DOI: 10.61782/fa.2025.0487

FORUM ACUSTICUM
EURONOISE 2025

INTERACTING MULTIPLE MODEL TARGET TRACKING ALGORITHM FOR UNDERWATER MANEUVERING TARGETS BASED ON TRANSFORMER

Yu Liu[1] Tianhang Ji[1,2*] Xiaochuan Ma[1,2]
Xiaomei Wang[3] Chao Feng[1] Pengzhuo Li[1,2] Yubo Hu[1,2]
[1] Institute of Acoustics, Chinese Academy of Sciences, Beijing, P. R. China
[2] University of Chinese Academy of Sciences, Beijing, P. R. China
[3] National Satellite Ocean Application Service, Beijing, P. R. China

ABSTRACT

The Interacting Multiple Model (IMM) algorithm has gained significant attention as an effective approach for maneuvering target tracking. However, when applied to underwater scenarios characterized by sparse observations, this method suffers from intrinsic model transition delays that significantly degrade tracking precision during maneuver phases. To address this limitation, we present a novel hybrid architecture integrating transformer-based neural networks with conventional tracking methodologies. Our core contribution lies in developing a temporal context-aware probability adaptation module through pattern extraction from historical state estimations, which effectively mitigates latency in model set adaptation. Monte Carlo simulations demonstrate statistical enhancement with 10% improvement in tracking precision and 20% reduction in model transition latency. The results indicate that the proposed algorithm outperforms the traditional IMM in both tracking error and response speed.

Keywords: *Interacting Multiple Model Filter, Transformer, target tracking*

1. INTRODUCTION

Due to the complexity and uncertainty of maneuvering target motion patterns, this task poses significant technical challenges. Over the past decades, various classical methods based on Bayesian tracking theory have been developed to improve tracking accuracy, including the Kalman Filter (KF) [1], Extended Kalman Filter (EKF) [2], Unscented Kalman Filter (UKF) [3], and Particle Filter (PF) [4, 5]. Under ideal conditions, these methods provide reliable tracking performance when the motion model assumptions hold. However, in real-world scenarios, accurately modeling target motion is challenging, especially in highly maneuverable states where abrupt motion changes occur. As a result, a single motion model often fails to adapt effectively, leading to degraded tracking performance.

To address this challenge, researchers have proposed various strategies to enhance the tracking performance of maneuvering targets, such as the input estimation algorithm [6], the Singer model [7], and the Interacting Multiple Model (IMM) algorithm [8]. Among these, the IMM algorithm has emerged as the mainstream approach for maneuvering target tracking due to its superior adaptability and robustness. By incorporating multiple motion models and their corresponding filters, the IMM algorithm estimates the target state through a weighted fusion mechanism, achieving a balance between tracking accuracy and computational complexity. However, the conventional IMM algorithm still has inherent limitations. Its tracking performance depends on a predefined set of motion models, making model selection critical. If the number of models is insufficient, it may fail to describe the target's motion characteristics accurately, leading to reduced tracking accuracy. Conversely, an excessive number of models significantly increases computational over-

*__Corresponding author__: jitianhang@mail.ioa.ac.cn.

head, reducing the real-time performance of the algorithm [9]. Moreover, in high-dimensional state spaces associated with complex motion patterns, the conventional IMM struggles to model target motion effectively, further impairing tracking accuracy. In addition, the IMM algorithm relies on the Bayesian update mechanism to compute the probability of each filter and estimate the state. However, when the target undergoes a maneuver, the algorithm often requires a considerable amount of time to accurately recognize the new motion pattern, resulting in substantial estimation delays that degrade tracking performance. This issue is particularly pronounced in high-speed maneuvering target tracking scenarios.

With the rapid advancement of deep learning technologies, particularly recurrent neural networks (RNNs) [10] and long short-term memory (LSTM) networks [11], these models have demonstrated unique advantages in addressing sequential problems, offering novel solutions to the challenges of the IMM algorithm [12–16]. These networks can extract temporal features from measurement data to estimate target states, while bidirectional LSTMs further aid in correcting trajectory deviations [14]. Although LSTM and its variants have shown certain advantages in state estimation tasks [15], their performance is often constrained when processing long sequences [16]. In contrast, the Transformer models input-output dependencies entirely through an attention mechanism, eliminating the need for an RNN structure and significantly enhancing sequential modeling capabilities. Moreover, since Transformers support parallel computation, they not only achieve higher computational efficiency than LSTM but also simultaneously capture both local and global dependencies within sequences, thereby further improving tracking accuracy.

To address the aforementioned challenges, we propose a Transformer-based Interacting Multiple Model (Transformer-IMM) tracking method. This approach leverages the Transformer to predict the switching of maneuvering targets between different motion models, replacing the model probability update step in the conventional IMM algorithm to improve the timeliness of model transitions. Subsequently, the improved IMM algorithm is employed to track the target state. Simulation results demonstrate that the proposed method reduces motion mode switching delays while enhancing the accuracy of target state estimation.

2. PROBLEM STATEMENT

Establish a state space model of a nonlinear system:

$$\begin{cases} X_k = F_{k-1}X_{k-1} + G_{k-1}w_{k-1} \\ z_k = H_k(X_k) + v_k \end{cases} \tag{1}$$

Where $X_k = [x_k, y_k, \dot{x}_k, \dot{y}_k] \in R^4$ represents the state vector. $w_k \in R^n$ is the process noise, $v_k \in R^m$ is the measurement noise, w_k and v_k are uncorrelated with each other.

Assume that both the process noise and measurement noise follow a Gaussian distribution.

$$p(w_k) = N(w_k; 0, Q_k) \tag{2}$$

$$p(v_k) = N(v_k; 0, R_k) \tag{3}$$

The target is tracked using the Constant Velocity (CV) model and the Constant Turn (CT) model, with their respective state transition matrices given by:

$$F_k^{CV} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$F_k^{CT} = \begin{bmatrix} 1 & 0 & \frac{\sin(\omega_k T)}{\omega_k} & \frac{(\cos(\omega_k T)-1)}{\omega_k} \\ 0 & 1 & \frac{(1-\cos(\omega_k T))}{\omega_k} & \frac{\sin(\omega_k T)}{\omega_k} \\ 0 & 0 & \cos(\omega_k T) & -\sin(\omega_k T) \\ 0 & 0 & \sin(\omega_k T) & \cos(\omega_k T) \end{bmatrix} \tag{5}$$

where T is the sampling interval, ω_k represents the target's turn rate.

The noise gain matrix for both the CV model and the CT model is given by:

$$G_k = \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ 0 & \frac{1}{2}T^2 \\ T & 0 \\ 0 & T \end{bmatrix} \tag{6}$$

The measurement equation can be expressed as:

$$\begin{aligned} z_k &= H_k(X_k) + v_k \\ &= \begin{bmatrix} \sqrt{x_k^2 + y_k^2} \\ \arctan\left(\frac{y_k}{x_k}\right) \end{bmatrix} + \begin{bmatrix} v_r \\ v_\theta \end{bmatrix} \end{aligned} \tag{7}$$

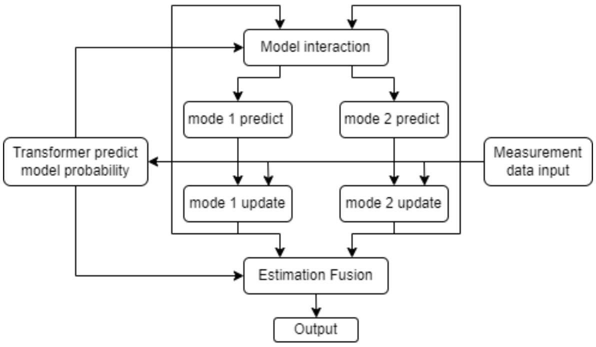

Figure 1. Transformer-IMM framework

3. PROPOSED ALGORITHM

3.1 Overall Architecture

In traditional Interactive Multiple Model (IMM) algorithms for maneuvering target tracking, the significant delay in model probability estimation often results in high tracking errors. To address this issue, we integrate the Transformer architecture into the IMM framework, leveraging its self-attention mechanism to effectively capture long-term dependencies and enhance the recognition of target motion patterns. With its strengths in nonlinear modeling and efficient filtering computation, the Transformer facilitates faster and more accurate estimation of model probabilities, thereby reducing model-switching latency and improving both tracking accuracy and real-time performance. In the proposed Transformer-IMM algorithm, the Transformer is employed to predict the model probabilities of the target, and these predictions are incorporated into the IMM update process to optimize the overall estimation. Unlike conventional IMM methods that directly update model probabilities based solely on observation data, our approach introduces a feed-forward prediction mechanism via the Transformer to provide a more accurate initial model probability prior to the interaction update. This predictive enhancement effectively improves estimation accuracy. The IMM algorithm is implemented in accordance with the approach described in [17]. The structure of the proposed Transformer-IMM algorithm is illustrated in Figure 1.

Since the Transformer model relies on a fixed-length input window for time series prediction, it may not effectively predict the initial phase of the target's motion. Therefore, during the initial stage of the target's trajectory, we still employ the traditional IMM algorithm to ensure prediction accuracy.

Assuming the target has n motion modes, the algorithm steps are as follows:

1) Input observation data , Use the Transformer to compute the target's model probabilities, obtaining the model probabilities as:

$$\mu_k^j = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} a_i \in \{0,1\}, \forall i = 1, 2, \cdots, n \tag{8}$$

Where μ_k^j denotes the model probability predicted by the Transformer, which is used to guide the IMM computation process, thereby enabling the interaction update in IMM to better align with the target's motion mode.

2) Compute the mixed state estimate $\widehat{X}^{0j}_{k-1|k-1}$ and the mixed covariance estimate $P^{0j}_{k-1|k-1}$ for each model.

3) Apply UKF to each model for filtering, obtaining the filtered output results $\widehat{x}^{j}_{k|k}$ and $P^{j}_{k|k}$.

4) Fuse the outputs of each model by weighting them according to their model probabilities to compute the final target state estimate $\hat{X}_{k|k}$ and $P_{k|k}$.

5) Repeat steps 1) – 4).

3.2 Transformer Neural Network

The Transformer consists of a positional encoding structure, N stacked encoders, N stacked decoders, and an output module. The encoder comprises a multi-head self-attention layer, a feedforward fully connected network, and two residual connection with layer normalization. Similarly, The decoder contains of a masked multi-head attention layer, a multi-head attention layer for encoder-decoder interaction, a feedforward fully connected network, and three residual connections with layer normalization. The output module consists of a linear fully connected layer and a Softmax activation function. Figure 2 illustrates the framework of the Transformer model.

3.2.1 Positional Encoding

Since the Transformer employs a self-attention mechanism, it can process input sequences in parallel. However, this prevents it from directly capturing the temporal order of the input sequence. Positional Encoding (PE) is introduced to incorporate positional information into the input sequence. Transformer typically adopts a fixed positional encoding approach, where position encodings are generated using sine and cosine functions of

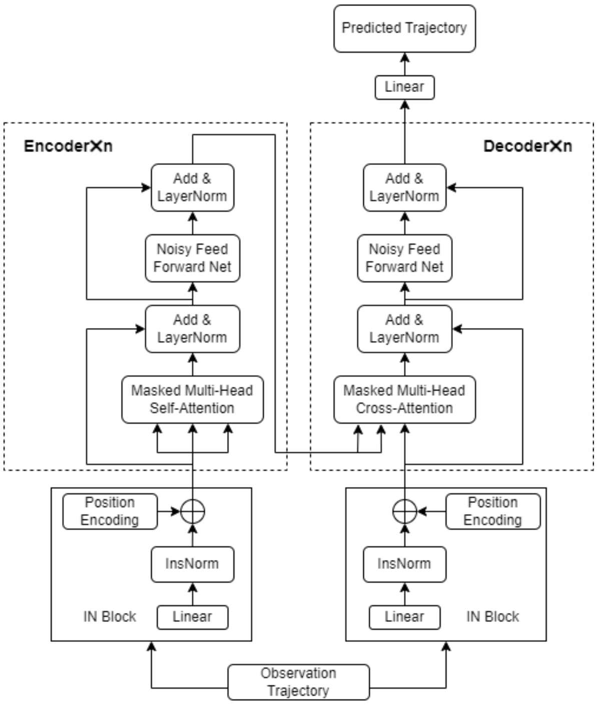

Figure 2. the framework of the Transformer model

different frequencies. These encodings vary across different time steps. At time step k, the positional encoding is given by:

$$\tilde{Z}_{e/d} = ZW + b + PE \tag{9}$$

$$PE(k) = \left[\sin(\rho_0 k),\ \cos(\rho_0 k),\ \cdots,\ \sin(\rho_{d/2} k),\ \cos(\rho_{d/2} k)\right] \tag{10}$$

$$\rho_i = \frac{1}{1000^{2i/d}} i = 0, 1, \cdots, (\mathrm{d}/2) - 1 \tag{11}$$

Where $Z \in R^{T\times 2}$ represents the input data sequence, $W \in R^{2\times d}$ is the weight matrix, and $b \in R^{T\times d}$ is the bias vector, which is used to project the observation vector into a d-dimensional representation space. $PE \in R^{T\times d}$ denotes the absolute positional encoding. $\tilde{Z}_e \in R^{T\times d}$ and $\tilde{Z}_d \in R^{T\times d}$ represent the state encodings of the encoder and decoder, respectively. The learnable parameters of the positional encoding are not shared.

3.2.2 Multi-Head Attention Mechanism

The multi-head attention mechanism is used to extract global features from the input sequence and enhance the Transformer's ability to process different information dimensions. Compared to traditional RNNs, which can only handle local and synchronous dependencies, multi-head attention can simultaneously capture both short-term and long-term dependencies. Both the encoder and decoder incorporate multi-head attention mechanisms, but their input variables differ depending on the application scenario. For convenience, the input sequence is denoted as $S \in {}^{T\times d}$, and it is assumed that this mechanism consists of M self-attention heads.

First, the input sequence is mapped separately into the query matrix $Q \in {}^{T\times d_k}$, the key matrix $K \in {}^{T\times d_k}$, and the value matrix $V \in {}^{T\times d_k}$ The computation is performed as follows:

$$\begin{cases} Q_i = S * W_{Q,i} \\ K_i = S * W_{K,i} \quad i = 1 \cdots M \\ V_i = S * W_{V,i} \end{cases} \tag{12}$$

Where $W_{Q,i}, W_{K,i}, W_{V,i} \in {}^{d\times d_k}$ are linear projection matrices.

The output of the single-head attention can then be computed as follows:

$$Head_i = \mathrm{Softmax}(Q_i \cdot {K_i}^T / \sqrt{d_k}) \cdot V_i \tag{13}$$

Where d_k is the scaling factor, used to prevent excessively large inner products from affecting gradient computation, and it satisfies $d_k = d/M$.

Finally, the outputs of the individual attention heads are concatenated and projected to the final dimension to compute the output of the multi-head attention mechanism

$$Head = Concat(Head_1, \cdots, Head_M) \cdot W_0 \tag{14}$$

The main difference between the masked multi-head attention mechanism and the standard multi-head attention mechanism lies in the introduction of a mask when computing single-head attention. This ensures that future information is not accessed during sequence generation. The computation is performed as follows:

$$Head_i = \mathrm{Softmax}(mask(Q_i \cdot {K_i}^T / \sqrt{d_k})) \cdot V_i \tag{15}$$

Where $mask(\cdot)$ is typically an upper triangular matrix.

Additionally, in the multi-head attention mechanism of the encoder and the masked multi-head attention mechanism of the decoder, the computation of Q, K and V is based on the input module or the output from the previous layer of the encoder (or decoder). In the decoder's multi-head attention mechanism, K and V are computed based on the final output of the encoder, while Q is derived from the output of the masked multi-head attention mechanism in the decoder.

3.2.3 Feedforward Fully Connected Network

The feedforward fully connected network (FFN) applies independent nonlinear transformations to the features at each time step. By utilizing a two-layer fully connected structure, it enhances information abstraction, improves feature representation, and strengthens the model's nonlinear modeling capability. FFN complements the self-attention mechanism, enabling the Transformer not only to capture global dependencies within sequences but also to enhance information processing at individual time steps. FFN consists of two sublayers: the first layer applies an activation function, while the second layer performs a linear transformation:

$$\mathrm{FFN}(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2 \qquad (16)$$

Where x is the input to the FFN, $W_1 \in R^{d \times d_f}$, $W_2 \in R^{d_f \times d}$, $b_1 \in R^{d_f}$ and $b_2 \in R^d$ are learnable parameters. $\mathrm{ReLU}(\,\cdot\,)$ is the ReLU activation function.

3.2.4 Residual Connection and Layer Normalization

Residual connection and layer normalization work together in the Transformer to enhance training stability and optimization efficiency. The residual connection employs skip connections to directly transmit information, preventing gradient vanishing, accelerating convergence, and preserving input features to minimize information loss. Meanwhile, layer normalization normalizes the features at each time step, stabilizing data distribution, accelerating convergence, and improving generalization capability.

$$O = \mathrm{LayerNorm}(x + \mathrm{SubLayer}(x)) \qquad (17)$$

Where $\mathrm{LayerNorm}(\cdot)$ represents layer normalization, and $\mathrm{SubLayer}(\cdot)$ denotes a specific sublayer within the Transformer.

3.2.5 Output Model

The output module consists of a linear transformation followed by a Softmax activation function, which can be expressed as:

$$\mathrm{Output} = \mathrm{Softmax}(xW_0 + b_0) \qquad (18)$$

Where x is the output of the decoder, $W_0 \in R^d$ and $b_0 \in {}^T$ are learnable parameters.

4. SIMULATION

4.1 Training the Transformer Model

The Transformer network was trained on 200,000 sets of maneuvering target motion trajectories. Details of the experimental parameters are provided in Table 1. The model was trained for a total of 100 epochs, with each batch containing 128 training samples. The loss function used was cross-entropy loss, and the optimizer was ADAM with an initial learning rate of 0.001. Additionally, the learning rate decay factor was set to 0.02 per epoch.

Table 1. Trajectory Dataset Parameters

Contents	Range
Distance from sonar	$[0, 10km]$
Velocity of target	$[0, 10m/s]$
Maneuvering turn rate	$[-5°/s, 5°/s]$
Deviations of distance noise	$[5m, 10m]$
Deviations of azimuth noise	$[0.5°, 1°]$

4.2 Simulation Setup

The total simulation sampling time is 100 seconds, with a sampling interval of T=2s. The target motion models at different time steps are shown in Table 2, while the target motion trajectory and measurements are illustrated in Figure 3.

Table 2. Target Motion Models at Different Time Steps

Time/s	Target Motion Model
1-68	CT(1.5°/s)
70-118	CV
120-200	CT(-3°/s)

The process noise covariance matrix Q_k , measurement noise covariance matrix R_k , initial target state information X_0, initial state covariance matrix P_0, transition probability Π and initial model probability μ_0 are set as follows:

$$Q_k = diag[(0.01m/s)^2, (0.01m/s)^2, (0.02°/s)^2] \qquad (19)$$

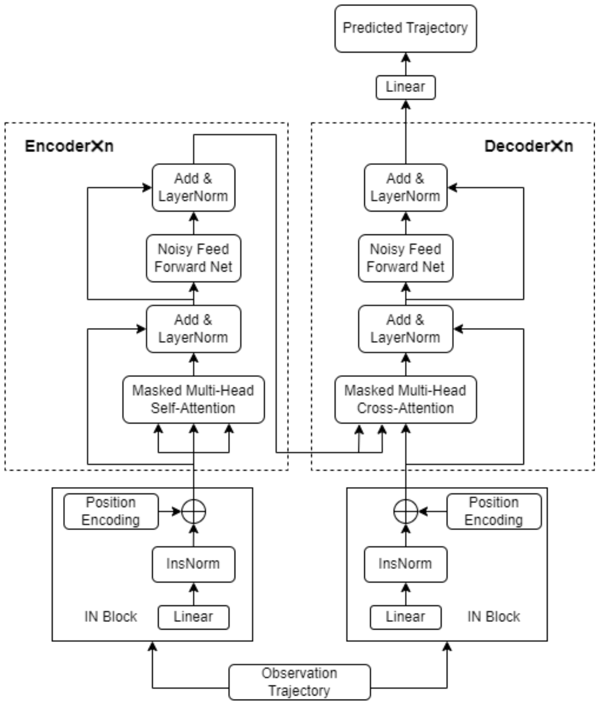

Figure 3. Target Motion Trajectory and Measurements

$$R_k = diag[(5m)^2, (0.5^\circ)^2] \tag{20}$$

$$X_0 = [305m, 512, 3m/s, 6m/s, 0] \tag{21}$$

$$P_0 = diag[(20m)^2, (20m)^2, (5m/s)^2, (5m/s)^2] \tag{22}$$

$$\Pi = \begin{bmatrix} 0.98 & 0.01 & 0.01 \\ 0.01 & 0.98 & 0.01 \\ 0.01 & 0.01 & 0.98 \end{bmatrix} \tag{23}$$

$$\mu_0 = [0.34 \;\; 0.33 \;\; 0.33] \tag{24}$$

To analyze the performance of the filter, the root mean square error (RMSE) of position, velocity, and turn rate, along with the average RMSE (ARMSE), are selected as performance metrics to evaluate the filter's accuracy and consistency.

$$\mathrm{RMSE}_{\mathrm{pos}}(k) = \sqrt{\frac{1}{M}\sum_{s=1}^{M}\left((x_k^s - \hat{x}_k^s)^2 + (y_k^s - \hat{y}_k^s)^2\right)} \tag{25}$$

$$\mathrm{RMSE}_{vel}(k) = \sqrt{\frac{1}{M}\sum_{s=1}^{M}\left(\left(v^s_{x(k)} - \widehat{v}^s_{x(k)}\right)^2 + \left(v^s_{y(k)} - \widehat{v}^s_{y(k)}\right)^2\right)} \tag{26}$$

$$\mathrm{ARMSE}_{\mathrm{pos}} = \sqrt{\frac{1}{MT}\sum_{k=1}^{N}\sum_{s=1}^{M}\left((x_k^s - \hat{x}_k^s)^2 + (y_k^s - \hat{y}_k^s)^2\right)} \tag{27}$$

$$\mathrm{ARMSE}_{vel} = \sqrt{\frac{1}{MT}\sum_{k=1}^{N}\sum_{s=1}^{M}\left(\left(v^s_{x(k)} - \widehat{v}^s_{x(k)}\right)^2 + \left(v^s_{y(k)} - \widehat{v}^s_{y(k)}\right)^2\right)} \tag{28}$$

Where M=1000 represents the number of Monte Carlo simulation runs, and N denotes the number of tracking steps in each Monte Carlo run.

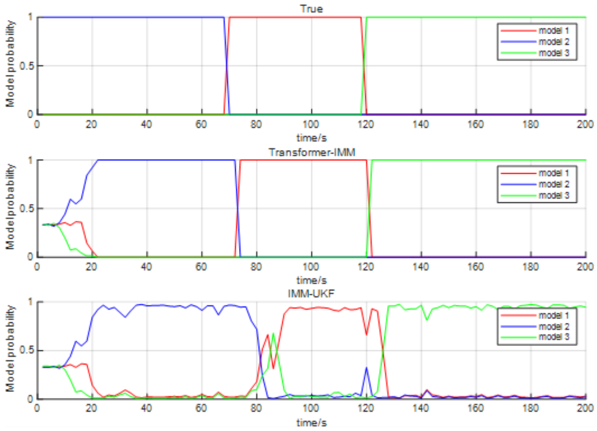

Figure 4. Model Probabilities of the Proposed Algorithm and Traditional IMM

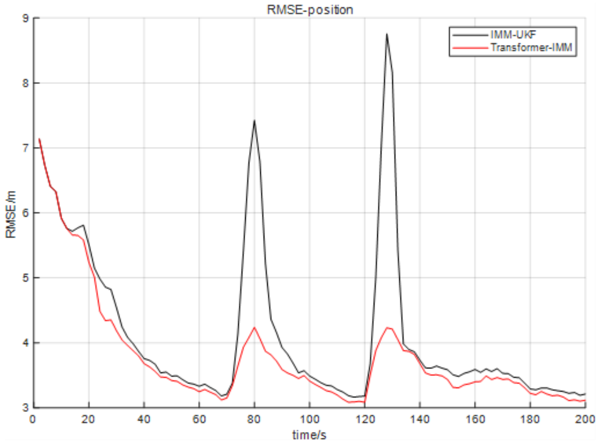

Figure 5. Comparison of Position Root Mean Square Error (RMSE) Plot

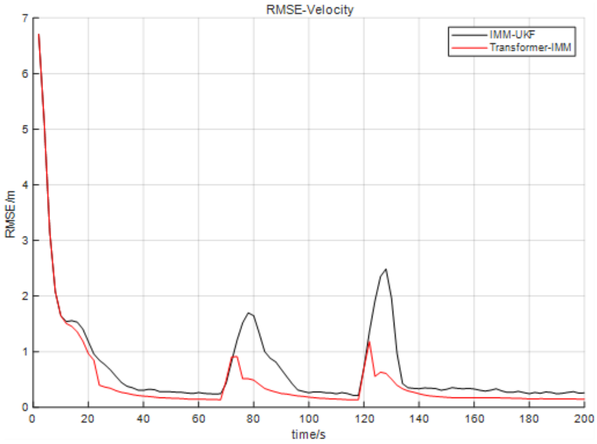

Figure 6. Comparison of Velocity Root Mean Square Error (RMSE) Plot

Table 3. Position and Velocity ARMSE

Filter	Position ARMSE(m)	Velocity ARMSE(m/s)
IMM-UKF	4.3392	1.2048
Transformer-IMM	3.8831	1.0327

4.3 Simulation Results

The performance of Transformer-IMM and IMM-UKF in target tracking tasks was analyzed in the simulation. Figure 4 presents the predicted model probabilities of both algorithms alongside the ground truth model probabilities. The results indicate that the proposed algorithm identifies the target's motion model more quickly and accurately. Specifically, the target undergoes its first maneuver at 68s and the second at 118s. The proposed algorithm detects these maneuvers at 72s and 120s, whereas the traditional IMM algorithm identifies them at 80s and 126s, demonstrating a slower response.

Figure 5 and Figure 6 illustrate the position RMSE and velocity RMSE of both algorithms. It can be observed that throughout the tracking process, the proposed algorithm consistently achieves lower errors than IMM-UKF. Notably, when a maneuver occurs, IMM-UKF experiences a significant increase in error, whereas Transformer-IMM maintains a relatively smaller error growth.

Table 2 presents the ARMSE results for both algorithms. The position ARMSE of Transformer-IMM is 3.8831 m, which is 10.5% lower than that of IMM-UKF, while the velocity ARMSE is 1.0327 m/s, representing a 14.3% reduction compared to IMM-UKF.

5. CONCLUSION

To enhance the tracking performance of underwater maneuvering targets, we propose a Transformer-based IMM algorithm that accelerates the identification of maneuvering target motion models, improving recognition accuracy and tracking precision. Simulation results demonstrate that, compared to the traditional IMM algorithm, the proposed method exhibits superior performance in both tracking accuracy and response speed. In future work, we will further validate the reliability of the proposed algorithm by testing it on more complex maneuvering target trajectories and evaluating its practical effectiveness in real experimental environments.

6. ACKNOWLEDGMENTS

This work is supported by the National Key R&D Program of China under Grant 2022YFC3101900.

7. REFERENCES

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

[2] E. Cortina, D. Otero, and C. E. D'Attellis, "Maneuvering target tracking using extended kalman filter," *IEEE transactions on aerospace and electronic systems*, vol. 27, no. 1, pp. 155–158, 1991.

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

[4] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 425–437, 2002.

[5] J. H. Kotecha and P. M. Djuric, "Gaussian sum particle filtering," *IEEE Transactions on signal processing*, vol. 51, no. 10, pp. 2602–2612, 2003.

[6] Y. Bar-Shalom, T. E. Fortmann, and P. G. Cable, "Tracking and data association," 1990.

[7] R. A. Singer, "Estimating optimal tracking filter performance for manned maneuvering targets," *IEEE Transactions on Aerospace and electronic systems*, no. 4, pp. 473–483, 1970.

[8] R. R. Pitre, V. P. Jilkov, and X. R. Li, "A comparative study of multiple-model algorithms for maneuvering target tracking," in *Signal Processing, Sensor Fusion, and Target Recognition XIV*, vol. 5809, pp. 549–560, SPIE, 2005.

[9] X.-R. Li and Y. Bar-Shalom, "Multiple-model estimation with variable structure," *IEEE Transactions on Automatic control*, vol. 41, no. 4, pp. 478–493, 1996.

[10] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[11] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[12] C. Gao, H. Liu, S. Zhou, H. Su, B. Chen, J. Yan, and K. Yin, "Maneuvering target tracking with recurrent neural networks for radar application," in *2018 International Conference on Radar (RADAR)*, pp. 1–5, IEEE, 2018.

[13] C. Gao, J. Yan, S. Zhou, P. K. Varshney, and H. Liu, "Long short-term memory-based deep recurrent neural networks for target tracking," *Information Sciences*, vol. 502, pp. 279–296, 2019.

[14] J. Liu, Z. Wang, and M. Xu, "Deepmtt: A deep learning maneuvering target-tracking algorithm based on bidirectional lstm network," *Information Fusion*, vol. 53, pp. 289–304, 2020.

[15] W. Yu, H. Yu, J. Du, M. Zhang, and J. Liu, "Deepgtt: A general trajectory tracking deep learning algorithm based on dynamic law learning," *IET radar, sonar & navigation*, vol. 15, no. 9, pp. 1125–1150, 2021.

[16] J. LIU, Y. ZHANG, and B. ZHANG, "Trajectory estimation algorithm for unmanned aerial vehicle based on lstm-kf," *Journal of Beijing University of Posts and Telecommunications*, vol. 45, no. 5, p. 121, 2022.

[17] X. Wu, Y. Liu, and X. Ma, "An underwater maneuvering target tracking algorithm based on ukf with adaptive sampling range," in *2023 42nd Chinese Control Conference (CCC)*, pp. 3133–3138, IEEE, 2023.