}(\hat{\boldsymbol{x}}(t_{k|k}), t_k) \\ \boldsymbol{k_2} = \boldsymbol{F}(\hat{\boldsymbol{x}}(t_{k|k}) + \frac{h}{2}\boldsymbol{k_1}, t_k + \frac{h}{2}) \\ \boldsymbol{k_3} = \boldsymbol{F}(\hat{\boldsymbol{x}}(t_{k|k}) + \frac{h}{2}\boldsymbol{k_2}, t_k + \frac{h}{2}) \\ \boldsymbol{k_4} = \boldsymbol{F}(\hat{\boldsymbol{x}}(t_{k|k}) + h\boldsymbol{k_3}, t_k + h) \end{cases} \tag{6}$$

where $h := t_{k+1} - t_k$ denotes the time step size. $\hat{\boldsymbol{x}}(t_{k+1|k})$ denotes the predicted state.

For the normal EKF, the error covariance can be calculated as follows:

$$\boldsymbol{P}(t_{k+1|k}) = \boldsymbol{F}(\hat{\boldsymbol{x}}(t_{k|k}))\boldsymbol{P}(t_{k|k})\boldsymbol{F}^T(\hat{\boldsymbol{x}}(t_{k|k})) + \boldsymbol{Q} \tag{7}$$

In the CD-EKF framework, predicting the error covariance requires computing the intermediate state $\hat{\boldsymbol{x}}(t_{k+1/2})$. This is achieved using Mazzoni's interpolation rule, expressed as follows [31]:

$$\hat{\boldsymbol{x}}(t_{k+1/2}) = \frac{1}{2}\left[\hat{\boldsymbol{x}}(t_{k|k}) + \hat{\boldsymbol{x}}(t_{k+1|k}) - \frac{h^2}{4}\boldsymbol{J}(\hat{\boldsymbol{x}}(t_{k|k}))\boldsymbol{F}(\hat{\boldsymbol{x}}(t_{k|k}))\right] \tag{8}$$

where $t_{k+1/2} := t_k + h/2$ represents the midpoint of a time step.

Subsequently, the intermediate variable matrices $\boldsymbol{L}(t_{k+1/2})$ and $\boldsymbol{M}(t_{k+1/2})$ are computed as follows:

$$\boldsymbol{L}(t_{k+1/2}) := \left[\boldsymbol{I}_n - \frac{h}{2}\boldsymbol{J}(\hat{\boldsymbol{x}}(t_{k+1/2}))\right]^{-1} \tag{9}$$

$$\boldsymbol{M}(t_{k+1/2}) := \boldsymbol{L}(t_{k+1/2})\left[\boldsymbol{I}_n + \frac{h}{2}\boldsymbol{J}(\hat{\boldsymbol{x}}(t_{k+1/2}))\right] \tag{10}$$

where $\boldsymbol{I}_n$ denotes an n-dimensional identity matrix.

The square-root CD-EKF propagates only the square root of the error covariance, $\boldsymbol{S}(t_{k|k})$, over time, ensuring numerical stability and preserving the positive definiteness of the covariance matrix. The relationship between $\boldsymbol{S}(t_{k|k})$ and the error covariance $\boldsymbol{P}(t_{k|k})$ is given by the following:

$$\boldsymbol{P}(t_{k|k}) = \boldsymbol{S}(t_{k|k})\boldsymbol{S}^T(t_{k|k}) \tag{11}$$

By maintaining a positive definite error covariance matrix, the square-root CD-EKF enhances numerical robustness and ensures reliable state estimation.

During the time update step, $\boldsymbol{S}(t_{k+1|k})$ is predicted using the following formulation [11]:

$$Qr\begin{bmatrix} \boldsymbol{S}(t_{k|k})\boldsymbol{M}^T(t_{k+1/2}) \\ \boldsymbol{Q}^{\frac{T}{2}}(t_{k+1/2})\boldsymbol{G}^T(t_{k+1/2})\boldsymbol{L}^T(t_{k+1/2})\sqrt{h} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}(t_{k+1|k}) \\ \boldsymbol{0} \end{bmatrix} \tag{12}$$

where $\boldsymbol{0}$ represents the zero block, while Qr denotes the QR decomposition of the left-hand matrix, with only the upper triangular matrix retained.

During the measurement update step, the observation matrix $\boldsymbol{H}(t_{k+1})$ is computed as follows:

$$\boldsymbol{H}(t_{k+1}) := \frac{\mathrm{d}\boldsymbol{h}_{ms}(\hat{\boldsymbol{x}}(t_{k+1|k}))}{\mathrm{d}\hat{\boldsymbol{x}}(t_{k+1|k})} \tag{13}$$

For a conventional EKF, the Kalman gain $\boldsymbol{K}(t_{k+1})$ is determined as follows:

$$\boldsymbol{R}_{e,k+1} = \boldsymbol{R} + \boldsymbol{H}(t_{k+1})\boldsymbol{P}(t_{k+1})\boldsymbol{H}^T(t_{k+1}) \tag{14}$$

$$\boldsymbol{K}(t_{k+1}) = \boldsymbol{P}(t_{k+1|k})\boldsymbol{H}^T(t_{k+1})\boldsymbol{R}_{e,k+1}^{-1} \tag{15}$$

where $\boldsymbol{R}_{e,k+1}$ denotes the innovation covariance.

The state estimate $\hat{\boldsymbol{x}}(t_{k+1|k+1})$ is then updated as follows:

$$\hat{\boldsymbol{x}}(t_{k+1|k+1}) = \hat{\boldsymbol{x}}(t_{k+1|k}) + \boldsymbol{K}(t_{k+1})\boldsymbol{e}_{k+1} \tag{16}$$

where $\boldsymbol{e}_{k+1} := \boldsymbol{z}_{k+1} - \boldsymbol{h}_{ms}(\hat{\boldsymbol{x}}(t_{k+1|k}))$ represents innovation.

$\boldsymbol{P}(t_{k+1|k+1})$ can be calculated as follows:

$$\boldsymbol{P}(t_{k+1|k+1}) = \boldsymbol{P}(t_{k+1|k}) - \boldsymbol{K}(t_{k+1})\boldsymbol{H}(t_{k+1})\boldsymbol{P}(t_{k+1|k}) \tag{17}$$

In the square-root formulation of the CD-EKF, the notation $\overline{\boldsymbol{K}}(t_{k+1})$ is defined as follows:

$$\overline{\boldsymbol{K}}(t_{k+1}) := \boldsymbol{P}(t_{k+1|k})\boldsymbol{H}^T(t_{k+1})\boldsymbol{R}_{e,k+1}^{-\frac{1}{2}} \tag{18}$$

$\overline{\boldsymbol{K}}(t_{k+1})$, $\boldsymbol{R}_{e,k+1}$, and $\boldsymbol{S}(t_{k+1|k+1})$ are obtained using the following equations:

$$Qr\begin{bmatrix} \boldsymbol{R}^{\frac{1}{2}} & \boldsymbol{0} \\ \boldsymbol{S}(t_{k+1|k})\boldsymbol{H}^T(t_{k+1}) & \boldsymbol{S}(t_{k+1|k}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{e,k+1}^{\frac{1}{2}} & \overline{\boldsymbol{K}}^T(t_{k+1}) \\ \boldsymbol{0} & \boldsymbol{S}(t_{k+1|k+1}) \end{bmatrix} \tag{19}$$

where $\boldsymbol{R}^{\frac{1}{2}}$ denotes a lower triangular matrix produced from the Cholesky decomposition of the $\boldsymbol{R}$. $\hat{\boldsymbol{x}}(t_{k+1|k+1})$ can be calculated as follows:

$$\hat{\boldsymbol{x}}(t_{k+1|k+1}) = \hat{\boldsymbol{x}}(t_{k+1|k}) + \overline{\boldsymbol{K}}(t_{k+1})\boldsymbol{R}_{e,k+1}^{-\frac{T}{2}}\boldsymbol{e}_{k+1} \tag{20}$$

3. The Neural Network of Mamba-2 for BOT

3.1. Mamba-2 Network Architecture

The Mamba architecture introduces the SSM as the backbone [32], which represent a one-dimensional function $u(t) \in \mathbb{R} \to y(t) \in \mathbb{R}$ at time t, utilizing the hidden states $\boldsymbol{h}_{st}(t) \in \mathbb{R}^N$ to model the temporal dynamics of the process. These hidden states evolve over time according to the matrix parameters $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and are governed by the following ordinary differential equations (ODEs):

$$\begin{aligned} \boldsymbol{h}'_{st}(t) &= \boldsymbol{A}\boldsymbol{h}_{st}(t) + \boldsymbol{B}u(t) \\ y(t) &= \boldsymbol{C}\boldsymbol{h}(t) \end{aligned} \tag{21}$$

where $\boldsymbol{A}$ is the transition matrix, the $\boldsymbol{B}$ is the control matrix, and the $\boldsymbol{C}$ is the output matrix.

To achieve forward propagation of SSM within neural networks, discretization of SSM is required. The time step parameter Δ is introduced to convert the continuous parameters $\boldsymbol{A}, \boldsymbol{B}$ into the discrete counterparts $\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}$ using the zero-order hold (ZOH) model:

$$\begin{aligned} \bar{\boldsymbol{A}} &= \mathbf{exp}(\Delta \boldsymbol{A}) \\ \bar{\boldsymbol{B}} &= (\Delta \boldsymbol{A})^{-1}(\mathbf{exp}(\Delta \boldsymbol{A}) - \boldsymbol{I}) \bullet \Delta \boldsymbol{B} \end{aligned} \tag{22}$$

Subsequently, Equation (21) can be reformulated using (22) as follows:

$$\begin{aligned} \boldsymbol{h}_{st,k} &= \bar{\boldsymbol{A}}\boldsymbol{h}_{st,k-1} + \bar{\boldsymbol{B}}u_k \\ y_k &= \boldsymbol{C}\boldsymbol{h}_k \end{aligned} \tag{23}$$

When calculating input sequences $\boldsymbol{u}$ in parallel, the following formulation applies:

$$\begin{aligned} \bar{\boldsymbol{K}} &= (\boldsymbol{C}\bar{\boldsymbol{B}}, \boldsymbol{C}\bar{\boldsymbol{A}}\bar{\boldsymbol{B}}, \cdots, \boldsymbol{C}\bar{\boldsymbol{A}}^k\bar{\boldsymbol{B}}, \cdots) \\ \boldsymbol{y} &= \boldsymbol{u} * \bar{\boldsymbol{K}} \end{aligned} \tag{24}$$

where $k = 0, 1, \cdots, L$ represents the index and L represents the input sequence length. Additionally, $\boldsymbol{y} = (y_1, \cdots, y_L)$ is the output sequence and $\boldsymbol{u} = (u_1, \cdots, u_L)$ is the input sequence, while $K \in \mathbb{R}^L$ serves as the convolution kernel.

The original SSM operates as a linear time-invariant (LTI) system, which inherently restricts its capacity to effectively compress diverse textual contextual information. To address this limitation, the Mamba architecture introduces an innovative parameterization approach for SSM that incorporates an input-dependent selection mechanism, termed S6. This advancement enables dynamic adaptation of the SSM to varying input characteristics, thereby significantly enhancing its capability to perceive and compress essential contextual information. By replacing the SSM layers with state space dual (SSD) layers, the Mamba-2 architecture achieves significant improvements in computational efficiency through utilization of matrix multiplication units (e.g., tensor cores in GPUs or systolic arrays in TPUs) in modern hardware accelerators. This structural innovation enables the network to better align with the parallel computation paradigms of contemporary deep learning frameworks, thereby substantially enhancing training throughput and scalability.

The fundamental structure of the Mamba-2 block is illustrated in Figure 1. The core component of the Mamba-2 block is the SSM. The block begins by expanding the feature dimension through a linear projection layer, followed by a one-dimensional convolution layer that extracts local features. A normalization layer enhances stability after updating the hidden state via the SSM. Finally, a linear layer contracts the dimension to ensure consistency between input and output dimensions. The activation function, denoted by σ, is chosen as SiLU.

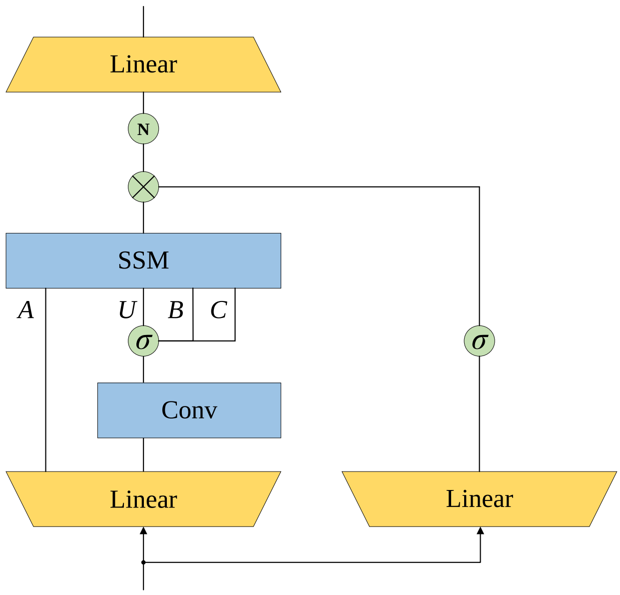


Figure 1. The architecture of the Mamba-2 block.

The compressed Mamba-2-based network for measurement noise estimation consists of dual linear layers at the input/output stages and only four basic Mamba-2 blocks in the intermediate processing sequence. This design significantly reduces computational overhead while preserving noise prediction capabilities. Since it is not designed for natural language processing, an embedding layer is not required for vocabulary processing. Instead, linear layers expand the input to the dimensional requirements of the Mamba-2 block and later contract it accordingly. The model has a dimension of 64, with a state dimension of 16 and a head dimension of 64. During training, the chunk size is set to 50.

3.2. Theoretical Basis for Innovation-Based Noise Estimation

The Mamba-2 module's input is the square of the innovation:

$$Input_k = \boldsymbol{e}_k^2 = (\boldsymbol{z}_k - \boldsymbol{h}_{ms}(\hat{\boldsymbol{x}}(t_k)))^2 \tag{25}$$

The choice of the squared innovation $\boldsymbol{e}_k^2$ as the input to the Mamba-2 network is theoretically motivated by its statistical relationship with the measurement noise variance $\boldsymbol{R}(t_k)$. Under the standard EKF measurement model, the innovation $\boldsymbol{e}_k$ satisfies the following:

$$\mathbb{E}[\boldsymbol{e}_k\boldsymbol{e}_k^T] \approx \boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k \tag{26}$$

where $\boldsymbol{H}_k$ and $\boldsymbol{P}_{k|k-1}$ are the observation matrix and the error covariance under the standard EKF. For scalar bearing measurements ($\dim(\boldsymbol{z}_k) = 1$), the squared innovation simplifies to the following:

$$\mathbb{E}[e_k^2] \approx \boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T + R_k \tag{27}$$

Equation (27) reveals that e_k^2's temporal dynamics encode the variation of R_k. Specifically, $\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T$ represents the projection of state prediction uncertainty onto the measurement space, which varies slowly as the filter converges. The residual variation in e_k^2 is therefore predominantly driven by changes in R_k, making the innovation sequence an effective input for measurement noise estimation. This relationship has been exploited in classical adaptive filtering. The Sage–Husa adaptive Kalman filter [13] estimates R_k via exponential smoothing:

$$\hat{R}_k = (1 - d_k)\hat{R}_{k-1} + d_k\left(e_k^2 - \boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T\right) \tag{28}$$

where $d_k = (1-b)/(1-b^k)$ and $b \in (0.95, 0.99)$ is a fixed forgetting factor. While effective for slowly varying noise, this linear moving-average estimator cannot capture rapid or nonlinear dynamics in R_k, and its fixed forgetting factor represents a compromise between responsiveness and smoothness. Furthermore, under nonlinear filtering, nonlinear errors in H will accumulate and lead to filter divergence.

VBAKFs [14,15] address this by placing an inverse-Wishart prior on R_k and estimating its posterior distribution through fixed-point iterations. However, these methods introduce two limitations: (1) the prior parameters $(\alpha_0, \beta_0, u_0, U_0)$ must be carefully tuned, with inappropriate configurations severely degrading performance, and (2) the inverse-Wishart assumption introduces posterior approximation errors.

The Mamba-2 network overcomes both limitations by learning a nonlinear temporal mapping $R_k = f_\theta(e_1^2, \ldots, e_k^2)$ directly from data. Its structured SSM maintains a hidden state $\boldsymbol{h}_{st,k}$ that serves as a learned, input-dependent memory. The S6 selection mechanism [28] adaptively determines which information to retain or discard based on the current input, enabling the following:

1. Adaptive memory length: High sensitivity to rapid R_k changes (the model learns to shorten its effective memory) and strong smoothing during stationary periods (the model learns to lengthen its effective memory).
2. No prior parameter dependence: Unlike VB methods, Mamba-2 requires no manually tuned prior parameters.
3. No distributional assumptions: The network learns the R_k dynamics empirically, avoiding the posterior approximation errors inherent in VB approaches.

In summary, using the innovation as input for Mamba-2 achieves generalization performance and is sufficient as an indicator to characterize the uncertainty of measurement noise.

3.3. Dataset Construction

This subsection presents a database construction method for training Mamba-2 to estimate measurement noise variance. The output represents the variance of the generated measurement noise.

By employing SSD as the SSM blocks, Mamba-2 retains the parallel processing capabilities of transformers. Consequently, the squared innovations of the entire trajectory can be directly fed into the Mamba-2 network for training.

Since obtaining ground-truth trajectories for BOT is challenging, the dataset is constructed based on generated noise, target trajectories, and filter tracking results. To minimize the computational cost of dataset generation, the transition equations of the trajectory generator and the filter must be linearized and discretized [33]. The standard DD-EKF is used for target tracking during dataset construction.

The linear transition equation is given by the following:

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{\omega}_k \tag{29}$$

where $\boldsymbol{F}$ denotes the transition matrix. $\boldsymbol{x}_k$ denotes the discrete state. $\boldsymbol{\omega}_k$ denotes the discrete process noise.

In the BOT scenario setting, the target's state can be expressed as $\boldsymbol{x}_k = [p_{x,k}, p_{y,k}, v_{x,k}, v_{y,k}]^T$, where the four components of the state vector $\boldsymbol{x}_k$ correspond to the position along the x-axis, position along the y-axis, velocity along the x-axis, and velocity along the y-axis, respectively. In the CV model, the target follows constant velocity. The CV model is defined as follows:

$$\boldsymbol{F}_{CV} = \begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{30}$$

where h denotes the time step.

In the CT model, the target follows a coordinated turn with a constant turn rate. The CT model is defined as follows:

$$\boldsymbol{F}_{CT} = \begin{bmatrix} 1 & 0 & \frac{\sin(\alpha h)}{\alpha} & \frac{\cos(\alpha h)-1}{\alpha} \\ 0 & 1 & \frac{1-\cos(\alpha h)}{\alpha} & \frac{\sin(\alpha h)}{\alpha} \\ 0 & 0 & \cos(\alpha h) & -\sin(\alpha h) \\ 0 & 0 & \sin(\alpha h) & \cos(\alpha h) \end{bmatrix} \tag{31}$$

where α denotes the turn rate.

In BOT, observations provide only bearing information. Therefore, the measurement equation is described as follows:

$$z_k = \arctan\left(\frac{p_{y,k} - p^o_{y,k}}{p_{x,k} - p^o_{x,k}}\right) + v_k \tag{32}$$

where z_k denotes the measured angle, and $v_k \sim N(0, \omega_k)$ represents the measurement noise. ω_k varies continuously, so within a trajectory, we typically represent ω_k using a cubic polynomial or a sine function that is greater than zero. $[p^o_{x,k}, p^o_{y,k}]^T$ represents the position coordinates of the observing drone.

To enhance the generalization capability of the Mamba-2 network, we need to ensure the dataset has diversity. When the motion mode is fixed to the CV or CT model, the trajectory generation is determined by its initial state, velocity and turn rate. Therefore, both the initial state, velocity and turn rate are randomized.

Refer to the generation method of the LAST dataset [34], assuming the detection range of the drone's radar is $[L_{min}, L_{max}]$; to prevent the target from exceeding this range, the initial distance between the target and the drone must fall within $[L_{min} + v_{max} \times T, L_{max} - v_{max} \times T]$, where T represents the total tracking duration, and v_{max} denotes the theoretical maximum speed of the target. Given the drone's position at [0, 0], the initial position state of the target is defined as follows:

$$\begin{cases} p_{x,0} = L_{random} \cos(\theta_p) \\ p_{y,0} = L_{random} \sin(\theta_p) \end{cases} \tag{33}$$

where θ_p denotes the angle between the direction from the drone to the target and the due east direction, which is uniformly sampled from -180° to 180°. L_{random} denotes the distance from the target to the drone, uniformly sampled from the interval $[L_{min} + v_{max} \times T, L_{max} - v_{max} \times T]$.

Likewise, the initial velocity state of the target is defined as follows:

$$\begin{cases} v_{x,0} = V_{random} \cos(\theta_v) \\ v_{y,0} = V_{random} \sin(\theta_v) \end{cases} \tag{34}$$

where θ_v denotes the angle between the direction of the target velocity and the due east direction, which is uniformly sampled from -180° to 180°. V_{random} denotes the initial velocity of the target, uniformly sampled from the interval $(0, v_{max}]$.

4. MSCDEKF Architecture

The MSCDEKF architecture is illustrated in Figure 2.

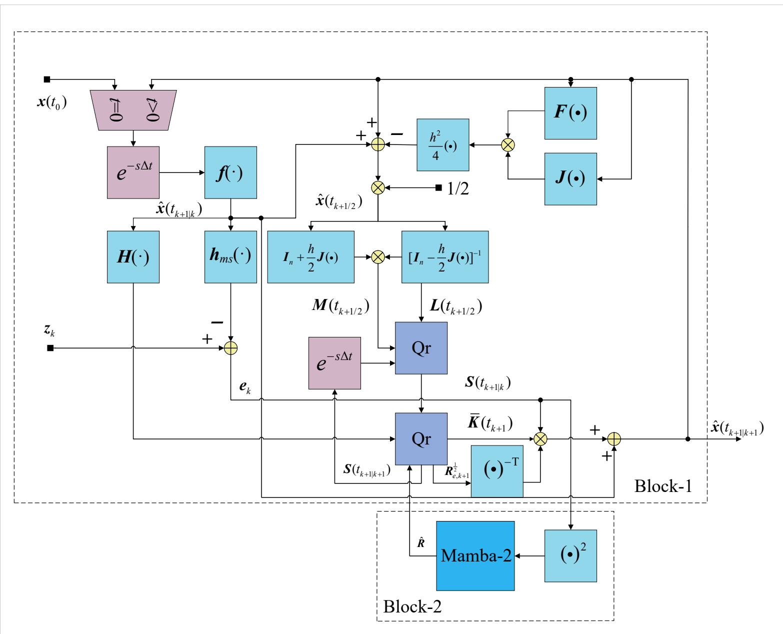


Figure 2. The architecture of the MSCDEKF.

Block I serves as the core component of the filter, responsible for continuously predicting the state and covariance while incorporating discrete updates based on measurement information. Block II comprises the neural network module, which sequentially estimates noise variance using innovations.

During the actual tracking process, the hidden state from the previous time step and the current innovation are fed into the Mamba-2 network in a manner similar to an RNN to estimate noise variance. To maintain the accuracy of state and error covariance predictions during the time update process, h is subdivided into m smaller time steps Δt, enabling continuous updates m times within each h.

The proposed MSCDEKF is formulated as follows (Algorithm 1):

Algorithm 1 MSCDEKF

Input: $\hat{\boldsymbol{x}}(t_1)$, $\boldsymbol{S}(t_1)$, h, Δt, $\boldsymbol{G}$, $\boldsymbol{Q}_1$, $\boldsymbol{R}$, $\boldsymbol{h}_{st} = None$, $\boldsymbol{z}_{2:K+1}$.
Output: The filtered state $\boldsymbol{x}(t_{k+1})$ and error covariance $\boldsymbol{S}(t_{k+1})$.

Initialize: $\boldsymbol{Q} \leftarrow \boldsymbol{Q}_1, m \leftarrow \Delta t/h$.
procedure TRAINING
  Train the Mamba-2 with the generated dataset.
end procedure
for $k = 1 \to K$ **do**
  procedure TIME UPDATE
    $\hat{\boldsymbol{x}}(t_{1,k+1}) \leftarrow \hat{\boldsymbol{x}}(t_k)$.
    $\boldsymbol{S}(t_{1,k+1}) \leftarrow \boldsymbol{S}(t_k)$.
    for $i = 1 \to m$ **do**
      Compute $\hat{\boldsymbol{x}}(t_{i+1,k+1})$ via Equation (6), where $h \leftarrow \Delta t$;
      $\hat{\boldsymbol{x}}(t_{i+1/2,k+1}) \leftarrow \frac{1}{2}\left[\hat{\boldsymbol{x}}(t_{i,k+1}) + \hat{\boldsymbol{x}}(t_{i+1,k+1}) - \frac{h^2}{4}\boldsymbol{J}(\hat{\boldsymbol{x}}(t_{i,k+1}))\boldsymbol{F}(\hat{\boldsymbol{x}}(t_{i,k+1}))\right]$;
      $\boldsymbol{L}(t_{i+1/2,k+1}) \leftarrow \left[\boldsymbol{I}_n - \frac{h}{2}\boldsymbol{J}(\hat{\boldsymbol{x}}(t_{i+1/2,k+1}))\right]^{-1}$;
      $\boldsymbol{M}(t_{i+1/2,k+1}) \leftarrow \boldsymbol{L}(t_{i+1/2,k+1})\left[\boldsymbol{I}_n + \frac{h}{2}\boldsymbol{J}(\hat{\boldsymbol{x}}(t_{i+1/2,k+1}))\right]$;
      $\begin{bmatrix} \boldsymbol{S}^T(t_{i+1,k+1}) \\ \boldsymbol{0} \end{bmatrix} \leftarrow Qr\begin{bmatrix} \boldsymbol{S}^T(t_{i,k+1})\boldsymbol{M}^T(t_{i+1/2,k+1}) \\ \boldsymbol{Q}^{\frac{T}{2}}(t_{i+1/2,k+1})\boldsymbol{G}^T(t_{i+1/2,k+1})\boldsymbol{L}^T(t_{i+1/2,k+1})\sqrt{h} \end{bmatrix}$;
    end for
    $\hat{\boldsymbol{x}}(t_{k+1|k}) \leftarrow \hat{\boldsymbol{x}}(t_{i+1,k+1})$.
    $\boldsymbol{S}(t_{k+1|k}) \leftarrow \boldsymbol{S}(t_{i+1,k+1})$.
  end procedure
  procedure MEASUREMENT UPDATE
    $\boldsymbol{H}(t_{k+1}) \leftarrow \mathrm{d}\boldsymbol{h}_{ms}(\hat{\boldsymbol{x}}(t_{k+1|k}))/\mathrm{d}\hat{\boldsymbol{x}}(t_{k+1|k})$.
    $\begin{bmatrix} \boldsymbol{R}_{e,k+1}^{\frac{T}{2}} & \overline{\boldsymbol{K}}^T(t_{k+1}) \\ \boldsymbol{0} & \boldsymbol{S}^T(t_{k+1|k+1}) \end{bmatrix} \leftarrow Qr\begin{bmatrix} \boldsymbol{R}^{\frac{1}{2}} & \boldsymbol{0} \\ \boldsymbol{S}^T(t_{k+1|k})\boldsymbol{H}_{k+1}^T & \boldsymbol{S}(t_{k+1|k}) \end{bmatrix}$.
    $\boldsymbol{e}_{k+1} \leftarrow \boldsymbol{z}_{k+1} - \boldsymbol{h}_{ms}(\hat{\boldsymbol{x}}(t_{k+1|k}))$.
    $\hat{\boldsymbol{x}}(t_{k+1|k+1}) \leftarrow \hat{\boldsymbol{x}}(t_{k+1|k}) + \overline{\boldsymbol{K}}(t_{k+1})\boldsymbol{R}_{e,k+1}^{-\frac{T}{2}}\boldsymbol{e}_{k+1}$.
  end procedure
  $\boldsymbol{x}(t_{k+1}) \leftarrow \hat{\boldsymbol{x}}(t_{k+1|k+1})$.
  $\boldsymbol{S}(t_{k+1}) \leftarrow \boldsymbol{S}(t_{k+1|k+1})$.
  Feed $\boldsymbol{e}_{k+1}^2$ and $\boldsymbol{h}_k$ to the Mamba-2 and obtain the estimated $\hat{R}$ and hidden state $\boldsymbol{h}_{k+1}$.
  $R \leftarrow \hat{R}$.
end for

5. Experiments

This section evaluates the performance of the proposed MSCDEKF in an UAV-based bearings-only tracking scenario. The objective is to estimate the two-dimensional state of a moving target, which may be a ground target or another aerial platform projected onto the horizontal plane. To emulate a typical surveillance mission where a fixed-wing UAV performs area search, we assume that the drone executes an S-shaped maneuvering trajectory to provide diverse observation geometries for the target, with its time-varying position given by the following:

$$\begin{cases} s_{x,k} = V_x t \\ s_{y,k} = A_y \sin(\omega_{dr} t) \end{cases} \tag{35}$$

where V_x denotes the constant forward speed along the x-axis, A_y is the amplitude of the lateral sinusoidal oscillation, and ω_{dr} is the angular velocity governing the maneuver. This serpentine flight pattern is commonly adopted in aerial surveillance missions, as it enhances observability by varying the bearing angle to the target over time. Two distinct target motion models, the CV and CT, are employed to assess the algorithm's accuracy under linear and nonlinear dynamics, respectively. Furthermore, to evaluate robustness against time-varying measurement noise, we deliberately increase the measurement noise variance in the CV scenario and amplify its temporal variation. This reduces the confounding influence of nonlinear target motion, enabling a more explicit examination of the noise effects. The measurement equation follows Equation (32). The time step h is set to 0.1 s, and Δt is set to 0.005 s for all experiments.

All computations are performed using Python 3.10 on an Intel(R) Xeon (R) Gold 5218 CPU @ 2.30 GHz and an NVIDIA(R) GeForce(TM) RTX-4090 GPU.

The root-mean-square error (RMSE) and average RMSE (ARMSE) are used to assess the performance of the MSCDEKF compared to other robust EKF methods for the BOT problem under time-varying process noise. These metrics are defined as follows:

$$\begin{cases} RMSE(t_k) = \sqrt{\dfrac{\sum_{n=1}^{M_C} (x^n(t_k) - \hat{x}^n(t_k))^2 + (y^n(t_k) - \hat{y}^n(t_k))^2}{M_C}} \\ ARMSE = \sqrt{\dfrac{\sum_{k=1}^{K} \sum_{n=1}^{M_C} (x^n(t_k) - \hat{x}^n(t_k))^2 + (y^n(t_k) - \hat{y}^n(t_k))^2}{M_C K}} \end{cases} \tag{36}$$

where $[x^n(t_k), y^n(t_k)]$ and $[\hat{x}^n(t_k), \hat{y}^n(t_k)]$ denote the true and estimated positions, respectively, in the nth Monte Carlo run. The number of Monte Carlo runs is set to $M_C = 200$. The number of the time steps is set to $K = 4500$.

The mean absolute error (MAE) is also used to assess the accuracy of the estimated measurement noise variance and is computed as follows:

$$MAE(t_k) = \frac{\sum_{n=1}^{M_C} |R^n(t_k) - \hat{R}^n(t_k)|}{M_C} \tag{37}$$

where $R^n(t_k)$ denotes the true variance of the measurement noise in the nth Monte Carlo run, and $\hat{R}(t_k)$ represents its estimated value in the nth Monte Carlo run.

To evaluate the stability of the algorithm, we introduce the variance of the RMSE as an indicator, which is calculated as follows:

$$\begin{aligned} variance_x &= \frac{\sum_{k=1}^{K} [RMSE_x(t_k) - \overline{RMSE}_x]^2}{K} \\ variance_y &= \frac{\sum_{k=1}^{K} [RMSE_y(t_k) - \overline{RMSE}_y]^2}{K} \end{aligned} \tag{38}$$

where $\overline{RMSE}_x$ and $\overline{RMSE}_y$ denotes the mean value of the $RMSE(t_k)$ on the x- and y-axes.

For training the Mamba model, a dataset comprising 50,000 trajectories was generated, each containing 4501 sample points. The AdamW optimizer [35] is employed for 100 epochs with a cosine-decay learning rate scheduler. The batch size is set to 10, with an initial learning rate of 0.001, decaying to a final learning rate of 0.00001. A weight decay of 0.01 is applied.

The performance of the MSCDEKF is compared against the EKF with the true measurement noise variance $R(t_k)$ (EKFTVM), and the EKF with the nominal measurement noise variance $\bar{R} = 0.009$ (EKFNVM), as well as other various robust EKF methods based on $\bar{R}$, including the Sage–Husa EKF (SHEKF), the variational Bayesian-based EKF (VBEKF), the novel variational Bayesian-based EKF (N-VBEKF), and the modified strong tracking sliding window variational adaptive EKF (MST-SWVAEKF). Meanwhile, we also introduce a normal extended Kalman filter with the Mamba-2 measurement-noise-adaptation module (MDDEKF) for comparison.

The decay parameter for VBEKF, N-VBEKF, and MST-SWVAEKF is set to $\rho = 1 - exp(-4)$. The number of iterations for the VBEKF and N-VBEKF is set to $N = 10$. The initial values of α and β for VBEKF are both 1. The initial estimates $\hat{u}_{0|0}$ and the $\hat{U}_{0|0}$ for N-VBEKF are 600 and 0.01, respectively. The A window size is $N = 10$. Additional parameters for the MST-SWVAEKF are configured as follows: $L_{min} = 4$, $L_{max} = 9$, $\kappa_{lower} = 0.9$, $\kappa_{upper} = 2$, scale factor $\alpha = 1$, $c_0 = 0$, and $\delta = 0.95$. The initial values $\hat{u}_{0|0}$ and $\hat{U}_{0|0}$ for the MST-SWVAEKF are set to 5 and $3\bar{R}$, respectively. All EKF variants utilize the same true process noise covariance. Theoretically, the EKFTVM achieves the highest estimation accuracy. To evaluate the performance of the square-root CD-EKF, we compare it with the EKFTVM under both the square-root CD model (EKFTVM-CD) and the DD model (EKFTVM-DD).

5.1. CV Experiments

In the CV scenario, the system model can be expressed by the following SDEs:

$$\mathrm{d}\boldsymbol{x}(t) = \check{\boldsymbol{F}}_{CV}\boldsymbol{x}(t)\mathrm{d}t + \boldsymbol{G}\mathrm{d}\boldsymbol{\omega}t \tag{39}$$

$$\check{\boldsymbol{F}}_{CV} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{40}$$

where $\boldsymbol{G} = \boldsymbol{I}_4$ is constant. The initial state is set to $\boldsymbol{x}(t_0) = [10, 10, 1, 1.1]^T$, and the initial error covariance matrix is $\boldsymbol{P}_{0|0} = \boldsymbol{I}_4$. The initial state noise is specified as $[0.02, 0.02, 0, 0]^T$. The true process noise covariance $\boldsymbol{Q}$ is also set to $\boldsymbol{I}_4$, while the true measurement noise variance $R_{CV}(t_k)$ is given by the following:

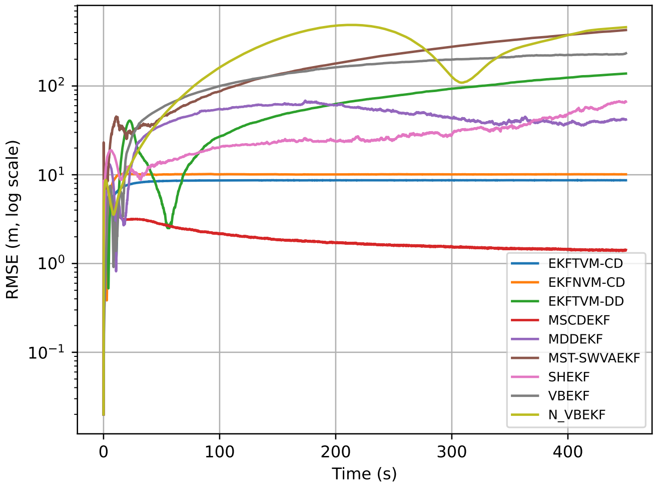
$$R_{CV}(t_k) = 0.2 + 0.05\cos(\frac{\pi t_k}{S_{CV}}) \tag{41}$$

where $S_{CV} = 450$ s represents the total duration of the CT numerical experiment for each Monte Carlo run. The ARMSE results for the CV scenario are summarized in Table 1.

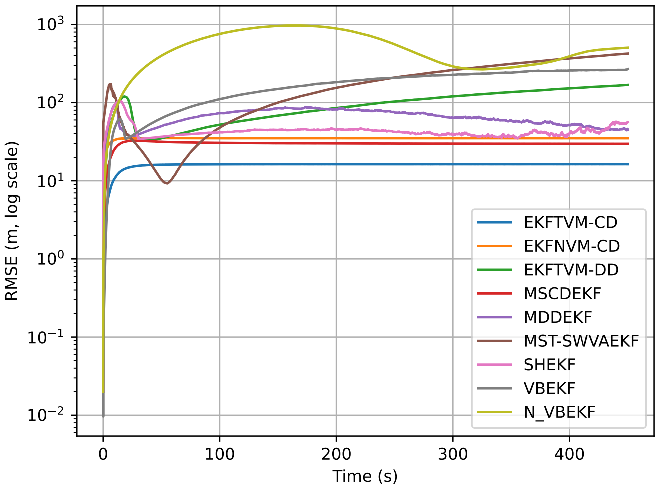
Table 1. ARMSE of the position obtained by various filtering methods in the CV-UAV-BOT experiment.

Filters	ARMSE (m)
EKFTVM-CD	18.15
EKFNVM-CD	36.29
EKFTVM-DD	132.64
VBEKF	253.85
N-VBEKF	671.40
MST-SWVAEKF	333.09
MSCDEKF	30.10
MDDEKF	82.16
SHEKF	55.04

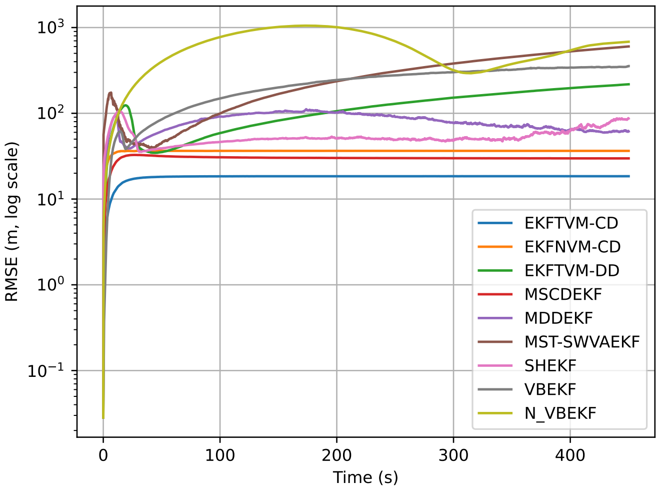
Figure 3 presents the per-step RMSE curves of all compared filters on a logarithmic scale.



(**a**)



(**b**)



(**c**)

Figure 3. RMSE of the position on a log scale obtained by different filtering algorithms in the CV-UAV-BOT experiment. (**a**) RMSE of Pos-X on a log scale. (**b**) RMSE of Pos-Y on a log scale. (**c**) RMSE of position on a log scale.

Table 2 reports the variance of the RMSE over the full simulation, quantifying the temporal stability of each filter.

Table 2. Variance of the RMSE obtained by various filtering methods in the CV-UAV-BOT experiment.

Filters	$variance_x$	$variance_y$
EKFTVM-CD	0.60	2.63
EKFNVM-CD	0.73	3.42
EKFTVM-DD	1665.38	1665.91
VBEKF	4224.99	5247.50
N-VBEKF	23,380.31	69,959.99
MST-SWVAEKF	14,895.45	16,035.84
MSCDEKF	0.42	6.52
MDDEKF	189.48	208.35
SHEKF	176.38	104.29

Figure 3 and Table 1 show the RMSE and ARMSE of the CV experiment result. The RMSE quantifies the average filtering error along the simulation time, and ARMSE reflects the overall error level of the entire simulation. Figure 3 is on a log scale. A clear three-tier performance emerges. The top tier consists of the CD-based methods: the EKFTVM-CD achieves the theoretical optimum (18.15 m) using the true $R_{CV}(t)$, while the MSCDEKF (30.10 m) attains the best accuracy among all methods without access to the true noise variance, outperforming EKFNVM-CD (36.29 m) by 17.1%. This confirms that Mamba-2's noise adaptation improves accuracy beyond the fixed covariance choice. The second tier comprises the DD-based adaptive methods, led by the SHEKF (55.04 m), whose direct moving-average estimation avoids VB parameter sensitivity. The third tier consists of the VB-based methods: VBEKF (253.85 m), MST-SWVAEKF (333.09 m), and N-VBEKF (671.40 m), whose large errors stem from inappropriate prior parameter configurations combined with posterior approximation errors. As shown in Figure 3, except the EKFTVM-CD, EKFNVM-CD, MSCDEKF, and MDDEKF, the RMSE curves of the remaining filters ultimately exhibit a divergent trend. In addition, the filters employing the square-root CD framework maintain relatively small fluctuations, shown in Table 2 where the EKFTVM-CD and MSCDEKF display smaller RMSE variances compared to the other filters. Because the square-root CD-EKF implementations offer superior accuracy and numerical stability. Figure 4 shows the MAE of the estimated measurement noise variance R on a log scale.

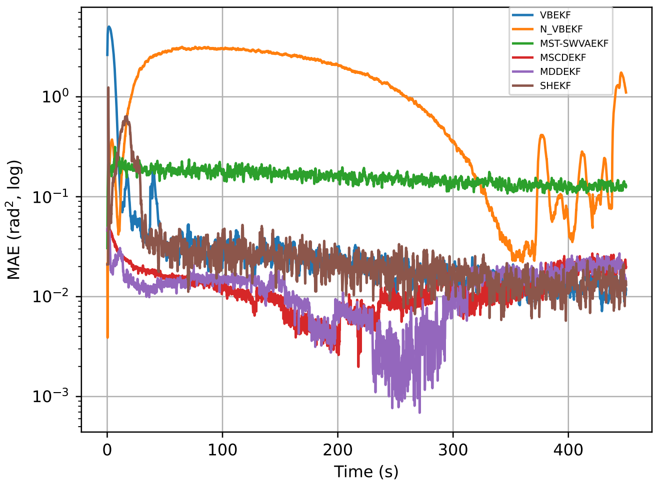


Figure 4. The MAE of the estimation of the measurement noise variance R on a log scale in the CV-UAV-BOT experiment.

In Figure 4, the MSCDEKF achieves the highest estimation accuracy of the measurement noise covariance. Because unlike VB-based methods, it is immune to the influence of initial parameter settings, and it is more capable of capturing the underlying dynamic variation of the measurement noise compared to the SHEKF . Among the DD filters, the MDDEKF and SHEKF achieve the best RMSE performance, with much lower errors than other DD filters. This is because the SHEKF's estimation of the measurement noise variance is directly based on a moving average of the innovation sequence. When the nonlinearity error is small, this estimation is accurate, which can be observed from Figure 4. Furthermore, as indicated in Table 2, the SHEKF exhibits greater stability than the EKFTVM-DD, which ensures its superior overall accuracy. For the MST-SWVAEKF and the N-VBEKF, the estimation of the measurement noise variance includes two components: one from the prior parameters and the other from the posterior distribution estimation. However, due to inappropriate parameter configurations, the two filters introduce an inherent bias in the initial estimation of the measurement noise variance. This initial prior error persists throughout subsequent estimation, ultimately influencing both accuracy and robustness. In comparison, VBEKF benefits from a more suitable parameter design, resulting in better accuracy and robustness than the MST-SWVAEKF and N-VBEKF. To assess statistical reliability, Table 3 reports the 95% confidence intervals computed from $n = 200$ independent Monte Carlo runs for the CV scenario.

Table 3. ARMSE with 95% confidence intervals, CV scenario ($n = 200$ Monte Carlo runs).

Method	95% CI (m)
EKFTVM-CD	[18.12, 18.18]
EKFNVM-CD	[36.22, 36.36]
EKFTVM-DD	[131.23, 134.05]
VBEKF	[250.74, 256.95]
N-VBEKF	[664.40, 678.40]
MST-SWVAEKF	[320.67, 345.51]
MSCDEKF	[30.04, 30.16]
MDDEKF	[77.32, 87.00]
SHEKF	[49.22, 60.86]

To assess statistical reliability, we computed 95% confidence intervals (CIs) from $n = 200$ independent Monte Carlo runs for each method (show in Table 3). The proposed MSCDEKF achieves the narrowest CI among all non-oracle methods (width 0.12 m), demonstrating highly consistent tracking performance across independent runs with negligible sensitivity to random seed variation. By contrast, the SHEKF—the best DD-based adaptive method—exhibits a CI width of 11.64 m, nearly two orders of magnitude wider, indicating substantial run-to-run variability. The CD-formulation methods (EKFTVM-CD and EKFNVM-CD) yield CI widths below 0.15 m, 20–50× narrower than their DD counterparts, attributable to the superior numerical stability of the square-root CD formulation. The VB-based methods exhibit correspondingly wider intervals (VBEKF 6.21 m, N-VBEKF 14.00 m, MST-SWVAEKF 24.84 m), reflecting their sensitivity to prior parameter settings.

5.2. CT Experiments

In the CT scenario, the system model is described by the following SDEs:

$$\mathrm{d}\boldsymbol{x}(t) = \boldsymbol{F}_{CT}(\boldsymbol{x}(t), t)\mathrm{d}t + \boldsymbol{G}\mathrm{d}\boldsymbol{\omega}t \quad (42)$$

$$\boldsymbol{F}_{CT}(\boldsymbol{x}(t), t) = [\dot{x}(t), -\Psi\dot{y}(t), \dot{y}(t), \Psi\dot{x}(t)]^T \quad (43)$$

where $\Psi = 0.01$ denotes the constant turn rate. Clearly, the CT state model is inherently nonlinear and continuous.

The drift matrix $\boldsymbol{G}$ is assumed to be constant, with $\boldsymbol{G} = \boldsymbol{I}_4$. The true process noise covariance $\boldsymbol{Q}$ is set to $\boldsymbol{I}_4$, while the true measurement noise variance $R_C T(t_k)$ is given by the following:

$$R_{CT}(t_k) = 0.02 + 0.01\cos(\frac{\pi t_k}{S_{CT}}) \tag{44}$$

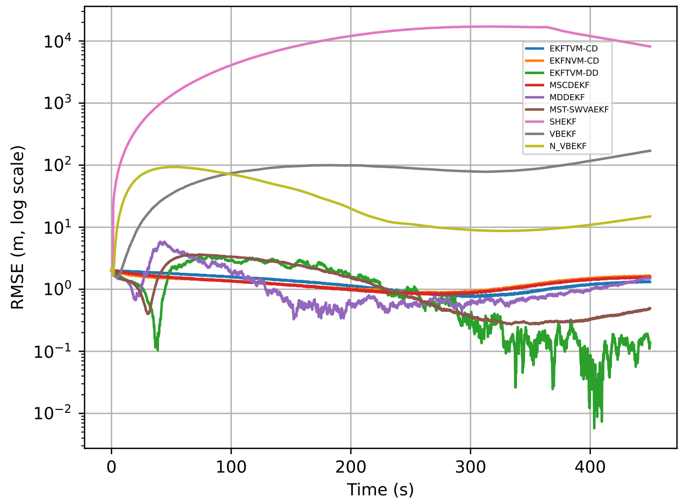
where $S_{CT} = 450$ s represents the total duration of the CT numerical experiment for each Monte Carlo run.

The true initial state is defined as follows: $\boldsymbol{x}(t_0) = [30, 15, 0, 0.1]^T$. The initial error covariance matrix is set to $\boldsymbol{P}_{0|0} = \boldsymbol{I}_4$. The initial state noise is specified as $[20, 0, 0, 0]$. The ARMSE results for the CT scenario are summarized in Table 4. Figure 5 presents the per-step RMSE curves on a logarithmic scale.

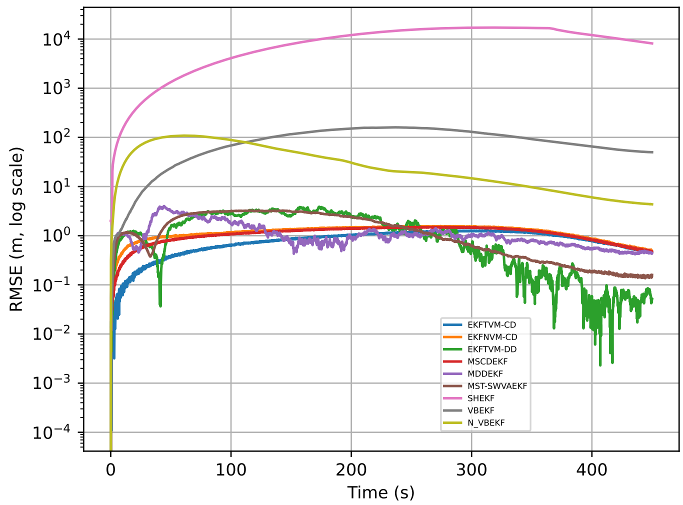
Table 4. ARMSE of the position obtained by various filtering methods in the CT-UAV-BOT experiment.

Filters	ARMSE (m)
EKFTVM-CD	1.59
EKFNVM-CD	1.77
EKFTVM-DD	2.51
VBEKF	137.76
N-VBEKF	66.56
MST-SWVAEKF	2.61
MSCDEKF	1.71
MDDEKF	2.15
SHEKF	14,832.22

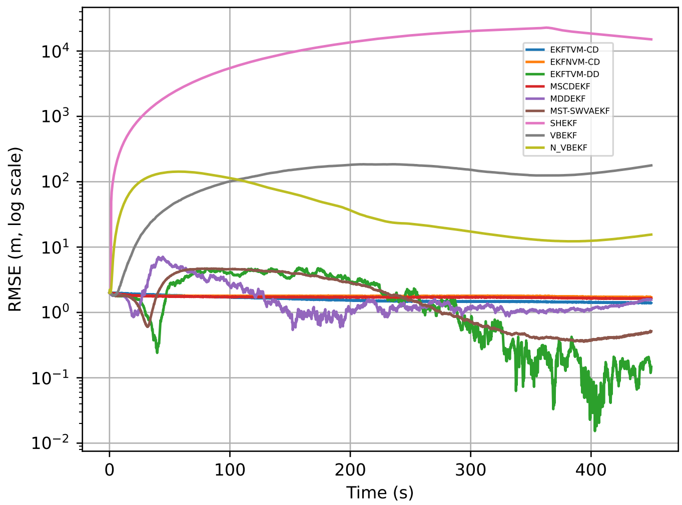
Figure 5 and Table 4 jointly characterize the position estimation accuracy of all compared filters in the CT-UAV-BOT experiment. As shown in Figure 5a–c, the RMSE curves on a logarithmic scale reveal a clear four-tier performance hierarchy within this nonlinear scenario. The EKFTVM-CD, MSCDEKF, and EKFNVM-CD form the top tier, with their RMSE curves stabilizing at approximately 1–2 m throughout the simulation and exhibiting minimal mutual separation. Their corresponding ARMSE values are 1.59 m, 1.71 m, and 1.77 m, respectively (shown in Table 4). Notably, the MSCDEKF achieves the second-lowest ARMSE without obtaining the true noise variance, and its RMSE curve is very close to that of the theoretically optimal EKFTVM-CD, indicating that Mamba-2-based noise adaptation approaches optimality under the nonlinear BOT. The EKFNVM-CD attains 1.77 m, only marginally behind MSCDEKF (1.71 m), indicating that the CD framework dominates the accuracy gains in this scenario while the Mamba-2 module contributes a modest but consistent 3.4% refinement, without requiring any parameter tuning or prior knowledge of the noise statistics. The second tier consists of the MDDEKF, EKFTVM-DD, and MST-SWVAEKF. Their corresponding ARMSE values are 2.15 m, 2.51 m, and 2.61 m, respectively. They achieve good accuracy but exhibit visible oscillations, particularly the EKFTVM-DD, whose RMSE fluctuates between 0.1 and 3 m, because discrete non-square-root filters struggle to maintain stability in nonlinear scenarios. The third tier comprises the VBEKF and N-VBEKF, whose ARMSE values are 137.76 m and 66.56 m. Their ARMSE values are two orders of magnitude larger than those of the top tier, as their prior parameter settings introduce systematic biases through the nonlinear tracking. Finally, the SHEKF constitutes the fourth tier with an ARMSE of 14,832.22 m. Its RMSE curve diverges monotonically to the top of the logarithmic chart (10^4 m), representing a severe failure triggered by the accumulation of linearization errors during CT tracking.



(**a**)



(**b**)



(**c**)

Figure 5. RMSE of the position on a log scale obtained by different filtering algorithms in the CT-UAV-BOT experiment. (**a**) RMSE of Pos-X on a log scale. (**b**) RMSE of Pos-Y on a log scale. (**c**) RMSE of position on a log scale.

Table 5 provides a quantitative measure of temporal filtering stability through the variance of RMSE over the entire 450 s simulation.

Table 5. Variance of the RMSE obtained by various filtering methods in the CT-UAV-BOT experiment.

Filters	$variance_x$	$variance_y$
EKFTVM-CD	0.13	0.11
EKFNVM-CD	0.07	0.11
EKFTVM-DD	1.23	1.46
VBEKF	1151.34	2305.11
N-VBEKF	829.05	1155.61
MST-SWVAEKF	1.34	1.34
MSCDEKF	0.08	0.12
MDDEKF	1.18	0.56
SHEKF	33,178,220.15	22,260,320.99

Table 5 provides a quantitative measure of temporal filtering stability through the variance of RMSE over the entire 450 s simulation. The MSCDEKF achieved the lowest variance of 0.08 along the x-axis and a variance of 0.12 along the y-axis, approaching the theoretical optimum of EKFTVM-CD (0.11). This stability stems from the combination of two mechanisms. The square-root covariance propagation preserves the positive definiteness of the error covariance matrix and prevents numerical degradation during continuous-time integration. At the same time, the Mamba-2 module's noise variance estimation avoids the drastic fluctuations under nonlinear dynamics. In contrast, the SHEKF's variance is too large (3.3×10^7 for x, 2.2×10^7 for y). Its RMSE exhibits jumps of several kilometers in magnitude between adjacent 0.1 s time steps. Among the DD-based filters, the MST-SWVAEKF exhibits the best stability (1.34 for both axes), demonstrating that its sliding-window smoothing and fading factor mechanisms effectively mitigate the adverse impact of linearization errors on filter consistency. Since VB-based methods do not rely exclusively on the innovation sequence for estimation, their prior parameter settings instead prevent the noise estimation error from escalating continuously in nonlinear scenarios. The MST-SWVAEKF's fading factor mitigates the adverse effects of linearization errors. Meanwhile, the smoothing operation enhances the overall stability of the filter, shown in Table 5. As a result, although its estimation error of the measurement noise variance is larger than that of other VBEKF variants, it still achieves superior state estimation accuracy. Figure 6 shows the MAE of the estimated R on a log scale, revealing the noise adaptation capability of each method.

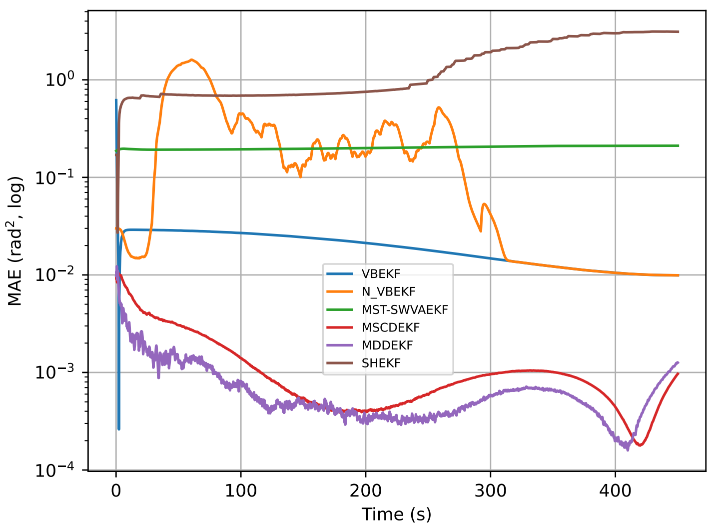


Figure 6. The MAE of the estimation of the measurement noise variance R on a log scale in the CT-UAV-BOT experiment.

Figure 6 presents the MAE of the estimated measurement noise variance R on a log scale, revealing the noise adaptation capability of each method under the nonlinear CT-UAV-BOT. The MSCDEKF and MDDEKF consistently achieve the lowest MAE, whose numerical range is approximately from 10^{-3} to 10^{-4} rad². They outperform all competing adaptive approaches by one to four orders of magnitude throughout the entire simulation. This demonstrates that the Mamba-2 module's state-space modeling captures the temporal dynamics of the noise process, enabling accurate variance prediction regardless of the quality of state estimates. The SHEKF exhibits an increasing MAE curve that rises from approximately 10^{-2} to over 3 rad², which shows that its linearization error continuously expands the R estimate, thereby amplifying the Kalman gain and causing divergence. In the VB-based methods, the divergence trend of its estimates is suppressed due to the effect of prior parameter regularization. Although the VBEKF and N-VBEKF maintain bounded MAE (approximately 10^{-2} rad²), their state estimation accuracy is compromised. The Mamba-2-based methods (MSCDEKF and MDDEKF) maintain similar noise estimation MAE in both the CV and CT scenarios, confirming that the network's noise perception is scenario-invariant. This indicates that Mamba-2 can serve as a noise estimation module across diverse UAV-BOT operating conditions without scenario-specific retraining. Table 6 reports the 95% confidence intervals for the CT scenario.

Table 6. ARMSE with 95% confidence intervals, CT scenario ($n = 200$ Monte Carlo runs).

Method	**95% CI (m)**
EKFTVM-CD	[1.5893, 1.5907]
EKFNVM-CD	[1.7664, 1.7735]
EKFTVM-DD	[2.4576, 2.5624]
VBEKF	[136.81, 138.71]
N-VBEKF	[35.67, 97.45]
MST-SWVAEKF	[2.5290, 2.6910]
MSCDEKF	[1.7085, 1.7115]
MDDEKF	[2.0942, 2.2059]
SHEKF	[13,764.75, 15,899.69]

Table 6 includes the 95% confidence intervals for the CT scenario. The MSCDEKF achieves an exceptionally narrow CI (width 0.0030 m), confirming that Mamba-2-based noise adaptation remains highly stable under coordinated turn nonlinearity. The CI widths reveal a clear three-tier stratification: CD methods (MSCDEKF, EKFTVM-CD, EKFNVM-CD) exhibit widths below 0.01 m; DD methods (MDDEKF, MST-SWVAEKF, EKFTVM-DD) show widths of 0.1–0.2 m; and VB methods yield substantially wider intervals (VBEKF 1.90 m, N-VBEKF 61.79 m), with N-VBEKF nearly two orders of magnitude wider than the DD tier. The SHEKF undergoes catastrophic failure with a CI width of 2134.95 m, quantitatively confirming the divergence trend.

5.3. Computational Efficiency Analysis

Table 7 presents the average per-step running time of each compared filtering method, measured over $M_K = 200$ Monte Carlo runs of $K = 4501$ time steps each.

As shown in Table 7, the DD-EKF has a faster per-step running time than the CD-EKF. The VB-based methods require approximately 0.6–0.8 ms/step due to fixed-point iterations. Among them, the N-VBEKF (0.63 ms) is faster than the VBEKF (0.82 ms) because the inverse-Wishart prior formulation avoids the per-iteration matrix symmetrization required by the VBEKF's inverse-Gamma prior. The CD-EKFs are approximately 6× slower than their DD counterparts due to the Runge–Kutta integration. Moreover, the MST-SWVAEKF offers faster performance compared to the other two VB methods due to its adaptive window size adjustment mechanism.

Table 7. Average per-step running time (ms/step).

Method	Running Time (ms/step)
EKFTVM-CD	0.580
EKFNVM-CD	0.556
EKFTVM-DD	0.091
VBEKF	0.816
N-VBEKF	0.631
MST-SWVAEKF	0.646
MSCDEKF	24.7
MDDEKF	24.1
SHEKF	0.134

The proposed MSCDEKF has a higher per-step cost (approximately 24 ms/step) in its current implementation. This overhead is primarily attributable to a per-step GPU cache clearing operation retained from the development phase. The Mamba-2 model is extremely compact, comprising only four Mamba-2 blocks with a hidden dimension of 64 and a state dimension of 16 (approximately 17K parameters). Removing the cache-clearing operations at each step can further reduce the computational time cost.

Even at the current 24 ms/step, the MSCDEKF supports a tracking update rate of approximately 41 Hz, well above the 10 Hz update rate typically employed in UAV tracking applications ($h = 0.1$ s). The mechanically scanned tracking radars provide updates at only 0.1–1.0 Hz [36]. Adaptive phased-array fire-control radars operate at 0.6–4.0 Hz for routine-to-engagement tracking [37]. The MSCDEKF at 41 Hz exceeds the above radar tracking frequencies and is sufficient to meet real-time application requirements.

6. Conclusions

This work presents a Mamba-2-based square-root CD-EKF algorithm for BOT tasks on UAVs. Compared to traditional RNNs, the Mamba-2 network incorporates the SSM, which captures long-term dependencies in sequences by maintaining a hidden state. This hidden state facilitates the transfer of information across time steps, enabling the network to retain past inputs and improve the long-term prediction of time-varying measurement noise variance. Additionally, the introduction of the square-root CD-EKF mechanism enhances the filtering process, improving estimation accuracy and numerical stability under the weak observability conditions inherent to BOT on UAVs and ensuring stable tracking performance. Experimental results show that the MSCDEKF achieves an ARMSE of 30.10 m in the CV scenario, outperforming the SHEKF (the best adaptive method based on DD) by 45.3% (55.04 m). In the CT scenario, the MSCDEKF attains an ARMSE of 1.71 m, which is 34.5% lower than that of the MST-SWVAEKF (2.61 m), the best DD-based competitor. Based on 200 Monte Carlo runs, the 95% confidence intervals are [30.04, 30.16] m and [1.7085, 1.7115] m, respectively. For R estimation, the MSCDEKF achieves substantially higher accuracy than all compared adaptive methods.

Future research will focus on extending this approach to BOT tasks by UAVs where measurement noise follows different distributions and cases where time-varying measurement noise includes outliers. In addition, we will investigate what maneuvering patterns drones should adopt to enhance the observability of BOT. Future efforts will also focus on implementing and validating the proposed method on actual fixed-wing UAV platforms to further demonstrate its practical effectiveness. The current work focuses solely on measurement noise adaptation; future research will extend the framework to jointly estimate both process noise and measurement noise by the Mamba-2 network, enabling the filter to respond to uncertainties arising from external disturbances and model mismatches simultaneously.

Author Contributions: Conceptualization, G.Z. and H.W.; methodology, G.Z. and H.W.; software, G.Z.; validation, H.W.; formal analysis, G.Z.; investigation, H.W.; resources, H.W.; data curation, A.M.; writing—original draft preparation, G.Z.; writing—review and editing, G.Z., H.W. and A.M.; visualization, Y.Z.; supervision, C.L.; project administration, H.W. and M.Y.; funding acquisition, H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 61703420 and Grant 62073337, the Youth Innovation Team of Shaanxi Universities under Grant 2023015, and the Natural Science Foundation of Shaanxi Province under Grant 2025JC-YBMS-749.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author(s).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BOT	Bearings-only tracking
CD	Continuous-discrete
SDEs	Stochastic differential equations
ODEs	Ordinary differential equations
RK	Runge–Kutta
EKF	Extended Kalman filter
MSCDEKF	Mamba-2 based square-root continuous-discrete extended Kalman filter
SHEKF	Sage–Husa extended Kalman filter
VBEKF	Variational Bayesian-based extended Kalman filter
N-VBEKF	Novel variational Bayesian-based extended Kalman filter
MST-SWVAEKF	Modified strong tracking sliding window variational adaptive extended Kalman filter
RMSE	Root mean square error
ARMSE	Average root mean square error

References

1. Wang, S.; Li, Y.; Qi, G.; Sheng, A. Diffusion Nonlinear Estimation and Distributed UAV Path Optimization for Target Tracking with Intermittent Measurements and Unknown Cross-Correlations. *Drones* **2023**, *7*, 473. [CrossRef]
2. Zhao, J.; Gui, R.; Dong, X. Generalized Labeled Multi-Bernoulli Filter-Based Passive Localization and Tracking of Radiation Sources Carried by Unmanned Aerial Vehicles. *Drones* **2024**, *8*, 96. [CrossRef]
3. Zhou, Z.; Hu, J.; Wu, X.; Liao, S.; Yuan, J. Fixed-Time Target Tracking and Encirclement Control for Multi-UAVs with Bearing-Only Measurements. *Drones* **2026**, *10*, 63. [CrossRef]
4. Yang, G.; Ma, W.; Fu, W.; Zhu, S.; Zhang, T. Distributed Consensus-Based Tracking with Inverse Covariance Intersection in Bearing-Only UAV Networks. *Drones* **2026**, *10*, 107. [CrossRef]
5. Farina, A. Target Tracking with Bearings—Only Measurements. *Signal Process.* **1999**, *78*, 61–78. [CrossRef]
6. Le Cadre, J.P.; Jaetffret, C. On the Convergence of Iterative Methods for Bearings-Only Tracking. *IEEE Trans. Aerosp. Electron. Syst.* **1999**, *35*, 801–818. [CrossRef]
7. Xie, Y.; Song, T.L. Bearings-Only Multi-Target Tracking Using an Improved Labeled Multi-Bernoulli Filter. *Signal Process.* **2018**, *151*, 32–44. [CrossRef]
8. Li, H.; Cui, G.; Kong, L.; Chen, G.; Wang, M.; Guo, S. Robust Human Targets Tracking for MIMO Through-Wall Radar via Multi-Algorithm Fusion. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2019**, *12*, 1154–1164. [CrossRef]

9. Arasaratnam, I.; Haykin, S.; Hurd, T.R. Cubature Kalman Filtering for Continuous-Discrete Systems: Theory and Simulations. *IEEE Trans. Signal Process.* **2010**, *58*, 4977–4993. [CrossRef]
10. Crouse, D. Basic Tracking Using Nonlinear Continuous-Time Dynamic Models. *IEEE Aerosp. Electron. Syst. Mag.* **2015**, *30*, 4–41. [CrossRef]
11. Kulikova, M.V.; Kulikov, G.Y. Square-Root Accurate Continuous-Discrete Extended Kalman Filter for Target Tracking. In *52nd IEEE Conference on Decision and Control*; IEEE: Piscataway, NJ, USA, 2013; pp. 7785–7790. [CrossRef]
12. Zeng, G.; Chen, S.; Wu, H.; Yang, M. Continuous-Discrete Extended Kalman Filtering Based on the Neural Ordinary Differential Equations Method. *Eng. Appl. Artif. Intell.* **2024**, *133*, 108611. [CrossRef]
13. Jiang, B.; Sheng, W.; Zhang, R.; Han, Y.; Ma, X. Range Tracking Method Based on Adaptive "Current" Statistical Model with Velocity Prediction. *Signal Process.* **2017**, *131*, 261–270. [CrossRef]
14. Sarkka, S.; Nummenmaa, A. Recursive Noise Adaptive Kalman Filtering by Variational Bayesian Approximations. *IEEE Trans. Autom. Control* **2009**, *54*, 596–600. [CrossRef]
15. Chang, G.; Chen, C.; Zhang, Q.; Zhang, S. Variational Bayesian Adaptation of Process Noise Covariance Matrix in Kalman Filtering. *J. Frankl. Inst.* **2021**, *358*, 3980–3993. [CrossRef]
16. Huang, Y.; Zhang, Y.; Wu, Z.; Li, N.; Chambers, J. A Novel Adaptive Kalman Filter with Inaccurate Process and Measurement Noise Covariance Matrices. *IEEE Trans. Autom. Control* **2018**, *63*, 594–601. [CrossRef]
17. Qiao, S.; Fan, Y.; Wang, G.; Mu, D.; He, Z. Modified Strong Tracking Slide Window Variational Adaptive Kalman Filter with Unknown Noise Statistics. *IEEE Trans. Ind. Inform.* **2023**, *19*, 8679–8690. [CrossRef]
18. Graves, A.; Mohamed, A.r.; Hinton, G. Speech Recognition with Deep Recurrent Neural Networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*; IEEE: Piscataway, NJ, USA, 2013; pp. 6645–6649. [CrossRef]
19. Greff, K.; Srivastava, R.K.; Koutnik, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A Search Space Odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 2222–2232. [CrossRef] [PubMed]
20. Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *Phys. D Nonlinear Phenom.* **2020**, *404*, 132306. [CrossRef]
21. Dey, R.; Salem, F.M. Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*; IEEE: Piscataway, NJ, USA, 2017; pp. 1597–1600. [CrossRef]
22. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)* ; Curran Associates, Inc.: Red Hook, NY, USA, 2017; pp. 6000–6010.
23. Shlezinger, N.; Whang, J.; Eldar, Y.C.; Dimakis, A.G. Model-Based Deep Learning. *Proc. IEEE* **2023**, *111*, 465–499. [CrossRef]
24. Xu, L.; Niu, R. EKFNet: Learning System Noise Statistics from Measurement Data. In *ICASSP 2021—2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; IEEE: Piscataway, NJ, USA, 2021; pp. 4560–4564. [CrossRef]
25. Xu, H.; Luo, H.; Wu, Z.; Wu, F.; Bao, L.; Zhao, F. Towards Predicting the Measurement Noise Covariance with a Transformer and Residual Denoising Autoencoder for GNSS/INS Tightly-Coupled Integrated Navigation. *Remote Sens.* **2022**, *14*, 1691. [CrossRef]
26. Revach, G.; Shlezinger, N.; Ni, X.; Escoriza, A.L.; Van Sloun, R.J.G.; Eldar, Y.C. KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics. *IEEE Trans. Signal Process.* **2022**, *70*, 1532–1547. [CrossRef]
27. Hao, P.; Karakuş, O.; Achim, A. RKFNet: A Novel Neural Network Aided Robust Kalman Filter. *Signal Process.* **2025**, *230*, 109856. [CrossRef]
28. Gu, A.; Dao, T. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv* **2024**. [CrossRef]
29. Dao, T.; Gu, A. Transformers Are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria, 21–27 July 2024; pp. 10041–10071.
30. Kulikova, M.V.; Kulikov, G.Y. NIRK-based Accurate Continuous-Discrete Extended Kalman Filters for Estimating Continuous-Time Stochastic Target Tracking Models. *J. Comput. Appl. Math.* **2017**, *316*, 260–270. [CrossRef]
31. Kulikov, G.Y.; Kulikova, M.V. Accurate State Estimation in Continuous-Discrete Stochastic State-Space Systems with Nonlinear or Nondifferentiable Observations. *IEEE Trans. Autom. Control* **2017**, *62*, 4243–4250. [CrossRef]
32. Huang, W.; Pan, J.; Tang, J.; Ding, Y.; Xing, Y.; Wang, Y.; Wang, Z.; Hu, J. ML-Mamba: Efficient Multi-Modal Large Language Model Utilizing Mamba-2. *arXiv* **2024**, arXiv:2407.19832. [CrossRef]
33. Zeng, G.; Wu, H.; Chen, S.; Yang, M. Modified Swin Transformer-Based Continuous-Discrete Extended Kalman Filtering for Bearings-Only Tracking With Unknown Noise Statistics. *IEEE Sensors J.* **2025**, *25*, 9842–9852. [CrossRef]
34. Liu, J.; Wang, Z.; Xu, M. DeepMTT: A Deep Learning Maneuvering Target-Tracking Algorithm Based on Bidirectional LSTM Network. *Inf. Fusion* **2020**, *53*, 289–304. [CrossRef]
35. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the 7th International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019; pp. 4069–4078.

36. Li, P.; Zhou, G. Target Tracking With Revisit Interval Uncertainty for Rotating Radars. *IEEE Trans. Aerosp. Electron. Syst.* **2024**, *60*, 7354–7366. [CrossRef]
37. Tuncer, O.; Cirpan, H.A. A Networked Radar Resource Management Approach Utilizing Target Priority and Maneuver for Cooperative Air Defense Fire Control Radars. *IEEE Access* **2023**, *11*, 136279–136291. [CrossRef]