sensors

MDPI

Communication

Transformer-Based Maneuvering Target Tracking

Guanghui Zhao [1,*], Zelin Wang [1], Yixiong Huang [1], Huirong Zhang [1] and Xiaojing Ma [2]

[1] School of Artificial Intelligence, Xidian University, Xi'an 710071, China
[2] School of Electronic Confrontation, National University of Defense, Hefei 230037, China
* Correspondence: ghZhao@xidian.edu.cn

Abstract: When tracking maneuvering targets, recurrent neural networks (RNNs), especially long short-term memory (LSTM) networks, are widely applied to sequentially capture the motion states of targets from observations. However, LSTMs can only extract features of trajectories stepwise; thus, their modeling of maneuvering motion lacks globality. Meanwhile, trajectory datasets are often generated within a large, but fixed distance range. Therefore, the uncertainty of the initial position of targets increases the complexity of network training, and the fixed distance range reduces the generalization of the network to trajectories outside the dataset. In this study, we propose a transformer-based network (TBN) that consists of an encoder part (transformer layers) and a decoder part (one-dimensional convolutional layers), to track maneuvering targets. Assisted by the attention mechanism of the transformer network, the TBN can capture the long short-term dependencies of target states from a global perspective. Moreover, we propose a center–max normalization to reduce the complexity of TBN training and improve its generalization. The experimental results show that our proposed methods outperform the LSTM-based tracking network.

Keywords: attention mechanism; maneuvering target tracking; recurrent neural network; transformer-based network

Citation: Zhao, G.; Wang, Z.; Huang, Y.; Zhang, H.; Ma, X. Transformer-Based Maneuvering Target Tracking. *Sensors* **2022**, *22*, 8482. https://doi.org/10.3390/s22218482

Academic Editor: Gemine Vivone

Received: 14 October 2022
Accepted: 1 November 2022
Published: 4 November 2022

1. Introduction

With the rapid development of the electronic information industry, target tracking technology has been increasingly used in the military and civilian fields. The target tracking task aims to estimate the state of the target based on data measured by sensors. It can be classified into maneuvering and non-maneuvering target tracking, where "maneuvering" refers to the case in which the target suddenly changes its motion state. For the tracking of maneuvering targets, the interactive multi-model (IMM) algorithm, which uses multiple models to fit complex motion states, is considered [1]. Therefore, many tracking algorithms proposed subsequently were based on the IMM [2–4]. However, IMM-based algorithms are associated with the mismatch problem between the set of models and the target motion states. Furthermore, when the motion state of the target changes, a specific number of observations must be accumulated, resulting in the model estimation delay problem [5].

The development of deep neural networks, especially recurrent neural networks (RNNs) with memory ability, provides novel ideas to solve the problems of IMM-based algorithms [6–9]. The RNN [10] and long short-term memory (LSTM) networks [11] can estimate the state from the observation at each time step [6,12]. Nevertheless, the LSTM and RNN can only process the input sequence sequentially, resulting in long-distance memory fading problems [9]. Thus, the LSTM and RNN may reduce the correlation between trajectory points at different locations, which subjectively influences the modeling of maneuvering states. In addition, trajectory datasets are usually collected in a fixed-range coordinate system and preprocessed with min–max normalization [8,13,14]. However, the same maneuvering state in the dataset may correspond to trajectories with different initial positions, which increases the complexity of network learning. Moreover, the fixed distance range reduces the generalization of the network.

In this study, to accurately model and estimate the states of maneuvering targets, we propose a transformer-based network (TBN). Specifically, our proposed network applies the transformer network as an encoder to extract global features of the observation sequence. Simultaneously, 1D convolutional networks are applied as a decoder to estimate the state sequence from the features. Compared with the LSTM network, which processes observations sequentially, the TBN associates the observations at all positions and applies an attention mechanism to model their dependencies [15]. Thus, the features of the observations can be represented independently without regard to their position in the sequence [16–18]. Therefore, the TBN has better feature representation and global memory ability than LSTM [18]. Moreover, a learnable positional embedding is added to the input of the TBN to explore the temporal features of the observation sequence. Finally, a novel center–max normalization is applied by the TBN to improve generalization. Compared with the min–max normalization, our proposed center–max normalization transforms the trajectories from a fixed to a relative coordinate system with the initial observation point as the origin. The experimental results demonstrate that center–max normalization considerably increases the generalization of the TBN to trajectories with different distance ranges. Furthermore, center–max normalization also promotes the tracking performance of the TBN by reducing the complexity of trajectory learning.

2. Problem Formulation

Based on the previous research on maneuvering target tracking [4,7,8,19], we mainly considered point targets tracked by radar in the X-Y plane. Meanwhile, the problem of target birth and death was not considered in this study. Therefore, we assumed that z_k is the observation vector and x_k is the state vector at the kth time step. Specifically, $x_k = \left[c_{x,k}, c_{y,k}, v_{x,k}, v_{y,k}\right]$ denotes the coordinates and corresponding velocities in the two-dimensional scene, and $z_k = [\theta_k, d_k]$ denotes the azimuth and distance of the radar observation.

We intend to build a maneuvering target tracking model based on a deep neural network. The input to the model is the observation sequence $z_{1:K} = \{z_1, z_2, \ldots, z_K\}$, and the output is the estimated state sequence $\hat{x}_{1:K} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_K\}$, where K is the total number of time steps. Given that target tracking is a regression problem, we used the root-mean-squared error (RMSE) between the normalized ground-truth sequence $x^*_{1K} = \{x^*_1, x^*_2, \ldots, x^*_K\}$ and the estimated sequence $\hat{x}^*_{1:K} = \{\hat{x}^*_1, \hat{x}^*_2, \ldots, \hat{x}^*_K\}$ as the loss function [9] to evaluate the model:

$$Loss = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(\hat{x}^*_k - x^*_k\right)^2}. \tag{1}$$

In practice, obtaining a sufficient number of trajectories is difficult. Thus, we simulated segmented trajectories based on the state-space model (SSM) [20].

The SSM defines the state transition equation and observation equation as:

$$\begin{cases} x_k = Fx_{k-1} + n_k \\ z_k = h(x_k) + u_k \end{cases} \tag{2}$$

where F is the transition matrix and n_k is the transition noise. h is the nonlinear observation, and u_k is the observed noise.

In this study, two motion states were considered: constant velocity (CV) and constant turn (CT), as mentioned in [8]. The transition matrix of CV and CT is defined as:

$$F_{CV} = \begin{bmatrix} 1 & 0 & \tau & 0 \\ 0 & 1 & 0 & \tau \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$F_{CT} = \begin{bmatrix} 1 & 0 & \frac{\sin(w\tau)}{w} & \frac{\cos(w\tau)-1}{w} \\ 0 & 1 & \frac{1-\cos(w\tau)}{w} & \frac{\sin(w\tau)}{w} \\ 0 & 0 & \cos(w\tau) & -\sin(w\tau) \\ 0 & 0 & \sin(w\tau) & \cos(w\tau) \end{bmatrix} \tag{4}$$

where w is the turn rate of the maneuvering target and τ is the sampling interval of the observations. According to [21], the transition noise $n_k = [n_{c,k}, n_{c,k}, n_{v,k}, n_{v,k}]$ is calculated from:

$$\begin{bmatrix} n_{c,k} \\ n_{c,k} \\ n_{v,k} \\ n_{v,k} \end{bmatrix} = \begin{bmatrix} \frac{\tau^2}{2} & 0 \\ \frac{\tau^2}{2} & 0 \\ 0 & \tau \\ 0 & \tau \end{bmatrix} \cdot \begin{bmatrix} \alpha_k \\ \alpha_k \end{bmatrix} \tag{5}$$

where $\alpha_k \sim \mathcal{N}(0, \sigma_a^2)$ is the Gaussian noise caused by the maneuvering acceleration with zero mean and standard deviation σ_a.

For radar tracking, Z_k is defined as:

$$\begin{bmatrix} \theta_k \\ d_k \end{bmatrix} = \underbrace{\begin{bmatrix} \arctan\frac{c_{y,k}}{c_{x,k}} \\ \sqrt{c_{x,k}^2 + c_{y,k}^2} \end{bmatrix}}_{h(x_k)} + \begin{bmatrix} u_{\theta,k} \\ u_{d,k} \end{bmatrix} \tag{6}$$
$$u_{\theta,k} \sim \mathcal{N}(0, \sigma_\theta^2), u_{d,k} \sim \mathcal{N}(0, \sigma_d^2)$$

where σ_θ is the standard deviation of the azimuth and σ_d is the standard deviation of the distance.

3. Proposed Model

In this section, we discuss the components of the TBN in detail. In Section 3.1, we introduce a trajectory normalization method named center–max normalization to improve generalization. In Section 3.2, the structure of the TBN is presented. In Section 3.3, we summarize the overall process of applying the TBN for maneuvering target tracking.

3.1. Center–Max Normalization

A trajectory of a maneuvering target is exhibited in Figure 1. The left of Figure 1 shows an observation sequence, which contains the distance and azimuth. To eliminate the dimensional difference between the observations, z_{1K} in the polar coordinates are converted to $\tilde{z}_{1K}$ in the X-Y plane coordinates:

$$\begin{bmatrix} \underbrace{\tilde{z}_{x,k} \\ \tilde{z}_{y,k}}_{\tilde{z}_k} \end{bmatrix} = \begin{bmatrix} d_k \cos(\theta_k) \\ d_k \sin(\theta_k) \end{bmatrix}. \tag{7}$$

Figure 1c shows a trajectory in the X-Y plane coordinates. The distance range and initial position of the targets may vary extensively; thus, we propose a center–max normalization mechanism to improve the generalization of the model and reduce the training complexity, as shown in Figure 2. This can be formulated as follows:

$$\tilde{z}_k^* = \frac{\tilde{z}_k - \tilde{z}_1}{D_{\max}}, k = 1, \ldots, K \tag{8}$$

where $\tilde{z}_k^*$ is the normalized observation at the kth time step, $\tilde{z}_1$ is the initial value of $\tilde{z}_{1:K}$, and $D_{\max}$ denotes the maximum distance that the targets can move within K time steps. In Equation (8), the observation sequence is normalized to $[-1, 1]$ by dividing by $D_{\max}$. Subtracting $\tilde{z}_1$, the observation sequence $\tilde{z}_{1:K}$ is represented in a relative coordinate system

with $\tilde{z}_1$ as the origin. Benefiting from center–max normalization, the TBN only needs to focus on learning different maneuvers of the target without considering the influence of the initial position. Therefore, the tracking of maneuvering targets by the TBN is not limited by the detection range. Correspondingly, the ground-truth state sequence $x_{1:K}$ is normalized as follows:

$$\begin{aligned} x^*_{1:K} &= \frac{x_{1:K} - c_x}{X_{\max}} \\ c_x &= [\tilde{z}_{x,1}, \tilde{z}_{y,1}, 0, 0] \\ X_{\max} &= [D_{\max}, D_{\max}, V_{\max}, V_{\max}] \end{aligned} \tag{9}$$

where $x^*_{1:K}$ is the normalized state sequence, c_x is the centering vector corresponding to $x^*_{1:K}$, $[\tilde{z}_{x,1}, \tilde{z}_{y,1}]$ is the position component of $\tilde{z}_1$, and $V_{\max}$ is the maximum speed of the simulation targets.

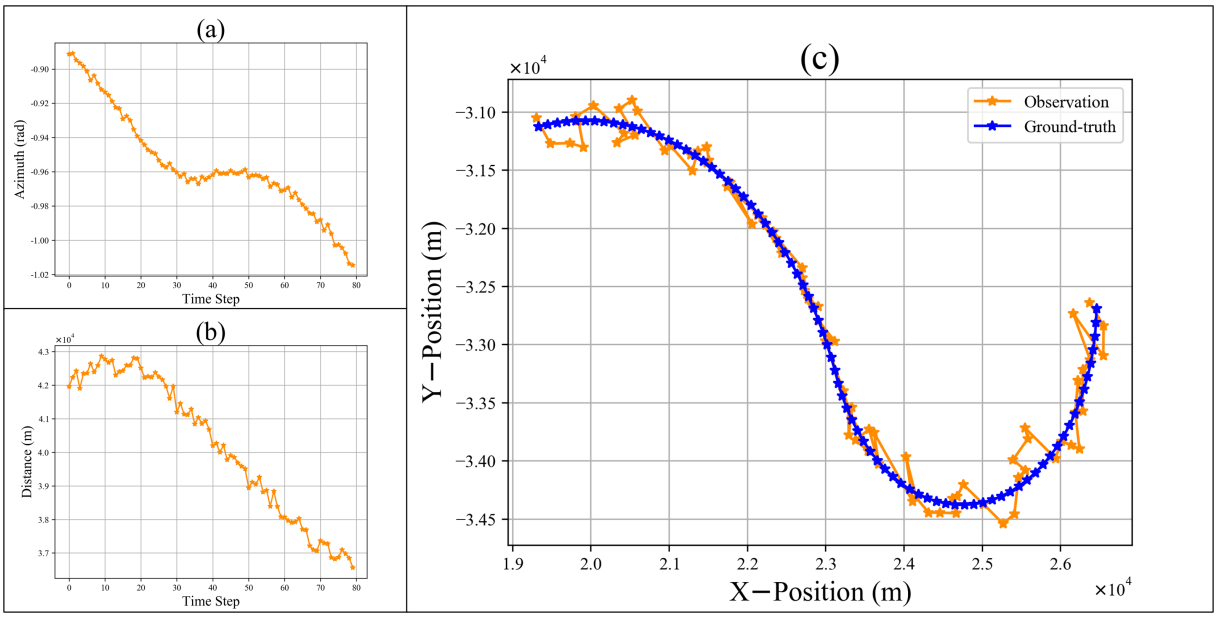

Figure 1. The observation and ground-truth of a trajectory. The (**a**) is the azimuth observation sequence of the trajectory. The (**b**) is the distance observation sequence of the trajectory. The (**c**) is the observation and ground-truth sequence in the X-Y plane coordinate system.

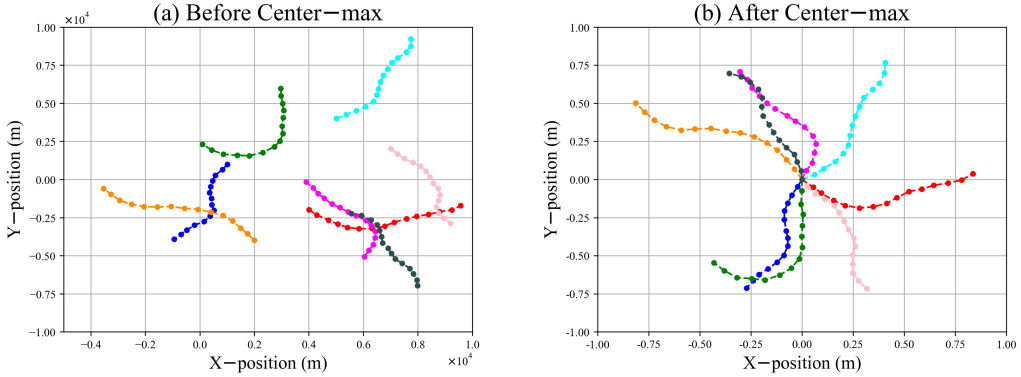

Figure 2. Center–max normalization. After center–max normalization, the distance ranges of the trajectories are transformed to $[-1, 1]$ and the differences in the initial positions of the trajectories are removed.

3.2. Proposed Network

In sequence modeling tasks, the LSTM network sequentially extracts features. However, the transformer network uses the self-attention mechanism to process input data in parallel, which can capture both local and global dependencies. Therefore, we innovatively introduced it to the target tracking task to comprehensively capture the internal law of target maneuvering. Our proposed TBN consists of positional encoding, N-stacked transformer encoder layers, and one convolutional decoder layer. Each transformer encoder layer contains multi-head self-attention, a feedforward fully connected network, and two residual connections after each of the previous blocks. For intuitive understanding, the entire architecture of the TBN is shown in Figure 3.

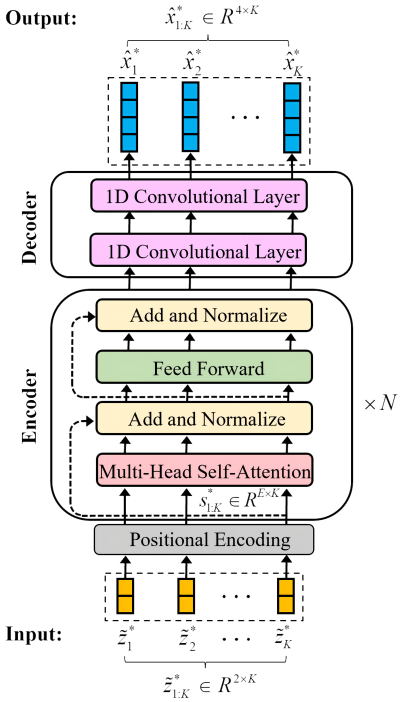

Figure 3. Architecture of the TBN. Input data are normalized observation sequences $\tilde{z}^*_{1:K}$. $\tilde{z}^*_{1:K}$ is first mapped to $s^*_{1:K}$, whose dimension is $E \times K$ by positional encoding. The encoder consists of N-stacked multi-head self-attention and fully connected feedforward layers, which aim at extracting the features of $s^*_{1:K}$. The decoder maps E-dimensional feature vectors to the normalized state sequence $\hat{x}^*_{1:K}$ by two 1D convolutional layers.

3.2.1. Positional Encoding

In natural language processing tasks, the transformer network adds positional encoding to the input tokens to represent their relative or absolute positions in the sequence [15]. However, in this study, the input to the TBN is numeric. Therefore, the learnable positional encoding mentioned [22] is added to the input of the network as follows:

$$s^*_{1:K}[i] = \begin{cases} w_i \tilde{z}^*_{1:K} + \varphi_i, \text{ if } i = 0 \\ \mathcal{F}\left(w_i \tilde{z}^*_{1:K} + \varphi_i\right), \text{ if } 1 \le i \le E \end{cases} \tag{10}$$

where $\mathcal{F}$ is the sine function, w_i and φ_i are learnable parameters that map $\tilde{z}^*_k$ to an E-dimensional representation space, and $s^*_{1:K}[i]$ is the encoding result of the ith subspace.

3.2.2. Multi-Head Self-Attention

Self-attention is the core of the TBN. First, the input encoding sequence $s^*_{1:K} \in R^{E\times K}$ of self-attention is linearly mapped into the sequences "query" (Q), "keys" (K), and "values" (V) as follows:

$$\begin{aligned} Q &= W_Q \cdot s_{1:K} \\ K &= W_K \cdot s_{1:K} \\ V &= W_V \cdot s_{1:K} \end{aligned} \tag{11}$$

where W_Q, W_K, and $W_V \in R^{E\times E}$ are learnable matrices.

Furthermore, Q, K, and V are split into M subsequences along dimension E, and M attention heads are obtained by the interaction of the elements at any two positions in each subsequence:

$$head_m = \text{soft}\max\left(\frac{Q_m^T K_m}{\sqrt{d_M}}\right)V_m, m = 1, \ldots, M \tag{12}$$

where $Q_m, K_m, V_m \in R^{E_m\times K}$ and $E_m = \frac{E}{M}$.

Finally, M attention heads are concatenated to compose the multi-head self-attention:

$$s_{attention} = Concat(head_1, \ldots, head_M). \tag{13}$$

Thus, the network is allowed to capture more information from different representation subspaces at different positions.

3.2.3. Feedforward Layer

After the multi-head self-attention, a feedforward layer consisting of two fully connected layers is used to linearly transform each position of $s_{attention}$.

In the decoder part, two 1D convolutional layers are used to output the final trajectory estimation $\hat{x}_{1:K}$, and the parameters of the network are trained by minimizing Equation (1) using the mini-batch gradient descent.

3.3. Maneuvering Target Tracking Based on the TBN

When the well-trained TBN is applied to track a complete trajectory, all observations are first segmented with window length $K = 10$ and step size $P = 5$. These segmented observation sequences are then normalized sequentially and passed to the TBN to estimate the corresponding state sequence set $\left\{\left(\hat{x}^{1+rP}_{1:K}\right)^*, r \in (0, \ldots, R-1)\right\}$, where $\left(\hat{x}^{1+rP}_{1:K}\right)^*$ denotes the normalized state sequence output at time step $(1 + rP)$ and R is the number of sequences. Subsequently, $\left(\hat{x}^{1+rP}_{1:K}\right)^*$ needs to be denormalized as follows:

$$\hat{x}^{1+rP}_{1:K} = \left(\hat{x}^{1+rP}_{1:K}\right)^* \odot X_{\max} + c^r_x, r = 0, \ldots, R-1 \tag{14}$$

where c^r_x is the centering vector of the rth state sequence. In addition, adjacent state sequences $\hat{x}^{1+rP}_{1:K}$ and $\hat{x}^{1+(r+1)P}_{1:K}$ are merged together. Let $\bar{x}^{1+rP}_{1:2K-P}$ denote the merge result of two above-mentioned state sequences, whose length is $2K - P$. Thus, the overlapped regions of $\bar{x}^{1+rP}_{1:2K-P}$ are calculated as follows:

$$\bar{x}^{1+rP}_{P:K} = 0.5\left(\hat{x}^{1+rP}_{P:K} + \hat{x}^{1+(r+1)P}_{1:K-P}\right). \tag{15}$$

Finally, all state sequences in the set $\left\{\left(\hat{x}^{1+rP}_{1:K}\right), r \in (0, \ldots, R-1)\right\}$ are merged in turn to obtain complete state estimates. Figure 4 illustrates the overall tracking process,

including the observations segmentation, center–max normalization processing, network estimation, denormalization processing, and segmented state sequences concatenation.

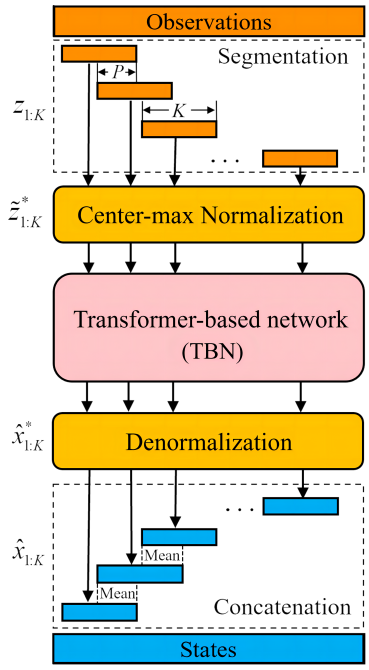

Figure 4. Structure of the transformer-based maneuvering target tracking. The observation sequences of targets are firstly segmented into subsequences $z_{1:K}$ of length K with step size P. After that, $z_{1:K}$ are converted to $\tilde{z}^*_{1:K}$ by center–max normalization. Then, the TBN infers the normalized trajectory $\hat{x}^*_{1:K}$ from $\tilde{z}^*_{1:K}$. In addition, $\tilde{z}^*_{1:K}$ are de-normalized to $\hat{x}_{1:K}$. Finally, the overlapped region of $\hat{x}_{1:K}$ is averaged and concatenated to obtain the estimation of the complete state sequences.

4. Experiments and Results

In this section, we list the parameters of the trajectory dataset and the TBN. Several experiments were designed to test the tracking performance of our proposed model.

4.1. Implementation Details

Dataset: We generated 300,000 trajectories based on the SSM as a dataset. The parameters of the trajectory dataset are listed in Table 1. In addition, we assumed normalization parameters: $D_{\max} = 3$ km, $V_{\max} = 300$ m/s, and targets were observed every 1 s.

Hyper–parameters: Our network consists of four encoder layers, with eight attention heads. The dimension of E was 512. The output dimensions of the 1D convolutional layer in the decoder were 64 and 4, respectively. The model was trained using the Adam optimizer [23] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\varepsilon = 10^{-9}$. The learning rate was linear warmed-up for the first 10 epochs and decayed subsequently based on the dynamic adjustment strategy mentioned in [15]. We trained 300 epochs with a batch size of 64 on a single NVIDIA TITAN Xp GPU.

Baseline: We compared the TBN+center–max normalization (TBN+CM) model with the IMM algorithm [19] and the LSTM+min–max normalization (LSTM+MM) tracking model [8]. As a comparison, we also built the LSTM+center–max normalization (LSTM+CM) model. The LSTM network consisted of four hidden layers with a dimension of 128, as mentioned in [8]. The same dataset was used to train the above networks.

Table 1. Parameters of the trajectory dataset.

Parameter	Value
Distance range	1 km∼10 km
Angle range	0°∼360°
Velocity range	−300 m/s∼300 m/s
Turn rate (w)	−10 °/s∼10 °/s
The standard deviation of acceleration noise (σ_a)	2 m/s^2∼8 m/s^2
The standard deviation of azimuth noise (σ_θ)	0.1°∼0.3°
The standard deviation of distance noise (σ_d)	5 m∼8 m

4.2. Results

We first compared the performances of the LSTM+MM, LSTM+CM, and TBN+CM based on a test set containing 20,000 segmented trajectories. The tracking results are listed in Table 2. In Table 2, the position and velocity RMSEs of the LSTM+CM are smaller than that of the LSTM+MM, which proves that our proposed center–max normalization improved the tracking capability of the network by reducing the complexity of trajectory learning. At the same time, the TBN+CM achieved the smallest position and velocity RMSE. Thus, it can be concluded that the TBN yields better performance than LSTM when tracking segmented trajectories.

Table 2. Numerical results of several methods for tracking segmented trajectories.

	RMSE of Position (m)	RMSE of Velocity (m/s)
LSTM+MM	16.27	6.75
LSTM+CM	14.43	5.14
TBN+CM	**13.50**	**3.64**

We then simulated a target with the initial states of [2 km, 2 km, 50 m/s, 0 m/s] and steering rates equal to 0° and conducted Monte Carlo simulations to generate a 60-step trajectory named A1. The target maneuvers had turn rates equal to −1° and 3° at the 10th step and the 40th steps, respectively. In addition, the standard deviations of acceleration, azimuth, and distance noise were set to 5 m/s^2, 0.2°, and 5 m. We evaluated the TBN+CM, LSTM+MM, LSTM+CM, and IMM algorithms on trajectory A1. The tracking results are listed in Table 3 and Figure 5.

Table 3. Numerical results of several methods for tracking trajectory A1.

	RMSE of Position (m)	RMSE of Velocity (m/s)
IMM	14.54	6.35
LSTM+MM	11.82	4.64
LSTM+CM	10.30	3.47
TBN+CM	**9.33**	**2.04**

Among the listed figures, Figure 5a shows how well the algorithms tracked the target. Figure 5b,c show the pointwise RMSE of trajectory A1. Furthermore, the average RMSEs of trajectory A1 are listed in Table 3. In Table 3, the RMSEs of the LSTM+CM are smaller than those of the LSTM+MM, which proves that our proposed center–max normalization improved the tracking capability of the network by reducing the complexity of trajectory learning. At the same time, the bolded results in Table 3 indicate that TBN+CM had the smallest tracking error. The experiments above demonstrated the superiority of the TBN+CM in tracking maneuvering targets.

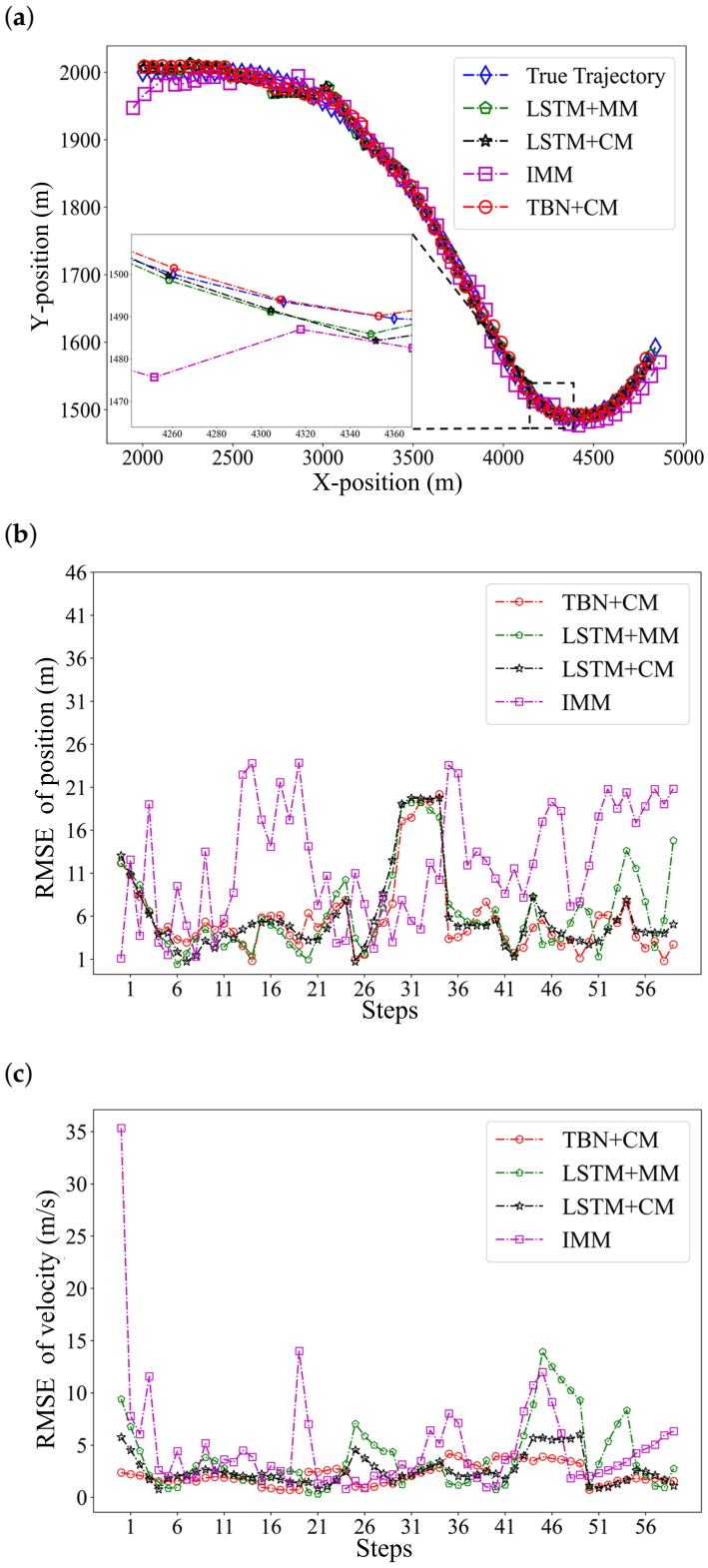

Figure 5. The result of tracking a maneuvering target by the TBM+CM, LSTM+MM, LSTM+CM, and IMM algorithms. (**a**) Tracking trajectory in the X-Y plane. (**b**) Pointwise position RMSE. (**c**) Pointwise velocity RMSE.

In addition, the initial position of trajectory A1 was moved to [12 km, 12 km] and [15 km, 15 km] to obtain trajectories A2 and A3. We conducted generalization experiments on trajectories A2 and A3, as listed in Table 4. The bolded results in Table 4

demonstrated that our proposed TBN+CM can generalize to tracking trajectories beyond the preset distance. However, the LSTM+MM led to tracking failure due to its fixed normalization mechanism.

Table 4. Results of tracking trajectories at different initial positions.

	RMSE of Position (m)		RMSE of Velocity (m/s)	
	TBN+CM	**LSTM+MM**	**TBN+CM**	**LSTM+MM**
A2	**9.94**	146.14	**2.08**	56.19
A3	**9.15**	295.71	**2.03**	78.92

5. Conclusions

In this study, we employed the attention mechanism of the transformer network to extract a comprehensive tracking of trajectories and finally developed a novel network named the TBN for radar target tracking missions. Furthermore, our proposed center–max normalization improved the generalization of the network by processing observations in a relative coordinate system. It can be seen from the experimental results that, when tracking maneuvering targets, our proposed TBN model obtained lower RMSEs of position and velocity than the LSTM model, and the TBN model can still work normally when the observation sequence is missing; however, the LSTM model will not be available. Therefore, our algorithm outperformed existing LSTM-based tracking networks and traditional algorithms.

Author Contributions: Investigation, G.Z., Z.W., Y.H. and H.Z.; methodology, G.Z., Z.W. and Y.H.; validation, Z.W. and Y.H.; visualization, Z.W.; writing—review and editing, Z.W., Y.H. and H.Z.; writing—original draft preparation, Z.W.; supervision, X.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Blom, H.A.; Bar-Shalom, Y. The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE Trans. Autom. Control.* **1988**, *33*, 780–783. [CrossRef]
2. Pulford, G.W.; La Scala, B.F. MAP estimation of target manoeuvre sequence with the expectation-maximization algorithm. *IEEE Trans. Aerosp. Electron. Syst.* **2002**, *38*, 367–377. [CrossRef]
3. Chen, H.; Chang, K. Novel nonlinear filtering & prediction method for maneuvering target tracking. *IEEE Trans. Aerosp. Electron. Syst.* **2009**, *45*, 237–249.
4. Ning, X.H.; Hui, X. Algorithm of maneuvering target tracking for video based on UKF and IMM. In Proceedings of the IEEE Conference Anthology, China, 1–8 January 2013; pp. 1–4.
5. Li, B.; Pang, F.; Liang, C.; Chen, X.; Liu, Y. Improved interactive multiple model filter for maneuvering target tracking. In Proceedings of the Proceedings of the 33rd IEEE Chinese Control Conference, Nanjing, China, 28–30 July 2014; pp. 7312–7316.
6. Gao, C.; Yan, J.; Zhou, S.; Varshney, P.K.; Liu, H. Long short-term memory-based deep recurrent neural networks for target tracking. *Inf. Sci.* **2019**, *502*, 279–296. [CrossRef]
7. Liu, J.; Wang, Z.; Xu, M. DeepMTT: A deep learning maneuvering target-tracking algorithm based on bidirectional LSTM network. *Inf. Fusion* **2020**, *53*, 289–304. [CrossRef]
8. Yu, W.; Yu, H.; Du, J.; Zhang, M.; Liu, J. DeepGTT: A general trajectory tracking deep learning algorithm based on dynamic law learning. *IET Radar Sonar Navig.* **2021**, *15*, 1125–1150. [CrossRef]
9. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**, arXiv:1803.01271.

10. Zimmermann, H.G.; Grothmann, R.; Schafer, A.M.; Tietz, C. Dynamical consistent recurrent neural networks. In Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, Montreal, QC, Canada, 31 July–4 August 2005; Volume 3, pp. 1537–1541.
11. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
12. Gao, C.; Liu, H.; Zhou, S.; Su, H.; Chen, B.; Yan, J.; Yin, K. Maneuvering target tracking with recurrent neural networks for radar application. In Proceedings of the 2018 IEEE International Conference on Radar (RADAR), Oklahoma City, OK, USA, 23–27 April 2018; pp. 1–5.
13. Ma, L.; Tian, S. A hybrid CNN-LSTM model for aircraft 4D trajectory prediction. *IEEE Access* **2020**, *8*, 134668–134680. [CrossRef]
14. Zhang, Z.; Ni, G.; Xu, Y. Ship trajectory prediction based on LSTM neural network. In Proceedings of the 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 12–14 June 2020; pp. 1356–1364.
15. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems 30*; MIT Press: Cambridge, MA, USA, 2017.
16. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
17. Kim, Y.; Denton, C.; Hoang, L.; Rush, A.M. Structured attention networks. *arXiv* **2017**, arXiv:1702.00887.
18. Shi, H.; Gao, S.; Tian, Y.; Chen, X.; Zhao, J. Learning Bounded Context-Free-Grammar via LSTM and the Transformer: Difference and Explanations. *Proc. Aaai Conf. Artif. Intell.* **2022**, *36*, 8267–8276. [CrossRef]
19. Magill, D. Optimal adaptive estimation of sampled stochastic processes. *IEEE Trans. Autom. Control.* **1965**, *10*, 434–439. [CrossRef]
20. Li, X.R.; Bar-Shalom, Y. Design of an interacting multiple model algorithm for air traffic control tracking. *IEEE Trans. Control. Syst. Technol.* **1993**, *1*, 186–194. [CrossRef]
21. Liu, J.; Wang, Z.; Xu, M. A Kalman estimation based rao-blackwellized particle filtering for radar tracking. *IEEE Access* **2017**, *5*, 8162–8174. [CrossRef]
22. Kazemi, S.M.; Goel, R.; Eghbali, S.; Ramanan, J.; Sahota, J.; Thakur, S.; Wu, S.; Smyth, C.; Poupart, P.; Brubaker, M. Time2vec: Learning a vector representation of time. *arXiv* **2019**, arXiv:1907.05321.
23. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.