GL-STGCNN: Enhancing Multi-Ship Trajectory Prediction with MPC Correction

Yuegao Wu, Wanneng Yv, Guangmiao Zeng, Yifan Shang and WeiQiang Liao *

School of Marine Engineering, Jimei University, Xiamen 361021, China; wyg729@foxmail.com (Y.W.); wnyu2007@jmu.edu.cn (W.Y.); miaogm.zeng@foxmail.com (G.Z.); 202112855021@jmu.edu.cn (Y.S.)
* Correspondence: wq_liao@jmu.edu.cn; Tel.: +86-0592-6181760

Abstract: In addressing the challenges of trajectory prediction in multi-ship interaction scenarios and aiming to improve the accuracy of multi-ship trajectory prediction, this paper proposes a multi-ship trajectory prediction model, GL-STGCNN. The GL-STGCNN model employs a ship interaction adjacency matrix extraction module to obtain a more reasonable ship interaction adjacency matrix. Additionally, after obtaining the distribution of predicted trajectories using the model, a model predictive control trajectory correction method is introduced to enhance the accuracy and reasonability of the predicted trajectories. Through quantitative analysis of different datasets, it was observed that GL-STGCNN outperforms previous prediction models with a 31.8% improvement in the average displacement error metric and a 16.8% improvement in the final displacement error metric. Furthermore, trajectory correction through model predictive control shows a performance boost of 44.5% based on the initial predicted trajectory distribution. While GL-STGCNN excels in multi-ship interaction trajectory prediction by reasonably modeling ship interaction adjacency matrices and employing trajectory correction, its performance may vary in different datasets and ship motion patterns. Future work could focus on adapting the model’s ship interaction adjacency matrix modeling to diverse environmental scenarios for enhanced performance.

Keywords: trajectory prediction; AIS; graph neural network; model predictive control; adjacency matrix

1. Introduction

With the growth of global trade, the density of ships on maritime routes continues to increase, leading to a higher frequency of ship interactions and, consequently, an elevated potential risk of collision accidents. However, with the continuous advancement of Internet of Things technology, numerous electronic devices are applied on ships to ensure maritime safety. Among them, the Automatic Identification System (AIS) serves as a ship’s automatic identification system utilizing radio technology to transmit crucial information to surrounding vessels and shore-based receiving stations. This information includes ship name, position, speed, heading, cargo type, etc. [1]. By collecting data, such as coordinates, heading, speed, and type over a certain period, a series of processing calculations can be performed to obtain the future trajectory positions of the ship. These position dates can be utilized for assessing collision risks, identifying abnormal ship behavior, optimizing route planning, and detecting potential faults and issues, thereby avoiding collisions and traffic congestion [2–5]. However, achieving accurate risk assessment and perception to ensure maritime safety crucially depends on obtaining sufficiently precise information about the future movements and positions of ships [6]. Therefore, enhancing the accuracy and reasonability of ship trajectory prediction is key to ensuring the safety and efficiency of maritime navigation.

In actual navigation, ships are often in crowded waters, engaging in frequent interactions with surrounding ships. Predicting ship trajectories solely based on historical trajectory data may lead to predictions that deviate from the actual situation [7]. Our aim is
to model the interactions between ships and combine this with the historical trajectory data of the vessels. This approach intends to produce trajectory predictions that closely align with real navigation outcomes, providing reliable support for the safe navigation of ships in congested waterways.

To enhance the modeling capability of network models for interactions between nodes in real-world scenarios, utilizing independently dynamic models to capture relationships between nodes can be beneficial. This approach enables a more accurate reflection of complex interaction scenarios. When studying the interaction relationships between targets, Neural Relational Inference (NRI) employs linear layers and embedding layers to obtain interaction relationships among multiple targets, discarding the previous method of determining interaction relationships solely based on the distance between targets [8]. When investigating the interaction intensity between targets, the Residual Attention Network (RAIN) model combines a Long Short-Term Memory Network (LSTM) with a Graph Attention Network (GAT). It represents the hidden states of LSTM as a graph structure and uses graph attention to extract correlations between hidden states, determining the interaction intensity between targets [9].

Considering that the outputs of neural networks typically exhibit randomness and may not explicitly understand the complexity and constraints of ship motion, there is often a problem of predicted trajectories violating the principles of kinematics. To improve the accuracy of trajectory prediction and ensure that the predicted results align with actual motion patterns, it is necessary to apply correction techniques to the predicted trajectories. Current methods for trajectory correction include those based on traffic rules, deep learning, Model Predictive Control (MPC), ant colony optimization algorithms, etc. [10–13]. Among these methods, MPC stands out as it not only considers the motion characteristics of targets but is also adaptable to various environments and task requirements.

In summary, to further improve the performance of multi-ship trajectory prediction, this paper conducts a thorough enhancement based on a Spatio Temporal Graph Convolutional Neural Network (STGCNN), designing a novel multi-ship trajectory prediction model called Graph Learning–Spatio-Temporal Graph Convolutional Neural Network (GL-STGCNN). By introducing a ship interaction adjacency matrix extraction module into GL-STGCNN, we address the limitation of modeling interaction adjacency matrices solely based on the distances between ships, enabling the model to flexibly capture dynamic interaction relationships among multiple ships. Although the introduction of predicted trajectory distributions allows for a more comprehensive understanding of the potential range of future ship movements, it simultaneously increases model uncertainty, making result interpretation and application more challenging. To address this issue, we integrate MPC into GL-STGCNN and adopt the concept of Velocity Obstacles (VO), correcting the predicted trajectory distribution from the network model output using a gradient descent-based approach. The aforementioned design enables GL-STGCNN to achieve both accuracy and motion feasibility in multi-ship trajectory prediction tasks.

The contributions of the paper are as follows.

The introduction of the ship interaction adjacency matrix extraction module overcomes the limitations of traditional methods that solely model interaction adjacency matrices based on distance. This innovation allows for a more flexible capture of dynamic interaction relationships among multiple ships.

The use of MPC and the VO method addresses the issue of predicted trajectories violating kinematic principles from the network model output. After sampling the probability distribution to obtain trajectory points, the trajectories are tracked through a gradient descent approach. This ensures the rationality and accuracy of the corrected trajectories.

GL-STGCNN excels in multi-ship trajectory prediction tasks through the aforementioned innovations. It not only enhances accuracy but also thoroughly considers the motion rationality of trajectories, bringing new breakthroughs to the research and application of ship trajectory prediction.
The remaining sections of this paper are as follows. In Section 2, we discuss the related works related to this article. Section 3 provides an introduction to the overall structure and relevant details of the model. In Section 4, we present a detailed description of the experimental design, including the dataset, loss functions, and other aspects. Section 5 involves testing the performance of the GL-STGCNN model and other relevant models using real ship trajectory datasets, followed by quantitative and qualitative analyses of the obtained results. Section 6 discusses the results obtained in the paper, and Section 7 concludes the entire manuscript.

2. Related Works

Ship trajectory prediction methods can be broadly categorized into two main classes: those based on traditional prediction methods and machine learning methods [14]. Traditional prediction methods for trajectory prediction tasks include techniques such as Kalman filters, hidden Markov models, Gaussian mixture models, Bayesian networks, etc. [15–18]. However, traditional prediction methods often rely on establishing probability models or regression models. When faced with a large number of targets and a substantial amount of data, these methods may not effectively handle interactions between multiple targets. They may struggle to accurately capture the characteristics of target behaviors and interactions between targets, resulting in suboptimal accuracy in trajectory prediction.

As a component of machine learning, neural networks exhibit increased expressive power and adaptability when addressing the challenge of predicting future ship trajectories. To incorporate interactions between targets in trajectory prediction, the Social-Long Short-Term Memory Network (Social-LSTM) utilizes the LSTM’s hidden states. It directly aggregates the hidden states of targets to model interactions, facilitating trajectory prediction for multiple targets engaged in interactions [19]. Furthermore, building upon Social-LSTM, trajectory classification is employed, individually training for each trajectory type, leading to improved predictive performance [20]. In tasks related to predicting traffic flow, the Graph-Long Short-Term Memory Network (Graph-LSTM) represents factors influencing traffic flow as a graph structure. By aggregating hidden states of different factors through this graph structure, Graph-LSTM achieves traffic flow prediction [21]. Improvements to Graph-LSTM involve representing the graph structure as a directed graph, establishing a feasible approach for one-way relationships between various factors [22,23]. In summary, the common approach to enhancing interactions between targets based on LSTM involves modifying the hidden layers of LSTM. However, data representing target interactions is often asymmetrical and falls into the category of irregularly structured data, making traditional LSTM, a type of recurrent neural network, less suitable for handling such irregular structures.

To avoid the impact of data structure shapes on the network model, the Social-Generative Adversarial Network (Social-GAN) introduces LSTM and performs pooling operations in the hidden layers of LSTM, enabling Social-GAN to handle target interaction data [24,25]. The Goal-Generative Adversarial Network (Goal-GAN) is applied to pedestrian trajectory prediction tasks, incorporating destination information into the training task through an end-to-end approach [26]. However, the goal of the Generative Adversarial Network (GAN) is to generate samples similar to real data distributions, and the model only aims to mimic the original samples as closely as possible [27]. In ship trajectory prediction tasks, we need neural networks to recognize the relationships between various variables. The model will struggle to capture the complexity and constraints of ship motion, leading to an inability to accurately understand the specific constraints of ship motion. Consequently, there may be issues with predicted trajectories violating the principles of ship kinematics.

To enable neural networks not only to handle irregular target interaction data but also to better capture complex interaction relationships among multiple targets, ensuring the output trajectories are task-relevant, the GAT has been applied. GAT employs an attention mechanism to automatically assign different weights to nodes during computation, elimi-
nating the need to predefine weights for each node [28], and it has been successfully used in traffic flow prediction [29]. Building upon GAT, an improvement for prediction tasks involves first converting time series into hidden states using LSTM and then inputting them into GAT for processing [30]. Since prediction tasks often involve time-series data, Temporal Graph Networks (TGN) have been proposed to handle dynamically continuous graph-structured data in the temporal dimension [31]; TGN performs excellently in traffic flow prediction where data change significantly over time [32]. To enhance the performance of TGN in processing input data, the Multitask Temporal Graph Network (MTGCN) introduces the GAT method on top of TGN [33]. STGCNN, based on TGN, enhances the utilization of spatial dimension information in graph-structured data through graph convolution operations in both time and space dimensions [34]. To better model the interactions between targets, the improvement of the Social-Spatio Temporal Graph Convolutional Neural Network (Social-STGCNN) involves modeling the interaction adjacency matrix based solely on the distances and other relationships between ships to predict their trajectories [35]. However, ship interactions are influenced not only by distance but also by other factors. Moreover, the interactions between ships may not be fixed and can change with time or task variations. Using a fixed interaction adjacency matrix may not adequately address this dynamic nature of interactions.

3. Multi-Ship Trajectory Prediction Model

The process of multi-ship trajectory prediction is illustrated in Figure 1. In a scenario involving interactions among multiple ships, the first step involves data preprocessing and the ship interaction adjacency matrix extraction module in GL-STGCNN. This transforms historical trajectories into a node matrix representing ship attributes and an adjacency matrix representing ship interactions. In this paper, the ship attributes we use include position, speed, and heading features. Subsequently, probabilistic distributions of predicted trajectories are obtained through spatiotemporal graph convolution and time extrapolation convolution.

Since probabilistic distributions cannot be directly applied to dynamic collision avoidance tasks, the final predicted trajectories are refined through MPC trajectory correction. The predicted trajectories are input into the MPC module, where, considering the predicted trajectories, surrounding environmental parameters, and the ship’s current motion state, control variables of the MPC are optimized and updated using a gradient descent approach. The optimized control variables are then fed into the ship’s control Mathematical Model Group (MMG), directing the ship’s next actions within the next t seconds while continuously updating the ship’s current state. In this paper, we choose to predict the future positions of ships for the next minute. Therefore, after updating the ship’s state and control variables iteratively 60 times, we obtain the final predicted trajectory.

The combination of GL-STGCNN ship trajectory prediction with the MPC model predictive control enhances the capabilities of the proposed ship trajectory prediction method in complex maritime environments. This approach considers not only the rapid changes in ship motion over short periods but also enables long-term ship trajectory prediction in a scientifically informed manner. Overall, this integration contributes to the improvement of maritime safety by addressing both short-term dynamic changes and providing scientifically grounded long-term ship trajectory predictions.
3.1. Interaction Intensity Extraction Module

As shown in Figure 2, the final output of the ship interaction intensity extraction module is an adjacency matrix representing the interaction intensity between nodes, where each node represents information about a specific ship. The process begins by encoding the positional information of vessels over a certain period using the LSTM network to obtain their hidden states. These hidden states undergo further processing through a fully connected layer to achieve a nonlinear mapping of the output tensor.

During the iterative processing phase, the nonlinear mapping tensor is replicated multiple times along the first dimension and multiplied by a matrix with matching dimensions. The output from the previous step is then modified to match the format of the iteration, allowing for multiple iterations as needed. The tensor output from the iterative processing steps is concatenated and further processed through a fully connected layer to obtain the final output, representing the adjacency matrix indicating the interaction intensity between ships.
3.2. STGCNN and TXPCNN

The time graph convolution module performs convolution in the time dimension to extract patterns and features between different time steps. Sharing parameters across different time steps helps capture patterns, features, and structures in sequential data, enabling the model to better understand and process time series data. To perform the specific operation, node matrix and adjacency matrix information about the graph are required. The node matrix is represented as $V_t = \{v_i^t | i \in \{1, \ldots, n\}\}$, with a shape of $(C, H, W)$, where $C$ represents the time steps, $H$ represents the number of nodes, and $W$ represents the node attributes. During convolution in the time dimension, the channels of the node matrix become the number of nodes $H$, and operations are performed on a two-dimensional plane of shape $(C, W)$.

The time extrapolation convolution module operates in the spatial dimension, performing convolution on the input in each channel. By setting the number of channels, it operates on the predicted time steps. For the node matrix with a shape of $(C, H, W)$, $C$ represents the time steps, $H$ represents the number of nodes, and $W$ represents the node attributes. During convolution in the spatial dimension, operations are performed on a two-dimensional plane of shape $(H, W)$. When performing graph neural network operations, given the positional motion information of vessels over a certain period $P_n^t = ((x_t^i, y_t^i) | t \in \{1, \ldots, T\})$ represents the ship’s position at time $t$, and $n$ is the number of vessels. This positional motion information can be represented as the structure of a graph $G_t = ((V_t, A_t) | t \in \{1, \ldots, T\})$, where $T$ denotes the time steps or the predicted time duration. Here, $V_t$ is the node matrix representing vessel positional motion information, and $A_t = \{a_{ij}^t | (i, j) \in \{1, \ldots, n\}\}$ is the adjacency matrix representing the interaction intensity between vessels. In this context, $a$ denotes the interaction intensity value between two ships. Based on the shape of the adjacency matrix, the $a$-directed adjacency matrix is a non-symmetric matrix.

The trajectory $P_t^n$ is processed using the ship interaction intensity extraction module to obtain the adjacency matrix between nodes. In general, $A_t^\wedge = A_t + I$, where $A_t$ is the adjacency matrix that does not consider self-influence, and $I$ is a diagonal matrix of the same shape as $A_t$. The calculation of the node matrix and the adjacency matrix is shown in Equation (1).

$$f(V_t, A_t) = \sigma(V_t \cdot A_t)$$  \hspace{1cm} (1)

In Equation (1), the node matrix is obtained from the processed node feature matrix, where $\sigma$ represents the activation function.
3.3. Predicted Trajectory Correction

As shown in Figure 3, the basic idea of trajectory correction is as follows. Firstly, a series of trajectory points is sampled from the probability distribution of predicted trajectories. Then, these sampled points need to be processed through spline curve fitting to obtain multiple points that closely approximate the fitted curve, with the number of points adjusted based on the predicted time length. These predicted points are then fed into the MPC module, integrating the predicted trajectory with the ship’s motion state. By utilizing gradient descent to optimize the cost function, the control variables are updated. These updated control variables are input into the ship’s motion model to track the trajectory points. Simultaneously, the ship’s current state is continually updated to achieve trajectory correction. This process ultimately produces the corrected predicted trajectory.

Figure 3. Trajectory correction schematic diagram.

3.3.1. Sampling from Trajectory Distribution

For the probability distribution of predicted trajectories, the coordinate information \((x, y)\) comprises two features, each with corresponding mean \(\mu\) and variance \(\sigma^2\). As for the sampling method from the distribution, firstly, two random numbers, \(U_1\) and \(U_2\), are randomly generated in the interval \([0, 1]\). Then, the Box–Muller transformation is utilized to convert the uniformly distributed random numbers into normally distributed random numbers.

\[
Z_0 = \sqrt{-2 \ln(U_1)} \cos(2\pi U_2) \tag{2}
\]

In Equation (2), \(Z_0\) is the sampled value from the standard normal distribution.

\[
X = \mu + Z_0 \cdot \sigma \tag{3}
\]

In Equation (3), \(X\) is the sampled value from the distribution with mean \(\mu\) and variance \(\sigma^2\).

Through the above steps, deterministic values can be sampled from the mean and variance of the trajectory distribution. Moreover, the aforementioned computational process can be easily implemented using the Pytorch (2.3.0) deep learning framework.

3.3.2. Motion Model

As shown in Figure 4, due to the lack of detailed motion parameters in AIS data, such as propeller speed and rudder angle, it implies that we cannot obtain information such as propeller speed and rudder angle. Establishing a comprehensive kinematic model for each individual ship poses significant challenges. Therefore, in constructing the kinematic model used to correct ship trajectories, only the current position and velocity of the ship are considered.
\[ \xi(t) = \begin{bmatrix} x \\ y \\ v \\ \alpha \end{bmatrix} \]

\[ u(t) = \begin{bmatrix} \dot{v} \\ \dot{\alpha} \end{bmatrix} \] (4)

Equation (4) neglects ship parameters and performance indicators. The established ship kinematics model only needs to consider the ship’s own position \((x,y)\), velocity \(v\), and heading angle \(\alpha\). \(\xi\) and \(u\) represent functions of the state variables and control variables, respectively.

**Figure 4. Principle of ship motion model.**

3.3.3. Linearization

\[ \dot{\xi} = f(\xi, u) \] (5)

In Equation (5), the derivative of the state variable \(\xi\) is a nonlinear function that cannot be directly represented and requires linearization processing.

\[ \dot{\xi} = f(\xi_r, u_r) + \frac{\partial f}{\partial \xi}(\xi - \xi_r) + \frac{\partial f}{\partial u}(u - u_r) \] (6)

According to the first-order Taylor expansion formula, since a reference point is needed for control, the expansion is carried out with respect to the reference target point \(\xi_r\), as shown in Equation (6).

\[ \xi' = \frac{\partial f}{\partial \xi} \xi + \frac{\partial f}{\partial u} u = A\xi + Bu \] (7)

The expanded error between the current point and the reference point, and its derivative, result in Equation (7).

\[ A = \begin{bmatrix} \frac{\partial f_1}{\partial \xi_1} & \frac{\partial f_1}{\partial \xi_2} \\ \frac{\partial f_2}{\partial \xi_1} & \frac{\partial f_2}{\partial \xi_2} \end{bmatrix} \] (8)

\[ B = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \end{bmatrix} \] (9)

The calculation result of \(A\) and \(B\) are given as shown in Equations (8) and (9), respectively, with the derivation provided therein.

3.3.4. Discretization

\[ \xi' = \frac{\xi(k+1) - \xi(k)}{T} = A\xi(k) + Bu(k) \] (10)

\[ \xi(k+1) = \tilde{A}\xi(k) + \tilde{B}u(k) \] (11)
In Equation (10), \( T \) represents the sampling time interval, which needs to be adjusted reasonably based on conditions to achieve the best performance. Equation (11) illustrates how to obtain the next state based on the previous state.

\[
\tilde{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{12}
\]

\[
\tilde{B} = \begin{bmatrix} T \sin(\alpha) & -Tv \sin(\alpha) \\ T \cos(\alpha) & -Tv \cos(\alpha) \end{bmatrix} \tag{13}
\]

As shown in Equations (12) and (13), since \( \tilde{A} \) consists of constants, its numerical values cannot be changed. The control process is completed solely by adjusting the parameters in \( \tilde{B} \) to minimize the error between the current point and the reference point.

\[
\alpha - 0.015\pi \leq \alpha \leq \alpha + 0.015\pi
\]

\[
0.99v \leq v \leq 1.01v \tag{14}
\]

As shown in Equation (14), during the trajectory correction process, it is necessary to impose some constraints on the model, such as constraints on the ship’s speed, heading angle, etc., to address the issue of unrealistic maneuvers in predicting ship trajectories. Considering the large inertia of the ship, it is difficult to reduce the speed in a short period. Therefore, it is necessary to carefully select the magnitude of changes in the constraint conditions. Through repeated experiments, it has been verified that during each iteration, within a time range of 1 second, the change in speed \( v \) should not exceed 1% of the previous speed, and the change in heading angle \( \alpha \) should not exceed 1.5% \( \pi \) based on the original heading angle. After trajectory tracking, this approach can achieve better trajectory correction effects.

3.3.5. Cost Function

In Figure 5, the distance from the ship to the target point is denoted as \( l \), and \( \theta \) represents the angle between the ship’s heading angle and the angle relative to the target point. Following the principles of the VO method, both \( \theta \) and \( l \) are incorporated into the loss function for controlling the ship’s arrival at the target point. The overall cost function is given by Equation (15), where the optimization process involves continuously minimizing the value of the cost function.

\[
f_{\text{cost}} = (-a \cos(\theta) + b)l \tag{15}
\]

In Equation (15), to control the ship approaching the target point, a relatively simple linear parameter \( l \) is used, ensuring that its gradient remains constant at all times and making it convenient for the analysis of composite functions. To maintain the ship’s heading, the parameters \( a \) and \( b \) in Equation (15) control the strength of heading maintenance. In this context, \( a \) is set to 1/2, and \( b \) is set to 5/2. Under such parameter settings, the gradient controlling the ship’s heading does not exhibit significant changes.

Figure 5. Principle of velocity obstacle method.
After combining the parameters of maintaining the heading and reducing the distance to the destination, the gradient value will vary within a range depending on the heading angle. This plays a significant role in using optimization problems to control the ship for trajectory tracking and implementing trajectory correction.

Figure 6 corresponds to Equation (15) in the original text. During the optimization process of MPC, different combinations of ship speed and heading angle result in varying distances and angles between the ship and the target point. Consequently, this leads to different output values of the cost function. By minimizing the output values of the cost function, the optimal heading angle and speed are determined. Figure 6 illustrates the values of the cost function generated by different distances and angles between the ship and the target point. A higher intensity of red color indicates a larger value of the cost function, whereas a deeper blue color indicates a smaller value. During the optimization process, MPC selects action plans that minimize the cost function values. Visualizing the image allows for a more intuitive observation of the decreasing trend of a cost function, moving from the highest value in the red region to the blue region, especially at the same distance ($l$), and selecting a smaller angle ($\theta$) will result in a smaller cost function value.

Figure 6. Visualization of the cost function.

4. Trajectory Prediction Experimental Design

In this section, we will systematically introduce the specific aspects of experimental design, starting from the selection of datasets and the definition of evaluation metrics. We will elaborate on the datasets we utilized, their characteristics, and elucidate the evaluation metrics chosen to assess model performance. Subsequently, we will discuss the selection of loss functions, outlining how we designed the loss functions to optimize the model during the training process. Finally, we will describe the process of setting model parameters, including aspects such as model architecture and the selection of hyperparameters, to ensure the effectiveness and robustness of the model.

4.1. Datasets and Evaluation Metrics

Currently, there is no publicly available dataset on ship interactions using AIS trajectory data. Therefore, this paper utilizes AIS data collected around the ports of San Francisco and Houston from the Marine Cadastre website [36]. We applied specific criteria to filter trajectory data that align with the task requirements, resulting in 1393 scenarios of multi-ship interactions. From these scenarios, one-tenth is chosen as the test set.

To assess the model’s performance, the authors employ the metrics of Average Displacement Error (ADE) and Final Displacement Error (FDE). Since the model’s output
predicts trajectories as bivariate Gaussian distributions, initial predictions cannot be directly compared with target values. We address this by sampling the initial predicted distribution to obtain trajectory points, subsequently using these points alongside target values to calculate ADE and FDE.

\[
ADE = \frac{1}{N \times M} \sum_{n=1}^{N} \sum_{t=1}^{M} \| \hat{p}_n^t - p_n^t \|^2 
\]

(16)

\[
FDE = \frac{1}{N} \sum_{n=1}^{N} \| \hat{p}_n^t - p_n^t \|^2 
\]

(17)

In Equations (16) and (17), \( p \) represents the positions of the points, \( t \) represents the time point, and \( n \) represents different ships.

4.2. Experimental Conditions

The algorithm in this paper was implemented by using the open-source neural network framework PyTorch (version 3.10). The computer configuration includes 1 GPU (GeForce RTX 3080), CPU (Intel(R) Core (TM) i9-9980HK CPU @ 2.40 GHz), and 64 GB RAM.

4.3. Loss Function

In a given scenario for a ship, where past coordinates are available, the goal is to predict the future trajectory. The true trajectory is represented as \((x_t, y_t) | t \in \{1, \ldots, M\}\), where \((x_t, y_t)\) describes the ship’s positions in 2D space over a period of time. It is assumed that the coordinates follow a bivariate Gaussian distribution. Thus, the true trajectory can be represented as \(P_t \sim (x_t, y_t, \sigma_t, \rho_t)\), where \(\sigma_t\) represents the variance of the coordinate distribution, and \(\rho_t\) represents the correlation between coordinates. The predicted trajectory is denoted as \(\hat{P}_t \sim (\bar{x}_t, \bar{y}_t, \bar{\sigma}_t, \bar{\rho}_t)\). Model training is performed by minimizing the negative log-likelihood, given by the expression in Equation (18).

\[
L = - \sum_{t=1}^{M} \log(P_x(x_t, y_t, \bar{\sigma}_t, \bar{\rho}_t)) \tag{18}
\]

In Equation (18), \((\bar{x}_t, \bar{y}_t)\) represent the mean of the predicted position distribution, \(\bar{\sigma}_t\) represents the variance of the predicted coordinate distribution, \(\bar{\rho}_t\) represents the correlation between the predicted coordinates, \(P\) represents the probability model of a Gaussian distribution, and \(P_t\) denotes the output probability of the probability model under given input conditions.

4.4. Model Parameter Settings

In Table 1, the model is trained with a batch size of 128, employs the Relu activation function, and utilizes the Adam optimizer with a learning rate of 0.001. The model undergoes a uniform training process, encompassing a total of 400 epochs. Additionally, to investigate the impact of different numbers of layers in STGCNN and TXPCNN on model performance, ablation experiments are conducted with the number of layers set to 1, 3, 5, and 7 for both STGCNN and TXPCNN, respectively.

As depicted in Figure 7, we evaluated the model’s performance in scenarios with different sea areas and vessel quantities, attempting to simulate diverse maritime traffic situations in the real world. Through these tests and analyses, we gain a comprehensive understanding of our model’s adaptability and accuracy in different environments, providing robust support for practical applications.
Table 1. Trajectory prediction model parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Numerical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>128</td>
</tr>
<tr>
<td>Activation function</td>
<td>Relu</td>
</tr>
<tr>
<td>STGCNN layer number</td>
<td>(1, 3, 5, 7)</td>
</tr>
<tr>
<td>TXPCNN layer number</td>
<td>(1, 3, 5, 7)</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Number of training epochs</td>
<td>400</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Figure 7. Electronic chart of target sea area.

5. Trajectory Prediction Experimental Analysis

In this section, we delve into a comprehensive analysis of trajectory prediction experiments, covering quantitative analysis, qualitative analysis, and the ship interaction adjacency matrix under different scenarios. Quantitative and qualitative analyses are crucial means to understand the performance and results of the models in trajectory prediction experiments, while the study of ship interaction adjacency matrices can reveal the interaction relationships among ships in various scenarios.

5.1. Trajectory Prediction Quantitative Analysis

In the quantitative analysis experiments in Table 2, the numbers of layers for STGCNN and TXPCNN are set to 1 and 5, respectively. The model performances on two different datasets are compared, and the results are presented in Table 2. Although the same prediction model may exhibit some performance differences on different datasets, the relative performance among different models usually remains consistent across datasets. Based on the average performance metrics, the error ratios of the Social-STGCNN model compared to other models (Social-LSTM, Social-GAN, GAT) are 0.76, 0.48, and 0.62, respectively. These values are in close agreement with the original paper Social-STGCNN [37], where the values are 0.58, 0.73, and 0.94. This indicates that we encountered no significant errors during the process of reproducing and comparing the algorithms. GL-STGCNN, compared to the model with the best performance in the table, RAIN [9], shows an increase of 31.8% in the ADE metric and 16.8% in the FDE metric. According to the last column of Table 2, which shows the time required for the model to complete one prediction, there is no significant difference in computational performance across different models.

The performance tests above only discussed the original model structure, without considering the improvement in model performance by adding the trajectory correction model.
Table 2. The predicted performance and computing performance of the model on different data sets.

<table>
<thead>
<tr>
<th>Model</th>
<th>Near Port of Houston (ADE/FDE)</th>
<th>Near Port of Los Angeles (ADE/FDE)</th>
<th>Average (ADE/FDE)</th>
<th>Inference Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRI</td>
<td>0.0656/0.0658</td>
<td>0.0767/0.0775</td>
<td>0.0711/0.0716</td>
<td>41.5</td>
</tr>
<tr>
<td>Social-Gan</td>
<td>0.0536/0.0578</td>
<td>0.0652/0.0691</td>
<td>0.0594/0.0634</td>
<td>93.4</td>
</tr>
<tr>
<td>GAT</td>
<td>0.0415/0.0482</td>
<td>0.0519/0.0591</td>
<td>0.0467/0.0536</td>
<td>57.2</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.0346/0.0399</td>
<td>0.0423/0.0439</td>
<td>0.0384/0.0419</td>
<td>13.7</td>
</tr>
<tr>
<td>Social-LSTM</td>
<td>0.0326/0.0373</td>
<td>0.0434/0.0479</td>
<td>0.038/0.0426</td>
<td>27.4</td>
</tr>
<tr>
<td>Graph-LSTM</td>
<td>0.0321/0.0354</td>
<td>0.0421/0.0428</td>
<td>0.0371/0.0391</td>
<td>26.8</td>
</tr>
<tr>
<td>MTGCN</td>
<td>0.0267/0.0343</td>
<td>0.0346/0.0354</td>
<td>0.0306/0.0348</td>
<td>39.5</td>
</tr>
<tr>
<td>RAIN-STGCNN</td>
<td>0.026/0.0312</td>
<td>0.036/0.0399</td>
<td>0.031/0.0355</td>
<td>48.9</td>
</tr>
<tr>
<td>RAIN-LSTM</td>
<td>0.0247/0.0276</td>
<td>0.0344/0.0364</td>
<td>0.0295/0.032</td>
<td>21.3</td>
</tr>
<tr>
<td>Social-STGCNN</td>
<td>0.0245/0.0251</td>
<td>0.0337/0.0378</td>
<td>0.0291/0.0314</td>
<td>45.6</td>
</tr>
<tr>
<td>NRI-LSTM</td>
<td>0.0241/0.0261</td>
<td>0.0317/0.0362</td>
<td>0.0279/0.0315</td>
<td>29.1</td>
</tr>
<tr>
<td>RAIN</td>
<td>0.0188/0.0203</td>
<td>0.0221/0.0236</td>
<td>0.0204/0.0219</td>
<td>47.6</td>
</tr>
<tr>
<td>GL-STGCNN</td>
<td>0.011/0.015</td>
<td>0.0167/0.0214</td>
<td>0.0139/0.0182</td>
<td>49.6</td>
</tr>
</tbody>
</table>

From Figure 8, it can be observed that as the number of vessels gradually increases, the model’s prediction accuracy shows a decreasing trend. This is due to the rising complexity of interaction problems as the number of targets increases, leading to a decrease in performance. However, through trajectory correction, the GL-STGCNN demonstrates better performance in scenarios with different numbers of vessels. The more vessels there are, the more significant the improvement in the model’s predictive performance. In scenarios with different numbers of vessels, trajectory correction can enhance the performance of GL-STGCNN by 44.5%.

This indicates that capturing the interactions between vessels becomes more challenging as the number of vessels increases. However, trajectory correction proves to be an effective method for improving trajectory prediction performance.

The ablation experiments in the paper Social-STGCNN indicate that the optimal model performance is achieved when the ratio of STGCNN layers to TXPCNN layers is set to 1:5. In the ablation experiments conducted in this paper on GL-STGCNN, various combinations of STGCNN and TXPCNN layers were explored, resulting in a total of 16 different configurations. The results of the ablation experiments are shown in Figure 9, and through detailed evaluation, it was observed that setting a lower number of TXPCNN layers often leads to a greater improvement in model performance. Specifically, when the number of STGCNN layers is five and the number of TXPCNN layers is one, the model performance reaches its optimum level.

This analysis emphasizes the significance of hierarchical structure configuration in achieving optimal performance in trajectory prediction tasks.
Figure 9. GL-STGCNN model ablation experiments with different layer combinations.

5.2. Trajectory Prediction Qualitative Analysis

The predicted trajectory distributions under different vessel motion modes and scenarios are illustrated in Figure 10. Overall, predicting trajectory distributions solely through GL-STGCNN can better reflect real vessel trajectories but still has some issues. In the straight-line navigation mode, the predicted trajectory distributions of vessel 4 in Figure 10a, vessel 5 in Figure 10b, and vessel 1 in Figure 10f exhibit some deviations from the true trajectories, typically showing a directional offset in the predicted trajectory distribution. Under the turning motion mode, the deviations between the predicted trajectory distributions and the true trajectories for vessel 1 in Figure 10a and vessel 1 in Figure 10c are more significant compared to the deviations under the straight-line navigation mode. This occurrence indicates the challenges faced by neural networks in understanding the motion constraints during vessel maneuvers. Due to the neural network’s difficulty in understanding the constraints of vessel motion, this issue becomes more pronounced in turning motion modes with larger vessel maneuver amplitudes, reflected in the deviation between the predicted trajectory distribution and the true trajectory for vessel 6 in Figure 10f. Additionally, as the number of vessels in the scene increases, the interaction relationships between vessels become more complex. Moreover, the frequency of turning, changing direction, and other maneuvers during navigation increases, leading to a gradual decline in corresponding prediction performance.

Figure 10. Distribution of predicted trajectories under different ship movement modes and ship number scenarios.
5.3. Ship Interaction Adjacency Matrix under Different Scenarios

The ship navigation data in Figure 11 is processed using the proposed GL-STGCNN, resulting in the corresponding ship interaction adjacency matrices in Figure 11, reflecting the interactive effects among ships during navigation.

![Ship interaction matrix extracted by GL-STGCNN in different scenarios.](image)

For the analysis of column weights in Figure 11a, the weight values range from 1.8 to 1.91, indicating similar levels of interaction among ships. However, the column corresponding to ship 2 has the relatively smallest weight, while the column for ship 4 has the relatively largest weight. This is because ship 2 initially occupies a central position in the scene, while ship 4 consistently moves away from surrounding ships.

For the analysis of column weights in Figure 11b, the weight values range from 0.08 to 0.18, suggesting significant differences in interaction levels among ships. The column for ship 1 has the relatively smallest weight, while the column for ship 4 has the relatively...
largest weight. This is due to ship 1 initially moving away from other ships, while ship 4 consistently approaches surrounding ships.

For the analysis of column weights in Figure 11c, the weight values range from 0.12 to 0.17, indicating moderate differences in interaction levels among ships. The column for ship 4 has the relatively largest weight, as ship 4 has the highest initial speed in the scene and consistently approaches ships 2, 3, and 5.

For the analysis of column weights in Figure 11d, the weight values range from 0.06 to 0.17, indicating significant differences in interaction levels among ships. The columns for ships 2 and 4 have the relatively smallest weight values. This is because ship 2 is initially farthest from other ships and follows a non-intersecting course, while ship 4 travels in a straight line without conflicting paths with surrounding ships.

For the analysis of column weights in Figure 11e, the weight values range from 0.07 to 0.15, indicating significant differences in interaction levels among ships. The columns for ships 4 and 6 have the relatively largest weight values, as ship 4 begins turning towards ships 3 and 5, and ship 6 starts turning towards ships 1 and 5.

For the analysis of column weights in Figure 11f, the weight values range from 0.07 to 0.17, indicating significant differences in interaction levels among ships. The columns for ships 3, 5, and 6 have the relatively largest weight values, as ships 3, 5, and 6 continuously maneuver and turn, with the smallest distances between them.

The analysis above utilizes weight analysis to quantify the interaction levels among different vessels, providing in-depth insights into the relationships between ships. This type of analysis contributes to understanding the ship interaction adjacency matrix output by the model, offering crucial insights for optimizing model performance and adapting to diverse scenarios.

The ship navigation data in Figure 10 is processed using Social-STGCNN, resulting in the corresponding ship interaction adjacency matrices shown in Figure 12, reflecting the interaction effects among ships during navigation. Since Social-STGCNN models the interaction effects between ships based on their distances, without considering relative speed and position, it faces challenges in capturing potential ship behaviors such as future turns and evasive maneuvers. For instance, in Figure 12b, Social-STGCNN incorrectly assumes mutual influence between ships 2, 3, and 4 and mutual influence between ships 1 and 5. However, in the actual scenario depicted in Figure 12b, ships 1 and 5 move away from each other, and their routes do not interfere.

In Figure 12e, due to the relatively uniform distribution of distances between ships, Social-STGCNN concludes that there is no significant difference in the interaction effects between ships. Nevertheless, in the real scenario presented in Figure 10e, ships exhibit frequent head-on navigation, an aspect not considered by Social-STGCNN when modeling the interaction adjacency matrix.

In Figure 12f, Social-STGCNN believes that there is little difference in the interaction effects between ships. However, in this complex scenario with a large number of ships and intricate maneuvering behaviors, Social-STGCNN incorrectly assumes little difference in mutual interactions among all ships.

Through the analysis of the content in Figures 11 and 12, it can be inferred that GL-STGCNN, compared to Social-STGCNN, excels in handling complex ship interaction behaviors. GL-STGCNN achieves this by extracting the ship interaction adjacency matrix to learn the relative positions and velocities between ships, providing a more accurate representation of the interaction effects among ships.

The corrected results of the predicted trajectory distributions under different ship motion patterns and ship quantity scenarios are illustrated in Figure 13, where the number of ships gradually increases and the performance of the corrected predicted trajectories does not exhibit a significant decline; overall, they demonstrate effective correction.
In the motion pattern of long-distance straight-line navigation in Figure 13a, the deviations between the corrected predicted trajectories and the true trajectories of ships 4 and 5 in Figure 13b and 1 in Figure 13f are further reduced. In the turning navigation mode in Figure 13a, the corrected predicted trajectories for ship 1, turning navigation mode in Figure 13c ship 1, and turning navigation mode in Figure 13f ships 2 and 7, although exhibiting some deviation from the true trajectories, accurately reflect the characteristic motion features of ships in turning mode after correction. In the turning motion mode, due to the significant deviation between the predicted trajectory distribution of ship 6 in Figure 10f and the true trajectory, the corrected predicted trajectory for ship 6 in Figure 13f does not accurately represent the true motion of the ship.

The effectiveness of the trajectory correction method proposed in this paper largely depends on the reliability of the predicted trajectory distribution. The more accurate and reliable the results of the predicted trajectory distribution, the more significant the correc-
tion effect after trajectory correction. This is because the accuracy of the predicted trajectory distribution directly influences the starting point, intermediate points, and endpoints during trajectory correction. Therefore, in trajectory prediction tasks, improving the accuracy of the initial predictions is crucial for enhancing the final correction effect.

Figure 13. Corrected predicted trajectories under common sailing scenarios with different numbers of ships.

6. Discussion

Based on the quantitative analysis results, GL-STGCNN outperforms previous trajectory prediction methods, achieving a 31.8% improvement in the ADE metric and a 16.8% improvement in the FDE metric. After trajectory correction using MPC, the performance of trajectory prediction improved by 44.5%. Qualitative analysis conducted in different ship motion modes and scenarios indicates a certain level of rationality in the ship interaction adjacency matrix modeled by GL-STGCNN in navigation scenarios. We observed variations in the predictive performance of GL-STGCNN across different maritime areas and motion modes. Moreover, the effectiveness of MPC trajectory correction is highly dependent on the reliability of the predicted trajectory distribution. The more accurate and reliable the initial predicted trajectory distribution, the more significant the improvement after trajectory correction. This may be attributed to the direct influence of the accuracy of the predicted trajectory distribution on the starting, intermediate, and ending points in trajectory correction.

7. Conclusions

In this study, we propose the GL-STGCNN multi-ship trajectory prediction model to enhance trajectory prediction accuracy in scenarios involving multiple ship interactions. By incorporating an adjacency matrix extraction module and leveraging MPC for trajectory correction, we successfully improve the accuracy and rationality of trajectory prediction. GL-STGCNN outperforms previous trajectory prediction methods, achieving a 31.8% improvement in average ADE and a 16.8% improvement in average FDE. Furthermore, trajectory correction using MPC results in a 44.5% performance enhancement based
on the initial predicted trajectory distribution. However, we observed variations in the trajectory prediction performance of GL-STGCNN across different datasets and motion modes of ships.

Future research efforts could focus on modeling more reasonable ship interaction adjacency matrices using deep learning approaches to achieve better results in diverse environmental settings. Additionally, selecting superior trajectory prediction models is crucial for enhancing trajectory correction effectiveness. Our study introduces a novel approach to ship trajectory prediction, aiming to provide valuable insights for practical applications.

**Author Contributions:** Y.W. and W.L.: conceptualization, writing, and methodology. G.Z. and Y.S.: data curation and visualization. W.Y.: preparation and modification. All authors have read and agreed to the published version of the manuscript.

**Funding:** This paper was supported by the National Natural Science Foundation of China [52171308]; the Natural Science Foundation of Fujian Province, China [2022[01333]; and the Natural Science Foundation of Fujian Province, China [2022[01813].

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

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

**Abbreviations**

The following abbreviations are used in this manuscript:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADE</td>
<td>Average Displacement Error</td>
</tr>
<tr>
<td>AIS</td>
<td>Automatic Identification System</td>
</tr>
<tr>
<td>FDE</td>
<td>Final Displacement Error</td>
</tr>
<tr>
<td>GAT</td>
<td>Graph Attention Network</td>
</tr>
<tr>
<td>GAN</td>
<td>Generative Adversarial Network</td>
</tr>
<tr>
<td>GL-STGCNN</td>
<td>Graph Learning-Spatio-Temporal Graph Convolutional Neural Network</td>
</tr>
<tr>
<td>Goal-GAN</td>
<td>Goal-Generative Adversarial Network</td>
</tr>
<tr>
<td>Graph-Lstm</td>
<td>Graph-Long Short-Term Memory Network</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short-Term Memory Network</td>
</tr>
<tr>
<td>MMG</td>
<td>Mathematical Model Group</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>MTGCN</td>
<td>Multitask Temporal Graph Network</td>
</tr>
<tr>
<td>NRI</td>
<td>Neural Relational Inference</td>
</tr>
<tr>
<td>RAIN</td>
<td>Residual Attention Network</td>
</tr>
<tr>
<td>STGCNN</td>
<td>Spatio-Temporal Graph Convolutional Neural Network</td>
</tr>
<tr>
<td>Social-GAN</td>
<td>Social-Generative Adversarial Network</td>
</tr>
<tr>
<td>Social-STGCNN</td>
<td>Social-Spatio-Temporal Graph Convolutional Neural Network</td>
</tr>
<tr>
<td>Social-Lstm</td>
<td>Social-Long Short-Term Memory Network</td>
</tr>
<tr>
<td>TGN</td>
<td>Temporal Graph Network</td>
</tr>
<tr>
<td>VO</td>
<td>Velocity Obstacle</td>
</tr>
</tbody>
</table>

**References**


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.