Hindawi
International Journal of Aerospace Engineering
Volume 2024, Article ID 9981332, 9 pages
https://doi.org/10.1155/2024/9981332

Research Article

A Strong Maneuvering Target-Tracking Filtering Based on Intelligent Algorithm

Jing Li,[1] **Xinru Liang**,[2] **Shengzhi Yuan**,[1] **Haiyan Li**,[1] **and Changsheng Gao**[2]

[1]*Naval University of Engineering, Wuhan, China*
[2]*Harbin Institute of Technology, Harbin, China*

Correspondence should be addressed to Shengzhi Yuan; yuanshengzhi_hy@sina.com

Received 18 April 2023; Revised 19 November 2023; Accepted 6 December 2023; Published 11 January 2024

Academic Editor: Francisco Ronay Lopez Estrada

In this paper, a variable-structure multimodel (VSMM) filtering algorithm based on the long short-term memory (LSTM) regression-deep Q network (L-DQN) is proposed to accurately track strong maneuvering targets. The algorithm can map the selection of the model set to the selection of the action label and realize the purpose of a deep reinforcement-learning agent to replace the model switching in the traditional VSMM algorithm by reasonably designing a reward function, state space, and network structure. At the same time, the algorithm introduces a LSTM algorithm, which can compensate the error of tracking results based on model history information. The simulation results show that compared with the traditional VSMM algorithm, the proposed algorithm can quickly capture the maneuvering of the target, the response time is short, the calculation accuracy is significantly improved, and the range of adaptation is wider. Precise tracking of maneuvering targets was achieved.

1. Introduction

Strong maneuvering target tracking (MTT) is an important research direction in the field of state estimation. At present, the filtering algorithms for maneuvering target tracking are mainly divided into two categories: the improved single-model algorithm dominated by the Kalman filter and the multimodel-tracking filtering algorithm represented by an interactive multimodel. Because of its strong model-covering ability, the multimodel filtering algorithm has a prominent phenotype in the field of strong maneuvering target tracking. However, there are also many shortcomings in the multimodel algorithm. For example, the interactive multimodel algorithm usually adopts a fixed model set, which is quite different from the changeable movement space in the actual motion.

To solve this problem, Li et al. proposed the concept of a variable-structure model filtering algorithm [1–3], which expanded the original model set through one or more adaptive models closely following the real model set, and used the directed graph to represent the transition between models to realize the variability of the model set. In reference [3], the peak error, steady-state error, and response time of the filtering algorithm under different model sets are studied via a deterministic scheme (DS). In this paper, it is proposed that too many models will provide less bias stability error, but the smaller model sets will also lose some jump accuracy. At the same time, affected by the performance of its own algorithm, the performance improvement of excessive model sets and complex topological relationships is not proportional to the complexity of the solution, and the response of the algorithm is slow.

Wang et al. [4] also proposed that expected mode augmentation (EMA) $\{13 + 1\}$ could improve the accuracy in a few cases compared with the EMA $\{9 + 1\}$ algorithm, but it sacrificed a large computational fast response capability compared with the EMA $\{9 + 1\}$ algorithm, which was not proportional to the accuracy improvement of the algorithm. Wang et al. [5] proposed too few models; for the strong maneuvering target, the coverage ability of the model set will be poor; and only when the target movement just falls within

the model set will the tracking effect be better. However, in the actual movement of the target, the change of the model is unknown and difficult to cover. At the same time, the division of the model depends on the prior division of the model set and the fixed topology.

Intellectualization of filtering is the hot spot direction of current complex filtering in the tracking problem. The cyclic neural network was first proposed by Lipton et al. [6]. It has outstanding performance in temporal data processing such as speech recognition, text generation, and machine translation.

Wei et al. [7] solved the problem of cooperative target hunting in the underwater environment by using the deep Q-learning (DQN) algorithm of neural networks. Fang et al. [8] adopted adaptive time slot (TS) and power allocation schemes to switch between different operating modes of machine communication devices, achieving optimization between peak age of information (AoI) and power consumption in energy harvesting- (EH-) assisted large-scale multiaccess networks. Zhang et al. [9] and Wang et al. [10] have achieved innovative applications of adaptive technology in nonorthogonal multiple access, ad hoc networks, and other fields. At the same time, because of the advantages of a recurrent neural network in processing a sequential sequence, it can be combined with the time series characteristics of filtering. Therefore, the introduction of the recurrent neural network to improve the filtering algorithm is a popular research field in the intelligent filtering algorithm. The paper proposes to combine the LSTM network with a hypersonic vehicle-tracking filter algorithm [11]. Kim et al. [12] developed a human posture estimation algorithm combining the Kalman filtering algorithm and recurrent neural network, which improved the accuracy of tracking results. Reinforcement learning is a process in which agents interact with the environment continuously under the driving of reward function to get the maximum cumulative reward and adopt the optimal strategy of learning. DQN has been widely used in the field of decision control because of its advantages [13, 14].

Compared with the traditional VSMM algorithm, the improvement of the R-DQN-based improved VSMM algorithm proposed in this article is as follows:

(1) Using the DQN algorithm to handle the selection and decision-making problems of the model set

(2) Use the additional LSTM algorithm for compensation. Compared with traditional algorithms, this algorithm adds three online-trained neural networks, so the computational complexity of the algorithm proposed in this article is significantly higher than that of traditional algorithms

The remainder of this article is structured as follows. The Variable-Structure Multiple-Model (VSMM) filtering and the Markov Decision Process (MDP) are detailed in Section 2. In Section 3, the L-DQN algorithm is introduced into the process of model selecting in VSMM, and an LSTM network is used to modify the error, followed by our experimental results and result analysis in Section 4. In the end, conclusions are reached in Section 5.

2. Problem Description

2.1. Variable-Structure Multiple-Model Filtering. The core of the VSMM method is the addition and deletion of the model set. When and how to make changes to the model set are included in the model set decision of the VSMM algorithm. The main steps of the algorithm are input interaction, filtering algorithm, model probability update, interactive output, and model set decision.

2.1.1. Step 1: Input Interaction [15].

$$\mu_{k-1}^{i,j} = \frac{p_{ij} \cdot \mu_{k-1}^{i}}{\sum_i p_{ij} \cdot \mu_{k-1}^{i}}, \tag{1}$$

where p_{ij} is the transition probability from model i to j, μ_{k-1}^{i} is the model probability of model i at $k-1$ time, and $\mu_{k-1}^{i,j}$ is the mixing probability of the model at $k-1$ time.

At the same time, the upper corner mark represents the model label, and the lower corner mark represents the time. The representations in the following filtering algorithms are the same and will not be repeated.

2.1.2. Step 2: Filtering Algorithm. The one-step prediction of state vector is

$$\hat{X}_{k/k-1}^{j} = F_{k-1}^{j} \cdot \hat{X}_{k-1}^{j}, \tag{2}$$

where F_{k-1}^{j} is the driving matrix.

The one-step prediction of error covariance is

$$P_{k/k-1}^{j} = F_{k-1}^{j} \cdot P_{k-1}^{j} \cdot {F_{k-1}^{j}}^{T} + Q_{k-1}^{j}. \tag{3}$$

Q_{k-1}^{j} represents the process noise covariance matrix.

The filter is

$$\hat{X}_{k}^{j} = \hat{X}_{k/k-1}^{j} + K_{k}^{j} \cdot r_{k}^{j}, \tag{4}$$

$$r_{k}^{j} = Z_{k}^{j} - \hat{Z}_{k/k-1}^{j}, \tag{5}$$

where K_{k}^{j} represents the Kalman filter gain matrix, r_{k}^{j} represents the filter residual, Z_{k}^{j} is the measurement vector, and $\hat{Z}_{k/k-1}^{j}$ is the estimate of the measurement vector.

The filtering covariance is

$$P_{k}^{j} = P_{k/k-1}^{j} - K_{k}^{j} g S_{k}^{j} g \left(K_{k}^{j}\right)^{T}. \tag{6}$$

The estimation of the observation equation is as follows: Kalman gain K_{k}^{j} is as follows:

$$\begin{aligned} S_{k}^{j} &= H_{k}^{j} P_{k}^{j} \left(H_{k}^{j}\right)^{T} + R_{k}^{j}, \\ K_{k}^{j} &= P_{k/k-1}^{j} \cdot \left(H_{k}^{j}\right)^{T} \cdot \left(S_{k}^{j}\right)^{-1}. \end{aligned} \tag{7}$$

S_k^j represents the filter residual covariance, and R_k^j represents the measurement noise covariance.

2.1.3. Step 3: Model Probability Update.

$$\mu_k^j = \frac{\Lambda_k^j \mu_{k-1}^{i,j}}{\sum_{i=1}^{r} \Lambda_k^i \mu_{k-1}^{i,j}}. \tag{8}$$

$\Lambda_k^j = N(r_k^j, 0, S_k^j)$ represents the likelihood function, which defines a Gaussian distribution (also known as normal distribution) with a variable r_k^j, the mean value as 0, and the variance as S_k^j.

$$\Lambda_k^j = \prod_{i=1}^{2} \left[\frac{1}{\sigma\sqrt{2\pi}} e^{-\left((x_i-\mu)^2/2\sigma^2\right)} \right], \quad \mu = 0, \sigma^2 = S_k^j. \tag{9}$$

2.1.4. Step 4: Interacting Output.

$$\begin{aligned} \widehat{X}_k &= \sum_j \widehat{X}_k^j \cdot \mu_k^j, \\ P_k &= \sum_j P_k^j + \left\{ \left[\widehat{X}_k^j - \widehat{X}_k \right] \cdot \left[\widehat{X}_k^j - \widehat{X}_k \right]^T \right\} \mu_k^j, \end{aligned} \tag{10}$$

where $\widehat{X}_k$ represents the state vector and P_k represents the error covariance.

2.1.5. Step 5: Model Set Decision. At present, there are mainly two ways of model decision adaptation: the likely model set (LMS) [1, 2] method and the expected mode augmentation (EMA) [3] method. The EMA method extends the expected model set to the existing model set, while the LMS method divides the model according to the importance of model probability, so as to approximate the accuracy of the model with the least model as possible. The LMS method takes the directed graph algorithm as the switching algorithm of the VSMM. It stipulates that only the adjacent models can be converted to each other, but the nonadjacent models cannot be converted to each other. The model set participating in the filtering calculation is selected according to the model probability to delete and add the model. The EMA algorithm relies on a fixed topology and selects the model based on the fixed topology. It is suitable for models with additivity and continuous model space. In each cycle, the existing model is weighted to obtain the expectation, and then, the dimension of the obtained expectation model is extended to the existing motion model.

2.2. Markov Decision Process. Through the expression formula given above, the filtering process itself has a Markov property. It is a Markov process, which can be represented by tuple $\langle S, P \rangle$, where S represents the finite state set and P represents the state transition matrix [16, 17].

The Markov Reward Process (MRP) adds reward R and attenuation coefficient γ (used to calculate the cumulative reward) based on the Markov process. The Markov Decision Process (MDP) adds the decision process based on the Markov Reward Process. Compared with MRP, MDP adds action set A, which is represented by tuple $\langle S, A, P, R, \gamma \rangle$. The mathematical expression is $P_{SS'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$. The reinforcement-learning process based on MDP is the theoretical basis of solving reinforcement-learning problems and the bottom mathematical model of reinforcement learning. The Markov nature of filtering and reinforcement-learning algorithm provides the feasibility for the combination of the two.

The reinforcement-learning decision problem can be understood as a nonlinear mapping, which is a mapping from the state to the action according to the strategy, $S \longrightarrow^{\pi} A$, where S is the state information of the environment, and A is the action instruction according to the strategy state, where strategy π represents the probability of selecting an action in a certain state in the process, and represents the set of execution probabilities of an action. The strategy calculation equation is as follows:

$$\pi(a|s) = p[A_t = a | S_t = s]. \tag{11}$$

In order to evaluate the return value of each strategy π, the cumulative reward function is defined, which is called the value function, and the calculation equation is defined as

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n R_{t+n+1} + \cdots = \sum_{k=0}^{T} \gamma^k R_{t+k+1}, \tag{12}$$

where R represents the reward function and γ represents the discount factor of reward.

The value function based on policy is divided into the state value function: it represents the long-term return in this state $v_\pi(s) = E_\pi[G_t | S_t = s]$, and the action value function: it represents the long-term return of taking the action in this state $q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a]$.

3. Variable-Structure Multimodel Filtering Based on L-DQN Algorithm

The structure diagram of the improved L-DQN algorithm is shown in Figure 1. The algorithm uses the DQN algorithm to make judgments and decisions on model set switching and uses LSTM to directly compensate the error of the filtering and positioning results of the target.

In the figure, white represents the original module of the VSMM algorithm, green represents the newly added submodule of L-DQN, and blue represents the improved module of L-DQN.

3.1. Markov Decision Process. The DQN algorithm combines the deep-learning algorithm with the reinforcement-learning algorithm and uses the deep neural network to replace the action value function. The algorithm uses the experience replay and the target network to ensure the nonlinear learning ability of the algorithm.

3.1.1. Design of Action Space. Different model sets are set as different labels, and different labels correspond to different

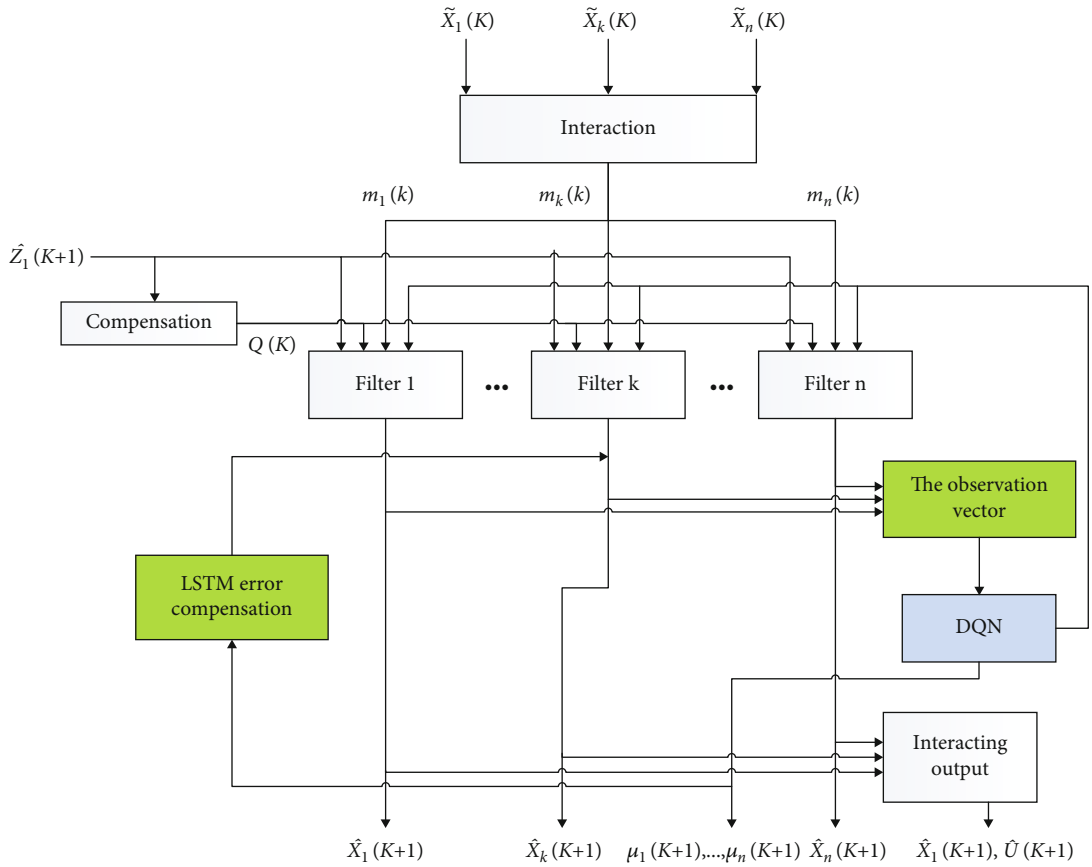

Figure 1: L-DQN algorithm.

actions; then, the selection of models could be converted into the selection process of agent actions. Since the size of the action space determines the coverage ability of the model set, during offline training, we put as many labels of the model set into the action space as possible and use the strategy value function to output the most possible action and its probability. The actions selected by the agent can be divided into two categories: one is the action of model selection and the other is the termination action, which is used to terminate exploration.

The exploration of action adopts a strategy, which can balance the relationship between exploration and utilization, and make a compromise between exploration and utilization with a certain probability [18].

$$\pi(a|s) \longleftarrow \begin{cases} 1-\varepsilon+\dfrac{\varepsilon}{|A(s)|} & a = \arg\ \max_a Q(s,a), \\ \dfrac{\varepsilon}{|A(s)|} & a \neq \arg\ \max_a Q(s,a), \end{cases} \tag{13}$$

where ε is the exploration factor, usually taken as $[0, 1]$; $|A(s)|$ is the set of optional actions of the agent; s is the current state of the agent; and $\pi(a|s)$ is the strategy adopted by the algorithm.

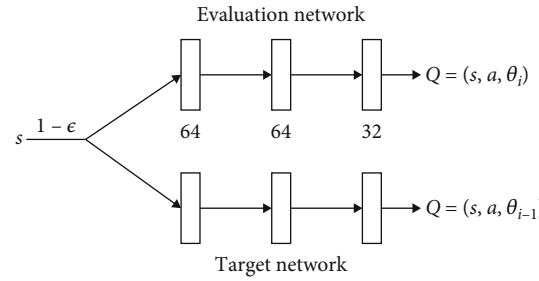

Figure 2: Network structure.

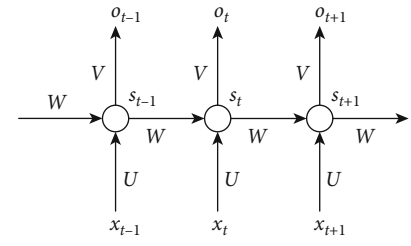

Figure 3: Network structure of RNN.

3.1.2. Design of State Space. In the process of target tracking, the model probability of the target is the basis for dividing the importance of the model. It can be seen from

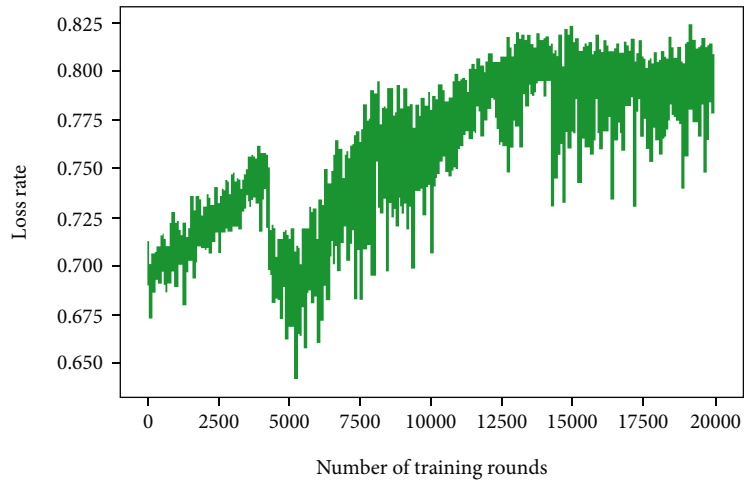

Figure 4: Reward function.

Equations (8) and (9) that the calculation of the model probability is mainly given by the filtered residual and the covariance of the residual. Therefore, we take the residual and covariance of the model as the observation of model decision-making, so that the agent can learn to make model decision through the residual and covariance of the model at the current time. At the same time, we take the output action probability value as the new probability of the model to obtain the accurate model probability.

3.1.3. Design of Reward Function. The reward function determines the agent's performance in the environment, which is the specific numerical value of the target task, and determines whether the optimal strategy can be learnt by the agent.

Moreover, the reward function influences the exploration-exploitation trade-off in reinforcement learning. A well-designed reward function can encourage the LSTM to explore different actions and states to discover optimal policies.

$$R_a(s) = \min\left(1, \max\left(-1, 1 - \frac{\left|f_a^i - f_t^i\right|}{\left|f_a^{\max} - f_a^{\min}\right|}\right)\right). \quad (14)$$

f_a^i is the mapping function of the real model set, and f_t^i is the mapping function of the agent selection action model set.

3.1.4. Network Structure. The strategy network of the DQN network is composed of two parallel networks with the same network structure. One is used to generate the current Q value, and the other is used to generate the target Q value. The two networks only have different network parameters. When calculating the target network parameters, they are the network parameters before several time steps of the current network. The network structure is shown in Figure 2.

One DQN network strategy network is made up of two identical parallel network structures: one is used to generate the current Q value and the other is used to generate a target Q value. Two networks have different network parameters. Calculating the target network parameters is also part of the calculation of the current network parameters of a single time step before.

Table 1: The scheme of maneuvering flight.

Flight time (s)	DS1	DS2	DS3
0-20	0	0	10
20-50	9	0	10
50-70	3	3	10
70-90	0	3	10
90-120	-3	3	10
120-140	-6	-6	-10
140-180	-3	-6	-10
180-200	0	-6	-10

3.2. Error Modification Network

3.2.1. Introduction to LSTM Network. RNN is a neural network used to process sequence data, which can fuse historical information with new input information. The schematic diagram of the network structure is shown in Figure 3.

$$O_t = g(V \cdot S_t) S_t = f(U \cdot X_t + W \cdot S_{t-1}). \quad (15)$$

After the RNN network receives the input x_t at time t, the value of the hidden layer is S_t and the output value is O_t. The key point is that the value of S_t depends not only on x_t but also on $S_{t\text{-}1}$.

As a result of gradient descent and gradient explosion in the process of long sequence training, it is difficult for RNN to learn the useful information far away from the processing information. Compared with traditional RNN, LSTM performs better in longer sequences.

The LSTM algorithm is used to design the error modification network as the regression network of tracking and positioning results in compensating the modeling nonlinear error and modeling uncertainty.

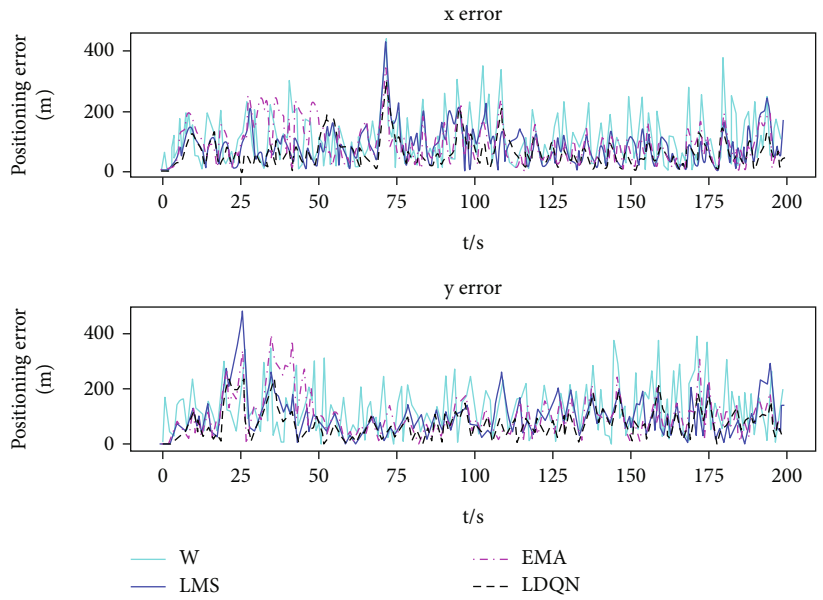

Figure 5: x-y tracking error of DS1.

3.2.2. Network Input/Output Selection. In the filtering equation, P_k is a quantitative description of the quality of the estimate, and the selection criteria of gain are as follows: the selection criteria of the gain matrix minimize the estimated mean square error matrix $P_k = E[\tilde{X}_k \tilde{X}^T{}_k]$, where $\tilde{X}_k = X_k - \widehat{X}_k$ is the estimation error [15]. Since the state covariance of the filter, that is, the mean square deviation matrix, is the value of each step filter estimation and reflection, and the gain matrix is the best way to minimize the mean square deviation matrix, the errors caused by model uncertainty and the nonlinearity of the driving matrix in the tracking filtering algorithm are mainly reflected in the filter gain and the filtered state covariance.

At the same time, according to the estimation Equation (4), the residual also determines the accuracy of target tracking.

Therefore, this paper uses the three variables mentioned in the above to describe the error compensation network as the input of error compensation for training. The improvement made is to embed the error modification network into the subfilter of the model with the highest probability, that is, the model corresponding to the action with the highest probability selected by the agent. Here, we call it the main filter and ignore the minor filter. Error modification is made for the filter at each hour. Under the condition that the intelligent decision of the model is correct, the nonlinear error of the model in the filtering algorithm is minimum, and the model description is more accurate.

The mathematical expression of the error modification network is as follows:

$$\left(\Delta x_k^{j^*} G_k\right) = \mathrm{LSTM}\left(r_k^j K_k^j P_{k-1}^j G_{k-1}\right). \tag{16}$$

4. Experimental Results and Result Analysis

4.1. Setting of Network Parameters. The setting of the reward function reflects whether the decision made by the agent at the current moment is correct or not. The establishment of the discount factor should consider not only the current return but also the future reward return. Therefore, the future reward return should be attenuated. In the initial moment of the simulation, the discount factor should be set as 0.9 to prevent the local optimality phenomenon and attenuate with the increase of step length as the interaction with the environment progresses.

At the initial stage of algorithm setting, the target collects experience before training. If much experience is not collected, the training will not be carried out. The training will not be conducted after each interaction with the environment but after 4 interactions with the environment.

Greedy decision-making: when using the strategy corresponding to the action value to estimate the action, strategy is used. The setting of the exploration factor decreases gradually with the progress of training. At step 5×10^4, $\varepsilon = 1$, then decrease linearly until $\varepsilon = 0.01$, and keep ε unchanged. In this way, each step of the agent's strategy is determined based on the maximum Q value of the current state during the training process.

The learning rate is $1e^{-5}$, and the RMSProp optimizer is used to train 20,000 rounds.

The data comes from the target's 2d motion track data, including turning motion, CA motion, CV motion, and their combinations. At the same time, the noise of the data is increased for the adopted training data to increase the amount of available training data and increase the adaptability of the algorithm. The noise we have added is white Gaussian noise, and the synthetic datasets consist of the sum of the measured data and the added noise.

4.2. Error Modification Network. Since the maneuvering of the aircraft is complex, the number of training rounds is increased. After 20,000 rounds of training, the output reward function image is shown in Figure 4.

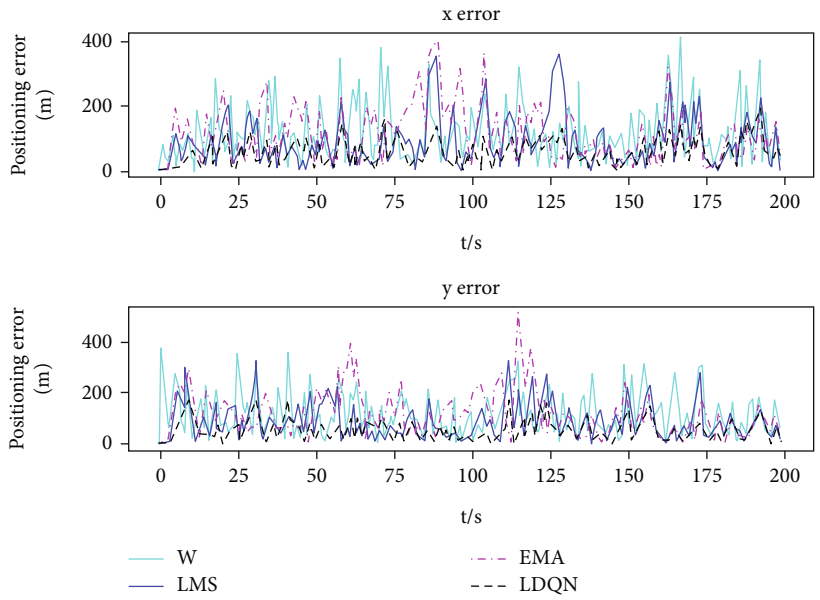

Figure 6: x-y tracking error of DS2.

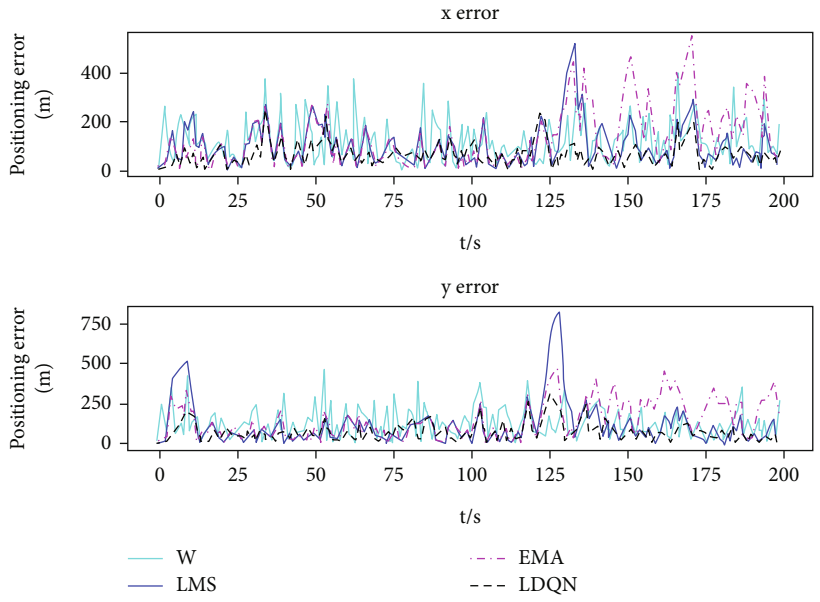

Figure 7: x-y tracking error of DS3.

4.3. Comparative Analysis. Three deterministic schemes are selected to evaluate the pros and cons of the improved algorithm proposed in this paper. Deterministic schemes 1, 2, and 3 are called DS1, DS2, and DS3, respectively, to study the peak error, steady-state error, and response time of the algorithm. DS1, DS2, and DS3 select the maneuvering tracking target model set to cover the target model set, the tracking target model set is more than the target model set, and the tracking target model set does not completely cover the target model set, so as to compare the advantages of the proposed algorithm in different simulation scenarios.

In order to design more complex maneuvers to verify the advantages and disadvantages of the algorithm, we assume that the target moving in the two-dimensional horizontal plane, z direction is zero. The initial position and speed of the target are set as (5000m, 5000m) and (500m / s, 400m / s), respectively, and the flight time is 200s.

The position and speed of the object are selected as the filtered state quantity, which is expressed as $X = [x\, y\, \dot{x}\, \dot{y}]^T$. The sampling interval is 1 s, the measurement data is the position $Z = [x\, y]^T$ of the object body, and the measurement matrix is

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{17}$$

Table 2: Tracking results.

Group		Average root-mean-square error		Accuracy of jumping		Response time (s)
		x (m)	y (m)	x (m)	y (m)	
DS1	LMS	79.29	92.91	419.49	470.14	0.12
	EMA	86.94	96.23	326.53	385.20	0.42
	L-DQN	58.95	64.21	281.89	237.23	0.03
DS2	LMS	77.95	87.78	359.05	335.85	0.14
	EMA	76.83	83.71	398.85	510.34	0.40
	L-DQN	56.85	59.69	185.01	173.37	0.03
DS3	LMS	94.67	106.99	522.04	818.92	0.41
	EMA	129.14	136.95	543.16	466.53	0.14
	L-DQN	58.17	62.46	243.01	286.81	0.02

Circular turning and uniform linear motion are selected to form the motion model set.

The CT turning model is as follows:

$$F = \begin{bmatrix} 1 & \frac{\sin(wt)}{w} & 0 & \frac{\cos(wt)-1}{w} \\ 0 & \cos(wt) & 0 & \sin(wt) \\ 0 & \frac{1-\cos(wt)}{w} & 1 & \frac{\sin(wt)}{w} \\ 0 & -\sin(wt) & 0 & \cos(wt) \end{bmatrix}. \tag{18}$$

The constant-velocity moving model is as follows:

$$F = \begin{bmatrix} 1 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{19}$$

The maneuver scheme is shown in Table 1, and the tracking model set in EMA and LMS algorithms is consistent with DS1.

The tracking result diagram of DS1 is shown in Figure 5.

The tracking result diagram of DS2 is shown in Figure 6.

The tracking result diagram of DS3 is shown in Figure 7.

In Figures 5–7, w indicates the positioning error.

The tracking results are summarized in Table 2.

Through comparative analysis, it can be concluded that when the tracking model set just covers the target motion model set, the EMA, LMS, and L-DQN algorithms proposed in this paper have a more accurate tracking accuracy. Meanwhile, compared with the first two schemes, the improved L-DQN effectively reduces the mean square error of tracking, improves the jumping accuracy of the model, and shortens the response time of the algorithm. For scheme DS2, it is simpler than the operation scheme of DS1, so the accuracy of the three is improved to a certain extent, but compared with DS1, the simulation shows that both the LMS and EMA algorithms lose more jumping accuracy in comparison to L-DQN. For the case that the target model falls outside the algorithm model set, in addition to the loss of jumping accuracy, there was also a certain degree of divergence in both cases, and the result of the tracking error being greater than the positioning error appeared.

The improved L-DQN can use offline training to solve the problem of model coverage and online response time and realize the positive correlation between response speed and model coverage ability and model accuracy.

According to the simulation results, it can be concluded that the calculation time required for the system to perform localization is much smaller than the time interval between each localization, which can meet the real-time requirements in the case of limited computing resources.

5. Conclusion

In order to improve the tracking accuracy of strongly maneuvering targets, this paper proposes a VSMM algorithm based on L-DQN. Based on the variable-structure filtering algorithm, the model selection of the target is mapped to the choice of the action labels, and reasonable state space observation variables, network structure, and reward function are designed to achieve the purpose of using the DQN algorithm to replace the traditional model decision. At the same time, the LSTM algorithm is introduced to compensate the tracking and positioning error. Experimental results show that the proposed algorithm can solve the problem of incomplete model coverage in the face of an unknown maneuver mutation, shorten the response time which is caused by too many model sets of the algorithm, and realize the positive correlation between the calculation accuracy and response speed of the algorithm. The algorithm does not depend on the fixed topology of the model and the inherent prior knowledge such as the fixed threshold, so it has good adaptability and stability.

The LSTM algorithm still has limitations in tracking maneuvering targets. LSTM models are designed to capture dependencies in sequential data, but they may struggle to capture the complex and nonlinear dynamics of maneuvering targets. Besides, sensitivity to input representation cannot be ignored. LSTM models heavily rely on the quality and representation of input features. If the input features

do not adequately capture the relevant characteristics of maneuvering targets, the LSTM model's performance may be limited.

The RL algorithm used in target tracking has several potential limitations. RL algorithms typically require a great number of interactions with the environment to learn effective policies. In target-tracking scenarios, collecting sufficient data can be challenging, especially if the target's behavior is rare or difficult to observe. Designing an appropriate reward function is crucial in RL. For target tracking, defining a reward function that accurately reflects the desired tracking behavior can be difficult. It may be challenging to strike a balance between rewarding the agent for successful tracking and penalizing it for incorrect or inefficient actions.

Due to the limitations in algorithm performance and other factors, the application of this algorithm in practical engineering is currently limited. However, the method that the L-DQN algorithm used in VSMM and the error modification network based on the LSTM network has certainly enhanced accuracy and reduced errors compared to traditional methods and can be gradually expanded to practical engineering after improvement.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation (NNSF) of China under Grant 62101579.

References

[1] X. R. Li and Y. Zhang, *Multiple-Model Estimation with Variable Structure: Likely Model Set Algorithm*, SPIE, Bellingham WA, 1998.

[2] X. Rong Li and Y. Zhang, "Multiple-model estimation with variable structure. V. Likely-model set algorithm," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 2, pp. 448–466, 2000.

[3] L. I. X. Rong, V. P. Jilkov, and J. Ru, "Multiple-model estimation with variable structure-part VI: expected-mode augmentation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 3, pp. 853–867, 2005.

[4] J. Wang, P. Dong, Z. Jing, and J. Cheng, "Consensus variable structure multiple model filtering for distributed maneuvering tracking," *Signal Processing*, vol. 162, pp. 234–241, 2019.

[5] Y.-Q. Wang, Z. Lu, and C. Yun-Ze, "Consensus-based distributed variable structure multiple model," *Acta Automatica Sinica*, vol. 47, no. 7, pp. 1548–1557, 2021.

[6] S. A. Lipton, Y.-B. Choi, Z.-H. Pan et al., "A redox-based mechanism for the neuroprotective and neurodestructive effects of nitric oxide and related nitroso-compounds," *Nature*, vol. 364, no. 6438, pp. 626–632, 1993.

[7] W. Wei, J. Wang, Z. Fang, J. Chen, Y. Ren, and Y. Dong, "3U: joint design of UAV-USV-UUV networks for cooperative target hunting," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 4085–4090, 2022.

[8] Z. Fang, J. Wang, Y. Ren, Z. Han, H. V. Poor, and L. Hanzo, "Age of information in energy harvesting aided massive multiple access networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 5, pp. 1441–1456, 2022.

[9] Y. Zhang, J. Wang, L. Zhang, Y. Zhang, Q. Li, and K.-C. Chen, "Reliable transmission for NOMA systems with randomly deployed receivers," *IEEE Transactions on Communications*, vol. 71, no. 2, pp. 1179–1192, 2023.

[10] J. Wang, L. Bai, J. Chen, and J. Wang, "Starling flocks inspired resource allocation for ISAC aided green ad hoc networks," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 1, pp. 444–454, 2023.

[11] T.-Y. Zheng, Y. Yu, and F.-H. He, "Trajectory estimation of a hypersonic night vehicle via L-EKF," *Journal of Harbin Institute of Technology*, vol. 52, no. 6, pp. 160–170, 2020.

[12] J. B. Kim, Y. Park, and I. H. Suh, "Tracking human-like natural motion by combining two deep recurrent neural networks with Kalman filter," *Intelligent Service Robotics*, vol. 11, no. 4, pp. 313–322, 2018.

[13] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[14] V. K. K. S. Mnih, "Playing Atari with deep reinforcement learning," 2013, https://arxiv.org/abs/1312.5602.

[15] Q. Yong-yuan, Z. Hong-Yue, and S.-H. Wang, *Kalman Filter and Principle of Integrated Navigation*, Northwestern Polytechnic University Press, 2012.

[16] S. M. Shah and V. S. Borkar, "Q-learning for Markov decision processes with a satisfiability criterion," *Systems & Control Letters*, vol. 113, pp. 45–51, 2018.

[17] T. Degris, P. M. Pilarski, and R. S. Sutton, *Model-free reinforcement learning with continuous action in practice*, American Control Conference, 2012.

[18] Y. Hong-Ge, Z. Wei, and H.-Q. Yang, "Joint regression object localization based on deep reinforcement learning," *Acta Automatica Sinica*, vol. 41, pp. 1–11, 2020.