RNN-UKF: Enhancing Hyperparameter Auto-Tuning in Unscented Kalman Filters through Recurrent Neural Networks

Zhengyang Fan
Intelligent Fusion Technology, Inc
Germantown, USA
zhengyang.fan@intfusiontech.com

Dan Shen
Intelligent Fusion Technology, Inc
Germantown, USA
dshen@intfusiontech.com

Yajie Bao
Intelligent Fusion Technology, Inc
Germantown, USA
yajie.bao@intfusiontech.com

Khanh Pham
Air Force Research Lab
Kirtland AFB, USA
khanh.pham.1@us.af.mil

Erik Blasch
Air Force Research Lab
Arlington, USA
erik.blasch.1@us.af.mil

Genshe Chen
Intelligent Fusion Technology, Inc
Germantown, USA
gchen@intfusiontech.com

***Abstract*—The Unscented Kalman Filter (UKF) stands out as a versatile and dynamic algorithm, celebrated for its prowess in estimating the states of nonlinear dynamical systems within uncertain environments. However, the accuracy of UKF state estimations hinges significantly on the thoughtful selection of pivotal hyperparameters, α, β, and κ, which are integral in shaping the distribution of sigma points around the current state estimate. Prevailing methods for tuning these parameters encompass heuristic approaches such as arbitrarily fix α at 0.001, κ at 0, and β at 2 for Gaussian noise, though the efficacy of such rules heavily hinges on the intricacies of the specific problem. Alternatively, the grid search technique seeks to optimize these hyperparameters, but it can become computationally burdensome, particularly when the search space is extensive and intricate. To navigate these hurdles, this paper introduces the RNN-UKF algorithm—a pioneering strategy that leverages recurrent neural networks (RNNs) to autonomously fine-tune UKF hyperparameters. The inherent adaptability of RNNs is harnessed to dynamically adjust the α, β, and κ parameters of the unscented transformation during each state estimation step, all aimed at minimizing the root mean squared error (RMSE). Demonstrated through numerical simulations, we provide compelling evidence that the RNN-UKF approach outperforms both heuristic rule of thumb and grid search techniques in terms of RMSE performance. Moreover, the RNN-UKF methodology showcases its superiority over the conventional extended Kalman filter (EKF) approach, particularly in scenarios characterized by substantial system noise.**

***Index Terms*—Unscented Kalman Filter, Recurrent Neural Networks, Hyperparameter Tuning, Deep Learning**

I. Introduction

The Unscented Kalman Filter (UKF) [1] is a variant of the traditional Kalman Filter [2] designed to overcome the limitations of linear models and Gaussian assumptions. The UKF is an effective nonlinear state estimation algorithm that provides robust and accurate solutions for systems with non-linear dynamics and non-Gaussian noise [3]. While the UKF offers improved performance as compared to the extended Kalman filter (EKF) in handling non-linear systems, it can be sensitive to the selection of the hyperparameters α, β, κ that control the spread of the sigma points around current state estimation. Poor choices in these hyperparameters can lead to convergence issues and inaccurate estimates, limiting the UKF's robustness in certain scenarios [4].

A natural hyperparameter selection scheme is to perform grid search [5], which involves defining a grid of possible hyperparameter values and exhaustively evaluating all possible combinations of these values to find the set of hyperparameters that produces the best performance according to a chosen evaluation metric. While grid search is simple and systematic, it has several drawbacks:

1) Computational Cost: Grid search explores all possible combinations, which can be extremely time-consuming and computationally expensive, especially when dealing with a large number of hyperparameters or a wide range of values.
2) Curse of Dimensionality: As the number of hyperparameters and the granularity of their values increase, the number of combinations grows exponentially, leading to a massive search space and potentially inefficient use of resources.
3) Limited to Defined Grid: If the optimal hyperparameters lie between the defined grid values, grid search may fail to identify them.

Another widely adopted rule of thumb method suggested in [1] is to select α equal to 0.001 and κ equal to 0 and β equal to 2 for Gaussian noise. However, the performance of the UKF using these hyperparameters are highly problem dependent and may result in poor estimation performance in some scenarios, e.g., in our Lorenz attractor and navigation use case in our numerical experiments (Section IV).

To overcome the limitations of grid search and rule of thumb methods, many research works have been proposed. In Julier et al. [6], for the formulation of the unscented transformation that features only the scaling parameter κ, the user is suggested to set κ equal to $3 - n$, where n is the dimension of the state vector. Since Julier's method also suggests a fixed set of hyperparameters, its performance quality is problem dependent. Sakai and Kuroda [7] propose a hyperparameter adjustment problem for the UKF and utilized a coordinate ascent algorithm to obtain optimal hyperparameters for accurate state estimation by assuming the availability of measurements of the true state vectors. Dunik et al. [8] deals with the UKF for state estimation of nonlinear stochastic dynamic systems with a special focus on the scaling parameter κ of the filter. They proposed an online adaptive technique for the scaling hyperparameter κ, which is based on choosing a κ that achieves the highest value of a criterion within a set of feasible κ at each time instant when a new measurement is available. However, Dunik's online approach imposes extra computational cost to the UKF at runtime, and Straka et al. [9] further explored and refined this method. But the numerical results presented showed that the UKF enhanced with the proposed approaches was still significantly slower than the regular UKF. Further analysis looked at methods to increase the speed with stochastic integration rules (SIR) [10] as compared to the UKF methods that utilize deterministic integration rules (DIRs) [11].

Bayesian optimization [12] is a popular tool for hyperparameter tuning for UKF in recent years. Turner and Rasmussen [13] developed an offline model-based optimization approach that treats sigma points placement in a UKF as a learning problem in a model-based view and demonstrate that learning to place the sigma points correctly from data can make sigma point collapse much less likely. Their approach utilizes Gaussian process optimization (GPO) to pick values for the hyperparameter of the UKF, together with an upper confidence bound acquisition criterion to guide the searching procedure. Since the approach is completely offline, it does not increase the runtime computational cost of the UKF. Scardua and Da Cruz [4] formulated the tuning of the UKF hyperparameters as an optimization problem and developed a stochastic search algorithm with standard model-based optimizer, such as GPO, to solve the optimization problem. The method is also performed in an offline fashion, hence does not increase the runtime when performing UKF. Recently, Bertipaglia et al. [14] developed a novel methodology to autotune UKF using a two stage Bayesian optimization method, based on a t-student process to optimize the process noise parameters of a UKF for vehicle sideslip angle estimation. Bertipaglia et al's method minimizes averaged sum of the states and measurement' estimation error for various vehicle maneuvers covering a wide range of vehicle behavior. The cubature Kalman filter (CKF) faces similar challenges as the UKF uses the unscented transform, while the CKF uses the spherical-radial cubature rule [15], [16].

Although Bayesian optimization is a powerful technique for hyperparameter tuning in UKF, it suffers from several drawbacks, which makes it not suitable for many real-world applications:

1) Sensitivity to Initial Points: The performance of Bayesian optimization can be sensitive to the initial set of points sampled. Poorly chosen initial points can lead to suboptimal results or prolonged tuning.
2) High-Dimensional Spaces: Bayesian optimization's effectiveness diminishes as the dimensionality of the search space increases. It struggles to efficiently explore and exploit in high-dimensional spaces.
3) Complexity of Acquisition Functions: Bayesian optimization relies on selecting acquisition functions to determine the next evaluation point. Choosing or designing appropriate acquisition functions for specific problems can be challenging and requires domain knowledge.

In recent years, there has been a notable surge in the empirical success of deep neural networks (DNNs) across a wide spectrum of real-world applications, including engineering [17]–[24], cyber security [25]–[27], geology [28] and sensing [29]. These data-centric parameterized models have demonstrated their aptitude for capturing intricate nuances inherent in complex processes, obviating the necessity for explicit domain characterization.

Numerous recent studies have delved into the synergies between DNNs, state space models, and filtering techniques. Pioneering a DNN-estimation intersection, Krishnan et al. [30] established a groundbreaking connection between DNNs and Kalman filters, devising a unified algorithm that efficiently learns a spectrum of Kalman filters through variational methods for deep generative model training. Building on this, Rangapuram et al. [31] introduced a per-time-series linear state-space model parametrized by a jointly-learned Recurrent Neural Network (RNN), preserving the desirable attributes of state space models such as interpretability and data efficiency while harnessing the capacity of deep learning to discern intricate patterns in raw data. Krishnan et al. [32] proposed a unified algorithm capable of efficiently learning a diverse range of linear and non-linear state space models, including variants incorporating deep neural networks to model emission and transition distributions. In a real-time context, Revach et al. [33] presented KalmanNet, a state estimator learning from data to perform Kalman filtering under non-linear dynamics with partial information. By integrating a dedicated recurrent neural network module within the Kalman filter, this approach maintains the data efficiency and interpretability of the classic algorithm while implicitly learning complex dynamics from the data. Numerous subsequent works, building upon [33], extended KalmanNet to high-dimensional settings and smoothing scenarios [34]–[37].

This paper introduces an alternative strategy for hyperparameter tuning, aiming to circumvent the need for explicit and precise knowledge of the dynamic model. This RNN-UKF approach addresses the challenges posed by the Bayesian optimization method for hyperparameter tuning. The RNN-UKF objective is to learn hyperparameters from data, lever-

aging deep learning techniques, particularly RNNs, known for their effectiveness in time series tasks within challenging environments. The use of various nonlinear techniques can also be combined with imagery analysis for space tracking [38], [39] and video public safety [40], [41].

The subsequent sections of this paper are structured as follows: Section 2 offers a concise overview of both the UKF and RNNs. In Section 3, we introduce our methodology, which combines neural networks and UKF for autonomous hyperparameter tuning. Section 4 provides an in-depth examination of our numerical experiment settings, accompanied by discussions. Finally, Section 5 serves as the conclusion for the paper.

II. Preliminaries

A. Unscented Kalman Filter

The focus on current estimation systems includes discrete-time state-space models for dynamic systems. In particular, our focus is on nonlinear dynamics models that incorporate additive noise. The system can be represented by two primary equations, the state evolution equation and the observation equation:

$$\begin{aligned} x_k &= f(x_{k-1}) + e_{k-1}, \quad e_{k-1} \sim N(0, Q) \\ y_k &= h(x_k) + v_k, \quad v_k \sim N(0, R) \end{aligned} \tag{1}$$

The state evolution equation $x_k = f(x_{k-1}) + e_{k-1}$ describes how the state at time k, denoted as x_k, evolves from the previous state x_{k-1}. This evolution is determined by a nonlinear function $f(\cdot)$. The system is subject to additive white Gaussian noise (AWGN) e_{k-1} with a covariance matrix Q.

The observation equation $y_k = h(x_k) + v_k$, describes how the observations at time k, represented by y_k, are generated from the current state x_k. The observation process is modeled by a nonlinear function $h(\cdot)$ and is corrupted by AWGN v_k with a covariance matrix R. It's essential to note that x_k, e_k and v_k are considered independent random variables.

In the classical UKF, predefined hyperparameters α, β and κ are employed to facilitate state estimation. It's important to note that the UKF can be regarded as a specific case within the broader category of Gaussian filters [42]. Gaussian filters, including the UKF, adopt a Kalman-like structure to tackle state estimation tasks in dynamic systems [42].

In the Kalman filter framework, the mean $\hat{x}^+_{k-1}$ and covariance $\hat{P}^+_{k-1}$ of the state estimation at time-step $k-1$ play a central role. The general equations governing the Gaussian filter can be divided into two phases: prediction and update.

Prediction

The prediction step begins with the mean $(\hat{x}^+_{k-1})$ and covariance (P^+_{k-1}), the estimation of mean and covariance in previous step, to predict mean $(\hat{x}^-_k)$ and the corresponding covariance (P^-_k) before a measurement update.

$$\begin{aligned} \hat{x}^-_k &= \int f(x_{k-1}) N(x_{k-1} | \hat{x}^+_{k-1}, P^+_{k-1}) dx_{k-1} \\ P^-_k &= \int (f(x_{k-1}) - \hat{x}^-_k)(f(x_{k-1}) - \hat{x}^-_k)^T \\ &\quad \times N(x_{k-1} | \hat{x}^+_{k-1}, P^+_{k-1}) dx_{k-1} + Q \end{aligned} \tag{2}$$

Update

The update step incorporate new measurement to correct the estimated mean $(\hat{x}^+_k)$ and covariance matrix (P^+_k).

$$\begin{aligned} \mu_k &= \int h(x_k) N(x_k | \hat{x}^-_k, P^-_k) dx_k \\ S_k &= \int (h(x_k) - \mu_k)(h(x_k) - \mu_k)^T N(x_k | \hat{x}^-_k, P^-_k) dx_k + R \\ C_k &= \int (x_k - \hat{x}^-_k)(h(x_k) - u_k)^T N(x_k | \hat{x}^-_k, P^-_k) dx_k \\ K_k &= C_k S_k^{-1} \\ \hat{x}^+_k &= \hat{x}^-_k + K_k(y_k - \mu_k) \\ P^+_k &= P^-_k - K_k S_k K_k^T \end{aligned} \tag{3}$$

In the given context, S_k denotes the covariance related to innovation, while C_k represents the cross-covariance. It's worth noting that different strategies for approximating the moment integrals in these equations (1)-(3) lead to various Gaussian filters [43], such as the CKF. The UKF distinguishes itself by employing the Unscented Transform (UT) as a means to approximately compute these moment integrals. The quality of estimates provided by the UKF is significantly influenced by the parameters of the UT. The next Section explores how the UT parameters impact the accuracy of the UKF's estimates in the following discussion.

The UT, as outlined in reference [1], serves as an approximation method for estimating the mean and covariance of a random variable y. This variable arises from a nonlinear transformation $h(\cdot)$ applied to a Gaussian random variable x with specific parameters. The distribution of x is described as $N(\bar{x}, P_x)$ and x resides in $\mathbf{R}^{n_x}$. To perform the estimation, dimensionless UT parameters, denoted as α, β and κ in $\mathbf{R}^3$, come into play. They are instrumental in calculating a set of $2n_x + 1$ predefined points, often referred to as "sigma points," along with their corresponding weights:

$$\begin{aligned} X_0 &= \bar{x} \\ X_i &= \bar{x} + \left(\sqrt{(n_x + \lambda) P_x}\right) \qquad i = 1, ..., n_x \\ X_i &= \bar{x} - \left(\sqrt{(n_x + \lambda) P_x}\right) \qquad i = n_x + 1, ..., 2n_x \end{aligned}$$

where $(\cdot)_i$ denotes the i^{th} column of the matrix, and λ is a scaling parameter given by

$$\lambda = \alpha^2 (n_x + \kappa) - n_x$$

The corresponding weights are

$$w_0^{(m)} = \frac{\lambda}{n_x + \lambda}$$
$$w_0^{(c)} = \frac{\lambda}{n_x + \lambda} + (1 - \alpha^2 + \beta)$$
$$w_i^{(m)} = w_i^{(c)} = \frac{1}{2(n_x + \lambda)} \quad i = 1, ..., 2n_x$$

Sigma points X_i are then used in the nonlinear transformation $h(\cdot)$, yielding

$$Y_i = h(X_i)$$

Finally, the first two moments of y are approximated as

$$\hat{y} \approx \sum_{i=0}^{2n_x} w_i^{(m)} Y_i$$
$$P_y \approx \sum_{i=0}^{2n_x} w_i^{(c)} (Y_i - \bar{y})(Y_i - \bar{y})^T$$

Figure 1 below illustrates the prediction and update steps in classical unscented Kalman Filters with fixed hyperparameters

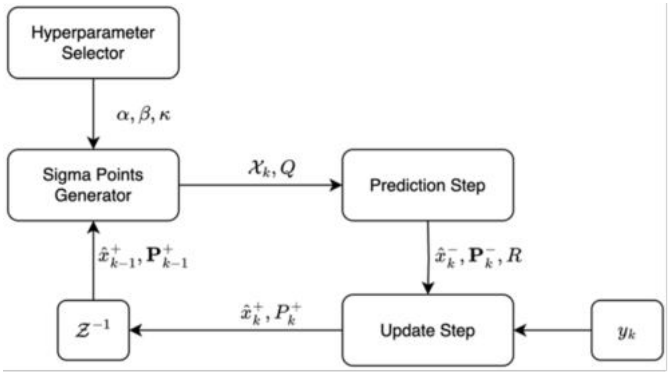

Fig. 1. Flow Chart for Classicl Unscented Kalman Filter

B. Recurrent Neural Network

RNN (Architecture) is a type of artificial neural network designed to work with sequences of data. Unlike traditional feedforward neural networks, which process individual data points independently, RNNs have the ability to capture temporal dependencies and patterns in sequential data. Hence, RNNs are particularly well-suited for tasks involving sequences, such as time series analysis [38].

The key characteristic of an RNN is its ability to maintain a hidden state that acts as a memory of the previous inputs it has seen in the sequence. A hidden state is updated at each time step, and it influences the predictions made at that time step. The hidden state allows RNNs to capture information from previous steps and carry it forward to influence future predictions. Figure 2 below shows the general RNN structure we utilized to generate hyperparameters.

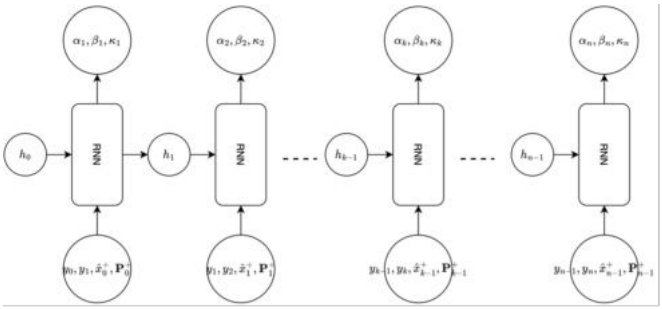

Fig. 2. Recurrent Neural Network Architecture

As shown in the Figure 2, at each time step k, the inputs of the network are $y_{k-1}, y_k, \hat{x}_{k-1}^+$ and P_{k-1}^+ with hyperparameters α_k, β_k and κ_k as output. The hidden states h_k are used to pass the information from previous time step to current time step.

Traditional RNNs have a limitation known as the "vanishing gradient" problem, where the influence of earlier time steps on later ones diminishes rapidly during training. To address this limitation, we use Gated Recurrent Unit (GRU) to include mechanisms that better manage the flow of information through the network's hidden states, enabling them to capture longer-term dependencies in sequences. Figure 3 shows the network architecture of one GRU unit, where σ and tanh are activation functions..

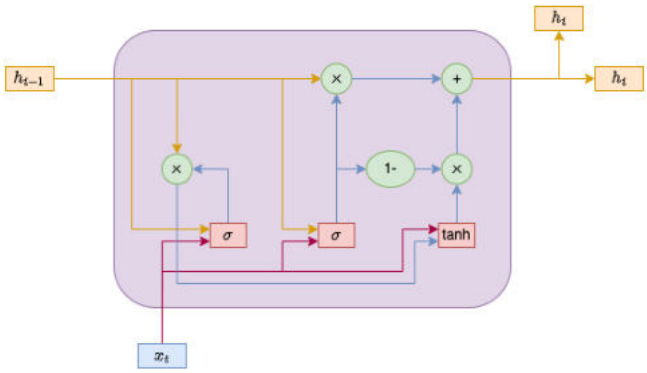

Fig. 3. Gated Recurrent Unit

GRUs incorporate gating mechanisms that help control the flow of information through the network's hidden states, allowing them to capture and retain long-term dependencies in sequential data.

The key components of a GRU are two gates: the reset gate and the update gate. These gates determine how much information is passed from the previous time step's hidden state and the current input to the current hidden state. The update gate (z_t) controls how much of the previous hidden state should be mixed with the current candidate hidden state. It takes into account the current input (x_t) and the previous

hidden state h_{t-1} and is expressed as

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

where σ is the sigmoid activation function, W_z is the weight matrix for the update gate and b_z is the bias term for the update gate.

The reset gate (r_t) determines how much of the previous hidden state should be ignored when computing the candidate hidden state and is expressed as

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

with W_r and b_r represent the corresponding weight matrix and bias.

The candidate hidden state $(\hat{h}_t)$ represents the information that could be included in the new hidden state, taking into account the current input and the previous hidden state, whereas the current hidden state (h_t) is a combination of the previous hidden state and the candidate hidden state, controlled by the update gate.

III. Methodology

A. Architecture Overview

Different from classical UKF which use a fixed set of α, β and κ during the whole estimation process, we allow these hyperparameters dynamically change over time k with α_k, β_k and κ_k, and the values for these parameters are generated by RNN. The inputs of RNN are current and previous measurements y_{k-1} and y_k, and previous state estimation $\hat{x}_{k-1}$ and corresponding estimation covariance matrix $\hat{P}_{k-1}$. Notice that these input values are known when we are performing state estimation at time step $k+1$. Figure 4 below illustrate the idea of the proposed method.

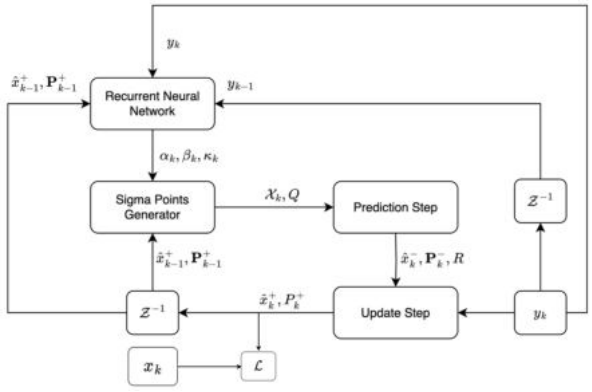

Fig. 4. Neural Network Assisted Hyperparameter Selection for UKF

B. Training Process

The automated hyperparameter selector is trained using the available labeled data set in a supervised learning manner. In particular, we use a RNN for generating the hyperparameter α_k, β_k and κ_k at each time step k rather than for directly producing the estimate $\hat{x}_k^+$. Namely, we compute the loss function $\mathcal{L}$ based on the state estimate $\hat{x}_k^+$, which is not the output of our RNN agent for hyperparameter selection. Since the loss-function vector takes values in continuous set $\mathbf{R}^m$, we use the squared-error loss,

$$\mathcal{L} = \|x_k - \hat{x}_k^+\|_2$$

where x_k represents the true latent state. By doing so, we build upon the ability to backpropagate the loss to the computation of the hyperparameters α_k, β_k and κ_k.

The data set used for training comprises N trajectories that can be of varying lengths. Namely, by letting T_i be the length of the i^{th} training trajectory, the data set is given by $\mathcal{D} = (Y_i, X_i)_1^N$, where

$$Y_i = [y_1^i, y_2^i, ..., y_{T_i}^i]$$
$$X_i = [x_0^i, x_1^i, ..., x_{T_i}^i]$$

By letting Θ denote the trainable parameters of the RNN, and let γ be a regularization coefficient, we then construct an ℓ_2 regularized mean-squared error loss measure

$$\ell_i(\Theta) = \frac{1}{T_i} \sum_{k=1}^{T_i} \|x_k^i - \hat{x}_k(y_k^i; \Theta)\|_2 + \gamma \|\Theta\|_2$$

To optimize Θ, we use a variant of mini-batch stochastic gradient descent in which for every batch indexed by s, we choose $M < N$ trajectories indexed by $i_1^s, i_2^s, ..., i_M^s$, computing the mini-batch loss as

$$\mathcal{L}_s(\Theta) = \frac{1}{M} \sum_{j=1}^{M} \ell_{i_j^s}(\Theta)$$

Since our RNN assisted hyperparameter selector is a recursive architecture with both an external recurrence and an internal RNN, we use the backpropagation through time (BPTT) algorithm to train it. Specifically, we unfold the selector across time with shared network parameters, and then compute a forward and backward gradient estimation pass through the network.

IV. Results

To demonstrate the RNN-UKF using the RNN, we choose the popular Lorenz attractor and a navigation use cases to demonstrate comparative performance. In our experiments, we use a two-layers long short-term memory (LSTM) network to select UKF hyperparameters.

A. Lorenz Attractor Use Case

The Lorenz attractor is a three-dimensional chaotic solution to the Lorenz system of ordinary differential equations in continuous time. The Lorenz synthetically generated system demonstrates the task of online tracking a highly non-linear trajectory and a real-world practical challenge of handling mismatches due to sampling a continuous time signal into discrete time.

In particular, the noiseless state-evolution of the continuous time process x_τ with $\tau \in \mathbf{R}^+$ is given by

$$\frac{\partial}{\partial \tau} x_\tau = A(x_\tau) \cdot x_\tau$$

with

$$A(x_\tau) = \begin{pmatrix} -10 & 10 & 0 \\ 28 & -1 & -x_{1,\tau} \\ 0 & x_{1,\tau} & -\frac{8}{3} \end{pmatrix}$$

We sample the noiseless process with sampling interval $\Delta\tau$ and assume that $A(x_\tau)$ can be kept constant in small neighborhood of x_τ:

$$A(x_\tau) \approx A(x_{\tau+\Delta\tau})$$

Then, the continuous time solution of the differential system is valid in the neighborhood of x_τ for a short time interval $\Delta\tau$:

$$x_{\tau+\Delta\tau} = exp(A(x_\tau) \cdot \Delta\tau) \cdot x_\tau$$

Finally, we take the Taylor series expansion of the above equation and a finite series approximation, which results in

$$F(x_\tau) = exp(A(x_\tau) \cdot \Delta\tau) \approx I + \sum_{j=1}^{J} \frac{(A(x_\tau) \cdot \Delta\tau)^j}{j!}$$

The resulting discrete time evolution process is thus given by

$$x_{t+1} = f(x_t) = F(x_t) \cdot x_t \tag{4}$$

Eq (4) represents the discrete time state evolution model with additional process noise is used for generating the simulated Lorenz attractor data with $J = 5$ Taylor order and $\Delta\tau = 0.02$ sampling interval. We also set $h(\cdot)$ to be the identity transformation, such that the observations are noisy versions of the true state. We generate 1000 training trajectories using the process model (4) and identity measurement model with random initial states under various noise scenarios.

Table I below presents the comparison of RNN-UKF (RUKF) with several other method in terms of root mean squared error (RMSE):

1) UKF-R: the UKF with hyperparameters suggested by [1] with $\alpha = 0.001, \beta = 2$ and $\kappa = 0$.
2) UKF-G: the UKF with hyperparameters chosen by grid search. 10 values of α are equally spaced over interval $[0.0001, 0.005]$, 10 values of β are equally spaced over interval $[0.1, 4]$ and 10 values of κ are equally spaced over interval $[-2, 2]$. Thus totally 1000 hyperparameter combinations are tried.
3) EKF: classical extended Kalman filter

Each RMSE reported in Table I is calculated by averaging 100 random testing sequences. Through the numerical experiments for this scenario, the measurement noise R is fixed to be diagonal matrix $diag([0.1, 0.1, 0.1])$. For low system noise case, the system noise Q is set to be $diag([0.001, 0.001, 0.001])$; For medium system noise case, the system noise Q is set to be $diag([0.1, 0.1, 0.1])$; For high system noise case, the system noise Q is set to be $diag([1, 1, 1])$.

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT FILTERING METHODS

	Low Sys Noise		Med Sys Noise		High Sys Noise	
	T=100	T=200	T=100	T=200	T=100	T=200
UKF-R	1.1508	1.2254	2.5188	3.5194	3.8278	4.0927
UKF-G	0.2043	0.2299	0.4219	0.4121	0.5729	0.6719
EKF	0.1458	**0.1406**	0.2271	0.2180	0.3081	0.2898
RUKF	**0.1427**	0.1418	**0.2075**	**0.1991**	**0.2866**	**0.2755**

Several insights can be observed from Table I:

1) The UKF-R has the worst performance over all testing cases. This conforms to our intuition since as observed in many applications, the performance of UKF is heavily dependent on the selection of hyperparameters.
2) The UKF-G always outperforms UKF-R. Since we conduct grid search to choose the optimal hyperparameter combinations over the grid, the performance improved. However, the performance of UKF-G is outperformed by EKF and RUKF. This might be due to that the grid does not cover the true optimal hyperparameters.
3) When system noise is low, the performance of EKF and RUKF are similar. However, when system noise increases, our method (RNN-UKF) always outperforms EKF, UKF-G and UKF-R.

B. Navigation Use Case

In our second navigation use case, we aim to localize an object that is moving according to the dynamic system of the form

$$x_{k+1} = F \cdot x_k + B(x_k) \cdot u_k \tag{5}$$

The state vector x is defined as follows:

$$x = [x_{pos}, y_{pos}, \theta, v]$$

where x_{pos} represents $x-$position of the agent; y_{pos} represents $y-$position of the agent; θ is the orientation of the agent; v is a constant velocity.

The control vector u is defined as $u = [v, \dot{\theta}]$ where v is the velocity of the agent and $\dot{\theta}$ is the rate of change of orientation (yaw rate) of the agent. Utilizing the major and minor axes in combination with sine and cosine functions, we can generate a non-linear ellipse model with state transition matrix F:

$$F = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

and control input matrix B:

$$B = \begin{pmatrix} \Delta t \cdot cos(\theta) & 0 \\ \Delta t \cdot sin(\theta) & 0 \\ 0 & \Delta t \\ 1 & 0 \end{pmatrix}$$

In this non-linear matrix B, Δt represents the time increment and cos, sin are the standard trigonometric function. This dynamic model represents a simple 2D motion model, describing the state vector that includes the x_{pos}, y_{pos}, orientation θ, and velocity v. It assumes constant velocity motion with a fixed yaw rate and employs a first-order linear approximation for state updates. The model takes inputs of velocity and yaw rate, incorporating them with the current state to predict the subsequent state. Additionally, there's an observation model that relates the true state to the measured state, introducing noise to simulate sensor measurements with inaccuracies:

$$y_{k+1} = H \cdot x_{k+1} \tag{6}$$

where

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Figure 5 below shows an example trajectory of the navigation use case, where the blue curve represents the true trajectory of the object moving according to system dynamic (5). The green dots represent the noisy measurements generated according to measurement model (6). The red curve represents the trajectory estimated using the proposed learning assisted UKF methodology.

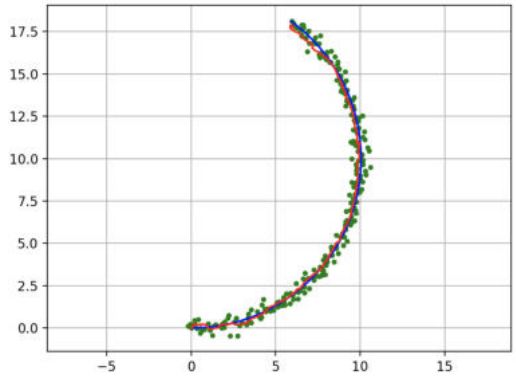

Fig. 5. Navigation Use Case

Similar to our previous Lorenz attractor use case, we compare the performance of the proposed learning RNN-UKF with other methods, including UKF-R, UKF-G and EKF, in terms of RMSE. Table 2 below shows the comparison results. Each RMSE reported in Table 2 is calculated by averaging 100 random testing sequences. Through the numerical experiments for this scenario, the system noise Q is fixed to be diagonal matrix $diag([0.1, 0.1, 0.023, 1.0])$. For low measurement noise case, the measurement noise R is set to be $diag([0.1, 0.1])$; For medium measurement noise case, the noise R is set to be $diag([0.7, 0.7])$; For high measurement noise case, the noise R is set to be $diag([2.5, 2.5])$.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT FILTERING METHODS

	Low Mea Noise		Med Mea Noise		High Mea Noise	
	T=100	T=200	T=100	T=200	T=100	T=200
UKF-R	7.1166	8.4568	15.3355	18.3112	17.7296	21.3769
UKF-G	0.7821	0.9812	2.4109	1.9222	9.1002	8.1723
EKF	**0.2359**	**0.2349**	0.2689	0.3091	2.9154	2.6091
RUKF	0.2366	0.2386	**0.2585**	**0.2477**	**2.1273**	**1.9166**

Several insights can be observed from Table II:

1) Similar to our previous use case, the UKF-R has the worst performance over all testing cases. This conforms to our intuition since as observed in many applications, the performance of UKF is heavily dependent on the selection of hyperparameters.
2) The UKF-G always outperforms UKF-R.
3) RUKF and EKF have similar performance in low measurement noise case. However, as measurement noise increase, the performance of the proposed RUKF is better than EKF and other methods.

V. CONCLUSION

This paper explored an innovative approach to hyperparameter tuning by combining neural networks and the unscented Kalman filter. Both the unscented Kalman filter and recurrent neural networks are relevant to estimation and time series tasks. Our proposed RNN-UKF for autonomous hyperparameter tuning leverages the synergies between these two powerful techniques, aiming to circumvent the challenges associated with explicit and precise knowledge of the dynamic model.

Through a detailed exposition of our numerical experiment settings, we demonstrated the efficacy of the RNN-UKF approach. The results showcased the capability of the RNN-UKF to dynamically learn hyperparameters from data, offering a promising alternative to traditional hyperparameter tuning methods.

In future work, further exploration and refinement of the combined approach could lead to enhanced performance across a broader array of applications. Another direction is to utilize the high fidelity modeling for support [40]. Additionally, investigating the scalability and robustness of the proposed methodology in high-dimensional settings and more complex scenarios could provide valuable insights for its practical implementation.

ACKNOWLEDGMENT

The authors acknowledge the support of the Air Force through the Contract Award Number: FA9453-22-C-A106. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Government.

REFERENCES

[1] E. A. Wan and R. 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)*. IEEE, 2000, pp. 153–158.

[2] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[3] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.

[4] L. A. Scardua and J. J. Da Cruz, "Complete offline tuning of the unscented kalman filter," *Automatica*, vol. 80, pp. 54–61, 2017.

[5] Z. Chen, C. Heckman, S. Julier, and N. Ahmed, "Weak in the nees?: Auto-tuning kalman filters with bayesian optimization," in *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, 2018, pp. 1072–1079.

[6] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on automatic control*, vol. 45, no. 3, pp. 477–482, 2000.

[7] A. Sakai and Y. Kuroda, "Discriminatively trained unscented kalman filter for mobile robot localization." *Journal of Advanced Research in Mechanical Engineering*, vol. 1, no. 3, 2010.

[8] J. Dunik, M. Simandl, and O. Straka, "Unscented kalman filter: aspects and adaptive setting of scaling parameter," *IEEE Transactions on Automatic Control*, vol. 57, no. 9, pp. 2411–2416, 2012.

[9] O. Straka, J. Duník, and M. Šimandl, "Unscented kalman filter with advanced adaptation of scaling parameter," *Automatica*, vol. 50, no. 10, pp. 2657–2664, 2014.

[10] O. Straka, J. Duník, M. Šimandl, and E. Blasch, "Randomized unscented transform in state estimation of non-gaussian systems: Algorithms and performance," in *2012 15th International Conference on Information Fusion*. IEEE, 2012, pp. 2004–2011.

[11] J. Dunik, O. Straka, M. Simandl, and E. Blasch, "Random-point-based filters: Analysis and comparison in target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 2, pp. 1403–1421, 2015.

[12] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[13] R. Turner and C. E. Rasmussen, "Model based learning of sigma points in unscented kalman filtering," *Neurocomputing*, vol. 80, pp. 47–53, 2012.

[14] A. Bertipaglia, B. Shyrokau, M. Alirezaei, and R. Happee, "A two-stage bayesian optimisation for automatic tuning of an unscented kalman filter for vehicle sideslip angle estimation," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 670–677.

[15] B. Jia, E. Blasch, K. D. Pham, D. Shen, Z. Wang, X. Tian, and G. Chen, "Space object tracking and maneuver detection via interacting multiple model cubature kalman filters," in *2015 IEEE Aerospace conference*. IEEE, 2015, pp. 1–8.

[16] Y. Bai, B. Yan, C. Zhou, T. Su, and X. Jin, "State of art on state estimation: Kalman filter driven by machine learning," *Annual Reviews in Control*, vol. 56, p. 100909, 2023.

[17] Z. Fan, K. Chang, A. K. Raz, A. Harvey, and G. Chen, "Sensor tasking for space situation awareness: Combining reinforcement learning and causality," in *2023 IEEE Aerospace Conference*. IEEE, 2023, pp. 1–9.

[18] Z. Fan, K.-c. Chang, R. Ji, and G. Chen, "Data fusion for optimal condition-based aircraft fleet maintenance with predictive analytics." *J. Adv. Inf. Fusion*, vol. In Press, 2023.

[19] Z. Fan, W. Li, and K.-C. Chang, "A bidirectional long short-term memory autoencoder transformer for remaining useful life estimation," *Mathematics*, vol. 11, no. 24, p. 4972, 2023.

[20] ——, "A two-stage attention-based hierarchical transformer for turbofan engine remaining useful life prediction," *Sensors*, vol. 24, no. 3, p. 824, 2024.

[21] Y. Bao, K. J. Chan, A. Mesbah, and J. M. Velni, "Learning-based adaptive-scenario-tree model predictive control with improved probabilistic safety using robust bayesian neural networks," *International Journal of Robust and Nonlinear Control*, vol. 33, no. 5, pp. 3312–3333, 2023.

[22] Y. Bao, H. S. Abbas, and J. Mohammadpour Velni, "A learning-and scenario-based mpc design for nonlinear systems in LPV framework with safety and stability guarantees," *International Journal of Control*, pp. 1–20, 2023.

[23] P. Cheng, Z. Xiong, Y. Bao, P. Zhuang, Y. Zhang, E. Blasch, and G. Chen, "A deep learning-enhanced multi-modal sensing platform for robust human object detection and tracking in challenging environments," *Electronics*, vol. 12, no. 16, p. 3423, 2023.

[24] Z. Fan, G. Chen, K. Chang, S. Khan, and M. Franco, "Explainable deep reinforcement learning for space situational awareness: Counterfactual explanation approach," in *2024 IEEE Aerospace Conference*. IEEE, 2024, pp. 1–10.

[25] F. L. Greitzer, W. Li, K. B. Laskey, J. Lee, and J. Purl, "Experimental investigation of technical and human factors related to phishing susceptibility," *ACM Transactions on Social Computing*, vol. 4, no. 2, pp. 1–48, 2021.

[26] W. Li, J. Lee, J. Purl, F. Greitzer, B. Yousefi, and K. Laskey, "Experimental investigation of demographic factors related to phishing susceptibility," in *Hawaii international conference on system sciences*, 2020.

[27] Z. Fan, W. Li, K. B. Laskey, and K.-C. Chang, "Investigation of phishing susceptibility with explainable artificial intelligence," *Future Internet*, vol. 16, no. 1, p. 31, 2024.

[28] W. Li, M. M. Finsa, K. B. Laskey, P. Houser, and R. Douglas-Bate, "Groundwater level prediction with machine learning to support sustainable irrigation in water scarcity regions," *Water*, vol. 15, no. 19, p. 3473, 2023.

[29] U. K. Majumder, E. P. Blasch, and D. A. Garren, *Deep learning for radar and communications automatic target recognition*. Artech House, 2020.

[30] R. G. Krishnan, U. Shalit, and D. Sontag, "Deep kalman filters," *arXiv preprint arXiv:1511.05121*, 2015.

[31] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," *Advances in neural information processing systems*, vol. 31, 2018.

[32] R. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[33] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. Van Sloun, and Y. C. Eldar, "Kalmannet: Neural network aided kalman filtering for partially known dynamics," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1532–1547, 2022.

[34] G. Revach, N. Shlezinger, T. Locher, X. Ni, R. J. van Sloun, and Y. C. Eldar, "Unsupervised learned kalman filtering," in *2022 30th European Signal Processing Conference (EUSIPCO)*. IEEE, 2022, pp. 1571–1575.

[35] X. Ni, G. Revach, N. Shlezinger, R. J. van Sloun, and Y. C. Eldar, "Rtsnet: Deep learning aided kalman smoothing," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 5902–5906.

[36] G. Choi, J. Park, N. Shlezinger, Y. C. Eldar, and N. Lee, "Split-kalmannet: A robust model-based deep learning approach for state estimation," *IEEE Transactions on Vehicular Technology*, 2023.

[37] X. Ni, G. Revach, and N. Shlezinger, "Adaptive kalmannet: Data-driven kalman filter with fast adaptation," *arXiv preprint arXiv:2309.07016*, 2023.

[38] B. Jia, K. D. Pham, E. Blasch, Z. Wang, D. Shen, and G. Chen, "Space object classification using deep neural networks," in *2018 IEEE Aerospace Conference*. IEEE, 2018, pp. 1–8.

[39] Y. Wang, X. Bai, H. Peng, G. Chen, D. Shen, E. Blasch, and C. B. Sheaff, "Gaussian-binary classification for resident space object maneuver detection," *Acta Astronautica*, vol. 187, pp. 438–446, 2021.

[40] E. Blasch, R. Xu, S. Y. Nikouei, and Y. Chen, "A study of lightweight DDDAS architecture for real-time public safety applications through hybrid simulation," in *2019 Winter Simulation Conference (WSC)*. IEEE, 2019, pp. 762–773.

[41] B. Liu, Y. Chen, E. Blasch, K. Pham, D. Shen, and G. Chen, "A holistic cloud-enabled robotics system for real-time video tracking application," in *Future Information Technology: FutureTech 2013*. Springer, 2014, pp. 455–468.

[42] K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE transactions on automatic control*, vol. 45, no. 5, pp. 910–927, 2000.

[43] Y. Wu, D. Hu, M. Wu, and X. Hu, "A numerical-integration perspective on gaussian filters," *IEEE Transactions on Signal Processing*, vol. 54, no. 8, pp. 2910–2921, 2006.