Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

High-accuracy target tracking for multistatic passive radar based on a deep feedforward neural network*

Baoxiong XU†, Jianxin YI†‡, Feng CHENG†, Ziping GONG, Xianrong WAN

Electronic Information School, Wuhan University, Wuhan 430072, China

†E-mail: jessiexu@whu.edu.cn; jxyi@whu.edu.cn; cwing@whu.edu.cn

Received June 17, 2022; Revision accepted Jan. 12, 2023; Crosschecked July 26, 2023

Abstract: In radar systems, target tracking errors are mainly from motion models and nonlinear measurements. When we evaluate a tracking algorithm, its tracking accuracy is the main criterion. To improve the tracking accuracy, in this paper we formulate the tracking problem into a regression model from measurements to target states. A tracking algorithm based on a modified deep feedforward neural network (MDFNN) is then proposed. In MDFNN, a filter layer is introduced to describe the temporal sequence relationship of the input measurement sequence, and the optimal measurement sequence size is analyzed. Simulations and field experimental data of the passive radar show that the accuracy of the proposed algorithm is better than those of extended Kalman filter (EKF), unscented Kalman filter (UKF), and recurrent neural network (RNN) based tracking methods under the considered scenarios.

Key words: Deep feedforward neural network; Filter layer; Passive radar; Target tracking; Tracking accuracy
https://doi.org/10.1631/FITEE.2200260 **CLC number:** TN95

1 Introduction

Target tracking is a fundamental and active research topic in many fields, such as radar, navigation, and smart driving systems. The primary purpose of target tracking is to accurately estimate the target state in motion space. High-accuracy tracking is important, especially for the detection of small and low-speed targets (e.g., unmanned aerial vehicles (UAVs)), which may help terminate the sabotage activities of UAVs and ensure better and safer airspace management. How to improve the tracking accuracy remains an important problem (Singer RA, 1970; Wang et al., 2009; Radmard et al., 2013; Choi et al., 2014; Jiang et al., 2015; Yi et al., 2015).

According to the approximation method of the posterior probability density function (PDF) (Smidl and Quinn, 2008; Mazuelas et al., 2013), the existing tracking algorithms can be categorized into two types, namely analytical Gaussian approximate tracking (Singer H, 2008; Saha et al., 2014; Ning et al., 2017) and random sampling approximate tracking (Gordon et al., 1993; Higuchi, 1997; Doucet et al., 2001; Schön et al., 2005; Bengtsson et al., 2008; Yin and Zhu, 2015). The first type is designed by approximating the PDF of the target state to a Gaussian or mixed Gaussian distribution. The extended Kalman filter (EKF) handles the nonlinearity of the tracking model with linear approximation around the predicted tracking point at each frame (Saha et al., 2014). The tracking state may be inaccurate or even divergent due to strong nonlinearity. The unscented Kalman filter (UKF) (Ning et al., 2017) is based on the unscented transformation. The approximation result of UKF is usually better than that of EKF. Singer H (2008) used the

‡ Corresponding author
* Project supported by the National Natural Science Foundation of China (Nos. 61931015, 62071335, and 61831009) and the Natural Science Foundation of Hubei Province, China (No. 2021CFA002)
ORCID: Jianxin YI, https://orcid.org/0000-0003-0585-0445

Gauss–Hermite (GH) integration rule to approximate the nonlinear part of the tracking model. The effectiveness of these analytical Gaussian approximate tracking approaches has been demonstrated in many studies (Singer H, 2008; Smidl and Quinn, 2008; Mazuelas et al., 2013; Saha et al., 2014; Ning et al., 2017). However, they have difficulties in dealing with arbitrary distributions in the tracking system.

The second type provides another effective state estimation tool for the nonlinear tracking system. Compared with the aforementioned algorithms, it approximates the conditional probability distribution of the state and realizes the transition of the conditional probability using the Bayesian theory (Gordon et al., 1993). Doucet et al. (2001) introduced the Monte Carlo (MC) simulation method into the particle filter (PF). A number of weighted samples (or particles) were used to approximate the posterior distribution. It may achieve high accuracy at the cost of unacceptable complexity and numerical problems, especially when the dimension of the state space becomes large (Bengtsson et al., 2008). Schön et al. (2005) decomposed the state vector into two disjoint components, including a linear state component and a nonlinear state component in marginalized particle filtering (MPF). They were estimated by a bank of Kalman filters (KFs) and by a PF, separately. This approach can improve the cost-effectiveness of the tracking algorithm. Higuchi (1997) revealed that PF and the genetic algorithm (GA) share a similar structure. Based on the observations, Yin and Zhu (2015) proposed genetic PF (GPF) by incorporating the genetic operators in PF to mitigate particle impoverishment. Nonetheless, most of these improved PFs have more complex strategies and suffer from a higher computational burden.

The aforementioned two types of methods are dependent on an accurate system model. Nevertheless, an accurate system model requires the extraction of more useful information from the measurement data. Since the measurement data are nonlinear and contain errors, target tracking based on an accurate model is challenging and difficult to adapt to application requirements. Hence, target tracking is evolving from model-driven to data-driven. In recent years, data-driven approaches have achieved great breakthroughs and have been applied to many fields (Oong and Isa, 2011; Ding et al., 2016). We note that a neural network (NN) is proven to have the potential to approximate the desired model with arbitrary accuracy by the universal approximation theorem (Hornik et al., 1989; Hornik, 1991). In fact, researchers have paid attention to the application of NNs in target tracking since 1994. Chin (1994) and Amoozegar and Sundareshan (1994) incorporated NN into KF to provide more information on manoeuvre targets. However, these NNs are too shallow to approximate the complex features of tracks due to the limited parameters in the networks. Gu et al. (2020) used deep reinforcement learning to track a mobile radiation source based only on the received signal strengths. Fatseas and Bekooij (2019) developed a deep convolutional NN to track multiple targets directly from frequency modulated continuous wave (FMCW) range-Doppler maps. Rassalna and Mishra (2020) used single shot detection (SSD) to process the images provided by the electro optic (EO) sensor. Then, the processing results and the data from the radar are used for tracking. However, these deep NN based tracking approaches are designed for the original signals and images. They are not suitable for radar tracking with measurements. Gao et al. (2018) used a recurrent neural network (RNN) to track manoeuvre targets. Nonetheless, RNN learning relies on back-propagation through time (BPTT). BPTT significantly increases the computational complexity of the learning, and even worse, it may cause many problems in learning, e.g., gradient vanishing and exploding (Bengio et al., 1994). Liu JX et al. (2020) mitigated the influence of target manoeuvres on tracking using the long short-term memory (LSTM) model. Unfortunately, LSTM can remember only finite dependency information and suffers from high computational complexity. Thus, they cannot solve the high-accuracy tracking problem well.

Compared with the aforementioned NNs, the learning of deep feedforward neural networks (DFNNs) (Goodfellow et al., 2016) is much easier and faster. It is preferable to use a feedforward structure to learn the trajectory characteristics. Nevertheless, DFNN cannot model the sequence dependency well due to the limitation of the network structure. Therefore, the recurrent feedback of RNN can be regarded as an infinite impulse response (IIR) filter. We use a high-order

finite impulse response (FIR) filter to approximate it. Specifically, we add a new FIR filter layer that is placed between the hidden layers of the DFNN network. In addition, this modified DFNN (MDFNN) can realize sliding window control of the length of the measurement sequences. Based on MDFNN, a novel target tracking algorithm is constructed for high-accuracy tracking. First, the target track database is built. Then, we design and train about 10 MDFNNs and select the best one. Finally, a dataset of multistatic passive radar is used to validate the proposed algorithm.

The main contributions of this paper are as follows:

1. We design a new filter layer for MDFNN. Target tracking needs to establish the long-term dependency of measurement in time series, but DFNN cannot model the dependency without a recurrence structure. Hence, an FIR filter is embedded between the hidden layers to solve the problem. The proposed target tracking algorithm can handle measurement sequences of any length by a filter layer that is formed by FIR, extract more target motion features, and achieve higher tracking accuracy.

2. An accurate regression model is established efficiently by exploiting DFNN/MDFNN for target feature extraction and state estimation. As mentioned above, compared with other recursive NNs, DFNN/MDFNN can achieve better generalization performance with a much higher learning rate. In addition, the influence of the measurement sequence size in the input layer on the tracking network is considered. The difference in measurement sequences affects the acquisition of target information and then influences the tracking performance. A tradeoff scheme is developed based on an experiment to select an appropriate sequence length.

2 Building the dataset of the target

In this paper, we build the dataset in the context of multistatic passive radars. A passive radar detection system (Griffiths and Long, 1986; Griffiths and Baker, 2005; Kuschel, 2013; Palmer et al., 2013) generally includes signal processing and data processing. Signal processing is mainly to obtain measurements that include the information of the target bistatic range, bistatic velocity, and azimuth. Data processing realizes target state estimation using measurements. In this section, the target tracking database is established. The samples in our database contain the measurements and real state of tracks, which constitute the input–output pairs for the high-accuracy target-tracking network. It is well known that the movement of the target can be simulated according to the corresponding model (Li and Jilkov, 2003), which means that the large amount of data required by the proposed tracking network in the training stage is usually not a problem. To make the tracks in our tracking database closer to reality, all the parameters used to generate the tracks are set according to the real scenarios.

The model of the track generator (MTG) is defined as follows:

$$\text{state equation: } \boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{v}_k, \tag{1}$$

$$\text{measurement equation: } \boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k, \tag{2}$$

where $\boldsymbol{x}_k = [x_k, \dot{x}_k, \ddot{x}_k, y_k, \dot{y}_k, \ddot{y}_k]^{\mathrm{T}}$ is the target state at time k, (x_k, y_k) are the positions of the target at time k, $(\dot{x}_k, \dot{y}_k)$ are the velocities of the target along the x and y directions at time k, and $(\ddot{x}_k, \ddot{y}_k)$ are the accelerations of the target along the x and y axes at time k. $\boldsymbol{z}_k$ is the corresponding measurement at time k. In Eq. (1), $\boldsymbol{f}$ is the state transition function, and the process noise $\boldsymbol{v}_k$ is assumed to be zero-mean additive white Gaussian noise. In Eq. (2), $\boldsymbol{h}$ is the nonlinear measurement function. The measurement noise $\boldsymbol{w}_k$ is also assumed to be additive zero-mean white Gaussian noise, and the covariance matrix of $\boldsymbol{w}_k$ conforms to

$$E\{\boldsymbol{w}_k \boldsymbol{w}_j^{\mathrm{T}}\} = \boldsymbol{R}_k \delta_{kj}, \tag{3}$$

where δ_{kj} is the Kronecker Delta function. According to MTG, the real state $\boldsymbol{x}_{1:K} = \{\boldsymbol{x}_k : k = 1, 2, \cdots, K\}$ of tracks is generated by Eq. (1), and the measurements $\boldsymbol{z}_{1:K} = \{\boldsymbol{z}_k : k = 1, 2, \cdots, K\}$ are generated by Eq. (2). The measurement is related to the radar system structure, which is analyzed as follows.

A multistatic passive radar system consists of N_{t} transmitters (i.e., $\boldsymbol{s}_{tn_{\mathrm{t}}}$, $n_{\mathrm{t}} = 1, 2, \cdots, N_{\mathrm{t}}$) and N_{r} receivers (i.e., $\boldsymbol{s}_{rn_{\mathrm{r}}}$, $n_{\mathrm{r}} = 1, 2, \cdots, N_{\mathrm{r}}$). Their positions are known and denoted by $\boldsymbol{s}_{tn_{\mathrm{t}}} = [x_{n_{\mathrm{t}}}, y_{n_{\mathrm{t}}}]^{\mathrm{T}}$ and $\boldsymbol{s}_{rn_{\mathrm{r}}} =$

$[x_{n_r}, y_{n_r}]^{\mathrm{T}}$. The measurement function $\boldsymbol{h}(\boldsymbol{x}_k)$ can be given by

$$\boldsymbol{h}(\boldsymbol{x}_k)=\begin{bmatrix}\boldsymbol{h}_1(\boldsymbol{x}_k)\\ \boldsymbol{h}_2(\boldsymbol{x}_k)\\ \vdots\\ \boldsymbol{h}_N(\boldsymbol{x}_k)\end{bmatrix}, \tag{4}$$

where $\boldsymbol{h}_n(\cdot)$ ($n=1,2,\cdots,N$, $N=N_{\mathrm{t}}\times N_{\mathrm{r}}$) is associated with the bistatic pair $\boldsymbol{s}_{tn_{\mathrm{t}}}$–$\boldsymbol{s}_{rn_{\mathrm{r}}}$. Generally, the measurement function of the n^{th} bistatic pair at time k can be expressed as

$$\boldsymbol{h}_n(\boldsymbol{x}_k)=\begin{bmatrix}r_{n,k}\\ \dot{r}_{n,k}\\ \varphi_{n,k}\end{bmatrix}=\begin{bmatrix}\left\|\boldsymbol{p}_k-\boldsymbol{s}_{tn_{\mathrm{t}}}\right\|+\left\|\boldsymbol{p}_k-\boldsymbol{s}_{rn_{\mathrm{r}}}\right\|\\ \left(\dfrac{\boldsymbol{s}_{tn_{\mathrm{t}}}-\boldsymbol{p}_k}{\left\|\boldsymbol{s}_{tn_{\mathrm{t}}}-\boldsymbol{p}_k\right\|}+\dfrac{\boldsymbol{s}_{rn_{\mathrm{r}}}-\boldsymbol{p}_k}{\left\|\boldsymbol{s}_{rn_{\mathrm{r}}}-\boldsymbol{p}_k\right\|}\right)^{\mathrm{T}}\boldsymbol{v}\\ \mathrm{atan2}(x_k-x_{n_r},y_k-y_{n_r})\end{bmatrix}, \tag{5}$$

where $\boldsymbol{v}$ is the target motion velocity, $r_{n,k}$ is defined as the bistatic range of the n^{th} bistatic pair at time k, $\boldsymbol{p}_k$ is the two-dimensional (2D) target position, denoted by $\boldsymbol{p}_k=[x_k,y_k]^{\mathrm{T}}$, $\dot{r}_{n,k}$ is the bistatic velocity of the n^{th} bistatic pair at time k, $\|\cdot\|$ denotes the Euclidean norm, $\varphi_{n,k}$ is the azimuth of the n^{th} bistatic pair at time k, and atan2$(\cdot,\cdot)$ is the four-quadrant inverse tangent. Assuming that $N=3$, the measurement function $\boldsymbol{h}(\boldsymbol{x}_k)$ at time k can be represented as

$$\boldsymbol{h}(\boldsymbol{x}_k)=[r_{1,k},r_{2,k},r_{3,k},\dot{r}_{1,k},\dot{r}_{2,k},\dot{r}_{3,k},\varphi_{1,k},\varphi_{2,k},\varphi_{3,k}]^{\mathrm{T}}. \tag{6}$$

In this paper, the transition function $\boldsymbol{f}$ in Eq. (1) is defined in a constant acceleration (CA) shape (Li and Jilkov, 2003). Using MTG, the measurements $\boldsymbol{z}_{1:K}=\{\boldsymbol{z}_k: k=1,2,\cdots,K\}$ and corresponding states $\boldsymbol{x}_{1:K}=\{\boldsymbol{x}_k: k=1,2,\cdots,K\}$ can be generated when the parameters of the model, i.e., the initial state $\boldsymbol{x}_0$, number of times K, process noise $\boldsymbol{v}_k$, measurement noise $\boldsymbol{w}_k$, and transition function $\boldsymbol{f}$, are set.

Before generating the samples of the target tracking database, we must determine the range of the target trajectory to ensure that they cover all possible UAV detection fields. In Fig. 1, the shaded area is the common detection area of the multistatic passive radar system. The boundary values in the detection area are determined according to the flight path of the UAVs. As shown in Table 1, the distance from the radar to the target covers the main detection range of the passive radar, i.e., 0.2 to 11 km. The velocity of the target is set in the range of 0–26 m/s, which covers all common velocities of UAVs as much as possible. The azimuth ranges from −60° to 60°.

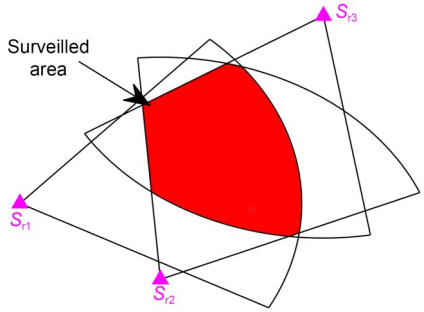

Fig. 1 Graphical illustration of the surveilled area for passive radar

Table 1 Ranges of target states

Parameter	Range
Distance from the passive radar	0.2–11 km
Speed of the UAVs	0–26 m/s
Azimuth from the passive radar to the UAVs	−60°–60°

Accordingly, the parameters of MTG are derived as follows:

1. Since a fixed length of track segments is required in the tracking network, the lengths of all the track segments $\mathrm{tr}_{\mathrm{segment}}$ are defined to be 100 s in the target tracking database. The length of a sample track is set to 1000 s. Hence, a sequence truncation stage is inserted and summarized in Algorithm 1. In practice, a long track can be considered a combination of several track segments.

2. The initial state $\boldsymbol{x}_0$ is calculated by using polar coordinate decomposition. To ensure that the entire tracks satisfy the passive radar detection ranges described in Fig. 1, the initial distance of the target is set to range from $200+V_{\max}\cdot\mathrm{tr}_{\mathrm{segment}}$ to $11\,000-V_{\max}\cdot\mathrm{tr}_{\mathrm{segment}}$, where $V_{\max}$=26 m/s is the maximum velocity of the UAV. Then, we uniformly sample a random distance R_{random} from this range to obtain the initial position (x_0, y_0) of the track in the x and y directions as

Algorithm 1 Input sequence setting for complex measurement data from the track segment of a given target in the training phase

Input: measurement sequence $\boldsymbol{z}_{1:K}$ and the corresponding state sequence $\boldsymbol{x}_{1:K}$.

// K is the total length of the sample track

Initialization: set $\varepsilon = 100$

// Length of the truncated subsequence

Num = $K-\varepsilon$ // Number of target subsequences

For n=1 to Num **do**

$M_\mathrm{i} = \boldsymbol{z}_{1:K}(:, n:\varepsilon)$

$M_\mathrm{o} = \boldsymbol{x}_{1:K}(:, n:\varepsilon)$

$\mathrm{Se_i}\{n\}=M_\mathrm{i}$ // Total subsequences

$\mathrm{Se_o}\{n\}=M_\mathrm{o}$

$\varepsilon = \varepsilon + 1$

End For

$$x_0 = R_{\text{random}} \cdot \sin\theta_\mathrm{R}, y_0 = R_{\text{random}} \cdot \cos\theta_\mathrm{R}, \tag{7}$$

where θ_R is the angle between the north and the direction from the radar, and it is uniformly sampled from 0 to π. Likewise, the initial velocities $(\dot{x}_0, \dot{y}_0)$ in the x and y directions are calculated as

$$\dot{x}_0 = V_{\text{random}} \cdot \sin\theta_\mathrm{V}, \dot{y}_0 = V_{\text{random}} \cdot \cos\theta_\mathrm{V}, \tag{8}$$

where V_{random} is the random velocity uniformly sampled from 0 to 26 m/s, and θ_V is the angle between the north and the initial direction of the target velocity, uniformly sampled from −π to π. Similarly, the initial accelerations $(\ddot{x}_0, \ddot{y}_0)$ are computed as

$$\ddot{x}_0 = A_{\text{random}} \cdot \sin\theta_\mathrm{A}, \ddot{y}_0 = A_{\text{random}} \cdot \cos\theta_\mathrm{A}, \tag{9}$$

where A_{random} is the random accelerated velocity uniformly sampled from 0.1 to 5 m/s^2, and θ_A denotes the angle between the north and the initial direction of the target acceleration, uniformly sampled from −π to π. Thus, the initial state of the track segment is $\boldsymbol{x}_0 = [x_0, \dot{x}_0, \ddot{x}_0, y_0, \dot{y}_0, \ddot{y}_0]^\mathrm{T}$.

3. According to Eqs. (1) and (2), the process noise $\boldsymbol{v}_k$ and measurement noise $\boldsymbol{w}_k$ are defined as

$$\boldsymbol{v}_k \sim \mathcal{N}(0, \boldsymbol{Q}_k), \boldsymbol{Q}_k = E\{\boldsymbol{v}_k \boldsymbol{v}_k^\mathrm{T}\} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_1 \end{bmatrix}, \tag{10}$$

$$\begin{cases} \boldsymbol{w} = [w_r, w_{\dot{r}}, w_\varphi]^\mathrm{T}, \quad w_r \sim \mathcal{N}(0, \sigma_r^2), \\ w_{\dot{r}} \sim \mathcal{N}(0, \sigma_{\dot{r}}^2), \quad w_\varphi \sim \mathcal{N}(0, \sigma_\varphi^2), \end{cases} \tag{11}$$

where $\mathcal{N}$ denotes a normal distribution. In Eq. (10), $\boldsymbol{Q}_k$ is the covariance matrix of $\boldsymbol{v}_k$, and $\boldsymbol{Q}_1$ is given by

$$\boldsymbol{Q}_1 = \begin{bmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{bmatrix} \tilde{q}, \tag{12}$$

where T is the sampling interval, $\tilde{q}$ is the power spectral density of the process noise, set to $\tilde{q}$=4 m^2/s^5 in this study. In Eq. (11), σ_{w_r} is the deviation of bistatic range noise randomly sampled in the range of [16, 30] m. $\sigma_{w_{\dot{r}}}$ and σ_φ are the deviations of the bistatic velocity and azimuth noise, randomly sampled in the ranges of [2, 4] m/s and [0.6°, 3°], respectively.

After obtaining all settings for MTG, 59 500 samples are generated based on the radar practical scenarios.

3 Proposed algorithm

In this section, the details of the proposed tracking algorithm are presented. Unlike traditional tracking algorithms, the proposed algorithm tracks the target based on an MDFNN that is trained off-line. The framework of the proposed algorithm is shown in Fig. 2. It contains the training and tracking stages. In the training stage, there are three main steps: (1) The original measurements and real state of tracks are processed to generate the normalized input–output pairs, which are suitable for training the tracking network. (2) To obtain a tracking network with the best generalization performance, the target measurement sequences are first decomposed into several groups. Then, a total of u (u=10) models are trained with these submeasurement sequences in each group and the corresponding real state of tracks. In the training models (i.e., MDFNNs), the cost function is defined by the mean square error (MSE). (3) The optimal tracking model is selected by cross-validation. In the tracking stage, the original measurement is first preprocessed using the bistatic target tracking (BTT) algorithm (Yi et al., 2015) to remove false measurements. The retained measurements are used to realize the one-step-ahead estimation of target states. Then, the entire target track is estimated by splicing the track segments. The details of the proposed tracking network are presented in the following.

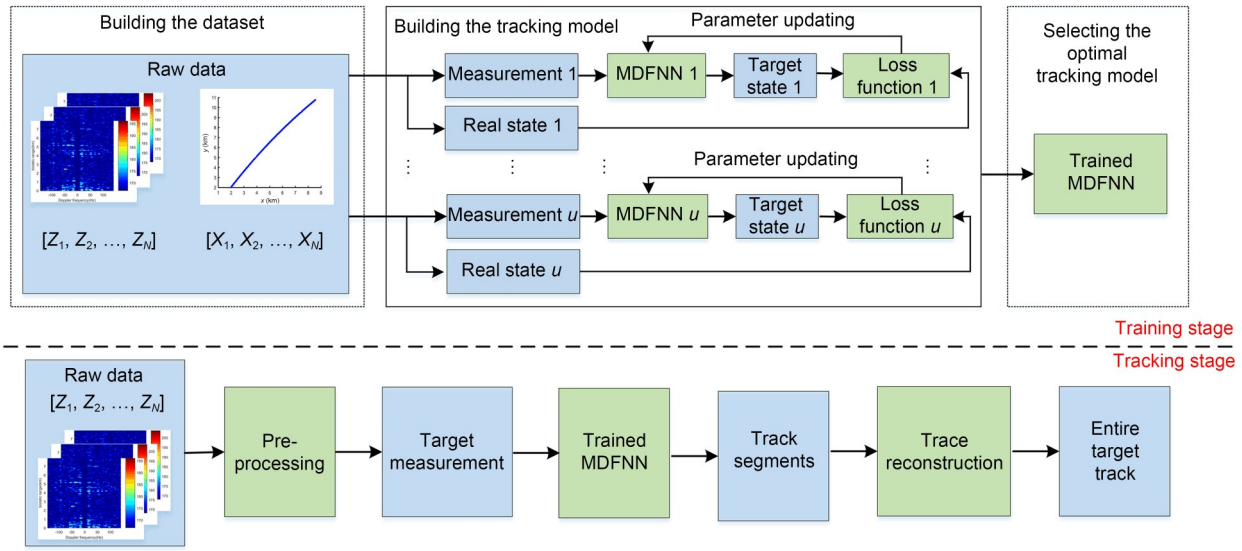

Fig. 2 Framework of the proposed tracking algorithm

The upper part shows the training stage where the preprocessed data are used to train the tracking network according to a designed cost function. The bottom part shows the tracking stage where the trained tracking network is used to estimate the target state

3.1 Preprocessing

There are two key problems when the measurement sequences are directly used to estimate the target state. The first problem is that the bistatic velocities and azimuths of input measurements are too small in comparison with the bistatic ranges. When they are fed into the tracking network and combined with bistatic ranges based on a random weight matrix, their data features are submerged by the bistatic range data. To relieve the problem, the min-max normalization technique is employed to modify the samples, which are suitable for training. Thus, the loss of the tracking network can be guaranteed to converge to a satisfactory small value. According to Eq. (6), input measurements $\boldsymbol{z}_{1:K}$ can be rewritten as

$$\begin{aligned}\boldsymbol{z}_{1:K} = [\,&\boldsymbol{r}_{1,1:K}, \boldsymbol{r}_{2,1:K}, \boldsymbol{r}_{3,1:K}, \dot{\boldsymbol{r}}_{1,1:K}, \dot{\boldsymbol{r}}_{2,1:K}, \dot{\boldsymbol{r}}_{3,1:K},\\ &\boldsymbol{\varphi}_{1,1:K}, \boldsymbol{\varphi}_{2,1:K}, \boldsymbol{\varphi}_{3,1:K}\,]^{\mathrm{T}},\end{aligned} \tag{13}$$

where $\boldsymbol{r}_{n,1:K}=[\,r_{n,1}, r_{n,2}, \cdots, r_{n,K}\,]$ $(n=1, 2, 3)$ denotes the bistatic range vector of the n^{th} bistatic pair, $\dot{\boldsymbol{r}}_{n,1:K} = [\,\dot{r}_{n,1}, \dot{r}_{n,2}, \cdots, \dot{r}_{n,K}\,]$ denotes the bistatic velocity vector of the n^{th} bistatic pair, and $\boldsymbol{\varphi}_{n,1:K}=[\,\varphi_{n,1}, \varphi_{n,2}, \cdots, \varphi_{n,K}\,]$ is the azimuth vector of the n^{th} bistatic pair. Measurement $\boldsymbol{z}_{1:K}$ is normalized to generate the input data $\boldsymbol{z}^{\mathrm{N}}_{1:K}$ for the tracking network as follows:

$$\begin{aligned}\boldsymbol{z}^{\mathrm{N}}_{1:K} = [\,&\boldsymbol{r}^{\mathrm{N}}_{1,1:K}, \cdots, \boldsymbol{r}^{\mathrm{N}}_{n,1:K}, \dot{\boldsymbol{r}}^{\mathrm{N}}_{1,1:K}, \cdots, \dot{\boldsymbol{r}}^{\mathrm{N}}_{n,1:K},\\ &\boldsymbol{\varphi}^{\mathrm{N}}_{1,1:K}, \cdots, \boldsymbol{\varphi}^{\mathrm{N}}_{n,1:K}\,]^{\mathrm{T}},\end{aligned} \tag{14}$$

where

$$\boldsymbol{r}^{\mathrm{N}}_{n,1:K} = [\,r^{\mathrm{N}}_{n,1}, r^{\mathrm{N}}_{n,2}, \cdots, r^{\mathrm{N}}_{n,K}\,], \tag{15}$$

$$\dot{\boldsymbol{r}}^{\mathrm{N}}_{n,1:K} = [\,\dot{r}^{\mathrm{N}}_{n,1}, \dot{r}^{\mathrm{N}}_{n,2}, \cdots, \dot{r}^{\mathrm{N}}_{n,K}\,], \tag{16}$$

$$\boldsymbol{\varphi}^{\mathrm{N}}_{n,1:K} = [\,\varphi^{\mathrm{N}}_{n,1}, \varphi^{\mathrm{N}}_{n,2}, \cdots, \varphi^{\mathrm{N}}_{n,k}\,], \tag{17}$$

$$r^{\mathrm{N}}_{n,i} = \frac{2(r_{n,i} - \min \boldsymbol{r}_{n,1:K})}{\max \boldsymbol{r}_{n,1:K} - \min \boldsymbol{r}_{n,1:K}} - 1, \tag{18}$$

$$\dot{r}^{\mathrm{N}}_{n,i} = \frac{2(\dot{r}_{n,i} - \min \dot{\boldsymbol{r}}_{n,1:K})}{\max \dot{\boldsymbol{r}}_{n,1:K} - \min \dot{\boldsymbol{r}}_{n,1:K}} - 1, \tag{19}$$

$$\varphi^{\mathrm{N}}_{n,i} = \frac{2(\varphi_{n,i} - \min \boldsymbol{\varphi}_{n,1:K})}{\max \boldsymbol{\varphi}_{n,1:K} - \min \boldsymbol{\varphi}_{n,1:K}} - 1, \tag{20}$$

where $n = 1, 2, 3$, and "max" and "min" are the maximum and minimum absolute values in the elements of the input vector, respectively. Meanwhile, the real state $\boldsymbol{x}_{1:K}$ of tracks is processed by the same normalization. The minimum and maximum values are determined. Obviously, the min-max normalization technique scales the real-time trained data $(\boldsymbol{z}_{1:K}, \boldsymbol{x}_{1:K})$ to values $(\boldsymbol{z}^{\mathrm{N}}_{1:K}, \boldsymbol{x}^{\mathrm{N}}_{1:K})$ between -1 and 1, which avoids data feature loss to a great extent. Thus, the tracking network can learn more target information.

The second problem is that false measurements are contained in radar detection, which may cause false trajectory output in the tracking stage. A preprocessing step is developed to modify the input of samples, which are suitable for tracking. Thus, the proposed tracking network can estimate the desired target

state. The preprocessing step is summarized in Fig. 3. On one hand, the field experimental measurements are disposed by the BTT algorithm with the transition matrix in the CA shape, where the false measurements contained by the field experimental measurements can be removed. Specifically, based on the framework of bistatic coordinates, association hypothesis decision and global association are first used to associate the measurements with each other and form the starting trajectory. Then the measurement of each scan is associated with the track, and KF is used to update the track states for the associated track. Finally, the raw measurement data associated with the bistatic track are outputted (Yi et al., 2015). On the other hand, the difference between the target measurements $\hat{\boldsymbol{z}}_{1:K}$ extracted is still large, and the min-max normalization technique is adopted to generate the final input data $\hat{\boldsymbol{z}}^{\mathrm{N}}_{1:K}$ for the high-accuracy network.

3.2 Target tracking network model

The proposed tracking network is used to learn the information of input sequence $\boldsymbol{z}^{\mathrm{N}}_{1:K}$ to output the estimates of state $\boldsymbol{x}_{1:K}$. Once the network is well trained, it can directly and precisely output the target state without any information of the state truth. As DFNN cannot map arbitrary measurement sequences from time k to K to the current state output at time K, MDFNN with an FIR filter added between the hidden layers is proposed to solve the above problem. MDFNN is developed with an input layer, three hidden layers, two filter layers, and an output filter. The structure of MDFNN is summarized in Fig. 4a. $\boldsymbol{h}_{i,k}$ (i=1, 2, 3) and $\tilde{\boldsymbol{h}}_{j,k}$ (j=1, 2) denote the output of the i^{th} hidden layer and j^{th} filter layer at time k, respectively. $\boldsymbol{W}_i$ (i=1, 2, 3, 4) is the weight matrix, and $\boldsymbol{W}_{\mathrm{F},i}$ (i=1, 2) is the weight matrix connecting the i^{th} filter layer to the $(i+1)^{\mathrm{th}}$ hidden layer. $\boldsymbol{c}_k$ denotes the coefficient vector connecting the hidden layer to the filter layer. Fig. 4b shows the sliding window (SW) processing of the filter layer. The filter layer encodes all measurement sequences from time k to $k+M$ into a fixed-size representation. Then, the output of the filter layer and the input of the current hidden layer are fed into the next hidden layer. The tracking network can also be called the M-order tracking network. Assume that the three hidden layers have l_1, l_2, and l_3 nodes in MDFNN, separately. The mathematical model of the tracking network can be expressed as follows:

$$\boldsymbol{h}_{1,k}=f_{\mathrm{h}}(\boldsymbol{W}_1^{\mathrm{T}}\boldsymbol{z}_k+\boldsymbol{\kappa}_1)=f_{\mathrm{h}}\left(\sum_{i=1}^{9}\boldsymbol{W}_1^{\mathrm{T}}(j,i)\boldsymbol{z}_k(i)+\boldsymbol{\kappa}_1(j)\right),\quad j=1,2,\cdots,l_1, \tag{21}$$

$$\begin{aligned}\tilde{\boldsymbol{H}}_{1,k}&=[\,\tilde{\boldsymbol{h}}_{1,1},\tilde{\boldsymbol{h}}_{1,2},\cdots,\tilde{\boldsymbol{h}}_{1,k}]\\&=[\,\boldsymbol{h}_{1,1},\boldsymbol{h}_{1,2},\cdots,\boldsymbol{h}_{1,k}][\boldsymbol{c}_1,\boldsymbol{c}_2,\cdots,\boldsymbol{c}_k],\end{aligned} \tag{22}$$

$$\boldsymbol{c}_k=\begin{cases}[c_1,c_2,\cdots,c_{k-L+1},c_{k-L+2},\cdots,c_k]^{\mathrm{T}}, & k\geq M,\\ [c_1,c_2,\cdots,c_k,\underbrace{0,\cdots,0}_{M-k}]^{\mathrm{T}}, & k<M,\end{cases} \tag{23}$$

$$\begin{aligned}\boldsymbol{h}_{2,k}&=f_{\mathrm{h}}(\boldsymbol{W}_2^{\mathrm{T}}\boldsymbol{h}_{1,k}+\boldsymbol{W}_{\mathrm{F},1}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{1,k}+\boldsymbol{\kappa}_2)\\&=f_{\mathrm{h}}\left(\sum_{i=1}^{l_1}\boldsymbol{W}_2^{\mathrm{T}}(j,i)\boldsymbol{h}_{1,k}(i)+\sum_{i=1}^{l_1}\boldsymbol{W}_{\mathrm{F},1}^{\mathrm{T}}(j,i)\tilde{\boldsymbol{h}}_{1,k}(i)+\boldsymbol{\kappa}_2(j)\right),\quad j=1,2,\cdots,l_2,\end{aligned} \tag{24}$$

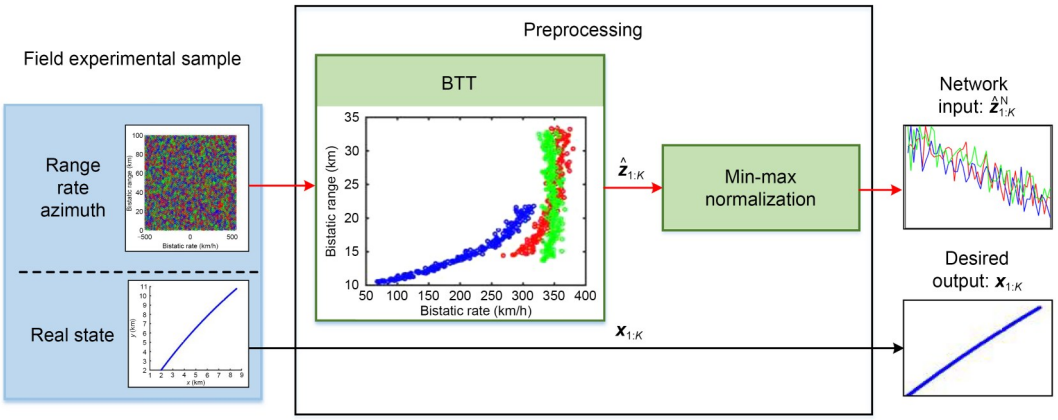

Fig. 3 Preprocessing for track data (BTT: bistatic target tracking)

After preprocessing, the normalized target measurement sequence is outputted as the network input

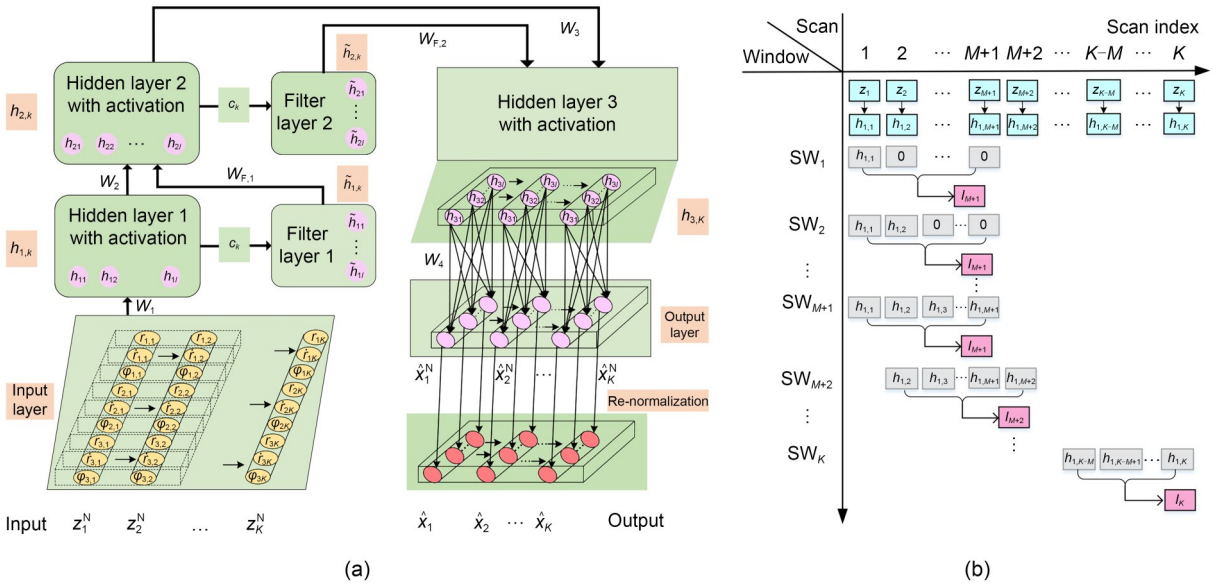

Fig. 4 Illustration of a modified deep feedforward neural network (MDFNN): (a) structure of MDFNN; (b) sliding window (SW) processing for window size *M*

In (a), the network consists of an input layer, hidden layers, filter layers, and an output layer, where the hidden layers perform a nonlinear transformation of input activation into output activation. The filter layers capture the long-term dependency in the measurement sequence. The input layer and output layer receive and output sequences, respectively. In (b), SW processing considers consecutive measurements and moves forwards by one scan each step to include the latest bistatic measurements

$$\begin{aligned}\tilde{\boldsymbol{H}}_{2,k} &= [\,\tilde{\boldsymbol{h}}_{2,1}, \tilde{\boldsymbol{h}}_{2,2}, \cdots, \tilde{\boldsymbol{h}}_{2,k}\,] \\ &= [\,\boldsymbol{h}_{2,1}, \boldsymbol{h}_{2,2}, \cdots, \boldsymbol{h}_{2,k}\,][\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_k\,], \end{aligned} \tag{25}$$

$$\begin{aligned}\boldsymbol{h}_{3,k} &= f_{\mathrm{h}}(\boldsymbol{W}_3^{\mathrm{T}}\boldsymbol{h}_{2,k} + \boldsymbol{W}_{\mathrm{F},2}^{\mathrm{T}}\tilde{\boldsymbol{h}}_{2,k} + \boldsymbol{\kappa}_3) \\ &= f_{\mathrm{h}}\left(\sum_{i=1}^{l_1}\boldsymbol{W}_3^{\mathrm{T}}(j,i)\,\boldsymbol{h}_{2,k}(i) + \sum_{i=1}^{l_1}\boldsymbol{W}_{\mathrm{F},2}^{\mathrm{T}}(j,i)\tilde{\boldsymbol{h}}_{2,k}(i)\right. \\ &\quad \left. +\boldsymbol{\kappa}_3(j)\right), j = 1, 2, \cdots, l_3, \end{aligned} \tag{26}$$

$$\begin{aligned}\hat{\boldsymbol{x}}_k &= f_{\mathrm{o}}(\boldsymbol{W}_4^{\mathrm{T}}\boldsymbol{h}_{3,k} + \boldsymbol{\kappa}_4) = f_{\mathrm{h}}\left(\sum_{i=1}^{l_2}\boldsymbol{W}_4^{\mathrm{T}}(j,i)\boldsymbol{h}_{3,k}(i) + \boldsymbol{\kappa}_4(j)\right), \\ &\quad j = 1, 2, \cdots, 6, \end{aligned} \tag{27}$$

where (j, i) denotes the element of the matrix in row j and column i, (i) represents the i^{th} element of the vector, $\boldsymbol{\kappa}_i$ $(i = 1, 2, 3, 4)$ denotes the offset vector of each layer of the tracking network, $\tilde{\boldsymbol{H}}_{i,k}$ $(i=1, 2)$ represents all outputs of the i^{th} filter layer from time 1 to k, $\hat{\boldsymbol{x}}_k^{\mathrm{N}}$ indicates the normalized regression prediction of $\boldsymbol{x}_k$ by the proposed tracking network at time k, and f_{h} and f_{o} are all elementwise activation functions. They are usually a saturating nonlinear function such as a logistic sigmoid function $s(\alpha)=1/(1+\mathrm{e}^{-\alpha})$ or a hyperbolic tangent function $\tanh(\alpha)=(\mathrm{e}^{\alpha}-\mathrm{e}^{-\alpha})/(\mathrm{e}^{\alpha}+\mathrm{e}^{-\alpha})$.

Let $\boldsymbol{W}=\{\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{W}_4\}$ denote the connection weight matrix of the input layer, hidden layer, and output layer, $\boldsymbol{W}_{\mathrm{F}}=\{\boldsymbol{W}_{\mathrm{F},1}, \boldsymbol{W}_{\mathrm{F},2}\}$ the weight matrix connecting the filter layer to the hidden layer, and $\boldsymbol{\kappa}=\{\boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \boldsymbol{\kappa}_3, \boldsymbol{\kappa}_4\}$ the offset of all layers. The target function of the tracking network can also be expressed as

$$\hat{\boldsymbol{x}}_k = \phi(\boldsymbol{z}_k;\, \boldsymbol{W}, \boldsymbol{W}_{\mathrm{F}}, \boldsymbol{c}_k, \boldsymbol{\kappa}), \tag{28}$$

where $\hat{\boldsymbol{x}}_k$ is the estimated state. It can be considered a composition of multiple nonlinear layers when unfolded in time. To obtain high-accuracy state estimation, $\hat{\boldsymbol{x}}_k$, $\boldsymbol{W}$, $\boldsymbol{W}_{\mathrm{F}}$, $\boldsymbol{c}_k$, and $\boldsymbol{\kappa}$ should be selected to minimize the error of the prediction. Thus, the cost function can be defined as

$$J = \sum_{i=1}^{K}\left\|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\right\|^2/2. \tag{29}$$

The parameters in Eq. (28) are usually estimated using the error back propagation algorithm (EBPA) based on stochastic gradient descent (Rumelhart et al., 1986). EBPA fully exploits the structure of MDFNN and backwards updates weights and thresholds from the output layer to the input layer.

3.3 Determination of the optimal tracking model

For the estimation of the target state, we use the proposed tracking network to learn from the target tracking dataset with multiple labels. Based on the MDFNN model, we mark the measurements without false alarms as input. Ten different tracking models are trained and evaluated to obtain a comparative idealist model. The whole process of determination involves four steps:

1. Training process: Generally, the training process is the decisive stage for tracking models. In this case, we use 85% of the target tracking dataset to train inherent parameters in Eq. (28). The regression relationship between the measurements and the real target state in the training dataset can be learned using the MDFNN algorithm. Thus, a trained tracking model can be employed to predict the target state.

2. Validation process: Once the training procedure of the MDFNN model is completed, cross-validation is applied to eradicate overfitting. To validate the effectiveness of the MDFNN model, 10% of the tracking dataset is used to cross-validate every trained tracking model. When the trained MDFNN model performs poorly on the validation dataset, we must further modify a series of parameters to improve the performance of the MDFNN model. Specifically, we retrain every trained tracking model and revalidate them separately based on the same retrained dataset. If MDFNNs achieve appropriate performance, the verification procedure will cease.

3. Testing process: We can acquire 10 MDFNN models as long as the training and validation procedures are completed. Accordingly, the performance of each MDFNN is further tested by using 5% of the measurement sequence dataset. As the testing dataset has never been used, this is a reliable and effective means to guarantee the estimation performance of the 10 MDFNN models.

4. Model selection: To further evaluate the adaptive capacity of the trained 10 MDFNN models, we choose some new measurement sequence datasets that have not been trained before to further test each acquired MDFNN. Then, an optimal MDFNN model is selected as the final target tracking model. Hence, this is a powerful way to compare and evaluate the performance of 10 MDFNN models. All universal procedures of training an MDFNN model are summarized in Fig. 5 (Liu H et al., 2019).

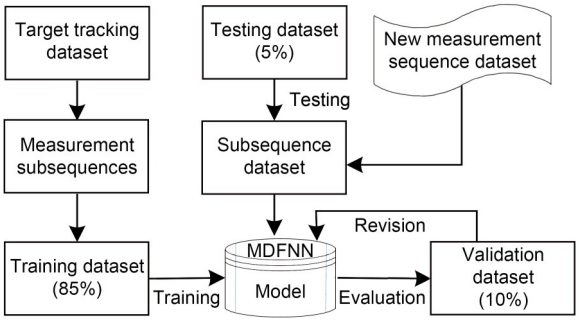

Fig. 5 Flowchart of training an MDFNN model

The steps contain feature extraction, training, testing, and evaluation in obtaining the target tracking model. When the final tracking model is recognized and confirmed, we input the target measurements into the trained MDFNN model to estimate the target state. This achieves the entire process from model-driven to data-driven.

In this study, the number of nodes of each layer N_h in the tracking network is set as in Table 2. The number of nodes is designed based on the empirical equation below:

$$N_h = N_s / \left[\lambda (N_i + N_o) \right], \tag{30}$$

where N_s denotes the sample size, N_i and N_o are the input size and output size, respectively, and λ is a constant that is usually set in the range of [1, 20] for a large sample. First, assuming that λ=5, 10, 20, we can obtain the number of nodes N_h, which is in the ranges of [396, 793] and [198, 396]. Then, the accurate number of nodes of each layer can be achieved by cross-validation.

Note that the cost function is usually nonconvex, which means that many local minima may exist.

Table 2 Number of nodes of each layer

Layer	Number of nodes
Input	9
Hidden 1	384
Filter 1	384
Hidden 2	225
Filter 2	225
Hidden 3	540
Output	6

Generally, we tend to find the quasi-optimum with lower computation complexity in the parameter space (Rumelhart et al., 1986).

4 Simulations and field experimental results

In this section, simulations are first conducted to verify the effectiveness of the proposed algorithm. Real data are then used to test the practical performance of the proposed method.

Based on the existing schemes and the proposed tracking algorithm, we design a framework for experimental comparison. As shown in Fig. 6, the framework consists of three modules. The first module is the tracking dataset building module, which aims to obtain target measurements from measurements with false alarms. The second module is a classic tracking scheme, which includes localization (spherical interpolation, SI) (Malanowski and Kulpa, 2012) with tracking (KF), EKF, and UKF based tracking methods. The third module is the machine learning module (i.e., classic regression model RNN and the proposed method MDFNN), which is used for extracting target features from the sample dataset and building a conclusive model. The MDFNN model is determined based on the high accuracy requirement, which is accurate in certain instances. The different tracking performances of the classic tracking methods, RNN, and the proposed tracking algorithm are tested and compared based on real experimental data.

For all the experiments, the root mean square error (RMSE) is employed for tracking performance evaluation. It is calculated as follows:

$$\mathrm{RMSE}_k = \sqrt{\frac{1}{M_\mathrm{C}} \sum_{i=1}^{M_\mathrm{C}} (\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k,i})^2}, \tag{31}$$

where RMSE_k is the RMSE at time k, $\boldsymbol{x}_k$ denotes the real value at time k, $\hat{\boldsymbol{x}}_{k,i}$ represents the estimated value of the i^{th} MC simulation at the same time, and M_C is the number of MC simulations. RMSE is an absolute measure of fitting for the proposed tracking algorithm. The lower the RMSE, the better the algorithm.

4.1 Ablation experiment

In this subsection, an ablation experiment is conducted to validate the effectiveness of the filter

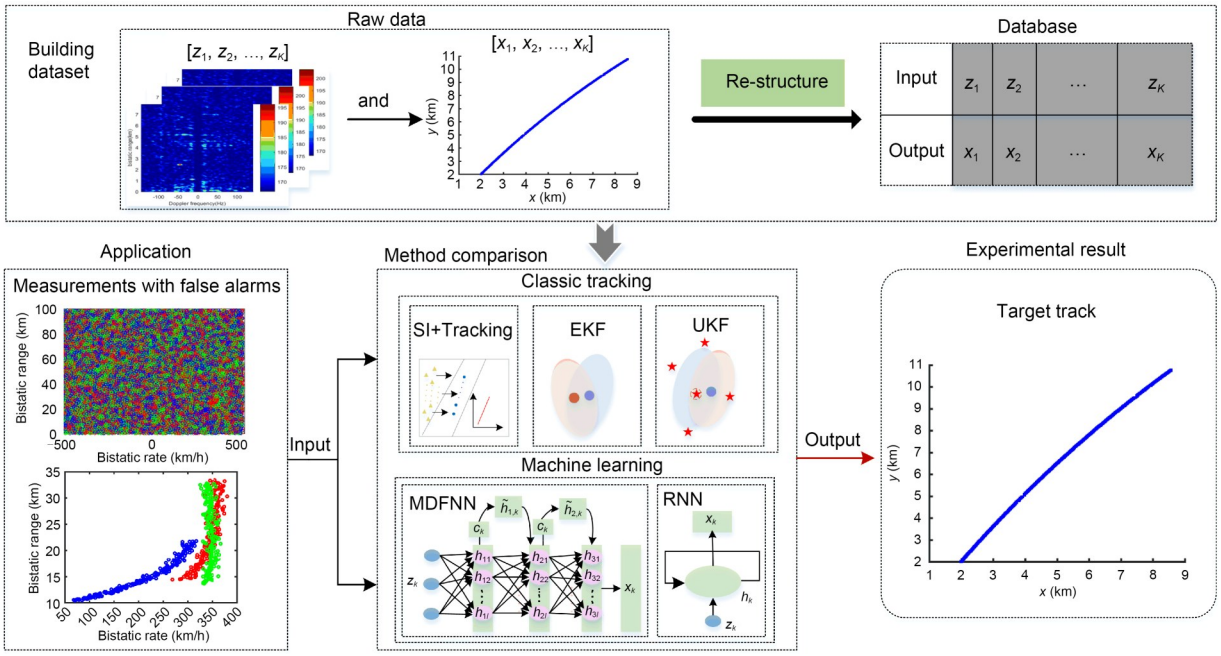

Fig. 6 Pipeline of the proposed target tracking method comparison framework, which includes three main modules, i.e., dataset building, classic tracking, and machine learning for target tracking (SI: spherical interpolation; EKF: extended Kalman filter; UKF: unscented Kalman filter; MDFNN: modified deep feedforward neural network; RNN: recurrent neural network)

layer in the tracking network. The tracking network without the filtering layer (i.e., DFNN) is trained with 20 000 epochs based on the target tracking database, which is identical to that in Section 2. The training results are shown in Fig. 7.

Fig. 7a shows the RMSEs of the tracking network based on MDFNN or DFNN in the x and y coordinates. Obviously, the RMSE of DFNN is larger than that of MDFNN. It seriously fluctuates after 48 s and is larger than 200 m, although it slightly decreases in some period of testing. As shown in Fig. 7b, the tracking network based on DFNN cannot correctly track the trajectory. As we further observe in Table 3, the RMSEs of the tracking network based on DFNN and the tracking network based on MDFNN are 545.79 m and 3.03 m, respectively. The tracking RMSEs of DFNN are so large that the passive radar has undoubtedly lost the target.

4.2 Selection of the sequence size

To evaluate the performance of the proposed approach, we have considered the case of three aerial targets, whose trajectories are shown in Fig. 8. In the simulations, a 2D space is considered, the accuracies of the bistatic range and bistatic velocity are 30 m and 2 m/s, respectively, and the azimuth accuracy is 3°. In our numerical results, the system performance is measured in terms of RMSE over 100 simulation runs. First, the performance of the proposed tracking network versus input sequence size is tested by tracking the trajectory of target 2 (Fig. 8). The tracking results for different sequence sizes are shown in Table 4. The values of RMSEs of the position slowly decline along with the increase in sequence size and fluctuate at approximately 1.28 m after ε=100. For a larger ε, the number of training parameters needs to increase, leading to a more complicated tracking network. As a result, considering a tradeoff between the acquisition of target information and the computation complexity, the input sequence size is set to 100.

Then, we examine the impact of the sliding window technique proposed in the filter layer on the performance of the tracking network described in Fig. 4. Table 5 gives the position RMSE of the tracking network over different window sizes for the abovementioned three trajectories (Fig. 8). The trends of the tracking RMSE of the three trajectories are basically the same with the change in filter order. The RMSEs of the three tracks are relatively small when M=20.

4.3 Performance comparison

For all tracking techniques, the corresponding parameters have been set to the values maximizing the performance. RNN, conventional EKF, and UKF are used for comparison.

The network structure of RNN refers to Gao et al. (2018), and the number of hidden units of the RNN network is set to 256. For the aforementioned scenario, Fig. 9 shows the RMSE versus time. The total RMSEs are given in Table 6. The results show that RNN and MDFNN perform much better than conventional tracking approaches. Among them, MDFNN outperforms RNN since it considers sliding window control on the length of the measurement sequences. Compared with

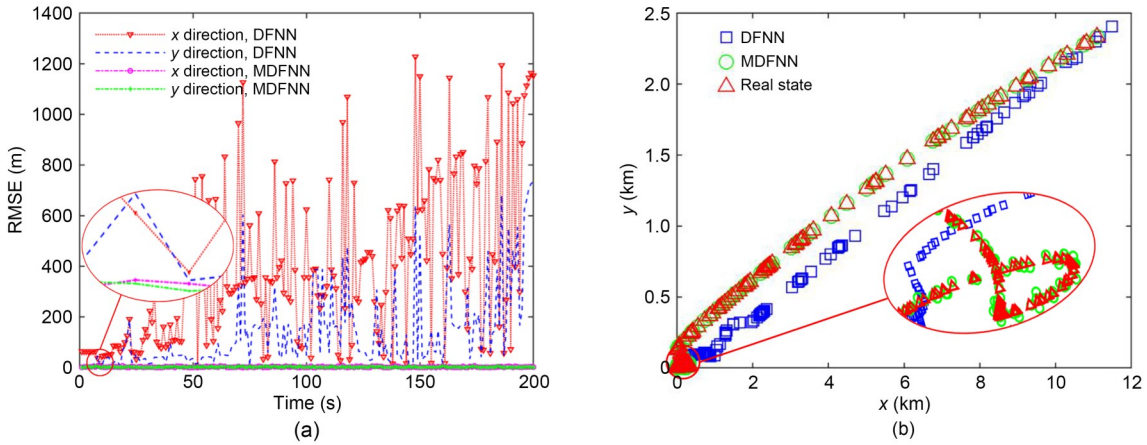

Fig. 7 Training results of the tracking network without the filter layer: (a) position RMSE of the tracking network based on DFNN or MDFNN in testing; (b) results of tracking by the tracking network based on DFNN or MDFNN

Table 3 Training results of the tracking network and the network without the filtering layer

Network	RMSE (m)
MDFNN	3.03
DFNN	545.79

Table 4 Means of the position tracking RMSE for the first trajectory with different sequence sizes

Sequence size ε	RMSE (m)	Sequence size ε	RMSE (m)
10	2.26	300	1.26
50	1.87	400	1.27
100	1.28	500	1.27
150	1.27	600	1.28
200	1.27	700	1.57

Table 5 Means of the position tracking RMSE for three trajectories with different filter orders M

M	RMSE (m)		
	Trajectory 1	Trajectory 2	Trajectory 3
10	1.45	1.07	2.54
20	1.29	0.97	2.33
30	1.50	1.12	2.18
40	1.93	1.24	2.60
50	2.65	1.45	3.32

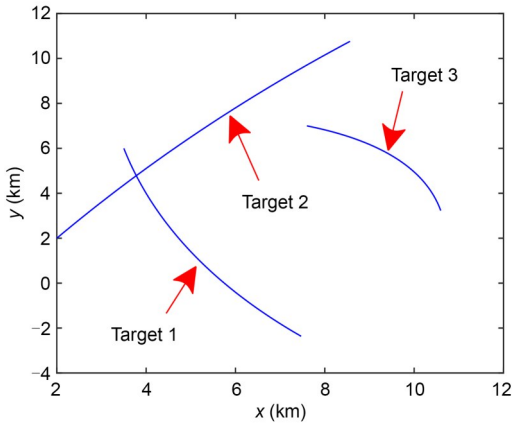

Fig. 8 Three aerial targets in Cartesian coordinates (simulation data)

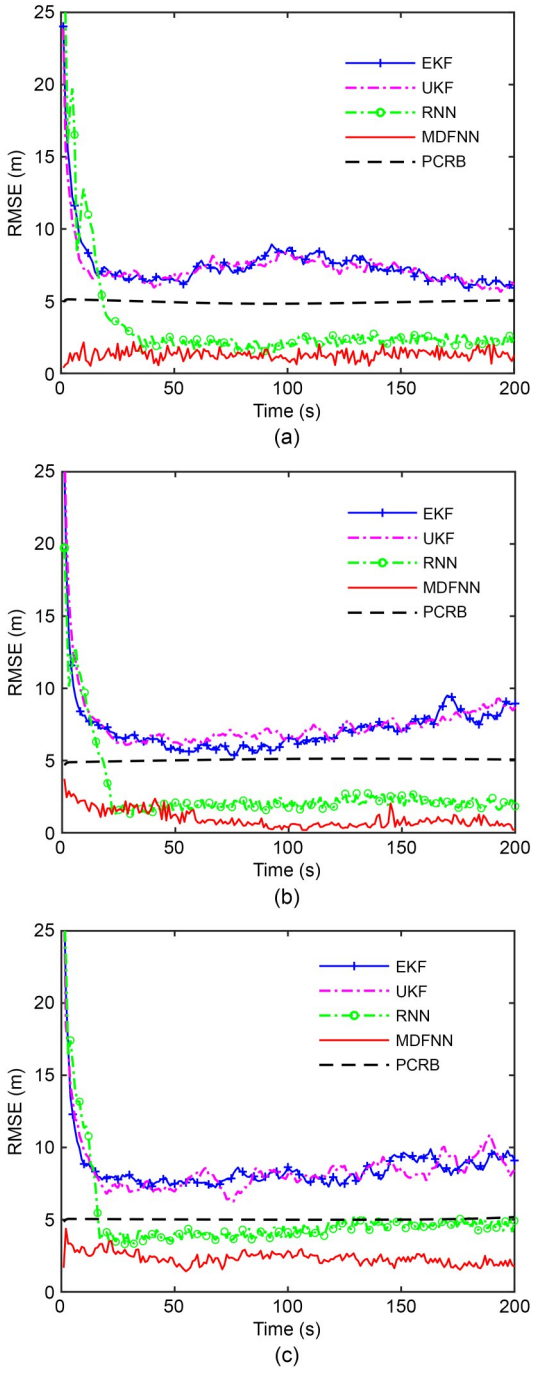

Fig. 9 Performance comparison among the proposed tracking algorithm, EKF, UKF, and RNN for three trajectories: (a) trajectory 1; (b) trajectory 2; (c) trajectory 3

the smaller RMSEs of EKF and UKF, the RMSEs of the proposed algorithm decrease by at least 82.33%, 86.51%, and 72.49% for trajectories 1, 2, and 3, respectively. The improved accuracy can be attributed to the combined effect of having a larger dataset for training and testing.

In addition, the posterior Cramer–Rao bound (PCRB) of the three target tracks is shown in Fig. 9. The method proposed by Tichavsky et al. (1998) is used to calculate PCRB. In the 1T-3R passive radar system, which consists of one transmitter and three receivers, all the RMSEs of the proposed method are

Table 6 Means of the position tracking RMSE for three trajectories with different tracking methods

Method	RMSE (m)		
	Trajectory 1	Trajectory 2	Trajectory 3
EKF	7.47	7.19	8.54
UKF	7.30	7.55	8.47
RNN	3.26	2.73	5.03
MDFNN	1.29	0.97	2.33

superior to the corresponding PCRB. This verifies the high accuracy of the proposed method. The possible reasons for this are mainly as follows: (1) Based on the analysis in Section 3.2, the process of target tracking can be considered as finding the optimal parameter of a regression model. The curve fitted by the proposed algorithm is smoother than the trajectory tracked by traditional methods. (2) PCRB is a lower bound on the MSE of an unbiased estimator. The proposed algorithm is not limited to unbiased estimators. It provides a possible processing means for satisfying the requirement of high accuracy.

According to the results achieved for this example, the performance of the proposed tracking algorithm is considerably improved compared to EKF and UKF. Instead of seeking a traditional method of approximation, the proposed tracking algorithm shows its capability to reduce the tracking error by applying its own nonlinear structure and the ability to process nonlinear systems.

4.4 Field experimental results

In this subsection, the effectiveness of the proposed algorithm is demonstrated using the experimental data acquired from an experimental passive radar system developed at Wuhan University in Hubei Province, China. As shown in Fig. 10, there is one transmitter and three receivers. The transmitter is located at the Wuhan Guishan Digital Radio and Television Tower. The three receivers are all located at Wuhan University, and the baseline distances to the transmitting station are 7.56, 8.12, and 8.28 km, separately. The target is a DJI drone, and its flight altitude is 100 m. The experimental data were acquired in July 2017. The selected duration was about 200 s. We extracted the corresponding Global Position System (GPS) information from the drone as a reference. The four tracking schemes mentioned in the framework (Fig. 6) were tested using the experimental data.

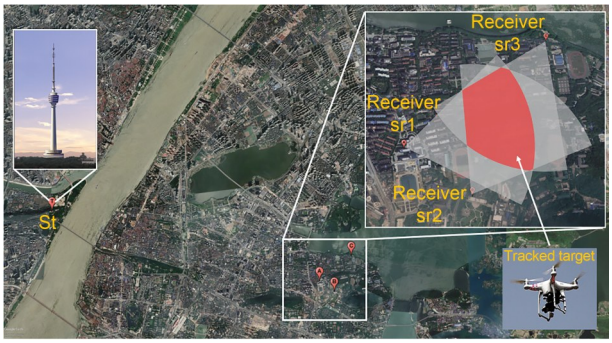

Fig. 10 Geometry of one transmitter, three receivers, and the area of interest

For multistatic passive radar detection, the influence of the target height on each receiver may offset each other. Hence, the effect of the target altitude on the accuracy is analyzed by simulating the field detection scenarios of multistatic passive radar. Ten thousand tracks are randomly simulated under the field detection area (Fig. 10) to investigate the influence of the target altitude on the accuracy. Fig. 11 shows the 2D coordinate EKF tracking results for different target altitudes. The total RMSEs are given in Table 7.

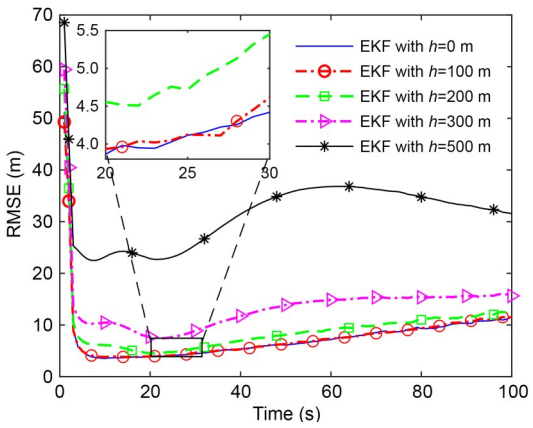

Fig. 11 Performance comparison with different target altitudes

Table 7 Means of the position tracking RMSE for 10 000 trajectories with different target altitudes

Altitude h (m)	RMSE (m)	$\lvert \text{RMSE} - \text{RMSE}_0 \rvert$ (m)
0	7.55	0
100	7.61	0.06
200	9.11	1.56
300	13.48	5.93
500	31.49	23.94

RMSE_0 denotes the tracking result with h=0 m. The position RMSE increases along with the increase in the target altitude h. Compared with RMSE_0, RMSE increases by 0.79%, 20.66%, 78.54%, and 317.09% for the four target altitudes of h=100, 200, 300, and 500 m, respectively. Hence, the effect of the target altitude on the tracking accuracy is relatively small for low-altitude detection.

Since the flight altitude of the target is 100 m in this study, the influence of the target altitude on the tracking performance is very weak according to the foregoing simulation results. The target altitude can be ignored here, and the real data can be used to verify the performance of the proposed algorithm by 2D coordinate tracking.

The processing results are plotted in Fig. 12. The target is tracked completely by the five tracking schemes. As shown in Fig. 12a, good agreement is achieved with GPS from a 2D view. The trajectory tracked by the proposed tracking algorithm is the closest to GPS compared with the four other tracking schemes. This validates the feasibility of the proposed algorithm. GPS information is used to calculate the RMSE of the trajectory. The position error is plotted in Fig. 12b. The track errors of SI+KF, EKF, and UKF seriously fluctuate in a dynamic process of tracking, and RNN tracking deviates significantly from the true state at the very start, while the errors of the proposed algorithm are relatively stable. This is because a deeper network structure can extract more target motion feature information from the measurement dataset. As we further observe in Table 8, the horizontal location accuracies of SI+KF, EKF, and UKF are 12.25, 12.45, and 12.21 m, respectively, while those of RNN and MDFNN are 2.16 and 1.72 m, respectively. The proposed algorithm can track the targets with high accuracy, while classical methods present large track errors. The proposed algorithm can give more precise position information to counter-UAV systems.

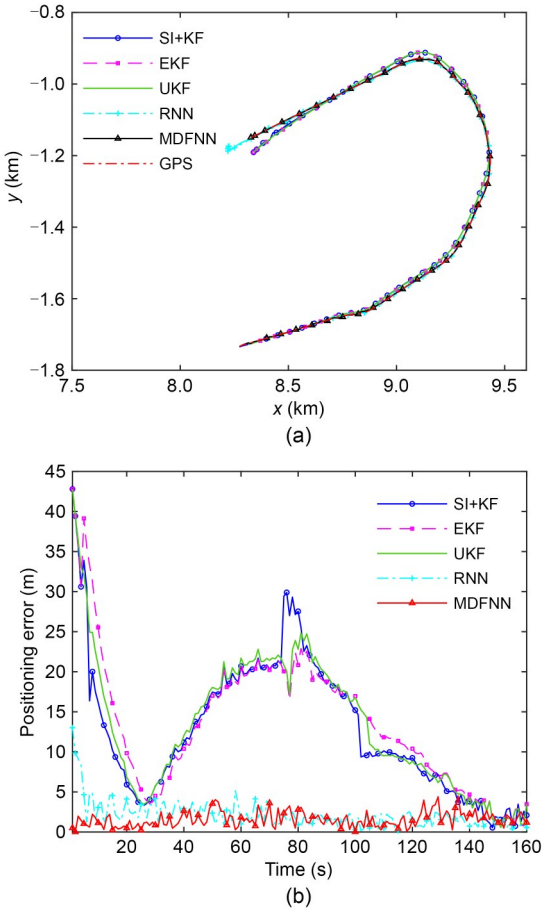

Fig. 12 Results of tracking in Cartesian coordinates (real data): (a) tracking results; (b) position deviation of the tracking track

Table 8 Means of the position RMSE of the track in the field experiment

Method	SI+KF	EKF	UKF	RNN	MDFNN
RMSE (m)	12.25	12.45	12.21	2.16	1.72

4.5 Computational cost

This subsection is devoted to analyzing the computational cost of the proposed algorithm and traditional approaches EKF (Saha et al., 2014) and UKF (Ning et al., 2017). The number of floating-point operations (FLOPs) is adopted as a measure of the model complexity with MDFNN and the computational complexity with traditional algorithms. Denote the dimension of the measurement and the dimension of the state by d_{m} and d_{s}, respectively. The computational complexities of EKF and UKF are $O(d_{\mathrm{s}}^3)$ and $O(d_{\mathrm{m}}^3)$, respectively.

Table 9 summarizes the model complexity of the proposed algorithm. The main procedures include the hidden layer $O(\varepsilon(d_{\mathrm{m}}+l_2)l_1+\varepsilon(d_{\mathrm{s}}+l_2)l_3)$ and filter layer $O((l_1+l_2)M^2)$. Thus, the total model complexity is

$$\mathrm{FLOPs}_{\mathrm{model}}=\varepsilon l_3 d_{\mathrm{s}}+\varepsilon l_1 d_{\mathrm{m}}+(l_1+l_3)\varepsilon+(l_1+l_2)M^2. \tag{32}$$

Table 9 Model complexity of the proposed algorithm in terms of matrix multiplication

Processing step	Complexity
1st hidden layer $\boldsymbol{W}_1^{\mathrm{T}}\boldsymbol{z}_k$	$S_1=\varepsilon d_{\mathrm{m}} l_1$
2nd hidden layer $\boldsymbol{W}_2^{\mathrm{T}}\boldsymbol{h}_{1,k}$	$S_2=\varepsilon l_2 l_1$
3rd hidden layer $\boldsymbol{W}_3^{\mathrm{T}}\boldsymbol{h}_{2,k}$	$S_3=\varepsilon l_3 l_2$
1st filter layer $\tilde{\boldsymbol{H}}_{1,k}$	$S_4=l_1 M^2$
2nd filter layer $\tilde{\boldsymbol{H}}_{2,k}$	$S_5=l_2 M^2$
Output layer $\hat{\boldsymbol{x}}_k$	$S_6=\varepsilon l_3 d_{\mathrm{s}}$
Total model complexity: $\mathrm{FLOPs}_{\mathrm{model}}=S_1+S_2+S_3+S_4+S_5+S_6$	

ε: sequence size

The model complexity is on the order of $O(d_{\mathrm{s}})$ and $O(d_{\mathrm{m}})$. It has a first-order relationship with both the state dimension and the measurement dimension. The computational cost of the proposed algorithm refers to the inference speed of the trained model. However, the inference speed is related not only to the model complexity but also to many other factors, such as the memory access cost, hardware characteristics, software implementation, and system environment (Ma et al., 2018). The model complexity cannot accurately measure the estimation speed of the proposed algorithm. Hence, experimentation is the most accurate way to evaluate the computational performance here.

We test the computational time of the proposed algorithm, EKF, and UKF by means of 100 experimental runs. For a fair comparison, all the algorithms are tested by the same Intel Core i7-4790 CPU at 4.0 GHz and 32.0 GB RAM. The test results show that the proposed algorithm, EKF, and UKF consume 31.0, 171.1, and 438.1 ms, respectively. Hence, the proposed algorithm is suitable for real-time implementation.

5 Conclusions

In this paper, we develop a high-accuracy target tracking algorithm for multistatic passive radar by exploiting the learning capabilities and easy structure of DFNN. First, a novel MDFNN tracking model based on a new filter layer is advocated for tracking recursion. We have shown that the tracking network based on MDFNN models the sequence dependency well by the ablation experiment, thereby having better reliability and tracking performance. Second, a reasonable measurement sequence size is selected to trade off the acquisition of target information and the computational complexity. Simulations and field data demonstrate that the proposed algorithm presents higher tracking accuracy compared to state-of-the-art methods. In future work, we will consider radar tracking problems with different measurement noise levels. In these cases, the proposed algorithm may need to be fine-tuned to achieve proper tracking results. In addition, we will try to use mixed machine-learning techniques to replace traditional target association modules.

Contributors

Baoxiong XU and Jianxin YI designed the research. Baoxiong XU and Feng CHENG processed the data. Baoxiong XU and Jianxin YI drafted the paper. Ziping GONG and Xianrong WAN helped organize the paper. Baoxiong XU and Jianxin YI revised and finalized the paper.

Compliance with ethics guidelines

Baoxiong XU, Jianxin YI, Feng CHENG, Ziping GONG, and Xianrong WAN declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Amoozegar F, Sundareshan MK, 1994. Target tracking by neural network maneuver modeling. Proc IEEE Int Conf on Neural Networks, p.3932-3937. https://doi.org/10.1109/ICNN.1994.374840

Bengio Y, Simard P, Frasconi P, 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans Neur Netw*, 5(2):157-166. https://doi.org/10.1109/72.279181

Bengtsson T, Bickel P, Li B, 2008. Curse-of-dimensionality revisited: collapse of the particle filter in very large scale systems. In: Nolan D, Speed T (Eds.), Probability and Statistics: Essays in Honor of David A. Freedman, Vol. 2. Institute of Mathematical Statistics, Beachwood, USA, p.316-334. https://doi.org/10.1214/193940307000000518

Chin L, 1994. Application of neural networks in target tracking data fusion. *IEEE Trans Aerosp Electron Syst*, 30(1): 281-287. https://doi.org/10.1109/7.250437

Choi S, Crouse DF, Willett P, et al., 2014. Approaches to Cartesian data association passive radar tracking in a DAB/DVB network. *IEEE Trans Aerosp Electron Syst*, 50(1):649-663. https://doi.org/10.1109/TAES.2013.120431

Ding J, Chen B, Liu HW, et al., 2016. Convolutional neural network with data augmentation for SAR target recognition. *IEEE Geosc Remote Sens Lett*, 13(3):364-368. https://doi.org/10.1109/LGRS.2015.2513754

Doucet A, de Freitas N, Gordon N, 2001. An introduction to sequential Monte Carlo methods. In: Doucet A, Freitas N, Gordon N (Eds.), Sequential Monte Carlo Methods in Practice. Springer-Verlag, New York, USA, p.3-14. https://doi.org/10.1007/978-1-4757-3437-9_1

Fatseas K, Bekooij MJG, 2019. Neural network based multiple object tracking for automotive FMCW radar. Int Conf on Radar, p.1-5. https://doi.org/10.1109/RADAR41533.2019.171248

Gao C, Liu HW, Zhou SH, et al., 2018. Maneuvering target tracking with recurrent neural networks for radar application. Int Conf on Radar, p.1-5. https://doi.org/10.1109/RADAR.2018.8557284

Goodfellow I, Bengio Y, Courville A, et al., 2016. Deep learning architectures. In: Deep Learning, Vol. 1. MIT Press, Cambridge, USA, p.16-21.

Gordon NJ, Salmond DJ, Smith AFM, 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc F-Radar Signal Process*, 140(2):107-113. https://doi.org/10.1049/ip-f-2.1993.0015

Griffiths HD, Baker CJ, 2005. Passive coherent location radar systems. Part 1: performance prediction. *IEEE Proc-Radar Sonar Navig*, 152(3):153-159. https://doi.org/10.1049/ip-rsn:20045082

Griffiths HD, Long NRW, 1986. Television-based bistatic radar. *IEE Proc F-Commun Radar Signal Process*, 133(7):649-657. https://doi.org/10.1049/ip-f-1.1986.0104

Gu JC, Wang HC, Ding GR, et al., 2020. UAV-enabled mobile radiation source tracking with deep reinforcement learning. Int Conf on Wireless Communications and Signal Processing, p.672-678. https://doi.org/10.1109/WCSP49889.2020.9299862

Higuchi T, 1997. Monte Carlo filter using the genetic algorithm operators. *J Stat Comput Simul*, 59(1):1-23. https://doi.org/10.1080/00949659708811843

Hornik K, 1991. Approximation capabilities of multilayer feedforward networks. *Neur Netw*, 4(2):251-257. https://doi.org/10.1016/0893-6080(91)90009-T

Hornik K, Stinchcombe M, White H, 1989. Multilayer feedforward networks are universal approximators. *Neur Netw*, 2(5):359-366. https://doi.org/10.1016/0893-6080(89)90020-8

Jiang L, Singh SS, Yıldırım S, 2015. Bayesian tracking and parameter learning for non-linear multiple target tracking models. *IEEE Trans Signal Process*, 63(21):5733-5745. https://doi.org/10.1109/TSP.2015.2454474

Kuschel H, 2013. Approaching 80 years of passive radar. Int Conf on Radar, p.213-217. https://doi.org/10.1109/RADAR.2013.6651987

Li XR, Jilkov VP, 2003. Survey of maneuvering target tracking. Part I: dynamic models. *IEEE Trans Aerosp Electron Syst*, 39(4):1333-1364. https://doi.org/10.1109/taes.2003.1261132

Liu H, Liu Z, Liu S, et al., 2019. A nonlinear regression application via machine learning techniques for geomagnetic data reconstruction processing. *IEEE Trans Geosci Remote Sens*, 57(1):128-140. https://doi.org/10.1109/TGRS.2018.2852632

Liu JX, Wang ZL, Xu M, 2020. DeepMTT: a deep learning maneuvering target-tracking algorithm based on bidirectional LSTM network. *Inform Fus*, 53:289-304. https://doi.org/10.1016/j.inffus.2019.06.012

Ma NN, Zhang XY, Zheng HT, et al., 2018. ShuffleNet V2: practical guidelines for efficient CNN architecture design. Proc 15th European Conf on Computer Vision, p.122-138. https://doi.org/10.1007/978-3-030-01264-9_8

Malanowski M, Kulpa K, 2012. Two methods for target localization in multistatic passive radar. *IEEE Trans Aerosp Electron Syst*, 48(1):572-580. https://doi.org/10.1109/TAES.2012.6129656

Mazuelas S, Shen Y, Win MZ, 2013. Belief condensation filtering. *IEEE Trans Signal Process*, 61(18):4403-4415. https://doi.org/10.1109/TSP.2013.2261991

Ning XL, Wang F, Fang JC, 2017. An implicit UKF for satellite stellar refraction navigation system. *IEEE Trans Aerosp Electron Syst*, 53(3):1489-1503. https://doi.org/10.1109/TAES.2017.2671684

Oong TH, Isa NAM, 2011. Adaptive evolutionary artificial neural networks for pattern classification. *IEEE Trans Neur Netw*, 22(11):1823-1836. https://doi.org/10.1109/TNN.2011.2169426

Palmer JE, Harms HA, Searle SJ, et al., 2013. DVB-T passive radar signal processing. *IEEE Trans Signal Process*, 61(8):2116-2126. https://doi.org/10.1109/TSP.2012.2236324

Radmard M, Karbasi SM, Nayebi MM, 2013. Data fusion in MIMO DVB-T-based passive coherent location. *IEEE Trans Aerosp Electron Syst*, 49(3):1725-1737. https://doi.org/10.1109/TAES.2013.6558015

Rassalna P, Mishra T, 2020. Target detection, tracking and threat evaluation in multi sensor system using machine learning. 3rd Int Conf on Intelligent Sustainable Systems, p.837-842. https://doi.org/10.1109/ICISS49785.2020.9315910

Rumelhart DE, Hinton GE, Williams RJ, 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533-536. https://doi.org/10.1038/323533a0

Saha M, Ghosh R, Goswami B, 2014. Robustness and sensitivity metrics for tuning the extended Kalman filter. *IEEE Trans Instrum Meas*, 63(4):964-971. https://doi.org/10.1109/TIM.2013.2283151

Schön T, Gustafsson F, Nordlund PJ, 2005. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Trans Signal Process*, 53(7):2279-2289. https://doi.org/10.1109/TSP.2005.849151

Singer H, 2008. Generalized Gauss–Hermite filtering. *AStA Adv Stat Anal*, 92(2):179-195.

https://doi.org/10.1007/s10182-008-0068-z

Singer RA, 1970. Estimating optimal tracking filter performance for manned maneuvering targets. *IEEE Trans Aerosp Electron Syst*, AES-6(4):473-483. https://doi.org/10.1109/TAES.1970.310128

Smidl V, Quinn A, 2008. Variational Bayesian filtering. *IEEE Trans Signal Process*, 56(10):5020-5030. https://doi.org/10.1109/TSP.2008.928969

Tichavsky P, Muravchik CH, Nehorai A, 1998. Posterior Cramer-Rao bounds for discrete-time nonlinear filtering. *IEEE Trans Signal Process*, 46(5):1386-1396. https://doi.org/10.1109/78.668800

Wang XZ, Musicki D, Ellem R, et al., 2009. Efficient and enhanced multi-target tracking with Doppler measurements. *IEEE Trans Aerosp Electron Syst*, 45(4):1400-1417. https://doi.org/10.1109/TAES.2009.5310307

Yi JX, Wan XR, Cheng F, et al., 2015. Deghosting for target tracking in single frequency network based passive radar. *IEEE Trans Aerosp Electron Syst*, 51(4):2655-2668. https://doi.org/10.1109/TAES.2015.130424

Yin S, Zhu XP, 2015. Intelligent particle filter and its application to fault detection of nonlinear system. *IEEE Trans Ind Electron*, 62(6):3852-3861. https://doi.org/10.1109/TIE.2015.2399396