Self-Attention Factor Graph Neural Network for Multiagent Collaborative Target Tracking

Cheng Xu, *Member, IEEE*, Ran Su, Ran Wang, *Graduate Student Member, IEEE*, and Shihong Duan, *Member, IEEE*

***Abstract*—Collaborative target tracking is an essential task in positioning systems, particularly in environments characterized by high dynamics, multisource heterogeneous data, and interactive multiagent scenarios. The challenge in such networks lies in the direct utilization of multisource heterogeneous data as feature input for models. Additionally, the presence of high-dynamic time-series data complicates the extraction of dependencies by the models. To address these issues, we introduce a novel approach that integrates a factor graph-based data fusion method with a graph neural network. This combination is designed to uncover potential dependencies between time-series data and positional information within dynamic networks. Furthermore, we employ a self-attention mechanism, enabling distance-agnostic autonomous selection of complex network features. This innovation allows the model to achieve enhanced accuracy performance while simultaneously reducing computational costs. We validated our approach through simulation experiments. The results demonstrated the method's effectiveness in fusing and selecting multisource heterogeneous information within collaborative networks. It also excelled in identifying potential relationships between feature information and positional data, showcasing the robustness and applicability of our proposed solution in challenging collaborative target tracking environments.**

***Index Terms*—Collaborative tracking, factor graph, factor graph neural network, graph neural network (GNN), multiagent network.**

I. Introduction

With the advent of the Internet of Things (IoT), accurate localization has become a cornerstone in the labyrinth of smart devices that pervade our daily lives. While the global positioning system (GPS) remains the bedrock of outdoor localization due to its high precision and cost-efficiency, its effectiveness is markedly reduced indoors, where satellite signals are scarce and prone to obstruction. In the quest for reliability within indoor and complex environments, alternative wireless positioning technologies, such as time of arrival (TOA) and time difference of arrival (TDOA), have risen to prominence [1], [2]. These methods, however, grapple with occlusions and non-line-of-sight conditions that are endemic to cluttered, dynamic collaborative networks [3]. Such conditions frequently degrade the accuracy of positioning, presenting a significant challenge in densely packed IoT ecosystems.

Compounding these challenges are the limitations of traditional inertial systems. While they offer notable instantaneous accuracy at a lower cost, their reliance on integration means that any drift in accelerometer or gyroscope readings can lead to increasing errors over time [4]. The inherent multisource heterogeneity of sensor data in dynamic IoT networks further complicates the landscape, posing a formidable barrier for traditional positioning methodologies to harness this wealth of information effectively.

In complex and highly dynamic collaborative networks, achieving effective collaborative distribution, data screening, and fusion poses significant challenges in network positioning. The method of multisource heterogeneous information fusion effectively harnesses data from various sensors, circumventing environmental or target perception limitations, thereby enhancing the system's external perception capabilities [5]. Wymeersch et al. [6] implemented a Bayesian approach based on message propagation for processing multisource heterogeneous data to determine agent locations. However, the application of this method to highly dynamic networks, especially those involving time-series data, is limited due to its high computational demand and the prerequisite of prior knowledge, such as offline labeling and positioning networks [7]. To transcend these constraints in current location technologies, there is a pressing need for a method capable of fusing multisource data and incorporating time-series information.

In recent years, graph-structure-based algorithms have gained widespread use in various graph-related learning tasks, including node classification, link prediction, and graph classification [8], [9], [10], [11]. Concurrently, the use of graph structures has been extended to positioning [7]. Graph structures, compared to traditional methods, can uncover additional node interrelations, thereby revealing more potential interaction information. In the domains of artificial intelligence and neural networks, stochastic models are typically represented as Bayesian networks or Markov random processes [12]. However, in real-world scenarios, time-series data often constitute incomplete observations of intricate underlying dynamic processes with high-dimensional states

Manuscript received 29 January 2024; accepted 16 February 2024. Date of publication 27 February 2024; date of current version 9 October 2024. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62101029; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515140071; and in part by the China Scholarship Council Award under Grant 202006465043 and Grant 202306460078. *(Corresponding author: Shihong Duan.)*

The authors are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China, and also with the Shunde Innovation School, University of Science and Technology Beijing, Foshan 528223, China (e-mail: xucheng@ustb.edu.cn; 2205372000@qq.com; wangran423@foxmail.com; duansh@ustb.edu.cn).

Digital Object Identifier 10.1109/JIOT.2024.3370830

that are not directly observable. For instance, human motion capture data provide specific marker positions, reflecting the complex kinematic and dynamic constraints of numerous joint angles [13]. Traditional graph structures, however, often fall short in depicting these dependencies.

Multiagent localization is a multifaceted process that encompasses the interaction between agents and the potential unobstructed variable information of these agents [14]. Even with the integration of multisource data, extracting useful information from large data sets remains a significant challenge. Mirowski and LeCun [13] introduced a factor graph approach tailored for time-series data to elucidate the relationship between time series and hidden states. Unlike traditional graph structures, factor graphs are extensively utilized to model independent random variables. For instance, approximate inference algorithms are applied in probabilistic graph models to reason about specified probability graphs [15], enabling a more accurate representation of the relationships between state variables and observable variables. In collaborative positioning scenarios that include timing information, factor graphs can ascertain higher order potential dependencies, thereby facilitating the analysis of correlations between time series and data features [16]. However, in multiagent cooperative networks, defining an effective prior model is challenging, often resulting in suboptimal approximate inference outcomes. Thus, efficiently processing data features in highly dynamic complex networks to enable automatic learning of inferential relationships and feature screening poses a formidable challenge.

Addressing this issue, Gori et al. [17] proposed the graph neural network (GNN) method, which learns potential variables and reasoning processes from data. This method only requires the provision of a graph structure with dependency relationships to better explore information within real data. Moreover, Zhang et al. [16] developed the factor GNN method, extending factor graphs to GNNs. This approach effectively tackles point cloud data classification but still struggles with the screening of complex input data. Veličković et al. [10] introduced a graph attention method that filters data by assigning weights to different graph nodes. However, this method requires recalculating graph attention when the graph structure changes, making it less suitable for the real-time demands of dynamic networks.

Furthermore, self-attention mechanisms have become crucial in various tasks, enabling the modeling of dependencies regardless of the distance between input and output sequences [18], [19], [20]. In terms of computational complexity, self-attention offers parallel computation advantages compared to networks like RNNs. This mechanism processes the input sequence into queries, keys, and values, subsequently deriving a weight coefficient by calculating the similarity between the query and the key. The final output is a weighted summation of the values and these weight coefficients. The parallel nature of this computation primarily lies in the similarity calculation between queries and keys, achieved through matrix multiplication. Since matrix multiplication is highly parallelizable, it significantly accelerates model computation. When the sequence length n is less than the representation dimension d, the self-attention layer outperforms recurrent layers, making it more efficient for short-sequence timing information [20]. In practical positioning scenarios, we often use timing information close to the current moment for computation. Therefore, the self-attention mechanism can address data screening challenges while reducing computational costs in dynamic networks.

In this article, we propose a collaborative localization model based on factor graph self-attention. This model leverages the synergistic integration of self-attention mechanisms and factor graphs within a GNN framework, enabling the effective screening of features from multiagent nodes. This approach addresses the challenges of multisource heterogeneous data fusion and screening and realizes multiagent localization through the use of factor GNNs to learn the reasoning process of data features. The primary contributions of our research are as follows.

1) *General Factor Graph-Based Framework for Collaborative Target Tracking:* We introduce a comprehensive framework that incorporates factor graphs within the realm of GNNs. This framework is specifically designed to navigate the unique challenges found in collaborative target tracking, such as dynamic environments and intricate agent interactions. It marks a significant advancement over conventional cooperative network approaches by enabling adaptive learning and meeting the real-time demands of dynamic scenarios.
2) *Advanced Data Fusion Method Utilizing Factor Graphs:* Our methodology presents an innovative data fusion approach, transforming heterogeneous data from a variety of sources into a structured graph format. This facilitates efficient processing of dynamic and complex data sets and enables the automatic discovery of complex inferential relationships, thus surpassing the constraints typically associated with traditional graph models.
3) *Integration of Self-Attention Mechanism in Factor GNNs:* We have successfully integrated a self-attention mechanism into the factor GNN architecture. This enhancement bolsters the model's proficiency in processing timing information from agents and efficiently filtering interaction features. Consequently, our model adeptly extracts pertinent data features within highly dynamic and complex network environments, assisting collaborative agents in actively selecting perceptual data to optimize their pose estimation. This functionality is particularly beneficial in scenarios with uncertain network conditions.

The remainder of this article is organized as follows. Section II introduces the related work and research, elaborating in detail on the merits of this study. Section III describes the localization method of factor GNNs based on self-attention, followed by experimental results and discussions in Section IV. Section V concludes this article with a comprehensive summary.

II. Related Work

A. Cooperative Localization

Collaborative localization involves both the self-observation of agents and the measurement of interactive information

TABLE I
STRENGTHS AND LIMITATIONS OF VARIOUS KINDS OF WORK IN COLLABORATIVE TARGET TRACKING APPLICATIONS

Methods	Strengths	Limitations
TOA	High precision	High cost, easy to block interference
MLE	Easy to implement	Noise probability distribution matching results in performance degradation
GNN	Large scale, high stability	The dynamic environment is not satisfied
BP	Obtain high level dependencies for dynamic environments	Require prior knowledge
Ours	Obtain high order dependencies in dynamic environments with high accuracy and low cost	Certain agent scale

between them. Current methods for localization primarily fall into two categories: optimization-based and learning-based approaches. Optimization-based methods include maximum-likelihood estimation [21], [22], the least-squares (LSs) method [6], multidimensional scaling [23], [24], [25], and Bayesian message propagation [6], [26], [27], among others. A comparative analysis of the methods presented in this article and the pros and cons of existing methods are delineated in Table I.

The maximum-likelihood estimation method treats noise as a definitive probability distribution [28], [29], but this can lead to significant performance degradation in the event of incorrect matching [7]. In the realm of machine learning, Xie and Yang [30] developed an indoor positioning method that combines random forest and deep learning. This method employs deep learning to train the channel propagation model offline and determine the orientation of agents during the online stage. Yan et al. [7] introduced a location method based on GNNs, which addresses the computational performance challenges in large-scale networks and provides enhanced accuracy and stability for static localization of large-scale agents. However, in highly dynamic cooperative networks where the agent target is typically dynamic and includes time series of historical moments, the GNN method struggles to effectively utilize time-series information.

Addressing this gap, Liu et al. [31] proposed a dynamic representation learning framework for network embedding on large-scale attributed networks. This approach models time-varying network features for dynamic attribute network embedding. In the offline learning stage, node embedding is generated by discovering potential node attributes and network structure to guide the learning subspace. In the online learning stage, dynamic network features are processed through timely and incremental updating of node embedding. In terms of heterogeneous data, Wang et al. [32] introduced the HAN method, incorporating the attention mechanism into the heterogeneous GNN. This method maps node features of different dimensions to a uniform dimension, effectively addressing the challenge of heterogeneous data processing and providing a clear explanatory framework. Building upon this, our paper adds a self-attention method to the factor GNN framework, thereby effectively resolving the heterogeneous data processing challenge in the cooperative localization environment.

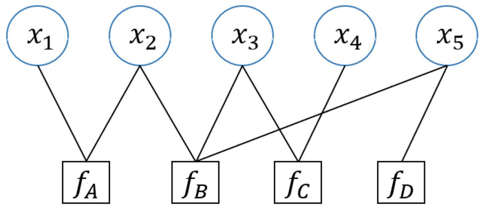

Fig. 1. Factor graph.

Addressing the complexities of high-dynamic environments, Patwardhan et al. [33] introduced a planning method based on confidence propagation, delving into robot planning challenges in high-speed, congested traffic settings. This method particularly addresses the issue of heterogeneous data in dynamic environments. By categorizing complex data from dynamic settings into distinct factor nodes according to their sources, the approach facilitates effective path planning. To capture long-term dynamic dependencies effectively, it is imperative for a model to maintain an internal state that adheres to dynamic constraints. To address this requirement, the concept of a dynamic factor graph has been proposed. In this context, factor graphs are applied to time-series data to map out the dependency relationships between state variables and observable variables, thereby unveiling higher order dependency relationships. Building on this foundation, Murai et al. [34] proposed a dynamic factor graph localization method that utilizes Gaussian confidence propagation. They developed a methodology that employs probabilistic factor graphs for perception and state estimation. By adopting Gaussian confidence propagation as the inference algorithm, this method enables agents to accurately estimate and adapt to dynamically collaborative scenes through self-organizing communication.

B. Factor Graph

Factor graphs, as shown in Fig. 1, are a class of graphical models used to infer tasks in probabilistic models by using graphs to represent dependencies between variables. The factor graph $G = (V, F, A)$ of bipartite graph is defined, which contains a group of variable nodes V, a group of factor nodes F, and a group of undirected edge A representing the connection relationship between variable nodes and factor nodes. Variable nodes represent hidden random variables and are usually represented by circles. Factor nodes represent probability-based dependencies between variable nodes, usually in square form.

In the graph, every node $i \in V$ is associated with a random variable x_i, and every node $j \in F$ is associated with a function f_i. There is an edge connecting nodes f_i to nodes x_i if and only if the factor node f_i depends on the variable node x_i. For example, a function $f(x_1, x_2, x_3, x_4, x_5)$ with the sum of variable nodes x_1, x_2, x_3, x_4, and x_5 factors the function into a product form

$$f(x_1, x_2, x_3, x_4, x_5) = f_A(x_1, x_2) f_B(x_2, x_3, x_5) f_C(x_3, x_4) f_D(x_5) \tag{1}$$

wherein, it includes factor node $F = \{f_A, f_B, f_C, f_D\}$ and variable node $V = \{x_1, x_2, x_3, x_4, x_5\}$.

Over the past few decades, factor graphs have gained widespread use in coding and stochastic modeling for defining probability graph models and modeling dependencies between random variables. To address coding challenges, [35] introduced the sum–product algorithm based on belief propagation. This algorithm effectively solved the decoding problem of LDPC codes, markedly enhancing their error correction capabilities. The sum–product algorithm [36], a message-passing algorithm, computes marginal distributions through a series of locally computed messages. It is adept at accurately or approximately calculating global functions [37]. Tanner [38] employed graphs to describe and generalize this encoding method, introducing the minimum sum algorithm. In recent years, factor graphs have emerged as an effective method for multisource information fusion in collaborative localization. For instance, Von Stumberg et al. [39] proposed a factor graph-based approach that addresses computational redundancy and asynchronous delay in multisource information processing, offering the advantage of plug-and-play functionality.

However, a significant challenge in practical scenarios is that we often obtain only an approximate value of the true distribution of the probability graph model, resulting in suboptimal outcomes. Additionally, message propagation technology, integral to factor graph methodologies, encounters limitations within cyclic graph structures. It is generally limited to calculating approximate values of the ideal posterior distribution, which can impede algorithm convergence [40]. Consequently, there has been growing interest in combining message propagation technology with graph networks in the coding field and classification tasks [16], [33], [40]. Satorras and Welling [40] proposed a method that merges traditional message propagation with GNNs, applying it to LDPC coding to address complex noise issues in Gaussian channels. Additionally, Zhang et al. [16] integrated the factor graph model with graph networks for classifying point cloud data. By utilizing graph network algorithms, models can learn potential variables and inference processes from data, requiring only the graph structure to provide variable dependencies. While factor graphs excel at capturing higher order potential dependencies in scenarios involving time-series information, their direct application to target tracking is hindered by the common necessity to approximate the true distribution of the probability graph model in real-world scenarios, leading to less than optimal results. Moreover, the message propagation technique faces considerable limitations in cyclic graph structures, typically computing only approximate values of the ideal posterior distribution, which may prevent algorithm convergence and thus diminish the effectiveness of factor graph approaches in meeting the specific requirements of target tracking. Therefore, the integration of factor graph models with graph networks can enable models to actively learn the reasoning process of agents, discover potential motion features from time series and agent characteristics, and consequently achieve more accurate target localization.

C. Attention Mechanism

In the domain of collaborative positioning, the complexity of a vast array of input data features and the accumulation of time-series information pose challenges in computational performance and associated costs for models. To address these challenges effectively, the attention mechanism has emerged as a standard in many sequential tasks. Its key advantage is the ability to process inputs of any size and focus on the most relevant features within those inputs [10]. The self-attention mechanism connects information from different locations within the same sequence, facilitating the computation of sequence representations. This approach has seen successful applications in various tasks, including reading comprehension, summary generation, and text implication, by enabling the selection of effective features from massive data sets, thus enhancing model performance while reducing computational costs [41], [42], [43], [44].

Graph attention mechanisms have gained widespread use in graph structures. Veličković et al. [10] introduced an attention mechanism that filters data by assigning weights. However, the incorporation of double computation and time series in dynamic graph structures tends to increase computation time. Methods such as those proposed by Kosaraju et al. [45], Alahi et al. [46], and Gupta et al. [47] initially employ the long short-term memory model [48] to capture each agent's track features over time. These features are then fed into an interactive model GNN [8] to identify interactions between agents. However, such methods can lead to computational resource wastage, and issues like gradient vanishing and exploding can adversely impact the model's performance and accuracy.

To overcome these challenges, Schwartz et al. [49] proposed applying the graph attention mechanism to factor graphs for classifying image-based dialog. This innovative approach has paved new paths for utilizing attention mechanisms within factor graphs. Inspired by these methods, this article applies the self-attention mechanism to the factor graph network. This application enables the screening and extraction of features from accumulated time-series data and the selection of effective features from numerous inputs, thereby enhancing the performance of the collaborative localization model.

III. Self-Attention Factor Graph Neural Network

In this section, we define the symbolic definition of data items in the collaboration localizatiion scenario, and then the data fusion of multisource heterogeneous data in the highly dynamic network is introduced based on the factor graph. Factor graphs help us capture dependencies between information. We further extend the factor graph into factor GNN, and realize the autonomous feature screening and learning inference process of agents based on self-attention, so as to achieve cooperative localization, as shown in Fig. 2.

A. Problem Formulation

This article assumes that the collaboration scenario is a 2-D wireless network space, assume the current time is

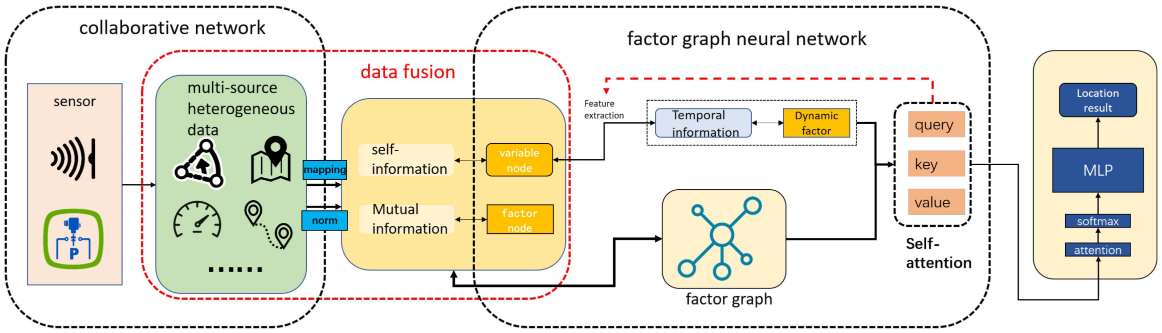

Fig. 2. Self-attention factor GNN.

$T(T > 0)$, $S_a = \{1, 2, \ldots, N\}$ represents the directory of the agent collection, where N is the number of agents. $X^t = \{x_1^t, x_2^t, \ldots, x_N^t\}$ represents the joint state of all N agents in $t(t < T)$, including agent position p, velocity v, orientation angle a, and interagent distance data d; $x_i^t = \{c_1, c_2, \ldots, c_M\}$ represents agent measurement data at time T, including agent velocity v, orientation angle a, and distance between agents d at the current time. Let $Y^T = \{y_1^T, y_2^T, \ldots, y_N^T\}$ represent the predicted state of all agents at time T. The objective of this article is to locate the position information of all agents at the current moment based on the measurement data of multiple agents at the current moment X^T and the observation state of historical moment in the collaborative network $X_{\text{his}} = \{X^t\}_{t=0}^{T-1}$, expressed as $\rho_\sigma(Y^{T+1}|X_{\text{his}}, x^T)$.

B. Multisource Data Fusion

Aiming at the problem that multisource heterogeneous data of agents cannot be directly used as model input in collaborative scenarios, this article proposes a solution. First, we define the interaction characteristics of agents as self-information factor nodes, the characteristics of agents themselves as variable nodes, and the hidden feature information obtained based on self-attention time-series data as dynamic factor nodes. Then, we set up the form of factor graph, fused multisource heterogeneous data, and carried out feature mapping. By this means, we transform the raw data into agent characteristics acceptable to the model. In this way, we can effectively process multisource heterogeneous data in collaborative scenarios and improve the accuracy and reliability of the model.

For the measurement data $x_i^T = \{c_1, c_2, \ldots, c_M\}$ obtained by the sensor at time T of the multiagent, according to the data dependency relationship, it is defined as the measurement data between agents obtained by communication sensors, namely mutual information X_{mut}^T, and the measurement data of the agent itself obtained by the sensor inside the agent, namely self-information X_{self}^T. Where, the self-information $X_{\text{self}}^T = \{X_i\}_{i=0}^N$ only represents the data of agent i, including the acceleration information and orientation angle of agent i at time T, which is defined as a variable node in this article. Mutual information $X_{\text{mut}}^T = (X_{ij})_{i=0,j=0}^{i,j=N}$ contains the ranging information between agent i and agent j at time T, which is dependent on the data of agent ij at a single time, which is defined as an internal factor node in this article. The original data obtained by these sensors cannot be calculated directly

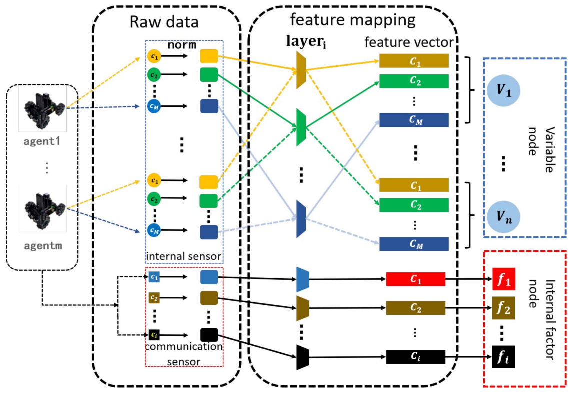

Fig. 3. For multisource heterogeneous raw data, different feature mapping layers are used to transform it into standard input features, and the data sources are classified into different nodes of factor graph.

as the input of the model due to the heterogeneity of the data resulting in different feature dimensions. Therefore, for different kinds of feature data c_i in x_1^t, this model uses multiple feature mapping layers φ_i and standardized functions θ_i to transform data features of different dimensions into the same d-dimension feature vector C_i, and then uniformly process them through standardized functions.Finally, all feature vectors are mapped to a d-dimensional feature space for calculation, as shown in Fig. 3.

Sensor data c_i in collaborative network contains speed information, angle information, and distance information, respectively. According to the data dimensions of different structures, this article defines the feature mapping function

$$\varphi_{\text{i}} \leftarrow \phi_i\left(w_i^{D_i \times d}, d\right) \tag{2}$$

where $w_i^{D_i \times d}$ is the network parameter. Feature mapping function can transform multisource heterogeneous data into unified dimension input features, which is convenient for model calculation. However, due to the different input data units and orders of magnitude of different structures, direct input will lead to poor performance of the model. In order to deal with this problem, this article obtained k samples in simulation scenarios for data in collaborative scenarios, and defined standardized functions using Gaussian distribution

$$\theta_i \leftarrow \frac{Q - \sum_{i=1}^{k} Q_i}{\sqrt{k} \times \sqrt{\sum_{i=1}^{k}\left(Q_i - \frac{\sum_{i=1}^{k} Q_i}{k}\right)^2}} \tag{3}$$

where Q_i is sample data. According to the defined feature mapping function and standardized function, the multisource heterogeneous data fusion function of this model is as follows:

$$C_i = \theta_i\left(\varphi_i(c_i, {w_i}^{D_i \times d}, d)\right). \tag{4}$$

According to different data dependencies, this model takes the interaction features between agents extracted from data as the internal factor node F_i of the factor graph, and takes the features of agents themselves as the variable node V of the factor graph. In addition, the model represents the communication relationship between agent i and agent j as an adjacency matrix A_{ij}

$$A_{ij} = \begin{cases} 0, \text{ no communication between agents } ij \\ 1, \text{ communication between agents } ij. \end{cases} \tag{5}$$

Then, the factor graph at time t is expressed as $G = (V, F_i, A)$. Such representation makes it easier for the model to deal with the communication relationship between agents, so that it can calculate and predict more accurately.

In the cooperative network of agents, the position of agents at the next moment may be affected by the historical motion state of agents. Therefore, in order to obtain the motion hiding state of the agent from the historical characteristics of the agent, we introduce dynamic factor nodes into the factor graph. This node connects the factor graph variable node of the current moment and the agent variable node of the historical moment to obtain the characteristics of the historical state of the agent. Specifically, for the agent characteristics $c_{\text{his}} = \{c^{T-k+1}, c^{T-k+2}, \ldots, c^T\}$ before the observed time T, a variable time length $p(p > 0)$ is selected to obtain the agent historical state characteristics $c_p = \{c^{T-p+1}, c^{T-p+2}, \ldots, c^T\}$, where $c^T = \{c_1^t, c_2^t, \ldots, c_N^t\}$ represents the historical data characteristics of all N agents at the time t. We can get the historical feature matrix mat_{his} by the column matrix of all the features of historical moments c^T. In order to learn the relationship between historical states and hidden motion states, this article adopts self-attention network to obtain hidden state S

$$S = \text{self}_{\text{attention}}(\mu_i) = \text{softmax}\left(\frac{{\mu_i}^T \mu_i}{\sqrt{d}}\right)\mu_i \tag{6}$$

$${\mu_i}^{(i \in 1,2,3)} = W_i(\omega_i, \text{mat}_{\text{his}}) \tag{7}$$

where W_i is trainable network, ω_i is the parameter of attention network, and S is the hidden state of timing information of variable nodes, as shown in Fig. 4.

In Section III-B, we elaborate on the construction and utility of the factor graph within our algorithm, highlighting its contributions to enhancing data processing efficacy and model accuracy. The factor graph encapsulates the dynamic factor nodes, which correlate with the variable nodes representing the model's hidden states S, across adjacent time points, thus forming a continuous time factor graph. Through this construction, our model transforms cooperative network data into input features that are conducive to the model's requirements, capturing essential feature dependencies. This factor graph creation process not only refines the data representation but also optimizes the model's performance.

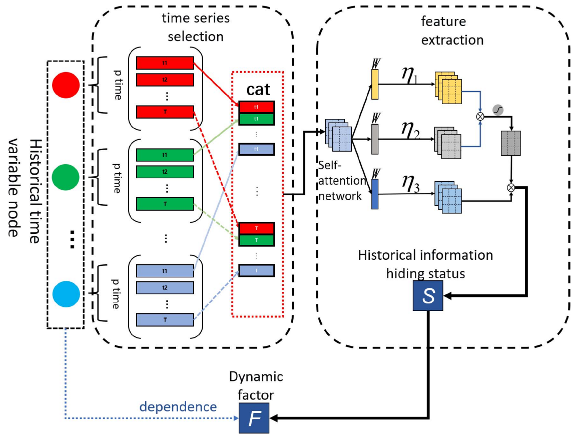

Fig. 4. For the obtained variable nodes, the hidden features of the history state are extracted from the attention network.

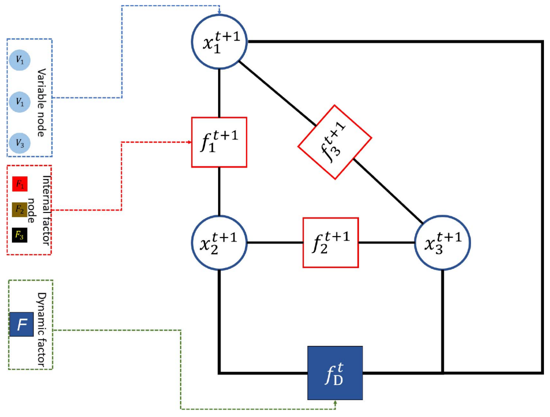

Fig. 5. Established factor diagram.

C. Establishment of Factor Graph Neural Network

Through the above multisource heterogeneous data fusion process, we can establish a factor graph with continuous moments and obtain the connection dependence relationship between nodes. However, it is often difficult to obtain a priori model of feature inference in highly dynamic collaborative networks. Therefore, we introduce self-attention neural network to learn the feature inference relationship, and enable agents to actively select and screen the most beneficial feature information.

Fig. 5 introduces the process of establishing factor graph in this article by taking three agents as an example. For variable node V and internal factor F_i node obtained through information fusion at time t in the previous section, x_i^t is used in this article to represent characteristic information of variable node V of agent i at time t. f_{ij}^t represents the internal factor node F between agent i and agent j at time t, represents that the variables x_i^t and x_j^t are dependent, Then, there is an edge connecting the variable node V with the internal factor node F_i. For the dynamic factor node F_D obtained from the timing information of variable node. In this article, f_D^t represents the dynamic factor of the agent obtained according to the historical characteristic state at time t, which is dependent on variable nodes of the factor graph at adjacent moments. Then, there is

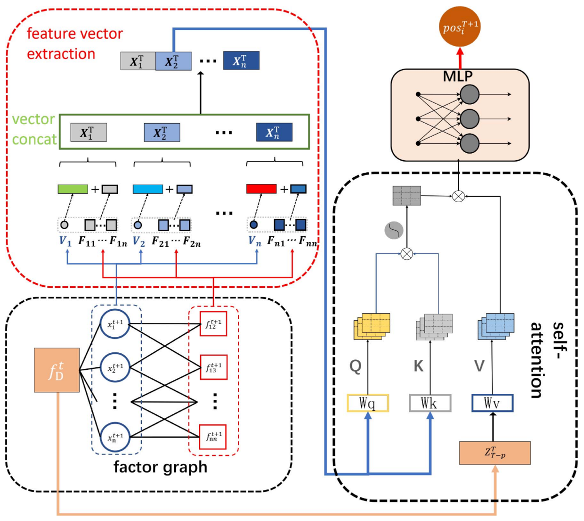

Fig. 6. By establishing factor GNN and based on self-attention, cooperative localization is realized.

an edge connecting all variable nodes V and dynamic factor nodes F_D.

For factor graph $G = (V, F_i, F_D, A)$. In order to obtain the motion trend at time $t+1$, it is necessary to take the observation state at the current time as the query object and search for similar motion trend in the state at the historical time. Therefore, this article takes the input characteristics and mutual information of the current moment V, F_i as the query value, and the dynamic factor of the historical moment F_D as the index and the queried value for self-attention calculation, as shown in Fig. 6. For the characteristic information c_i of variable node i and the internal factor nodes $c_{F_i} = \{c_{F_i}^1, c_{F_i}^2, \ldots, c_{F_i}^{m_i}\}$ of self-information connected to it, m_i is the number of factor nodes connected to variable node i. Calculate the query value of the input feature at time $t+1$, and obtain the query value through matrix splicing and multilayer neural network

$$Q = W^{(1)}\Big(\omega_Q, W^{(2)}(\text{cat}(c_i, c_{F_i}^1, c_{F_i}^2, \ldots, c_{F_i}^{m_i}), \omega_Q)\Big) \quad (8)$$

where $W^{(1)}, W^{(2)}$ is the trainable network, ω_Q is the network parameter, and *cat* is the eigenmatrix concat function. For the dynamic factor node feature c_{F_D} of the hidden state S of historical motion, as the queried value at time $t+1$, the index and value of self-attention network are calculated, respectively

$$K = W^{(3)}(\omega_K, c_{F_D}) \quad (9)$$
$$V = W^{(4)}(\omega_V, c_{F_D}). \quad (10)$$

Based on the obtained values of Q, K, and V, self-attention network calculation is carried out to obtain the position feature information at time $t+1$

$$Y_{\text{pred}} = \text{MLP}\left(W^{(5)}\left(\text{softmax}\left(\frac{Q^T K}{\sqrt{d}}\right)V\right), \omega\right) \quad (11)$$

where $W^{(5)}$ is a trainable network, ω is a network parameter, multilayer perceptron (MLP) is fully connected MLP. In order to optimize the network weight W, root mean-square error

Algorithm 1 Training

```
Input: $D_i$(i ∈ $M$), c$_i$ ∈ $x_i^t$, $Q_i$, Time information $X_{his}$ and label $Y^T$
  for $i = M, \ldots, 1$ do
      Create $\varphi_i$ by (2).
      Create $\theta_i$ by (3).
      GET $(V_c, F_k) = C_i$ by (4).
  end for
  Obtain the adjacency matrix by
      $A_{ij} = \begin{cases} 0, \text{ no communication between agents } ij \\ 1, \text{ communication between agents } ij. \end{cases}$
  repeat
      Create $math_{his} = \text{concat}(X_{his}, p)$.
      Add $S$ to $F_D$ by (6) and (7).
      Create Factor Graph network $G = (V, F_i, F_D, A)$.
      for $i = N, \ldots, 1$ do
          Get $c_i$ from $V$.
          Get $c_{F_i}^m$ from $F_i$.
          Get $c_{F_D}$ from $F_D$.
          Create $Q$,$K$,$V$ from (8), (9) and (10).
          Calculates the agent information.
          Get $Y_{pred}$ by (11)
      end for
  until argmin($L$) = RMSE($|Y^T - Y_{pred}|^2$)
```

(RMSE) can be minimized. The constrained optimization objective of the loss function is

$$\text{argmin}(L) = \text{RMSE}\Big(|Y - Y_{\text{pred}}|^2\Big). \quad (12)$$

In this article, the ADAM optimizer is used to train the network model. In the training process, the mean-square error (MSE) between the model's prediction results of the target location and the actual location information is minimized on the training set, as shown in Algorithm 1. Finally, the prediction is made on the test set and the positioning result of the target position is obtained.

IV. Experiment and Analysis

In this section, we use a data set of simulation scenarios to evaluate the performance of the model in this article and experiment with the following research questions (RQs).

RQ 1: What scale of multiagent network does this model apply to, and how does the model performance change with the number of agents? (Fig. 7).

RQ 2: Does the hidden state extracted from the time series of the model in this article affect the final position estimation accuracy of the model, and how to choose the length of the time window? (Fig. 8).

RQ 3: How does the communication threshold between agents affect the accuracy of the model? Under what communication conditions can the proposed model achieve optimal performance? (Fig. 9).

RQ 4: What is the impact of each data module, including data fusion, internal factor and dynamic factor, on the accuracy of the model? (Fig. 10).

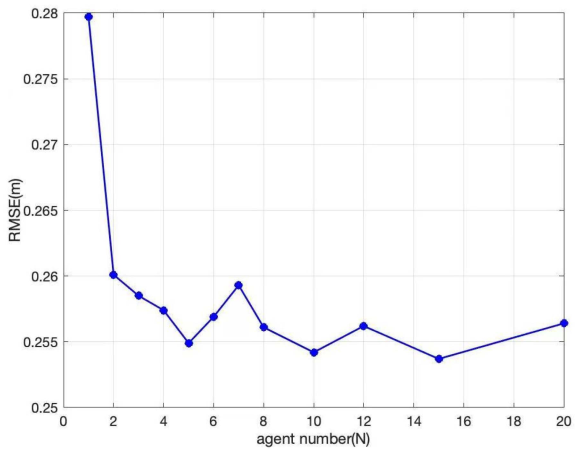

Fig. 7. Model accuracy under different number of agents.

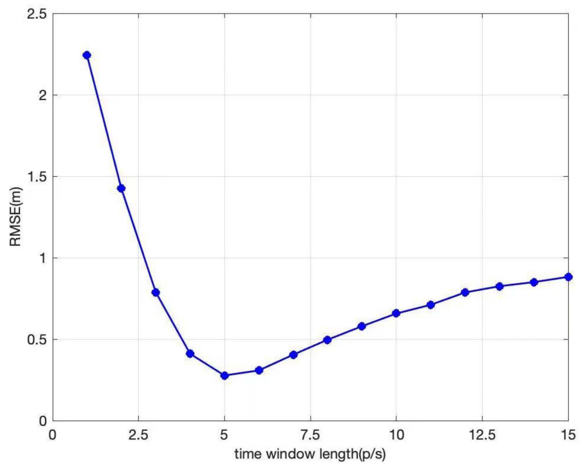

Fig. 8. Model experiment results under different window sizes.

RQ 5: Compared with existing collaborative network localization methods, how does the proposed model perform? (Fig. 12).

A. Setups

Prior to initiating model training, we established a well-defined training procedure. On the hardware front, all simulations were executed on a server computer furnished with a 13-core 3.40-GHz CPU and an NVIDIA 4080 GPU with 12 GB of memory. Regarding the software environment, we employed a training regimen of 100 epochs for all methods under consideration, with a batch size of 15 for each epoch. The optimization was performed using the Adam optimizer, set at a learning rate of 0.001. For the parameter experiments, we selected an array of continuous parameter values to evaluate the average performance of the trained model across 100 generated test data sets. In the ablation and comparative experiments, we applied the optimal parameter values derived from the initial parameter experiments to facilitate a fair comparison across various noise conditions.

We generated the data set through the simulation scene. First, the initial coordinates of N agents were randomly generated in a square area of 100 m * 100 m. The value of N was set according to different experimental parameters, ranging from 1 to 20. For each agent i, we set a fixed step size a_i and a fixed angle b_i, and add measurement errors when it is used as sensor data, which conform to the Gaussian distribution. where, $a_i^{\text{nois}} = a_i + n_{a_i}, b_i^{\text{nois}} = b_i + n_{b_i}, n_{a_i} \sim N(0, \sigma_{a_i}^2), n_{b_i} \sim N(0, \sigma_{b_i}^2)$.

In the process of data set generation, we asked all agents to walk 25 time steps in the simulation scene according to their fixed step size and angle. At each time step, we calculate the distance dis_{ij}^t between the agents as one of the sensor data, and add the measurement error $\text{dis\_nois}_{ij}^t = \text{dis}_{ij}^t + n_{\text{dis}_{ij}^t}, n_{\text{dis}_{ij}^t} \sim N(0, \sigma_{\text{dis}_{ij}^t}^2)$ when adding the sensor data.

B. Numerical Simulations

1) With Different Numbers of Agents: Aiming at problem 1 in the experimental part, we tested the positioning accuracy changes of the proposed model under different number of agents. Experimental results are shown in Fig. 7. RMSE is used as the measurement standard of positioning accuracy. Our experimental analysis indicates that the model's positioning accuracy becomes notably stable with an average error of 0.2561 m when the agent count is above eight. This stability continues with increasing agent numbers, suggesting that the interaction among a larger set of agent features is beneficial up to a point. The chosen interval of [4, 7] for the historical moment window size stems from a comprehensive evaluation of the model's performance across various configurations. It represents a pragmatic range within which the factor graph network can effectively utilize collaborative information to discern the agents' motion states without incurring the drawbacks of processing a potentially overwhelming feature set.

In conclusion, the model presented in this article can show stable average performance with the increase of the number of agents in the multiagent co-operating scenario. Compared with the traditional static positioning method, the proposed model can realize the dynamic positioning of a certain scale of agents, and can meet the actual requirements of high dynamic collaborative positioning scenarios.

2) Effect of Different Window Sizes: Fig. 8 shows the positioning performance of this model based on historical time-series data under different window sizes. We use RMSE as a measure of positioning accuracy. The experimental results show that when the length of the historical time window is less than 2, the positioning accuracy of the model is the lowest, reaching 1.424 m. With the increase of window length, the positioning accuracy of the model is gradually improved, and the minimum error of 0.2768 m is reached when the window length is 5. However, as the window size further increases, the performance of the model begins to slowly decline. When the window length is less than 2, due to the small amount of historical data, the model cannot effectively obtain the historical motion state, resulting in poor performance. With the increase of window length, the historical state that can be obtained by the model increases, which improves the prediction effect of the motion trend of the agent. However, when the window length is further increased, the experimental

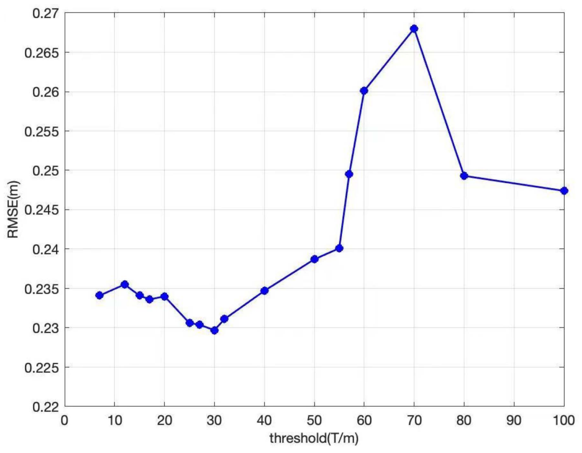

Fig. 9. RMSE under different communication thresholds.

results show that the more distant historical data has a smaller effect on the agent position prediction at the current moment. In addition, the number of input historical features is large, which makes it difficult for the model to accurately extract the history hidden state from the large historical data, resulting in lower test accuracy.

In conclusion, the proposed model can effectively utilize historical data information and improve the positioning accuracy of collaborative agents by extracting the hidden state of historical motion. In terms of the selection of time window length, a reasonable window length value should be selected in the interval [4, 7] according to the experimental results.

3) Effect of Different Communication Thresholds: In the location scenario, different communication thresholds T play an important role in data fusion and filtering. The main research in this section is to evaluate the positioning accuracy of this model under different threshold selection conditions. When the distance between two agents is greater than T, the agents do not communicate with each other. When the distance between agents is less than or equal to T, communication is connected. Fig. 9 shows the results of RMSE under different values of T in this model. It can be seen that when the communication threshold T is less than 20, the MSE RMSE is 0.2341 m. As T gradually increases to 30, RMSE decreases to the minimum value 0.2297 m. When the threshold value T is further increased to greater than 55, the average error of the model reaches the maximum value of 0.2524 m.

Our analysis has revealed that when the communication threshold window length is less than 25, the model is constrained by an insufficient historical data pool, which hampers its performance. As the window length extends beyond this lower threshold, the model gains access to an increasing amount of historical state information, thereby enhancing its ability to predict the agents' motion trajectory more effectively. Conversely, we observed that expanding the window length beyond 32 yields diminishing returns. This is due to the fact that older historical data exerts a progressively minor influence on the current position prediction of the agent. Additionally, a larger set of input historical features could overwhelm the model, complicating its task of accurately extracting the

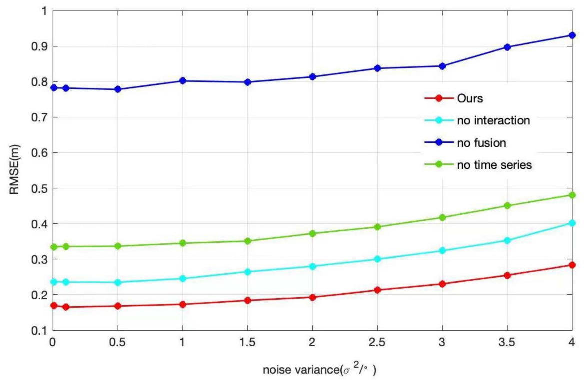

Fig. 10. RMSE under different noise variances.

relevant historical states from a vast data trove, which our tests showed can lead to a decline in accuracy.

It is this nuanced relationship between window length and historical data utility that guided our selection of the interval [25, 32]. This range represents the empirical sweet spot where the model benefits from sufficient historical information to enhance predictive accuracy without being encumbered by an overload of less influential data points.

4) Ablation Study: In order to investigate the influence of different modules in the model in this article on the results, we conducted ablation experiments. We conducted experiments on data fusion module, historical time-series data module and interaction data module between agents under different noise variance conditions, and the results are shown in Fig. 10 and Table II. With the increase of noise variance, the RMSE of all methods increases gradually. The model in this article has the best performance in terms of positioning accuracy, and can also show better performance when the noise variance is high. In contrast, the model without the information fusion module performed the worst. The results of ablation experiment show that the data fusion module has the greatest influence on the model. Considering that the heterogeneous nature of the input data leads to different orders of magnitude and units of data, the convergence effect of the model is poor without data fusion. The historical data module and the interagent data module also greatly improve the model performance. When there is no historical data, the model only uses the interactive information between the current motion model and the agent to predict, resulting in poor prediction effect. However, when there is no interagent cooperative information, the model only uses the historical motion trend and current motion state to predict, without interagent information cooperative correction, the effect is also inferior to the model in this article.

C. Physical Experiment

In the physical experiment, we set up four auto-robots equipped with UWB and IMU sensors in a 10 m $\times$ 10 m field, as shown in Fig. 11. The robots were allowed to move in random walk motion, and their sensor information and position data were collected during the experiments. The ground-truth positions were captured using the NOKOV optics motion tracking system [50]. We evaluated the algorithm's

TABLE II
AVERAGED LOSS (RMSE) OF ALL METHODS UNDER DIFFERENT ANGLE NOISE CONDITIONS

Models/angle noise(σ^2)	0.5	1	1.5	2	4
GCN	1.0346	1.1644	1.1076	1.0678	1.1294
GAT	0.4838	0.4870	0.5089	0.5290	0.5279
LS	0.5158	0.5202	0.5428	0.5441	0.6113
GTN	0.4103	0.4097	0.4172	0.4443	0.5491
Ours	0.1709	0.1785	0.1919	0.2118	0.3257

TABLE III
MULTIDIMENSIONAL COMPARISON OF DIFFERENT METHODS

Models	MSE (m)	Mean (m)	Computational time (s)
GCN	1.094	1.1046	29.214
GAT	0.5073	0.6540	27.344
LS	0.5468	0.5428	35.238
GTN	0.4461	0.5180	16.1215
Ours	0.2157	0.3376	16.734

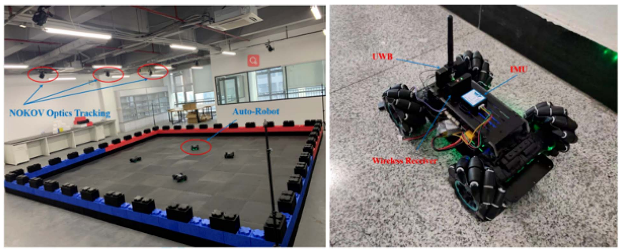

Fig. 11. (a) Scene for physical experiment. (b) Auto-robot and sensors used for the test.

performance based on the trajectory paths and localization errors of the agents.

To assess the effectiveness of our proposed model, we conducted comparative analyses with several collaborative localization methods, including graph attention networks (GAT), LS, graph convolutional networks (GCNs), and the recently introduced graph transformer networks (GTNs). Particularly, we utilized the GTN method as a comparative benchmark for the self-attention component within our model. In scenarios with varying noise variances and a fixed agent count of 10, we evaluated the performance of these methods, as depicted in the revised Fig. 12. Our method consistently exhibited superior MSE performance, especially with increasing noise variance. Notably, our model, along with the GTN and GAT methods, displayed a similar error trend. In contrast, the LS method showed a more gradual error increase with rising noise variance, while the GCN method demonstrated the least effective performance.

Table III presents a detailed comparison of our method against existing methods, considering metrics, such as MSE, Mean Error, and computational time. The data in Table III reveals that our model achieves lower MSE and Mean Error rates, thus surpassing both the GAT and GTN methods in terms of accuracy. We also assessed computational efficiency by measuring the model training time. The results confirmed that our self-attention mechanism reduces computational time,

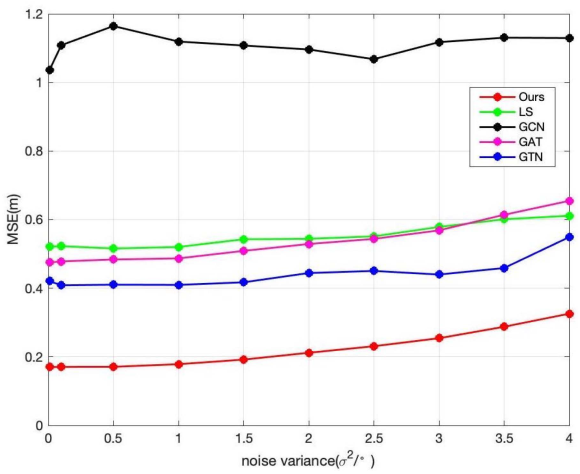

Fig. 12. Under different communication threshold, model experiment results.

enabling our model to function more efficiently compared to other attention and graph-based methods. This efficiency leads to a shorter startup time, enhancing the practical applicability of our model.

The experimental comparisons address research question 5 (RQ 5), illustrating that our model, along with GTN and GAT, follows a similar error trajectory. The LS method exhibits a steadier error increase as noise variance intensifies, while the GCN method is the least efficient. Importantly, our model outperforms the GTN method, which also incorporates an attention mechanism, in positioning accuracy. These findings validate that our self-attention method excels in data filtering, and our integration of this method within a factor graph-based neural network framework significantly improves localization accuracy. Compared to the static graph network approach of GCN, our model effectively utilizes historical dynamics to discern the hidden motion states of agents, facilitating more efficient data fusion and reasoning via the factor GNN in dynamic environments. Furthermore, our model's comparison with GAT highlights the efficacy of our self-attention method in capturing historical motion states and its dynamic adaptability to varying agent numbers, meeting the demands of dynamic collaborative scenarios.

V. CONCLUSION

In this article, we introduce a pioneering self-attention co-location method based on factor graphs, specifically engineered to integrate multisource heterogeneous data within a cooperative network. This approach utilizes a self-attention neural network to discern and analyze the inference processes among various types of information and actively filter agent data. Through an array of experiments, including parametric, ablation, and comparative studies, our model has proven to outperform traditional methods like GCNs, GATs, and MLPs. The results from these experiments underscore the influence of varying window sizes and communication thresholds on the model's performance, highlighting the strengths of our proposed approach.

However, our experiments have also uncovered certain limitations in the model. A notable issue is the model's reduced stability in accuracy when dealing with a small number of agents or when constrained by short startup times. This observation points to a need for enhancements in the model's generalization capabilities to ensure its effectiveness in a wider array of scenarios.

Moving forward, our future work will focus on overcoming this limitation. We plan to investigate strategies to boost the model's adaptability, especially in scenarios involving fewer agents and reduced operational durations. Possible solutions might include incorporating more sophisticated machine learning techniques for improved feature extraction and developing advanced training protocols to increase the model's robustness. In addition, we aim to explore the scalability of our model across diverse operational environments, with the goal of expanding its practical applicability and overall efficacy.

REFERENCES

[1] X. Li and F. Cao, "Location based TOA algorithm for UWB wireless body area networks," in *Proc. IEEE 12th Int. Conf. Dependable, Auton. Secure Comput.*, 2014, pp. 507–511. [Online]. Available: http://ieeexplore.ieee.org/document/6945742/

[2] A. Chehri, P. Fortier, and P.-M. Tardif, "On the TOA estimation for UWB ranging in complex confined area," in *Proc. Int. Symp. Signals, Syst. Electron.*, 2007, pp. 533–536. [Online]. Available: http://ieeexplore.ieee.org/document/4294530/

[3] C. Zhang, W. Wang, C. Xu, X. Sun, and J. Guo, "A TOA–based optimization positioning algorithm for non–line–of–sight errors," *J. Nanjing Univ. Posts Telecommun.*, vol. 42, no. 4, pp. 56–63, 2022.

[4] W. Hu and Y. Zhou, "Research on indoor localization algorithm based on UWB and IMU information fusion," *Manuf. Autom.*, vol. 45, no. 2, pp. 193–197, 2023.

[5] Z. Wang, Y. Wu, and Q. Niu, "Multi-sensor fusion in automated driving: A survey," *IEEE Access*, vol. 8, pp. 2847–2868, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8943388/

[6] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4802193/

[7] W. Yan, D. Jin, Z. Lin, and F. Yin, "Graph neural network for large-scale network Localization," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2021, pp. 5250–5254. [Online]. Available: https://ieeexplore.ieee.org/document/9414520/

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017, *arXiv:1609.02907*.

[9] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018, *arXiv:1706.02216*.

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. 6th ICLR*, 2018, pp. 1–12.

[11] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2019, *arXiv:1810.00826*.

[12] H. A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004. [Online]. Available: http://ieeexplore.ieee.org/document/1267047/

[13] P. Mirowski and Y. LeCun, "Dynamic factor graphs for time series modeling," in *Proc. Mach. Learn. Knowl. Discov. Databases Eur. Conf.*, 2009, pp. 128–143. [Online]. Available: http://link.springer.com/10.1007/9783642041747_9

[14] Y. Yuan, X. Weng, Y. Ou, and K. Kitani, "AgentFormer: Agent-aware transformers for Socio-temporal multi-agent forecasting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 9793–9803. [Online]. Available: https://ieeexplore.ieee.org/document/9710708/

[15] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, (Adaptive computation and machine learning). Cambridge, MA, USA: MIT Press, 2009.

[16] Z. Zhang, F. Wu, and W. S. Lee, "Factor graph neural network," 2019, *arXiv:1906.00554*.

[17] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2005, pp. 729–734. [Online]. Available: http://ieeexplore.ieee.org/document/1555942/

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016, *arXiv:1409.0473*.

[19] Y. Kim, C. Denton, L. Hoang, and A. M. Rush, "Structured attention networks," 2017, *arXiv:1702.00887*.

[20] A. Vaswani, "Attention is all you need," in *Proc. 31st Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6000–6010.

[21] N. Patwari, A. O. Hero, M. Perkins, N. S. Correal, and R. J. O'Dea, "Relative location estimation in wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2137–2148, Aug. 2003. [Online]. Available: http://ieeexplore.ieee.org/document/1212671/

[22] A. Simonetto and G. Leus, "Distributed maximum likelihood sensor network localization," *IEEE Trans. Signal Process.*, vol. 62, no. 6, pp. 1424–1437, Mar. 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6725647/

[23] J. A. Costa, N. Patwari, and A. O. Hero, "Distributed weighted-multidimensional scaling for node localization in sensor networks," *ACM Trans. Sens. Netw.*, vol. 2, no. 1, pp. 39–64, Feb. 2006. [Online]. Available: https://dl.acm.org/doi/10.1145/1138127.1138129

[24] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Trans. Sens. Netw.*, vol. 2, no. 2, pp. 188–220, May 2006. [Online]. Available: https://dl.acm.org/doi/10.1145/1149283.1149286

[25] P. Tseng, "Second-order cone programming relaxation of sensor network localization," *SIAM J. Optim.*, vol. 18, no. 1, pp. 156–185, Jan. 2007. [Online]. Available: http://epubs.siam.org/doi/10.1137/050640308

[26] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 809–819, Apr. 2005. [Online]. Available: http://ieeexplore.ieee.org/document/1413473/

[27] D. Jin, F. Yin, C. Fritsche, F. Gustafsson, and A. M. Zoubir, "Bayesian cooperative localization using received signal strength with unknown path loss exponent: Message passing approaches," *IEEE Trans. Signal Process.*, vol. 68, pp. 1120–1135, Jan. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8974236/

[28] F. Yin, C. Fritsche, D. Jin, F. Gustafsson, and A. M. Zoubir, "Cooperative localization in WSNs using gaussian mixture modeling: Distributed ECM algorithms," *IEEE Trans. Signal Process.*, vol. 63, no. 6, pp. 1448–1463, Mar. 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7015606/

[29] H. Chen, G. Wang, Z. Wang, H. C. So, and H. V. Poor, "Non-line-of-sight node Localization based on semi-definite programming in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 11, no. 1, pp. 108–116, Jan. 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6087384/

[30] H. Xie and H. Yang, "An indoor location method combining random forest with deep learning," *J. Shanghai Maritime Univ.*, vol. 41, no. 3, pp. 117–121, 2020.

[31] Z. Liu, C. Huang, Y. Yu, P. Song, B. Fan, and J. Dong, "Dynamic representation learning for large-scale attributed networks," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 1005–1014. [Online]. Available: https://dl.acm.org/doi/10.1145/3340531.3411945

[32] X. Wang et al., "Heterogeneous graph attention network," in *Proc. World Wide Web Conf.*, 2019, pp. 2022–2032. [Online]. Available: https://dl.acm.org/doi/10.1145/3308558.3313562

[33] A. Patwardhan, R. Murai, and A. J. Davison, "Distributing collaborative multi-robot planning with gaussian belief propagation," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 552–559, Feb. 2023. [Online]. Available: http://arxiv.org/abs/2203.11618

[34] R. Murai, J. Ortiz, S. Saeedi, P. H. J. Kelly, and A. J. Davison, "A robot Web for distributed many-device localization," *IEEE Trans. Robot.*, vol. 40, pp. 121–138, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10286058/

[35] "Low-density parity-check codes," in *Fundamentals of Codes, Graphs, and Iterative Decoding*, (The International Series in Engineering and Computer Science), vol. 714, Boston, MA, USA: Kluwer Academic Publ., 2002, pp. 137–175. [Online]. Available: http://link.springer.com/10.1007/0306477947_8

[36] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001. [Online]. Available: http://ieeexplore.ieee.org/document/910572/

[37] P. Chavali and A. Nehorai, "Distributed power system state estimation using factor graphs," *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2864–2876, Jun. 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7060661/

[38] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981. [Online]. Available: http://ieeexplore.ieee.org/document/1056404/

[39] L. Von Stumberg, V. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 2510–2517. [Online]. Available: https://ieeexplore.ieee.org/document/8462905/

[40] V. G. Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," 2021, *arXiv:2003.01998*.

[41] Y. Shi, B. Wang, Y. Yu, X. Tang, C. Huang, and J. Dong, "Robust anomaly detection for multivariate time series through temporal GCNs and attention-based VAE," *Knowl.-Based Syst.*, vol. 275, Sep. 2023, Art. no. 110725.

[42] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2016, pp. 2249–2255. [Online]. Available: http://aclweb.org/anthology/D16-1244

[43] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017, *arXiv:1705.04304*.

[44] S. Li, W. Chen, B. Yan, Z. Li, S. Zhu, and Y. Yu, "Self-supervised contrastive representation learning for large-scale trajectories," *Future Gener. Comput. Syst.*, vol. 148, pp. 357–366, Nov. 2023.

[45] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, S. H. Rezatofighi, and S. Savarese, "Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks," 2019, *arXiv:1907.03395*.

[46] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 961–971. [Online]. Available: http://ieeexplore.ieee.org/document/7780479/

[47] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2255–2264. [Online]. Available: https://ieeexplore.ieee.org/document/8578338/

[48] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: https://direct.mit.edu/neco/article/9/8/1735-1780/6109

[49] I. Schwartz, S. Yu, T. Hazan, and A. G. Schwing, "Factor graph attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 2039–2048. [Online]. Available: https://ieeexplore.ieee.org/document/8953801/

[50] "Nokov." Accessed: Jul. 21, 2023. [Online]. Available: https://en.nokov.com/direct

Cheng Xu (Member, IEEE) received the B.E., M.S., and Ph.D. degrees from the University of Science and Technology Beijing (USTB), Beijing, China, in 2012, 2015, and 2019, respectively.

He is currently working as an Associate Professor with USTB. He is supported by the Postdoctoral Innovative Talent Support Program from Chinese Government in 2019. His current research interests include swarm intelligence and multirobots network.

Dr. Xu is an Associate Editor of *International Journal of Wireless Information Networks*.

Ran Su is currently pursuing the master's degree with the University of Science and Technology Beijing, Beijing, China.

His research interests include swarm intelligence and Internet of Things.

Ran Wang (Graduate Student Member, IEEE) received the B.E. degree from Beijing Information Science and Technology University, Beijing, China, in 2013, and the M.S. degree from the University of Science and Technology Beijing, Beijing, in 2016, where she is currently pursuing the Doctoral degree.

Her research interests include swarm intelligence, distributed security, and Internet of Things.

Shihong Duan (Member, IEEE) received the Ph.D. degree in computer science from the University of Science and Technology Beijing (USTB), Beijing, China, in 2012.

She is an Associate Professor with the School of Computer and Communication Engineering, USTB. Her research interests include wireless indoor positioning, swarm intelligence, and Internet of Things.