



InferLoc: Hypothesis-Based Joint Edge Inference and Localization in Sparse Sensor Networks

XUEWEI BAI, YONGCAI WANG, HAODI PING, XIAOJIA XU, DEYING LI, and SHUO WANG, Renmin University of China, People's Republic of China

Ranging-based localization is a fundamental problem in the Internet of Things and unmanned aerial vehicle networks. However, the nodes' limited-ranging scope and users' broad coverage purpose inevitably cause network sparsity or subnetwork sparsity. The performances of existing localization algorithms are extremely unsatisfactory in sparse networks. A crucial way to deal with the sparsity is to exploit the hidden knowledge provided by the unmeasured edges, which inspires this work to propose a hypothesis-based Joint Edge Inference and Localization algorithm called *InferLoc*. InferLoc mines the Unmeasured but Inferable Edges (UIEs). Each UIE is an unmeasured edge, but it is restricted through other edges in the network to be inside a rigid component, so it has only a limited number of possible lengths. We propose an efficient method to detect UIEs and geometric approaches to infer possible lengths for UIEs in 2D and 3D networks. The inferred possible lengths of UIEs are then treated as multiple hypotheses to determine the node locations and the lengths of UIEs simultaneously through a joint graph optimization process. In the joint graph optimization model, to make the 0/1 decision variables for hypotheses selection differentiable, differentiable functions are proposed to relax the 0/1 selections, and rounding is applied to select the final length after the optimization converges. We also prove the condition when a UIE can contribute to sparse localization. Extensive experiments show remarkably better accuracy and efficiency performances of InferLoc than the state-of-the-art network localization algorithms. In particular, it reduces the localization errors by more than 90% and speeds up the convergence time more than 100 times than that of the widely used G2O-based methods in sparse networks.

CCS Concepts: • Networks → Network algorithms; • Software and its engineering → Software notations and tools; • Theory of computation → Design and analysis of algorithms;

Additional Key Words and Phrases: Network localization, edge inference, joint graph optimization model, sparse and noise networks

ACM Reference format:

Xuewei Bai, Yongcai Wang, Haodi Ping, Xiaojia Xu, Deying Li, and Shuo Wang. 2023. InferLoc: Hypothesis-Based Joint Edge Inference and Localization in Sparse Sensor Networks. *ACM Trans. Sensor Netw.* 20, 1, Article 8 (October 2023), 28 pages.

<https://doi.org/10.1145/3608477>

This work was supported in part by the National Natural Science Foundation of China Grant No. 61972404, 12071478; Public Computing Cloud, Renmin University of China; Blockchain Laboratory, Metaverse Research Center, Renmin University of China.

Authors' address: X. Bai, Y. Wang (corresponding author), H. Ping, X. Xu, D. Li, and S. Wang, Renmin University of China, Beijing 100872, People's Republic of China; emails: {bai_xuewei, ycw, haodi.ping, xuxiaojia, deyingli, shuowang18}@ruc.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

1550-4859/2023/10-ART8 \$15.00

<https://doi.org/10.1145/3608477>

1 INTRODUCTION

Network localization is a foundation problem in the Internet of Things and **Unmanned Aerial Vehicle (UAV)** networks [1–3]. The critical problem of network localization is to calculate the locations of agents based on the partially measured distances among the agents [4]. Various algorithms have been proposed for the network localization problem, including trilateration-based methods [5, 6], distance equation-based methods [7], graph optimization methods [8–10], and component stitching based methods [11–13]. More localization algorithms can be found in a recent survey [14].

The condition for a node or a network to be uniquely localized, known as the localizability problem, has also attracted great attention. It points out that only when the underlying graph is global rigid can all the nodes in the network be uniquely localizable [15, 16]. However, when the underlying measurement graph is sparse, localizability and localization accuracy are unsatisfactory because there are too few edge constraints to restrict the freedom of node locations [11–13].

When the distance measurements are sparse, the measured distances are too limited to constrain the nodes to be correctly localized. The location calculation algorithms [8–13] may converge to ambiguous results that differ significantly from the ground truth. An example is shown in Figure 1. In this example, Figure 1(a) gives the ground truth locations of the nodes, and Figure 1(b) shows the network localization result calculated by G2O [8], a widely used graph optimization method. Although G2O can successively minimize the least square residue error in the objective, the realized network formation differs significantly from the ground truth. Some components have serious flipping errors, such as nodes 19 and 29. This is because the algorithm itself cannot disambiguate flipping ambiguity when the ambiguous location solutions can also satisfy the measured distance constraints.

Various methods have been proposed to deal with sparse network localization. A common idea is to mine additional constraints by mining and exploiting the structural knowledge to deal with flipping ambiguity. Saha and Sau [10] added the inequality constraints of negative edges into the optimization problem to form a constrained optimization problem. Yang and Liu [16] inferred the implicit edges by finding vertex pairs shared by two rigid subgraphs. ARAP [13] and ASAP [17] methods divided the graph into patches to infer the lengths of the negative edges in each patch by the Triangle Inequality condition. WCS [12] further proposed weighted patch stitching to assign higher weights to denser patches with better local realization qualities.

Another class of idea is to disambiguate flipping ambiguity by analyzing the conditions under which the flipping ambiguity may occur. The **Unit Disk Graph (UDG)** [18] constraint is exploited to avoid flipping ambiguities [19]. The basic idea of using UDG is that the inter-distance between two nodes without distance measurement should be larger than the ranging radius. The flipping conditions are analyzed in the work of Ping et al. [20], which presented local flipping-free conditions and global flipping-free conditions based on the geometric characteristics of graph components. Kannan et al. [21, 22] presented the probabilistic analysis of flipping ambiguities. They considered the noise impacts and evaluated the flipping probabilities. Moore et al. [23] proposed the idea of a robust quadrilateral to avoid flipping caused by noises. More related work will be given in Section 2, and several typical methods have been summarized in Table 1.

The common ideas of existing methods tend to mine *exactly hidden knowledge* to improve sparse network localization. For example, the UDG constraint based approach [19] is based on an ideal assumption that an edge is not measured because two nodes are far away than the ranging radius. Using this regulation, the localization results in which the two nodes are closer than the ranging radius can be ruled out. The edge inference based methods (e.g., [16, 20]) infer edges that satisfy special conditions to infer a unique length for such an unmeasured edge. Although this inferred exact knowledge can help improve sparse network localization, such methods have two drawbacks:

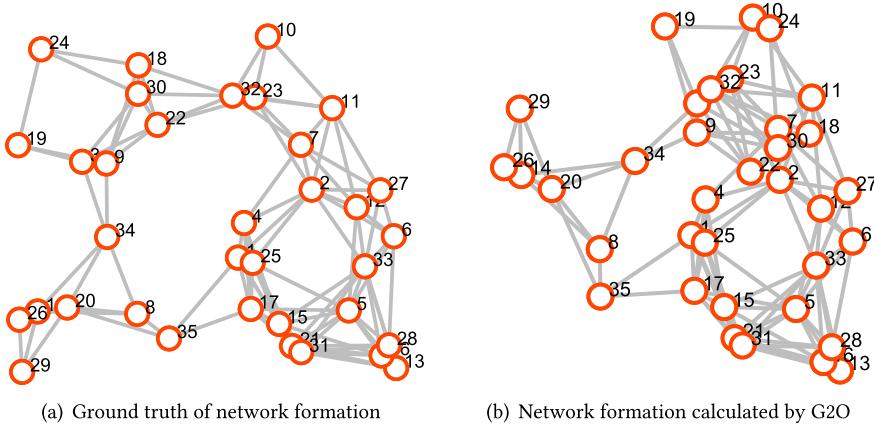


Fig. 1. An example network formation and its realization.

- (1) Relying solely on exact hidden knowledge can be overly assertive. For example, when communication blocking occurs, two nodes cannot measure inter-distance even if they are close enough. Using UDG constraints may lead to inaccurate results in such noisy cases.
- (2) The inferable exact knowledge is rare, and the exact knowledge conditions are hard to satisfy. Therefore, they have limited impacts in sparse network localization. For example, the number of unmeasured edges whose length can be uniquely inferred is limited.

Actually, in sparse networks, even if the length of a hidden edge cannot be exactly inferred, only if it is in a rigid graph, its possible lengths are limited. This is because each rigid graph has a limited number of possible realizations [24]. However, without the exact length, the knowledge about the limited number of possible lengths can still provide valuable information in sparse network localization. This article treats these possible discrete lengths as multiple hypotheses. It proposes a joint edge length inference and network localization method to automatically vote for the most compatible edge length and the localization results. Since the length of one UIE is restricted in multiple subgraphs and has a unique ground truth, the most consistent result of multiple subgraphs is generally close to the ground truth, even if the network is sparse.

This work utilizes the massive unmeasured edges whose possible lengths are not unique but limited. Such edges are called **Unmeasured but Inferable Edges (UIEs)**. The possible lengths of a UIE may differ in different rigid subgraphs containing it. We show that not all UIEs are useful. The condition of what kinds of UIEs can contribute to the InferLoc framework is investigated by proposing the penalty of inconsistency. Then InferLoc presents a joint learning model that jointly infers the lengths of UIEs and the locations of nodes. InferLoc has broad applicability and extracts much more valuable knowledge from sparse networks.

We present efficient methods to detect UIEs and infer possible lengths of UIEs. A 0/1 selection weight for each hypothesis in each subgraph is learned online to determine which hypothesized measurement is compatible with the other subgraphs and with the final localization. InferLoc also proposes the concept of qualified UIEs to enhance noise immunity. The key contributions of this article are as follows:

- (1) We propose a joint graph optimization model that utilizes multiple hypothesized lengths of UIEs to simultaneously determine the node locations and the lengths of the UIEs in both 2D and 3D sparse networks.
- (2) The conditions of what kinds of UIEs can contribute to the InferLoc framework are investigated, and the penalty of inconsistency is proposed.

- (3) We present efficient methods to detect UIEs and efficient geometric methods to infer hypothesized lengths of UIEs in three kinds of special subgraphs in 2D and 3D networks. Two kinds of these methods use local knowledge, and one kind uses multiple-hop knowledge.
- (4) A Levenberg-Marquardt optimization algorithm is exploited to conduct the joint optimization. It uses continuous switching functions to approximate the 0/1 selection variables, making the optimization objective differentiable. Rounding is applied to the continuous variables to select the final edge length.
- (5) Experiments show that InferLoc can achieve remarkably better accuracy and efficiency than the state-of-the-art localization methods in sparse networks. In particular, it can reduce location errors by more than 90% while improving convergence speed by more than 100. The inferred knowledge helps to reduce significantly the number of iteration steps of the graph optimization algorithm.

2 RELATED WORK

Various methods have been proposed in the literature to improve localization performances and tackle flipping ambiguity when the network is sparse. The literature related can be classified into three categories: (1) UDG-based flipping avoidance methods, (2) geometric conditions analysis based methods, and (3) edge inference methods.

2.1 UDG-Based Methods

The UDG constraint exploits hidden information that if the edge is unmeasured, the two end nodes are beyond the ranging scope [18]. It is mainly used to avoid flipping ambiguity under the ideal-ranging model assumption. Oliva et al. [6] proposed SELA, a “shadow edge” method using a UDG constraint to improve trilateration-based localization. Guo et al. [25] concentrated on the flipping ambiguity of micro-UAV networks. They proposed a bi-boundary model based on UDG for bilateration and a unique localization criterion to avoid flipping ambiguities in trilateration localization. Although these methods improve localization effectiveness, they are sensitive to noise because they do not consider the presence of noise. Lillis [26] and Cagirici [19] considered the UDG constraint in the presence of noise, which made the UDG constraint more realistic. However, the UDG constraint is based on an ideal communication model, which differs from that in practice. Moreover, UDG-based methods generally need the information of already localized nodes and a sequential localization manner. However, sequential localization has unsatisfactory localizability [16] and accuracy since its error accumulation and sequential localizability detection process [27].

2.2 Geometric Conditions Analysis Based Methods

Moore et al. [23] proposed the concept of *robust quadrilateral*. Some trilateration-based algorithms can reduce flipping ambiguities by selecting “robust quadrilaterals” [28]. Kannan et al. [21, 22] proposed a probabilistic analysis method to estimate the plane flipping probabilities to enhance the robustness criterion. Sun et al. [29] proposed a robust component-based localization method to evaluate the risks of the component’s local deformation and flipping. The problem of checking whether a straight line intersects with the range circles is called the **Existence of Intersecting Line (EIL)** problem. Wang et al. [30] pointed out that the EIL problem is equivalent to checking the flipping ambiguity. Liu et al. [31, 32] improved the EIL detection methods. However, these methods mainly use geometric methods to avoid the flipping ambiguities, which still rely on measured knowledge. They can only resolve the cases when measurements sufficiently determine the ambiguities.

Table 1. Comparison of InferLoc and Several Typical Methods in Sparse Network Localization

Method	Kinds of Solved UIEs	Noise	3D	Infer Edge Lengths	Independent of Regulation	Joint Learning
SELA [6]		✗	✗	✗	✗	✗
Implicit Edge [16]		✗	✓	✗	✗	✗
Ping et al. [20]		✗	✓	✓	✗	✗
InferLoc		✓	✓	✓	✓	✓

2.3 Edge Inference Based Methods

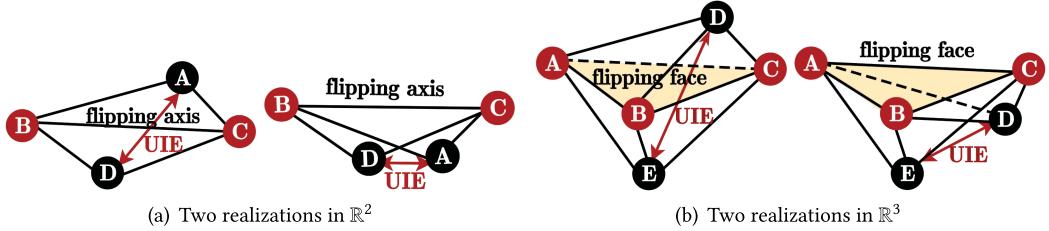
Saha and Sau [10] added the negative edge based inequality constraints into the optimization problem to form a constrained optimization problem, which can reduce the occurrence of some wrong flipping problems. Zhang et al. [13] and Cucuringu et al. [17] divided the graph into patches to infer the lengths of the negative edges in each patch by the Triangle Inequality condition. However, these methods only determine the range of the unmeasured edge, not the actual length of the edge. Yang and Liu [16] defined an unmeasured edge shared by two independent rigid components as *implicit edge*. They augmented the original graph with these implicit edges to improve the node localizability. They only studied the number of implicit edge lengths without further investigating the calculation of edge lengths. Ping et al. [20] utilized negative edge inference in special four-vertex, five-edge \mathcal{BFG} (basic flipping graphs). A method is proposed to infer the length of the unmeasured (negative) edge, which works as an additional measurement to improve the graph optimization accuracy. However, their method can only infer negative edges in specific subgraphs, and the accuracy is improved only when the exact lengths of the negative edges can be inferred. However, in practice, there are massive unmeasured edges whose exact lengths cannot be inferred precisely, which are not utilized. What is more, these methods cannot handle the case of flipping ambiguity due to the unpredictability of the environment.

We summarize the main differences between the proposed InferLoc with several typical methods for sparse network localization in Table 1. The compared methods are SELA [6], Implicit Edge [16], and Ping et al. [20]. These methods are compared from six aspects: (1) the kinds of UIEs that can be solved, (2) whether the effect of noise is considered, (3) whether they can be applied in 3D, (4) whether they can accurately find the lengths of UIEs and thus improve sparsity, (5) whether they are independent of handcraft regulation, and (6) whether this method uses jointly learning.

The comparison shows that existing methods exploit particular UIEs, whereas InferLoc can utilize general UIEs. The UIE used by the previous methods is only a tiny fraction of the overall UIEs, which wastes a large amount of information. InferLoc significantly improves the usage of available information, thus improving localization accuracy to a greater extent. Existing methods mainly separate the disambiguating and the localization phases. InferLoc conducts these two tasks jointly. Overall, InferLoc advances itself by utilizing general kinds of UIEs, working in 3D, adding inferred edge lengths, noise tolerance, independent of handcrafted regulations, and jointly edge length and node location learning. Experiments also show that InferLoc not only remarkably improves location accuracy but also dramatically improves the efficiency of graph optimization.

Table 2. Summarization of Abbreviations

Abbreviation	Full Name	Abbreviation	Full Name
UAV	Unmanned aerial vehicle	SGC	Special graph component
UIE	Unmeasured but inferable edge	\mathcal{BFG}	Basic flipping graph
UDG	Unit disk graph	\mathcal{EFG}	Extended flipping graph
EIL	Existence of intersecting line	\mathcal{NFC}	Non-flipping component
NEI	Negative edge inference	RE	Residual error

Fig. 2. Example of rigid graph that has two realizations in \mathbb{R}^2 and \mathbb{R}^3 .

The mined hypothesized knowledge significantly reduces the number of iteration steps in graph optimization.

3 PROBLEM MODEL AND ALGORITHM ARCHITECTURE

The underlying problem of network localization based on inter-node distance measurements is generally modeled as a graph realization problem [14]. This section describes the problem model and some basic concepts. For the convenience of readers, the abbreviations used in the article are summarized in Table 2. Only the generic graph is considered. Note that a graph is generic if the vertex coordinates are algebraically independent over the rationals [33].

A distance graph is denoted by $\mathcal{G}(V, E, \mathbf{d})$ in \mathbb{R}^d ($d = 2$ or 3), where V represents the agents and \mathbf{d} represents the partially measured distance matrix. Note that the distance measurements are noisy. An edge $(i, j) \in E$ represents a distance measurement between agent i and j . The goal is to calculate the node coordinates $X = \{x_i | i \in V\}$ so that the calculated distances $\|x_i - x_j\|, (i, j) \in E$ are as consistent as possible with the distance measurements $d_{ij} \in \mathbf{d}$. The problem can be formulated as follows.

$$X^* = \arg \min_X \sum_{(i, j) \in E} (\|x_i - x_j\| - d_{ij})^2 \quad (1)$$

3.1 Unmeasured but Inferable Edges

Notice that even if $(i, j) \notin E$ (i.e., $d_{ij} \notin \mathbf{d}$), we can still get some information from these unmeasured edges. Even if the graph is not global rigid (i.e., cannot be realized uniquely [34]), it may still be restricted to only a limited number of discrete realizations [35, 36] when it satisfies the rigid condition. Since the number of graph realizations is limited, the possible lengths of an unmeasured edge in the rigid subgraphs must be limited. As an example shown in Figure 2, both the 2D graph in Figure 2(a) and the 3D graph in Figure 2(b) have only two possible realizations given the measured edge constraints. So the unmeasured edge (A, D) in Figure 2(a) and edge (D, E) in Figure 2(b) have only two possible lengths in all possible realizations. Such a phenomenon exists widely in more complex graphs. For the edge where this phenomenon exists, we call it a UIE.

Definition 1 (Unmeasured but Inferable Edge). An unmeasured edge (i, j) between node i and j is called an *unmeasured but inferable edge* (UIE) in \mathcal{G} if it satisfies the following conditions:

- (1) The unmeasured edge has a finite number of possible lengths in \mathcal{G} .
- (2) All the possible lengths of the unmeasured edge are inferable.

Definition 1 describes the attributes of a UIE, but it cannot serve as a method to find a UIE. Before a method is given, we give some definitions to present the method clearly.

Definition 2 (Rigid Graph). A graph is *rigid* if and only if it contains a spanning Laman Graph [37] as a subgraph.

Definition 3 (Redundant Rigid Graph). A graph $\mathcal{G}(V, E, \mathbf{d})$ with realization in \mathbb{R}^2 and \mathbb{R}^3 is *redundantly rigid* if and only if it remains rigid after the removal of any one edge $(i, j) \in E$.

Theorem 4 proposes a sufficient condition to identify UIEs.

THEOREM 4. *In \mathbb{R}^2 , considering a generic graph \mathcal{G} , suppose a graph component $C \in \mathcal{G}$ is redundant rigid, then all the unmeasured edges in C are UIEs.*

PROOF. A graph C is redundant rigid if it remains rigid after removing any single edge from it, and this component does not have flex ambiguities. By a maximum flow algorithm [38], we can find all the two-vertex cuts in \mathcal{G} that can cause flipping ambiguities, which are called the *flipping separators*. Suppose k separators can cause flipping ambiguity of C . Then the total number of possible realizations of C is 2^k . All of these realizations are theoretically inferable by stitching the global rigid components on different sides of the flipping axes. So the possible lengths of each unmeasured edge in C have a limited number and are inferable. \square

Note that Theorem 4 gives only a sufficient condition. Some UIEs can be found by detecting redundant rigid components. Although the Pebble Game algorithm can detect the redundant rigid components in polynomial time [37], calculating all the UIEs is still time consuming. UIEs can be further found in simpler SGCs, in which the possible lengths can be calculated efficiently, which will be detailed in Section 4. Moreover, not all kinds of UIEs can contribute to network localization. Therefore, we first suppose we have some methods to efficiently identify a set of UIEs and present the InferLoc model. Then we theoretically investigate what kinds of UIEs can contribute to improving sparse network localization. Algorithms to solve the joint optimization problem will be presented in Section 5.

3.2 InferLoc Model Using Hypothesized Lengths of UIEs

InferLoc exploits that a UIE may have different possible lengths in different rigid subgraphs containing it. This forms multiple sets of possible lengths for the same UIE. To distinguish from $(i, j) \in E$, $\widehat{(i, j)}$ is used to indicate a UIE between i and j . We call the subgraphs in which $\widehat{(i, j)}$ has a limited number of possible lengths the **Special Graph Components (SGCs)** of $\widehat{(i, j)}$, denoted by a graph set S_{ij} . The number of possible lengths of $\widehat{(i, j)}$ in $G_l \in S_{ij}$ is denoted by n_l . Then the optimization objective in Equation (1) is rewritten as Equation (2).

$$\begin{aligned}
X^* = \arg \min & \sum_{(i, j) \in E} (\|x_i - x_j\| - d_{ij})^2 \\
& + \sum_{\widehat{(i, j)} \in U} \sum_{G_l \in L_{ij}} \sum_{k=1}^{n_l} \omega_k^l (\|x_i - x_j\| - \hat{d}_{ij}^k)^2 \\
\text{s.t. } & \begin{cases} \sum_{k=1}^{n_l} \omega_k^l = 1 \\ \omega_k^l = 0 \text{ or } 1 \end{cases} \quad \forall \widehat{(i, j)} \in U, \forall G_l \in L_{ij}, \forall k = 1, \dots, n_l
\end{aligned} \tag{2}$$

In Equation (2), U denotes the utilized UIEs. $\omega_k^l \in \{0, 1\}$ is a decision variable determining which hypothesized length in the l th SGC is selected in the result. The selection is exclusive (i.e., $\sum_{k=1}^{n_l} \omega_k^l = 1$), so the constraints of UIEs are called *switchable constraints*. We will introduce the designed method to make ω_k^l differentiable in Section 5.

Remark 1. The InferLoc model does not use UDG constraints. Therefore, it does not need to assume $\|x_i - x_j\| > R$, if $(i, j) \notin E$.

3.3 What Kind of UIE Can Contribute?

Since many UIEs can be selected, the first question is what kind of UIE can contribute to the localization in Equation (2). Since a $\widehat{(i, j)}$ may be in multiple SGCs, let G_l denote an SGC component where multiple possible lengths of $\widehat{(i, j)}$ are inferred. It can contribute to the localization only when $\widehat{(i, j)}$ can be disambiguated in the final optimization result in the InferLoc model. We claim that not all UIEs can be disambiguated by joint optimization.

THEOREM 5. *In a generic graph $\mathcal{G}(V, E, \mathbf{d})$, suppose $\widehat{(i, j)} \in G_l$, only when i and j are also connecting to other nodes outside G_l in \mathcal{G} , can the $\widehat{(i, j)}$ be disambiguated by the joint optimization.*

PROOF. Suppose i and j are not connecting to any other node outside G_l , then all the edge constraints that restrict the length of $\widehat{(i, j)}$ are in G_l . Since multiple possible lengths of $\widehat{(i, j)}$ are inferred based on edges in G_l , and no additional outside information can help to resolve the ambiguities of the lengths of $\widehat{(i, j)}$, the length ambiguity of $\widehat{(i, j)}$ cannot be resolved by the edge information in \mathcal{G} . \square

3.4 Penalty of Inconsistency

Based on Lemma 5, UIEs that are in multiple overlapping SGCs are possible to disambiguate. They contribute to localization by adding costs to the objective if inconsistent lengths are selected in different subgraphs. In Equation (2), the hypothesized lengths of UIEs in multiple SGCs add two kinds of costs to the objective: (1) *residue cost*, which evaluates the inconsistency with the predicted edge length $\|x_i - x_j\|$, and (2) *selection cost*, which evaluates the inconsistency for UIE length selection in different SGCs. Ideally, suppose the node locations are correctly calculated and the hypothesized lengths are correctly selected. In that case, the selected UIE lengths should be well consistent with the correct node locations, namely $\sum_{G_l \in L_{ij}} \sum_{k=1}^{n_l} \omega_k^l (\|x_i - x_j\| - \hat{d}_{ij}^k)^2 \approx 0$. In such cases, the different SGCs agree at a common edge length.

Otherwise, if the hypothesized lengths for $\widehat{(i, j)}$ are wrongly selected in some SGCs in the final optimization result, the wrong selection will incur additional selection costs. For clarity, we replace L_{ij} by L for a specific $\widehat{(i, j)}$. Suppose $\{\hat{d}_1, \hat{d}_2, \dots, \hat{d}_L\}$ are the selected lengths of the L SGCs for a specific $\widehat{(i, j)}$ in the optimization result. Theorem 6 proves the relationship between selection cost and the difference among the selected lengths.

THEOREM 6. *The cost added to the objective by the wrong selection is denoted as \mathcal{P} , and $\mathcal{P} \geq \frac{\sum_{m=1}^{L-1} \sum_{n=m+1}^L (\hat{d}_m - \hat{d}_n)^2}{L}$.*

PROOF. Given the length selection in L SGCs, the cost of $\widehat{(i, j)}$ in Equation (2) is $\mathcal{P} = (\|x_i - x_j\| - \hat{d}_1)^2 + \dots + (\|x_i - x_j\| - \hat{d}_L)^2$. By considering $\|X_i - X_j\|$ as one variable, \mathcal{P} takes the smallest value

when $\|x_i - x_j\| = \frac{\sum_{m=1}^L \hat{d}_m}{L}$. Thus, we have the following.

$$\begin{aligned}\mathcal{P} &\geq \left(\frac{\sum_{m=1}^L \hat{d}_m}{L} - \hat{d}_1 \right)^2 + \cdots + \left(\frac{\sum_{m=1}^L \hat{d}_m}{L} - \hat{d}_L \right)^2 \\ &= \frac{L \sum_{m=1}^L (\hat{d}_m)^2 - (\sum_{m=1}^L \hat{d}_m)^2}{L} \\ &= \frac{(L-1) \sum_{m=1}^L (\hat{d}_m)^2 - \sum_{m=1}^{L-1} 2\hat{d}_m(\hat{d}_{m+1} + \cdots + \hat{d}_L)}{L} \\ &= \frac{\sum_{m=1}^{L-1} \sum_{n=m+1}^L (\hat{d}_m - \hat{d}_n)^2}{L}\end{aligned}$$

□

A square-level penalty regarding the length difference will be added to the optimization objective if the selected hypothesis lengths in different subgraphs are inconsistent in the final result. The larger the length differences across different SGCs, the higher a penalty this UIE adds. Note that the exclusive selection weight ω_k^l is the length selection variable in different subgraphs. So whatever the estimated edge length $\|x_i - x_j\|$ is, the penalty of inconsistency will tend to push the selection variables to select consistent UIE lengths in different SGCs. Using the idea in the next section, we seek an effective method to identify multiple SGCs for a UIE.

3.5 Special Graph Components

The efficiency is also essential for InferLoc, determined by the method to find the UIEs. Although UIEs can be found by detecting redundant rigid components according to Theorem 4, such a method detects too many UIEs. It is unnecessary to exhaustively detect all UIEs because adding a portion of additional internode distance information can significantly improve the network localization performance in sparse networks. Further, according to Lemma 5, not all UIEs can contribute to localization. To detect UIEs efficiently and effectively, we propose to detect and infer possible lengths of UIEs in three kinds of particular graph components. These three SGCs can be used independently or together.

4 UIE EXTRACTION AND INFERENCE

This section first introduces the methods to detect \mathcal{BFG} and \mathcal{EFG} (extended flipping graph) using local information and the UIE edge length inference methods in \mathcal{BFG} and \mathcal{EFG} . Direct consistency checking for length voting in \mathcal{BFG} is also presented. Then, the partition and evaluation of \mathcal{NFC} s (non-flipping component) and the approximated UIE edge length inference in \mathcal{NFC} s are presented.

4.1 Edge Length Inference in \mathcal{BFG}

\mathcal{BFG} proposed in 2D by Ping et al. [20] is an efficiently detected subgraph to infer possible lengths for UIEs. We extend it to 3D.

Definition 7 (Basic Flipping Graph in 2D [20]). A quadrangle with four vertices and five edges, which is rigid but not global rigid in \mathbb{R}^2 , is called a *basic flipping graph* (\mathcal{BFG}) in 2D, whose example is shown in Figure 2(a).

Definition 8 (Basic Flipping Graph in 3D). A hexahedron with five vertices and nine edges, which is rigid but not global rigid in \mathbb{R}^3 , is called a *basic flipping graph* (\mathcal{BFG}) in 3D, whose example is shown in Figure 2(b).

Note that each \mathcal{BFG} has two and only two possible ambiguous realizations [20].

4.1.1 Detecting Qualified \mathcal{BFG} s. \mathcal{BFG} s can be easily detected using local information. For an unmeasured edge (i, j) , if i and j share two neighbors and the two neighbors are connected, then these four vertices form a \mathcal{BFG} in 2D. Similarly, if i and j share three neighbors and the three neighbors are connected, then the subgraph of these five vertices forms a \mathcal{BFG} in 3D.

Many \mathcal{BFG} s can be found in the graph. A selection method is further proposed to select the reliable \mathcal{BFG} s whose formations are less impacted by the ranging noises. We refer to the “robust quadrilateral” [23] to extract only the qualified \mathcal{BFG} s. Each \mathcal{BFG} can be divided into two sub-triangles. A method to select the qualified \mathcal{BFG} s in \mathbb{R}^2 and \mathbb{R}^3 is given in Definition 9 and Definition 10, respectively.

Definition 9 (Qualified \mathcal{BFG} in \mathbb{R}^2 [23]). A \mathcal{BFG} is called *qualified* if the edge and sub-triangles in the \mathcal{BFG} satisfy the condition that

$$d_{min}(\sin \theta_{min})^2 > \sigma,$$

where d_{min} is the shortest edge and θ_{min} is the minimal angle in the sub-triangles.

Definition 10 (Qualified \mathcal{BFG} in \mathbb{R}^3). A \mathcal{BFG} is called *qualified* if the area and sub-tetrahedrons in the \mathcal{BFG} satisfy the condition that

$$s_{min}(\sin \alpha_{min})^2 > \gamma,$$

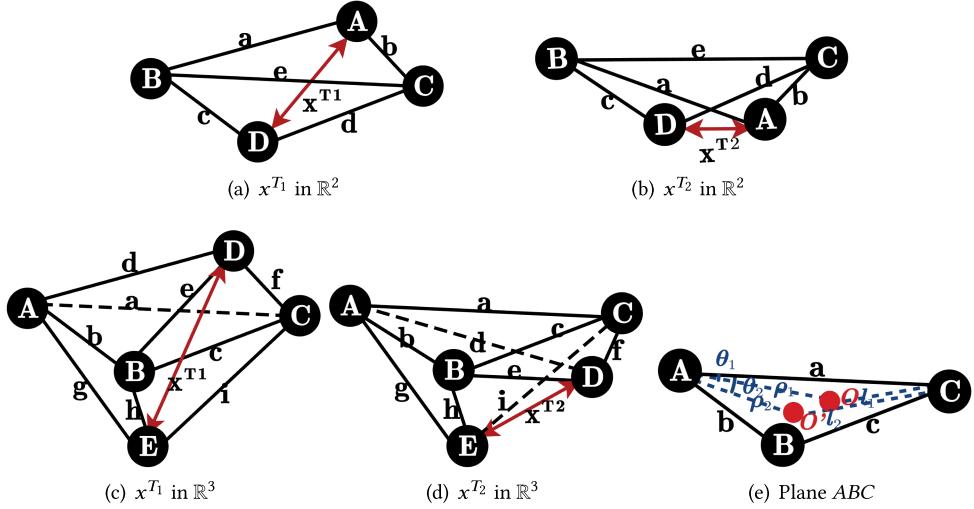
where s_{min} is the smallest area and α_{min} is the minimal dihedral angle in the sub-tetrahedrons.

4.1.2 Possible Length Inference. Given a qualified \mathcal{BFG} in 2D or in 3D, the UIE related to this \mathcal{BFG} has only two possible lengths, which are denoted as x^{T_1} and x^{T_2} . Denote the five measured edges in the \mathcal{BFG} as a, b, c, d, e as shown in Figure 3(a). The two possible lengths for the UIE can be calculated using the geometrical method.

$$\begin{aligned} x^{T_1} &= \sqrt{a^2 + c^2 - 2ac \times \cos(\alpha + \beta)} \\ x^{T_2} &= \sqrt{a^2 + c^2 - 2ac \times \cos(|\alpha - \beta|)} \\ \alpha &= \arccos\left(\frac{a^2 + e^2 - b^2}{2ae}\right) \\ \beta &= \arccos\left(\frac{c^2 + e^2 - d^2}{2ce}\right) \end{aligned} \tag{3}$$

In 3D, denote the nine measured edges in the \mathcal{BFG} as $a, b, c, d, e, f, g, h, i$ as shown in Figure 3(c) and Figure 3(d). The plane ABC is shown in Figure 3(e). O is the projection of node D on the plane, and O' is the projection of node E on the plane. ρ_1, ρ_2, l_1, l_2 represent the four calculated edges. V_1 denotes the volume of the tetrahedron $DABC$, and V_2 denotes the volume of the tetrahedron $EABC$. h_1 and h_2 represent the height of $DABC$ and $EABC$, respectively. Then the two possible lengths in 3D are as follows:

$$\begin{aligned} x^{T_1} &= \sqrt{\omega^2 + (h_1 + h_2)^2} \\ x^{T_2} &= \sqrt{\omega^2 + (h_1 - h_2)^2}, \end{aligned} \tag{4}$$

Fig. 3. Calculation of the ambiguous realizations in \mathcal{BFG} .

where

$$\begin{aligned}
 \omega_1 &= \sqrt{\rho_1^2 + \rho_2^2 - 2\rho_1\rho_2 \cos(\theta_1 - \theta_2)}, \quad \omega_2 = \sqrt{\rho_1^2 + \rho_2^2 - 2\rho_1\rho_2 \cos(\theta_1 + \theta_2)} \\
 \rho_1 &= \sqrt{d^2 - h_1^2}, \quad \rho_2 = \sqrt{g^2 - h_2^2} \\
 \theta_1 &= \arccos\left(\frac{\rho_1^2 + a^2 - l_1^2}{2\rho_1 a}\right), \quad \theta_2 = \arccos\left(\frac{\rho_2^2 + a^2 - l_2^2}{2\rho_2 a}\right) \\
 l_1 &= \sqrt{f^2 - h_1^2}, \quad l_2 = \sqrt{i^2 - h_2^2} \\
 h_1 &= \frac{3V_1}{S}, \quad h_2 = \frac{3V_2}{S} \\
 V_1 &= \frac{4d^2 e^2 f^2 - d^2 D_1^2 - e^2 E_1^2 - f^2 F_1^2 + D_1 E_1 F_1}{12} \\
 D_1 &= e^2 + f^2 - c^2, \quad E_1 = d^2 + f^2 - a^2, \quad F_1 = d^2 + e^2 - b^2 \\
 V_2 &= \frac{4g^2 h^2 i^2 - g^2 D_2^2 - h^2 E_2^2 - i^2 F_2^2 + D_2 E_2 F_2}{12} \\
 D_2 &= h^2 + i^2 - c^2, \quad E_2 = g^2 + i^2 - a^2, \quad F_2 = g^2 + h^2 - b^2 \\
 S &= \sqrt{p(p-a)(p-b)(p-c)} \\
 p &= \frac{a+b+c}{2}.
 \end{aligned} \tag{5}$$

Remark 2. When doing 3D calculations, it is necessary to judge whether the projection O of node D on the plane ABC is inside the triangle ABC or outside the triangle ABC according to the edge information:

- (1) If O is inside the triangle ABC , $\omega = \omega_1$.
- (2) If O is outside the triangle ABC , $\omega = \omega_2$.

4.1.3 Directly Length Voting by Consistency Checking. If $k \geq 2$ \mathcal{BFG} s are detected regarding one UIE, $2k$ possible lengths are obtained. Because the UIE has a ground truth length, each \mathcal{BFG} should have an inferred length close to this ground truth. Because each \mathcal{BFG} has two inferred lengths, if one of them in each \mathcal{BFG} finds agreement while the other cannot find consistent parties, it is highly suggested that the agreed lengths are measurements from the ground truth. Suppose each \mathcal{BFG} has a highly consistent inferred length with a difference less than ε , and the other inferred length is obviously different from the others (with a difference larger than σ , where $\varepsilon \ll \sigma$). In this case, we can confidently vote the consistent length of the k \mathcal{BFG} s as the inferred length of the UIE without inputting them into the joint optimization model. The length of the UIE can be inferred directly by $d = \frac{d_1 + d_2 + \dots + d_k}{k}$, where d_1 to d_k are the consistent lengths in the k \mathcal{BFG} s.

ALGORITHM 1: Edge Length Inference in \mathcal{BFG}

```

1 Input:  $\mathcal{G} = (V, E, d)$ 
2 for  $(i, j) \in E$  do
3   find all  $N(i, j) = N(i) \cap N(j)$ ;
4   if  $|N(i, j)| \geq 2$  then
5     for  $\{s, t\} \subseteq N(i, j)$  do
6       if  $(s, t) \notin E \&& d_{min} \sin^2 \theta_{min} > \sigma$  then
7          $U = U \cup U_{st}$ ;
8          $\{\mathcal{BFG}(s, t)\} = \{\mathcal{BFG}(s, t)\} \cup \{v_i, v_j, v_s, v_t\}$ ;
9         calculate the length of  $U_{st}$  by Equation (3);
10         $D(s, t) = D(s, t) \cup \{x^{T_1}, x^{T_2}\}$ ;
11 Output:  $U, D(U), \{\mathcal{BFG}(U)\}$ 

```

The whole algorithm of qualified \mathcal{BFG} detection and UIE length inference in \mathbb{R}^2 is shown in Algorithm 1. The complexity is $O(m\Delta^2)$, where m is the number of edges and Δ is the maximum node degree. Note that $\Delta \ll n$ in sparse graphs. The algorithm in \mathbb{R}^3 is similar to Algorithm 1, which will not be repeated.

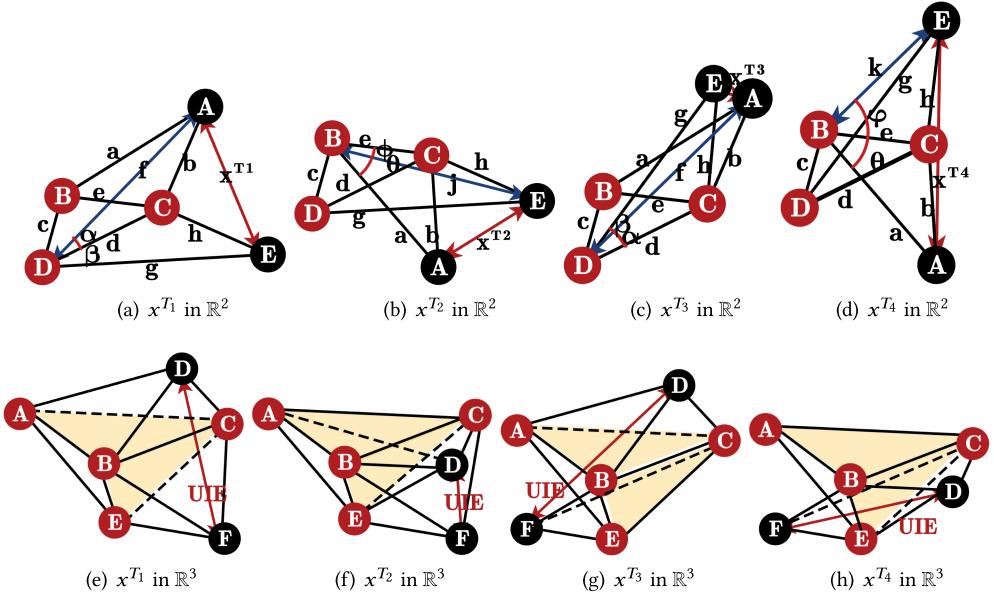
4.2 Edge Length Inference in \mathcal{EFG}

Definition 11 (Extended Flipping Graph). A graph that can be decomposed into multiple \mathcal{BFG} s is called an *extended flipping graph* (\mathcal{EFG}).

Examples for \mathcal{EFG} in \mathbb{R}^2 and \mathbb{R}^3 are shown in Figure 4. For simplicity of edge inference, the minimal \mathcal{EFG} , which can be decomposed into two \mathcal{BFG} s, is considered in both \mathbb{R}^2 and \mathbb{R}^3 .

The minimal \mathcal{EFG} can be detected using local information. For two \mathcal{BFG} s in \mathbb{R}^2 , if \mathcal{BFG}_i and \mathcal{BFG}_j share a connected triangle $\{a, b, c\}$ and edge $\widehat{(i, j)}$ is unmeasured, then these five vertices form a \mathcal{EFG} in \mathbb{R}^2 . Similarly, if \mathcal{BFG}_i and \mathcal{BFG}_j share a connected tetrahedron $\{a, b, c, d\}$ and edge $\widehat{(i, j)}$ is unmeasured, then these six vertices form a \mathcal{EFG} in \mathbb{R}^3 . Considering the existence of noise, a minimal \mathcal{EFG} is called *qualified* if its two \mathcal{BFG} s are qualified \mathcal{BFG} s. The number of ambiguous realizations and the calculation method are shown in Theorem 12.

THEOREM 12. *In \mathbb{R}^2 , considering a minimal \mathcal{EFG} as shown in Figure 4, there are four and only four distinct realizations that satisfy the seven edge constraints. Without loss of generality, the possible lengths of the UIE \widehat{AE} using the edge notations in Figure 4 can be calculated as follows. The calculation*

Fig. 4. Four ambiguous realizations in the minimal \mathcal{EFG} .

method in 3D is shown in Equation (17).

$$\begin{aligned}
 x^{T_1} &= \sqrt{f^2 + g^2 - 2fg \cos(\alpha + \beta)} \\
 x^{T_2} &= \sqrt{a^2 + j^2 - 2aj \cos(\theta - \phi)} \\
 x^{T_3} &= \sqrt{f^2 + g^2 - 2fg \cos(\beta - \alpha)} \\
 x^{T_4} &= \sqrt{k^2 + g^2 - 2kg \cos(\theta + \varphi)} \\
 \alpha &= \text{arc cos } \frac{f^2 + d^2 - b^2}{2fd}, \beta = \text{arc cos } \frac{d^2 + g^2 - h^2}{2dg} \\
 \theta &= \text{arc cos } \frac{a^2 + e^2 - b^2}{2ae}, \phi = \text{arc cos } \frac{e^2 + j^2 - h^2}{2ej} \\
 \varphi &= \text{arc cos } \frac{e^2 + k^2 - h^2}{2ek}
 \end{aligned} \tag{6}$$

Similar derivations can be applied in \mathbb{R}^3 , shown in the appendix. The detection of the qualified $\mathcal{EFG}s$ is based on the \mathcal{BFG} detection result. By Algorithm 1, qualified $\mathcal{BFG}s$ can be obtained first. Then, for any two qualified $\mathcal{BFG}s$ ($\mathcal{BFG}(s, p)$ and $\mathcal{BFG}(s, q)$) whose separators share a common vertex s , a qualified \mathcal{EFG} can be formed if $p \in \mathcal{BFG}(s, q)$ and $q \in \mathcal{BFG}(s, p)$.

The whole process of \mathcal{EFG} detection and UIE length inference in \mathbb{R}^2 is given in Algorithm 2. The process is applied for every two $\mathcal{BFG}s$ obtained in Algorithm 1, so the complexity is $O(N^2)$, where N is the number of $\mathcal{BFG}s$. The algorithm in \mathbb{R}^3 is similar to Algorithm 2, which will not be repeated.

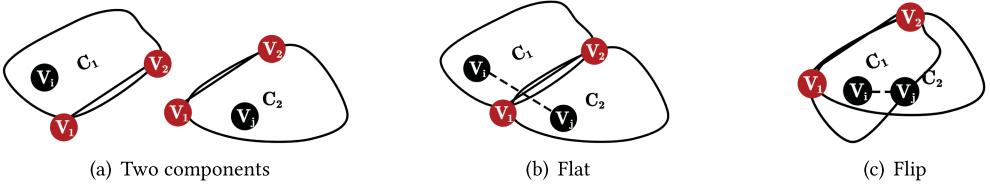


Fig. 5. Two different stitching methods.

ALGORITHM 2: Edge Length Inference in \mathcal{EFG}

```

1 Input:  $\{\mathcal{BFG}(U)\}$ 
2 for  $\{\mathcal{BFG}(s, p), \mathcal{BFG}(s, q)\} \in \{\mathcal{BFG}\}$  do
3   if  $p \in \mathcal{BFG}(s, q)$  and  $q \in \mathcal{BFG}(s, p)$  then
4      $\{\mathcal{EFG}(i, j)\} = \{\mathcal{EFG}(i, j)\} \cup \{\mathcal{BFG}(s, p) \cup \mathcal{BFG}(s, q)\};$ 
5     Calculate possible lengths of  $U_{ij}$  by Equation (6);
6      $D(i, j) = D(i, j) \cup \{x^{T_1}, x^{T_2}, x^{T_3}, x^{T_4}\};$ 
7 Output:  $U, D, \{\mathcal{EFG}(U)\}$ 

```

4.3 Edge Length Inference Across \mathcal{NFC} s

In addition to the locally detected UIEs in \mathcal{BFG} s and \mathcal{EFG} s, we consider UIEs across NFGs. A three-vertex-connected component is an \mathcal{NFC} in \mathbb{R}^2 , and a four-vertex-connected component is an \mathcal{NFC} in \mathbb{R}^3 . This is because flipping ambiguities will not happen in $(d+1)$ -vertex-connected components in \mathbb{R}^d , which is proved in the work of Ping et al. [20]. The \mathcal{NFC} s can be partitioned by an SPQR-tree algorithm [38] in \mathbb{R}^2 and can be partitioned by a k -vertex-connected-component partition algorithm [39] in higher dimensions.

Considering two three-vertex-connected components C_1 and C_2 in \mathbb{R}^2 , suppose there is a binary cut $\{v_1, v_2\}$ separating them as shown in Figure 5. We call the binary cut $\{v_1, v_2\}$ (with the edge (v_1, v_2) if the edge exists) a *flipping separator*.

Definition 13 (Flipping Separator). Given a rigid graph $\mathcal{G} = (V, E)$ in \mathbb{R}^d , if there is a d -vertex cut set V' , where $|V'| = d$ and $\mathcal{G}[V \setminus V']$ is disconnected, we call $\mathcal{G}[V']$ a *flipping separator*.

There are two ways to stitch the two \mathcal{NFC} s in the final graph realization, namely flat stitching and flipping stitching, which are shown in Figure 5(b) and (c). So for two arbitrary vertices $i \in C_1$, $j \in C_2$ and $i \notin \{v_1, v_2\} \& j \notin \{v_1, v_2\}$, $\widehat{(i, j)}$ has two possible lengths. Such UIEs are called *approximated UIEs* because \mathcal{NFC} s may still contain flex ambiguities. Although the global rigid components can confidently exclude flex ambiguities, they are generally distributed sparsely in the sparse graphs and rarely share a flipping separator. So UIEs can hardly be found between global rigid components. To exploit UIEs between components, \mathcal{NFC} s and approximated UIEs are utilized. UIEs are extracted between \mathcal{NFC} s, and the potential lengths are inferred geometrically.

The critical steps of \mathcal{NFC} detection, UIE selection, and UIE length inference are as follows:

- (1) *Components detection:* SPQR tree is applied in \mathbb{R}^2 and 4-VCC is applied in \mathbb{R}^3 to find \mathcal{NFC} s and flip separators between \mathcal{NFC} s.
- (2) *Components realization:* For each \mathcal{NFC} , the component-based graph realization method ARAP [13] is applied to realize the local formation of each component.
- (3) *Components evaluation:* Every \mathcal{NFC} is evaluated by the realization residue. Only the components that meet the standards will be considered to extract UIEs.

- (4) *Edge length inference*: Finally, in any two qualified \mathcal{NFC} s sharing a separator, two nodes with the largest hop count in the two \mathcal{NFC} s are selected to generate a UIE. The two possible lengths of the UIE are calculated by stitching the component realizations in two different ways (i.e., flat stitching and flipping stitching).

Residual Error (RE) is adopted as the metric for evaluating the realization quality of a component. The RE of each component can be evaluated by Equation (7), where RE_k represents the standard residual of the k th component. If $RE_k > \varepsilon_k$ (a threshold), the component is not qualified for generating UIEs since its local realization is not accurate enough.

$$RE_k = \frac{\sum_{(i,j) \in E_k} \|\hat{d}_{ij} - \tilde{d}_{ij}\|^2}{|E_k|} \quad (7)$$

Edge inference is conducted by stitching one component in two ways across the flipping axis with the other component. The component C_1 is assumed to be fixed, so the coordinates of j (i.e., x_j) are fixed. There are two possible coordinates for i , which are denoted as x_i^1 and x_i^2 , respectively, which are symmetrical about the flipping separator $\{v_1, v_2\}$. Therefore, the two possible edge lengths for $\widehat{(i,j)}$ are calculated as follows. For space limitation, how x_i^1 and x_i^2 are calculated will be omitted.

$$d_{ij}^1 = \|x_j - x_i^1\|^2, d_{ij}^2 = \|x_j - x_i^2\|^2 \quad (8)$$

5 INFERLOC

After extracting and inferring possible lengths for the qualified UIEs, the possible lengths are utilized as hypothesized edge constraints to be input into the InferLoc model Equation (2). Then the problem is how to solve the joint optimization problem.

A difficulty is that the 0/1 variable ω_k^l is not differentiable and cannot be directly optimized by graph optimization methods like Levenberg-Marquardt. We use a continuous function of a continuous variable $s_{ij}^l \in \mathbb{R}$ to replace each 0/1 variable ω_k^l . The set $S = \{s_{ij}^l\}$ is called the *continuous selection variables*. A switching function $\Psi(s)$ is utilized to map a continuous variable $s \in \mathbb{R}$ to be 0 or 1, which finishes the length selection. Then Equation (2) is rewritten as follows:

$$\begin{aligned} X^*, S^* = \arg \min & \sum_{(i,j) \in E + U_c} \|\hat{d}_{ij} - \tilde{d}_{ij}\|_{\Omega_{ij}}^2 \\ & + \sum_{(i,j) \in U} \sum_{l \in L} \sum_{k=1}^{n_l} \Psi^k(s_{ij}^l) \|\hat{d}_{ij}^k - \tilde{d}_{ij}\|_{\Lambda_{ij}}^2, \end{aligned} \quad (9)$$

where E represents the measured edges, U represents the selected UIEs, and U_c represents the directly voted well-consistent UIEs in \mathcal{BFGs} . \tilde{d}_{ij} denotes the Euclidean distance calculated by the estimated coordinates (i.e., $\tilde{d}_{ij} = \|x_i - x_j\|$). The function $\|\cdot\|_{\Omega}^2$ represents the squared Mahalanobis distance with covariance Ω . For the UIE with two possible lengths, we let $\Psi^2(s) = 1 - \Psi^1(s)$. For the UIE with multiple possible lengths, a switching function that meets the requirements ($\sum_{k=1}^{n_l} \Psi^k(s) = 1$) can be constructed by transforming the function linearly. The optimization problem in Equation (9) can be solved by the Levenberg-Marquardt algorithm using iterative optimization. Two switching functions are considered. The *sigmoid* function and its derivative are as follows.

$$\begin{aligned} \Psi^{sigmoid}(s_{ij}) &= sig(s_{ij}) = \frac{1}{1 + e^{-s_{ij}}} \\ sig'(s_{ij}) &= sig(s_{ij}) \cdot (1 - sig(s_{ij})) \end{aligned} \quad (10)$$

The derivative is non-zero and easy to compute. However, the *sigmoid* function asymptotically converges toward 0 and 1 but never exactly reaches those values. As shown in Equation (11), a piecewise linear function is also investigated.

$$\Psi^{linear}(s_{ij}) : \mathbb{R} \rightarrow [0, 1] = \begin{cases} 0 & s_{ij} < -0.5 \\ s_{ij} + 0.5 & -0.5 \leq s_{ij} \leq 0.5 \\ 1 & s_{ij} > 0.5 \end{cases} \quad (11)$$

Ψ^{linear} shows better convergence speed in experiments than $\Psi^{sigmoid}$, because the gradient $\nabla\Psi^{linear}$ is 0 when the Ψ^{linear} approaches 0 and 1 and the gradient $\nabla\Psi^{linear} = 1$ is constantly steep when the Ψ^{linear} between 0 and 1, which speeds up the convergence. The Levenberg-Marquardt iteration algorithm is used to solve the InferLoc model Equation (9). Given an initial state X_0 , the quadratic objective can be approximated by first-order Taylor approximation. The process to solve Equation (9) by the Levenberg-Marquardt algorithm is expressed as follows:

$$\begin{aligned} \Delta X &= -(\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{b} \\ X &= X + \Delta X, \end{aligned} \quad (12)$$

where μ represents the damping coefficient, \mathbf{I} is the identity matrix, $\mathbf{H} = \mathbf{J}^T\Omega\mathbf{J}$ is the Hessian matrix, and \mathbf{J} is the Jacobi matrix. \mathbf{J} can be expressed in a two-block form. The first block \mathbf{J}_d is the Jacobi matrix about the measurements, and the second block \mathbf{J}_s is the Jacobi matrix about the variables s (i.e., $\mathbf{J} = (\mathbf{J}_d, \mathbf{J}_s)^T$).

6 PERFORMANCE EVALUATION AND ANALYSIS

Simulations are conducted in Matlab2020b and AirSim [40], a 3D simulation platform specifically designed for UAV networks. The code runs on a Windows 11 Intel Core i5-10210U CPU @ 1.60 GHz. In simulations, n nodes are deployed randomly in $L \times L$ areas in 2D and $L \times L \times L$ areas in 3D. The sparsity of the measurement graph is mainly controlled by the maximum ranging radius r . Two nodes can measure inter-distance within the ranging radius r . For $L = 100$, we vary r in the range [10, 20] to make the average node degree about 3 to 7, which is highly sparse and is sparser than the experiment settings of existing works [12, 13, 20]. The ranging noise is assumed zero-mean Gaussian distributed ($\Omega \sim \mathcal{N}(0, \sigma^2)$). The σ varies in [1, 5]. For localization error evaluation, **Mean Square Error (MSE)** (i.e., $e = \frac{\sum_{i=1}^n \|x_i - \hat{x}_i\|}{n}$) is used, where x_i is the ground truth and \hat{x}_i is estimation result of InferLoc.

6.1 Compare Contributions of \mathcal{BFG} , \mathcal{EFG} , and \mathcal{NFC}

We first visualize the contributions of \mathcal{BFG} , \mathcal{EFG} , and \mathcal{NFC} in InferLoc. For clarity, a small-scale sparse network of 50 nodes with $r = 18$ and $\sigma = 5$ is instantiated in Figure 6. The blue diamonds represent the ground truth locations, and the red star markers show the localization results. The gray lines are measured edges, and the green lines show localization errors. The shorter the green lines, the more accurate the localization results are.

Figure 6(a) shows the result of G2O in a 2D network, which has large errors. Figure 6(b) through (d) show the results of InferLoc using only the UIEs in \mathcal{BFG} , \mathcal{EFG} , and \mathcal{NFC} , respectively. The yellow lines represent the correctly inferred UIEs in different kinds of subgraphs (i.e., \mathcal{BFG} , \mathcal{EFG} , and \mathcal{NFC} , respectively). $Num(C)$ is defined as the number of formation C . InferLoc infers more UIEs in \mathcal{BFG} and \mathcal{EFG} than that in \mathcal{NFC} . This is because $Num(\mathcal{BFG}) \geq Num(\mathcal{EFG}) \geq Num(\mathcal{NFC})$ in the same graph. So the location accuracies using UIEs in \mathcal{BFG} and \mathcal{EFG} are better than those using \mathcal{NFC} . The inferred edge lengths are the longest in \mathcal{NFC} . Figure 6(e) and (f) compare the localization results of G2O and InferLoc in sparse 3D networks. It can be seen

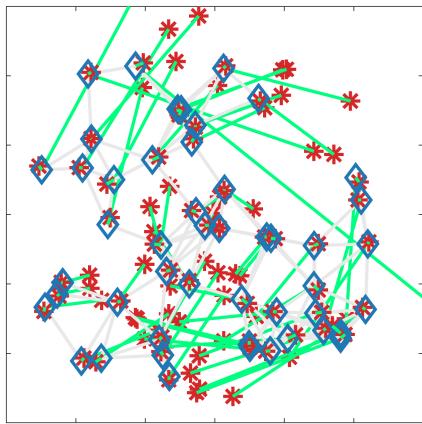
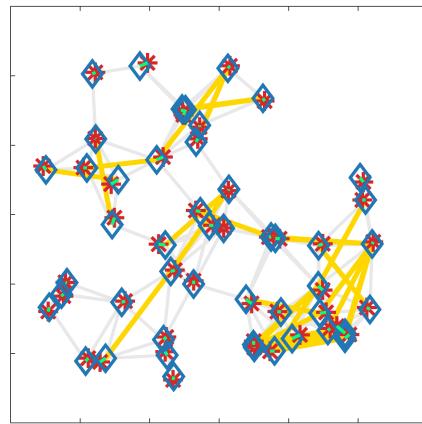
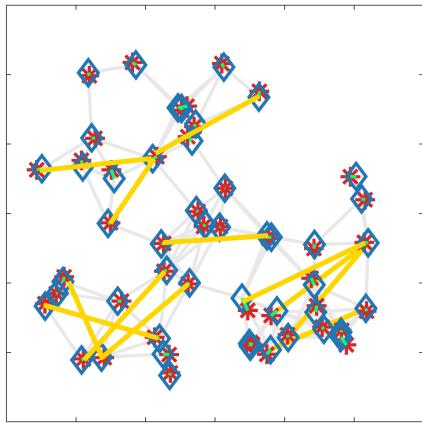
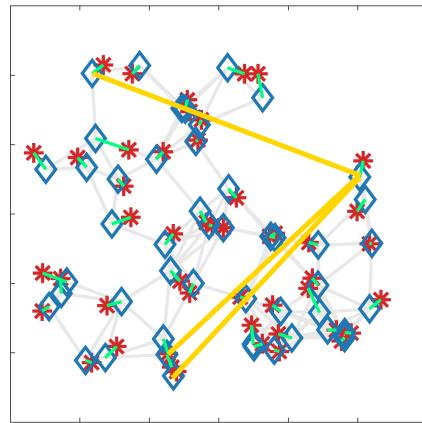
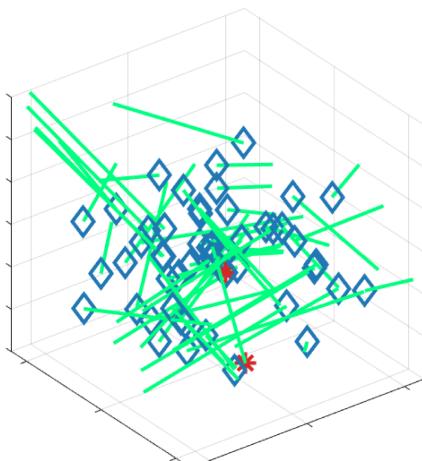
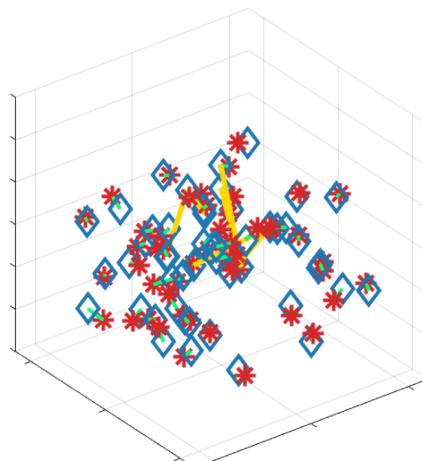
(a) G2O in \mathbb{R}^2 (b) $\mathcal{B}\mathcal{F}\mathcal{G}$ InferLoc in \mathbb{R}^2 (c) $\mathcal{E}\mathcal{F}\mathcal{G}$ InferLoc in \mathbb{R}^2 (d) $\mathcal{N}\mathcal{F}\mathcal{C}$ InferLoc in \mathbb{R}^2 (e) G2O in \mathbb{R}^3 (f) InferLoc in \mathbb{R}^3

Fig. 6. The error of G2O and InferLoc.

Table 3. Validity of \mathcal{BFG} , \mathcal{EFG} , and \mathcal{NFC} InferLoc

MSE of G2O	\mathcal{BFG}		\mathcal{EFG}		\mathcal{NFC}		
	S/N	MSE	S/N	MSE	S/N	MSE	
$\sigma = 1$	25.92	843/784	1.31	1,933/1,770	1.39	472/423	1.47
$\sigma = 3$	27.57	332/306	3.63	1,202/1,083	3.69	472/413	3.74
$\sigma = 5$	30.87	293/267	6.09	574/517	6.13	472/398	6.16

that the improvement of localization accuracy is more noticeable when utilizing InferLoc in 3D networks because 3D network localization requires more dense edge information.

Table 3 further summarizes the validity of \mathcal{BFG} , \mathcal{EFG} , and \mathcal{NFC} when the noise is different. S/N represents the ratio of the number of UIEs to the number of correctly inferred UIEs with different methods. For example, in the first row, the number of UIEs in \mathcal{BFG} in 100 experiments is 843 and \mathcal{BFG} InferLoc can correctly infer 784 UEs. So all of the \mathcal{BFG} , \mathcal{EFG} , and \mathcal{NFC} InferLoc can correctly infer a significant portion of UIEs. A portion of UIEs cannot be correctly inferred because of the similarity of ambiguous lengths. So even if the inference is wrong, it will not significantly impact the positioning accuracy. The average MSE of each method is shown in Table 3. The \mathcal{BFG} , \mathcal{EFG} , and \mathcal{NFC} InferLoc all greatly outperform G2O in localization accuracy.

6.2 Accuracy Comparison with Other Localization Algorithms

The average localization error of InferLoc is compared with the state-of-the-art network localization algorithms G2O [8] and SMACOF [41], and component stitching based network localization algorithms (designed for sparse networks), including ARAP [13] and WCS [12]. The cumulated density functions of localization errors with different settings are shown in Figure 7 when σ varies in {1, 3, 5} and r varies in {16, 20}.

We can see that G2O and SMACOF are sensitive to network sparsity compared with the component stitching methods. ARAP and WCS show robustness to network sparsity but still have significant errors when the network is highly sparse. InferLoc provides the best accuracy when the network is sparse. It also provides better accuracy in all settings for using the unmeasured edge information to deal with the lack of constraints in sparse networks.

6.3 Comparison with State-of-the-Art Methods in Sparse Networks

We visualize the comparison of InferLoc with state-of-the-art methods (G2O [8], ARAP [13], WCS [12], and Ping et al. [20]) for sparse network localization. Three sparse networks, which are skeletons of the letters “R”, “U”, and “C” are generated, and the localization errors of these four methods are calculated and visually compared.

It can be seen that G2O (Figure 8(a)) and ARAP (Figure 8(b)) both show significant localization errors in such sparse networks where each node has only a few neighbors. The reason is that G2O is not specially designed for sparse networks and does not have the generalized ability to identify and disambiguate flipping ambiguity. The edge inference of ARAP in the patch is ineffective in these graphs since many nodes are on the boundary, so it has significant errors. WCS (Figure 8(c)) performs better than G2O and ARAP but still has significant errors. Ping et al. (Figure 8(d)) infer negative edges, which shows great performance improvement, but the errors are still notable. InferLoc (Figure 8(e)) shows superior performance improvement over the other four methods in these sparse networks. The results validate the effective utilization of UIEs in InferLoc.

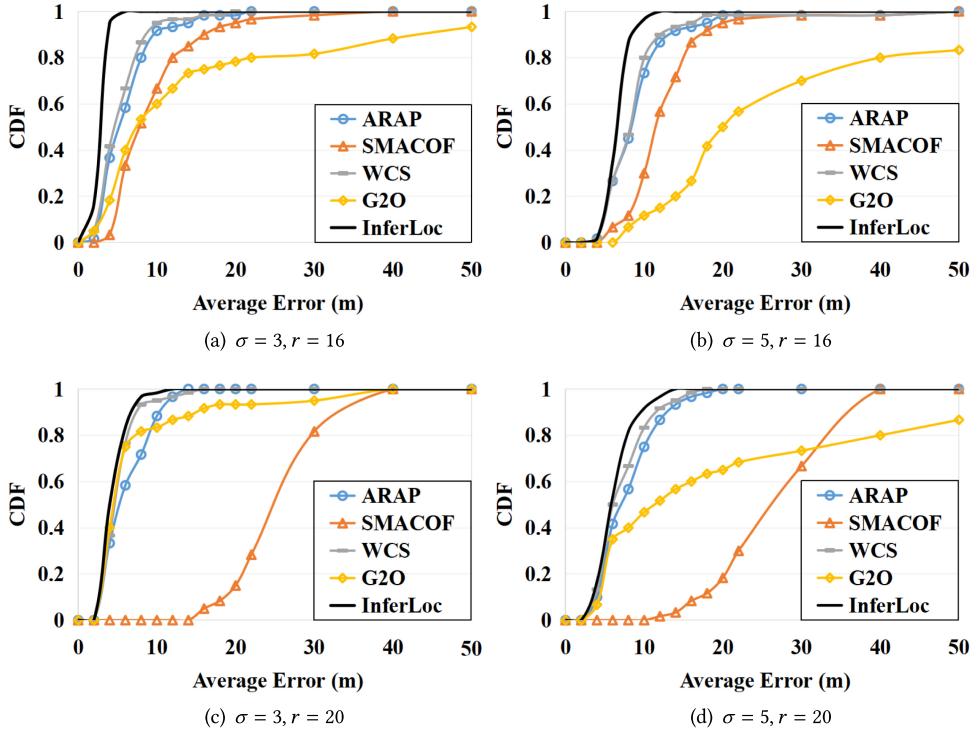


Fig. 7. The location error of different algorithms for networks with different connectivity and noise levels.

6.4 Accuracy for Sparse Formation Calculation

We further conduct formation tracking experiments in 3D networks, where the network topology varies to be sparser and sparser over time. Formation tracking simulations are conducted in AirSim [40], a UAV network simulator built on Unreal Engine. The 3D environment is created in Unreal 4.0. In the simulation, the formation of UAVs changes from a dense network to a sparse network. Initially, 50 UAVs are deployed randomly in a small 3D area, forming a dense network. Then each UAV begins to move randomly in each time slot. Due to random movement, the UAV network will become increasingly sparse for Brownian motion. The network formations at different time slots are shown in Figure 9, which are becoming increasingly sparse.

The localization algorithms are applied to calculate the UAV network topology when the sensing radius and ranging noises are set differently. In particular, we evaluate the cases when $(r = 16, \sigma = 3)$, $(r = 16, \sigma = 5)$, $(r = 20, \sigma = 3)$, and $(r = 20, \sigma = 5)$. The network topologies during 50 time slots are calculated using different algorithms in each setting. The results are shown in Figure 10(a) through (d). InferLoc performs best in all of these settings than in the other four methods. When $T < 30$, all methods perform better since the network has good density. When $T > 30$, the network becomes increasingly sparser, and InferLoc shows remarkably better robustness and accuracy than the other methods. Benefiting from edge inference, InferLoc shows much smaller error and variance than the other four methods.

6.5 Performance in Large-Scale Sparse Networks

To verify the effectiveness of InferLoc in large-scale networks, we test the networks with nodes $n = 100, 200, 300, 400$, and 500 , respectively, and the results are shown in Table 4. σ is the variance of ranging noises. For each setting of n , we calculate and average the performances of 100 randomly

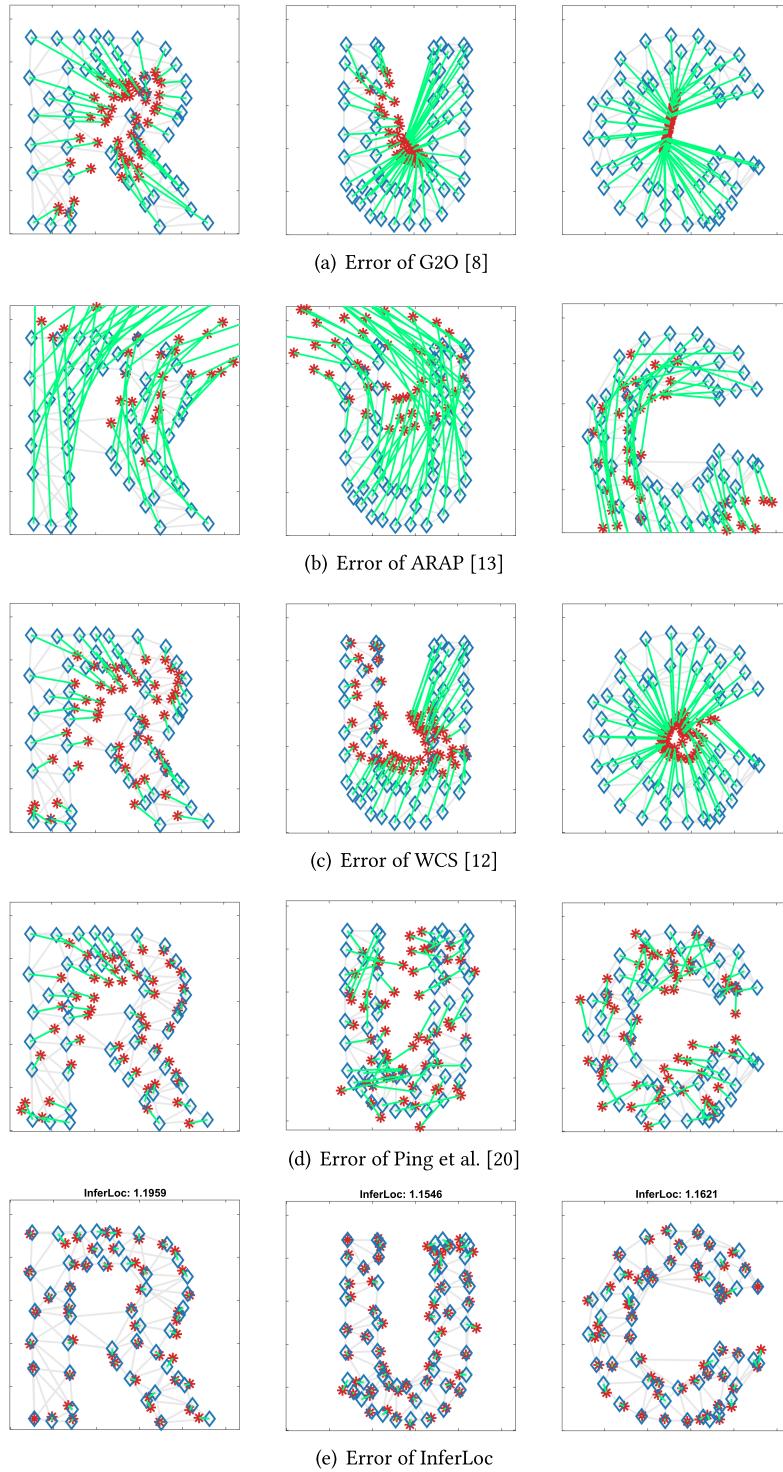


Fig. 8. The networks of shapes “R,” “U,” and “C.”

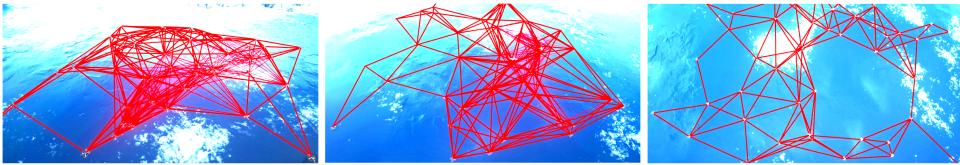


Fig. 9. The simulation environments by AirSim for formation tracking.

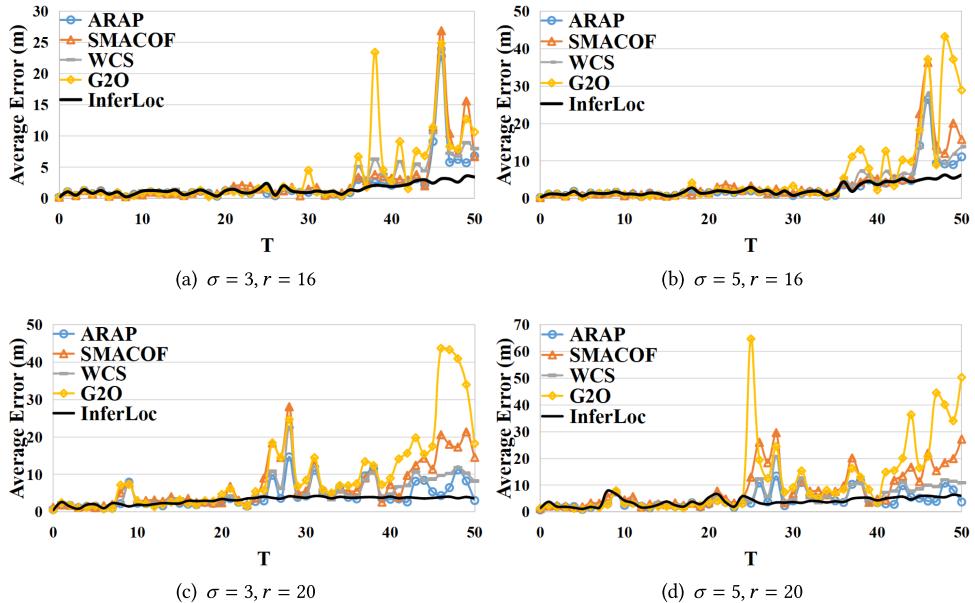


Fig. 10. The location error of different algorithms for networks with different connectivity and noise levels.

generated different graphs. We compare the average localization errors with that of the G2O [8] algorithm. The results show that InferLoc achieves significantly fewer localization errors than G2O in all network scales and noise ratio settings. The errors are reduced by more than 90% compared with G2O.

G2O has significant localization errors because, in sparse networks, even one wrong flipping can cause significant localization errors in the whole graph. This problem becomes more significant as the sparseness increases. InferLoc can infer many UIEs to add additional constraints and to identify the ambiguity of flipping. Therefore, InferLoc can provide correct guidance for graph formation calculations. Table 3 shows that the number of UIEs inferred in a network of 50 nodes can be up to several hundred. As the size of the network increases, the number of correctly identified UIEs increase with the network size accordingly. The knowledge provided by these UIEs helps reduce localization errors significantly. We can see from Table 4 that as the network size increases, the localization errors of G2O increase, but because more UIEs can be identified in larger size sparse networks, the network scale impacts the localization errors of InferLoc less. Therefore, InferLoc achieves better localization accuracy improvement in larger networks.

6.6 Computational Efficiency Compared to G2O

In this section, we compare the computational efficiency of InferLoc with that of G2O. The computational efficiency will be evaluated in terms of the time used before the algorithms converge, as

Table 4. Effectiveness of InferLoc in Large-Scale Networks

$n = 100$		$n = 200$		$n = 300$		$n = 400$		$n = 500$		
	G2O	InferLoc	G2O	InferLoc	G2O	InferLoc	G2O	InferLoc	G2O	InferLoc
$\sigma = 1$	27.39	1.05	35.28	1.21	47.65	1.32	69.33	1.33	79.15	1.32
$\sigma = 3$	45.57	3.53	52.37	3.49	65.21	3.74	82.45	3.88	89.37	3.91
$\sigma = 5$	66.87	6.31	78.81	6.77	88.02	6.93	92.31	7.02	99.97	7.11

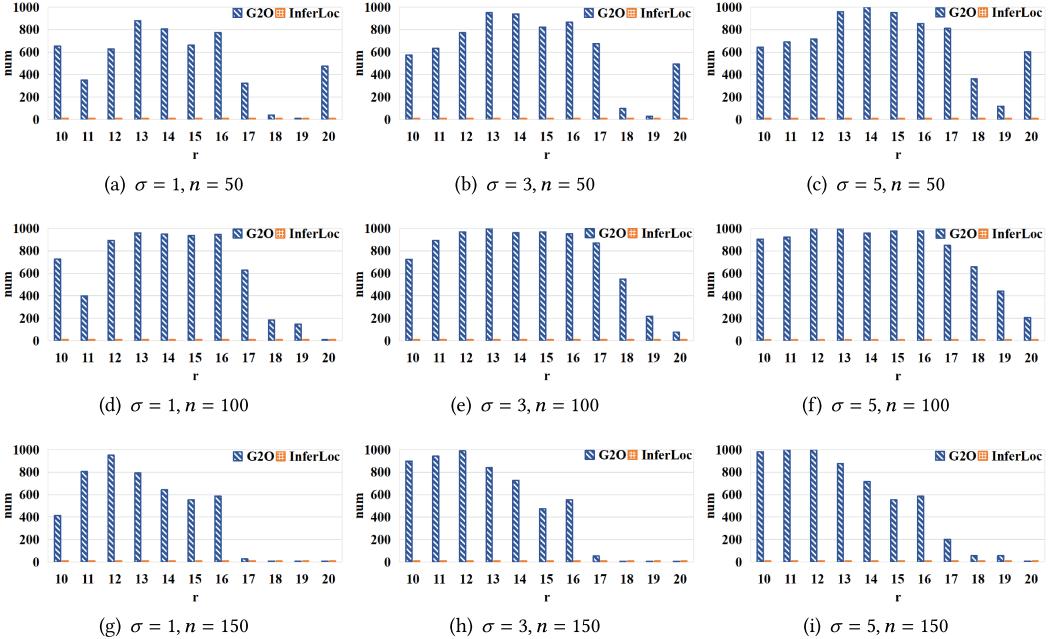


Fig. 11. The average number of iterations required by G2O and InferLoc under different network scales and noise levels.

well as the number of iterations. We repeat the experiment 100 times and take the average on the running time (or the number of iterations) in each experiment setting. The iteration is completed if the change between two iterations is less than a tiny threshold. Otherwise, the optimization will stop when it runs 1,000 iterations.

Figure 11 compares the average number of iterations required by G2O and InferLoc under different parameter settings. It is easy to see that the average number of iterations required by InferLoc in the sparse network (e.g., in networks of 150 nodes with $r \leq 16$) can be decreased almost hundreds of times more than that of the G2O algorithm. InferLoc can often converge within 10 iterations, and the average number of iterations of InferLoc is robust to different noise levels, ranging radius, and graph scales.

However, we should also note that when the graph is less sparse (e.g., in the networks of 150 nodes with $r \geq 17$), the average number of iterations can be very close to G2O. UIEs become less valuable in dense networks since the measured edges have provided enough constraints.

Benefiting from the substantial reduction in the number of iterations, Figure 12 shows that the average running time required by InferLoc in sparse networks is also significantly reduced compared with that of G2O. InferLoc reduces the overall running time from several seconds to

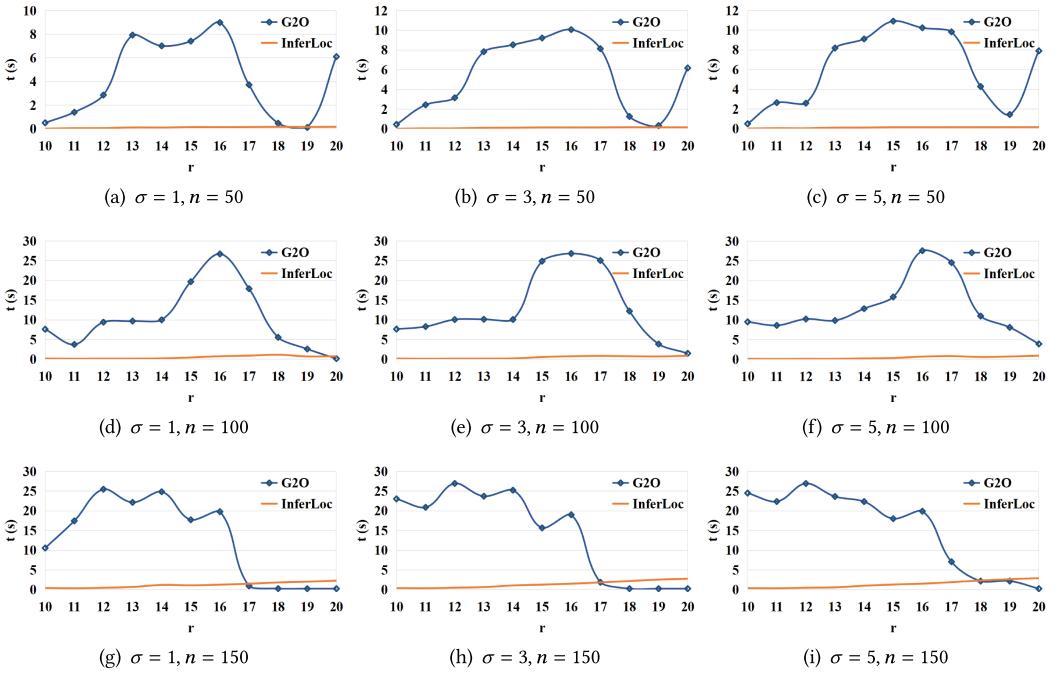


Fig. 12. The average running time required by G2O and InferLoc under different network scales and noise levels.

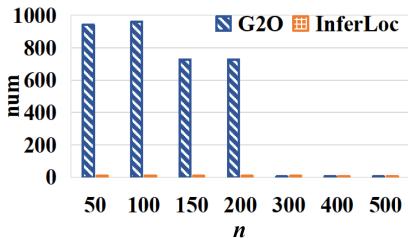


Fig. 13. The average number of iterations required by G2O and InferLoc in different scale networks.

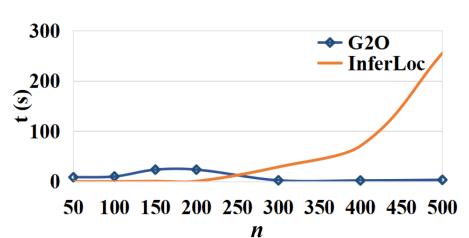


Fig. 14. The average running time required by G2O and InferLoc in different scale networks.

less than 1 second in the networks with 50 nodes, and from dozens of seconds to 1 to 2 seconds with 100 and 150 nodes in sparse networks.

We also evaluated the impacts of graph scale and noise on the number of iterations and running time. The results are shown in Figures 13 through 16. In the experiments of Figure 13 and Figure 14, we set $\sigma = 3$ and $r = 14$ and vary n from 50 to 500. Figure 13 shows that InferLoc needs fewer iterations in sparse network settings (i.e., when n is in the range of 50 to 200). When $n > 200$, the network becomes dense. The number of iterations required by InferLoc becomes close to that of G2O.

Figure 14 shows that the average running time of InferLoc is also much lower than that of G2O in sparse networks with $n \leq 200$. Its running time exceeds that of G2O when $n > 200$ (i.e., when the

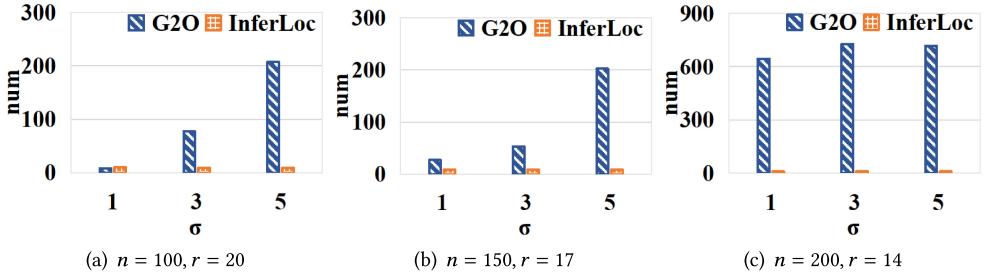


Fig. 15. The average number of iterations required by G2O and InferLoc under different noise levels.

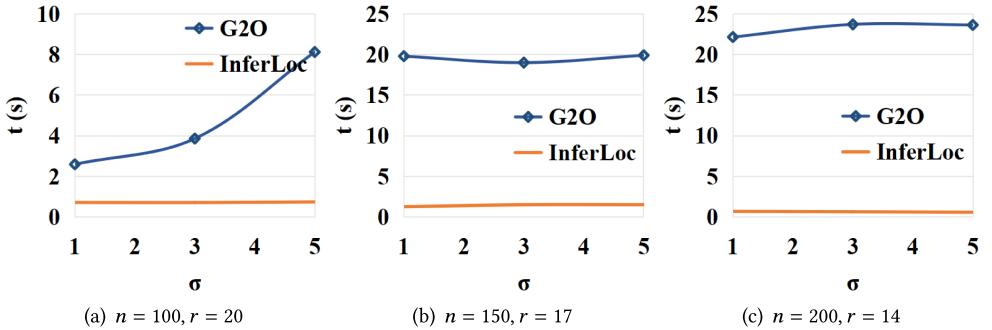


Fig. 16. The average running time required by G2O and InferLoc under different noise levels.

networks become dense). This is because much higher computation costs are needed to infer UIEs in dense networks. However, these UIEs provide less help in reducing the number of iterations since the measured edges are already very dense in these networks.

Figures 15 and 16 summarize the average number of iterations and running times of InferLoc under different noise levels. They show that the computation efficiency of InferLoc is highly robust to noise.

7 CONCLUSION AND DISCUSSION

This article presented InferLoc, a joint edge inference and network localization method to exploit general UIEs for sparse network localization. It can deal with the cases that existing handcraft-based methods cannot address and shows remarkable localization accuracy improvement in sparse networks over existing state-of-the-art methods. In particular, it can reduce localization errors by more than 90% and reduce convergence time by more than 100 times than that of the widely used G2O-based methods in sparse networks. It theoretically investigates what kinds of UIEs can contribute and designs a differentiable LM-based optimization framework to solve the joint optimization problem. For generating UIEs efficiently, geometric methods to infer UIEs in \mathcal{BFG} , \mathcal{EFG} , and \mathcal{NFC} in both 2D and 3D networks were presented. We also proposed methods for selecting qualified UIEs and direct voting methods for UIEs in \mathcal{BFG} s. Moreover, InferLoc is compatible with various existing methods, such as the edge inference methods that cannot precisely infer the exact edge length. In such cases, InferLoc can help infer the edge lengths through hypothesis-based joint graph optimization.

However, as shown in Figure 14, InferLoc is inefficient when the network is dense. In future work, we will seek a more effective method to dynamically select the number of UIEs. InferLoc can be adaptive to the network sparsity to improve computing efficiency while ensuring localization accuracy. We will also explore InferLoc in graphs with bearing and distance measurements.

APPENDIX

A CALCULATION METHOD OF UIE IN 3D \mathcal{EFG}

This section gives the calculation method of UIE in 3D \mathcal{EFG} . In \mathbb{R}^3 , considering a minimal \mathcal{EFG} as shown in Figure 4, there are four and only four distinct realizations that satisfy the 12 edge constraints. Without loss of generality, the possible lengths of the UIE DF using the edge notations in Figure 17(a) can be calculated as follows. The solution method appearing in Equation (5) will not be repeated and handled directly according to the known variables. The length of edge AF has two possible lengths, denoted as AF_1 and AF_2 . The lengths of OA, OB, OC , and OD can be calculated according to Equation (5). BF and CF are measurements, so we use two vertices to represent the length. The coordinates of nodes A, B, C , and D can be expressed as follows.

$$\begin{aligned} A & (OA \sin \theta, -OA \cos \theta, 0) \\ B & (OB, 0, 0) \\ C & (-OC \sin \alpha, OC \cos \alpha, 0) \\ D & (0, 0, OD) \\ \theta & = \frac{\pi}{2} - \frac{OA^2 + OB^2 - AB^2}{2OAOB} \\ \alpha & = \pi - \frac{OB^2 + OC^2 - BC^2}{2OB \cdot OC} \end{aligned} \tag{13}$$

After having the coordinates of D , the main goal is to solve the coordinates of F . Therefore, the coordinates of F can be set to $F(x, y, z)$. The following equations can be obtained according to the edge passing through the node F .

$$\begin{cases} BF^2 = (x - OB)^2 + y^2 + z^2 \\ CF^2 = (x + OC \sin \alpha)^2 + (y - OC \cos \alpha)^2 + z^2 \\ AF_1^2 = (x - OA \sin \theta)^2 + (y + OA \cos \theta)^2 + z^2 \\ AF_2^2 = (x - OA \sin \theta)^2 + (y - OA \cos \theta)^2 + z^2 \end{cases} \tag{14}$$

Solving Equation (14) can get Equation (15).

$$\begin{cases} L_1 = BF^2 - CF^2 - OB^2 + OC^2 \sin^2 \alpha + OC^2 \cos^2 \alpha \\ \quad = -2(OC \sin \alpha) x + 2yOC \cos \alpha \\ L_2 = -BF^2 + AF_1^2 - OB^2 - OA^2 \sin^2 \theta + OA^2 \cos^2 \theta \\ \quad = -2(OA \sin \theta) x + 2yOA \cos \theta \\ L'_2 = -BF^2 + AF_2^2 - OB^2 - OA^2 \sin^2 \theta + OA^2 \cos^2 \theta \\ \quad = -2(OA \sin \theta) x + 2yOA \cos \theta \end{cases} \tag{15}$$

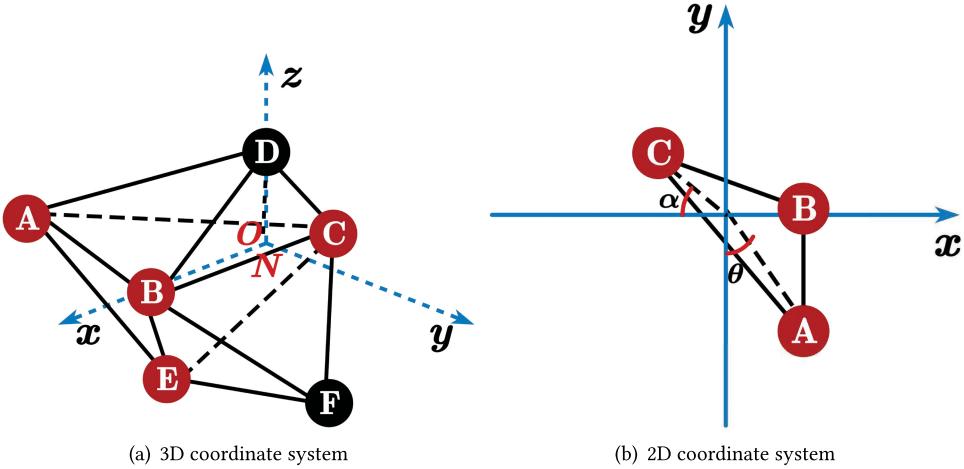


Fig. 17. The effect of establishing the coordinate system.

Solving the Equation (15) can then get all coordinates of F .

$$\begin{aligned}
 &F_1(x_1, y_1, z_1) \\
 &F_2(x_1, y_1, z_2) \\
 &F_3(x_2, y_2, z_3) \\
 &F_3(x_2, y_2, z_4) \\
 &\begin{cases} x_1 = \frac{OA \cos \theta L_1 - OC \cos \alpha L_2}{2(-OB + OA \sin \theta)OC \cos \alpha - 2(OB + OC \sin \alpha)OA \cos \theta} \\ y_1 = \frac{L_1 + (2OB + OC \sin \alpha)x}{2OC \cos \alpha} \end{cases} \\
 &\begin{cases} x_2 = \frac{OA \cos \theta L_1 - OC \cos \alpha L'_2}{2(-OB + OA \sin \theta)OC \cos \alpha - 2(OB + OC \sin \alpha)OA \cos \theta} \\ y_2 = \frac{L_1 + (2OB + OC \sin \alpha)x}{2OC \cos \alpha} \end{cases} \\
 &\begin{cases} z_1 = \sqrt{BF^2 - (x_1 - OB^2) - y_1^2} \\ z_2 = -\sqrt{BF^2 - (x_1 - OB^2) - y_1^2} \\ z_3 = \sqrt{BF^2 - (x_2 - OB^2) - y_2^2} \\ z_4 = -\sqrt{BF^2 - (x_2 - OB^2) - y_2^2} \end{cases}
 \end{aligned} \tag{16}$$

So the four possible lengths of DF are expressed as follows.

$$\begin{aligned}
 DF_1 &= \sqrt{x_1^2 + y_1^2 + (z_1 - h)^2} \\
 DF_2 &= \sqrt{x_1^2 + y_1^2 + (z_2 - h)^2} \\
 DF_3 &= \sqrt{x_2^2 + y_2^2 + (z_3 - h)^2} \\
 DF_4 &= \sqrt{x_2^2 + y_2^2 + (z_4 - h)^2}
 \end{aligned} \tag{17}$$

REFERENCES

- [1] Yuanpeng Liu, Yunlong Wang, Jian Wang, and Yuan Shen. 2020. Distributed 3D relative localization of UAVs. *IEEE Transactions on Vehicular Technology* 69, 10 (2020), 11756–11770.

- [2] Shuo Wang, Yongcai Wang, Xuewei Bai, and Deying Li. 2023. Communication efficient, distributed relative state estimation in UAV networks. *IEEE Journal on Selected Areas in Communications* 41, 4 (2023), 1151–1166. DOI : <http://dx.doi.org/10.1109/JSAC.2023.3242708>
- [3] Shuo Wang, Yongcai Wang, Deying Li, and Qianchuan Zhao. 2023. Distributed relative localization algorithms for multi-robot networks: A survey. *Sensors* 23, 5 (2023), 2399. DOI : <http://dx.doi.org/10.3390/s23052399>
- [4] Yongcai Wang, Tianyuan Sun, Guoyao Rao, and Deying Li. 2018. Formation tracking in sparse airborne networks. *IEEE Journal on Selected Areas in Communications* 36, 9 (2018), 2000–2014.
- [5] Zheng Yang, Yunhao Liu, and X-Y Li. 2009. Beyond trilateration: On the localizability of wireless ad-hoc networks. In *Proceedings of IEEE INFOCOM 2009*. IEEE, Los Alamitos, CA, 2392–2400.
- [6] Gabriele Oliva, Stefano Panzieri, Federica Pascucci, and Roberto Setola. 2015. Sensor networks localization: Extending trilateration via shadow edges. *IEEE Transactions on Automatic Control* 60, 10 (2015), 2752–2755.
- [7] Sayit Korkmaz and Alle-Jan van der Veen. 2009. Robust localization in sensor networks with iterative majorization techniques. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, Los Alamitos, CA, 2049–2052.
- [8] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. 2011. G2O: A general framework for graph optimization. In *Proceedings of 2011 IEEE International Conference on Robotics and Automation (ICRA'11)*. IEEE, Los Alamitos, CA, 3607–3613.
- [9] H. Ping, Y. Wang, and D. Li. 2020. HGO: Hierarchical graph optimization for accurate, efficient, and robust network localization. In *Proceedings of the 29th International Conference on Computer Communications and Networks (ICCCN'20)*. 1–9. DOI : <http://dx.doi.org/10.1109/ICCCN49398.2020.9209620>
- [10] Ananya Saha and Buddhadeb Sau. 2017. Network localization by non-convex optimization. In *Proceedings of the 7th ACM International Workshop on Mobility, Interference, and MiddleWare Management in HetNets*. 1–6.
- [11] Xiaoping Wang, Jun Luo, Yunhao Liu, Shanshan Li, and Dezun Dong. 2010. Component-based localization in sparse wireless networks. *IEEE/ACM Transactions on Networking* 19, 2 (2010), 540–548.
- [12] Tianyuan Sun, Yongcai Wang, Deying Li, Zhaoquan Gu, and Jia Xu. 2018. WCS: Weighted component stitching for sparse network localization. *IEEE/ACM Transactions on Networking* 26, 5 (2018), 2242–2253.
- [13] Lei Zhang, Ligang Liu, Craig Gotsman, and Steven J. Gortler. 2010. An as-rigid-as-possible approach to sensor network localization. *ACM Transactions on Sensor Networks* 6, 4 (2010), 1–21.
- [14] Tianyuan Sun, Yongcai Wang, and Deying Li. 2020. A survey and evaluation of graph realization algorithms. *Acta Automatica Sinica* 46, 4 (2020), 613–630.
- [15] Yuan Zhang, Shutang Liu, Xiuyang Zhao, and Zhongtian Jia. 2012. Theoretic analysis of unique localization for wireless sensor networks. *Ad Hoc Networks* 10, 3 (2012), 623–634.
- [16] Z. Yang and Y. Liu. 2012. Understanding node localizability of wireless ad hoc and sensor networks. *IEEE Transactions on Mobile Computing* 11, 8 (Aug. 2012), 1249–1260. DOI : <http://dx.doi.org/10.1109/TMC.2011.122>
- [17] Mihai Cucuringu, Yaron Lipman, and Amit Singer. 2012. Sensor network localization by eigenvector synchronization over the Euclidean group. *ACM Transactions on Sensor Networks* 8, 3 (2012), 1–42.
- [18] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. 1990. Unit disk graphs. *Discrete Mathematics* 86, 1-3 (1990), 165–177.
- [19] Onur Cagirici. 2016. Avoiding the flip ambiguities in 2D wireless sensor localization by using unit disk graph property. *arXiv preprint arXiv:1604.03396* (2016).
- [20] Haodi Ping, Yongcai Wang, Deying Li, and Tianyuan Sun. 2020. Flipping free conditions and their application in sparse network localization. *IEEE Transactions on Mobile Computing* 21, 3 (2020), 986–1003. DOI : <http://dx.doi.org/10.1109/TMC.2020.3015480>
- [21] Anushya A. Kannan, Baris Fidan, and Guoqiang Mao. 2011. Use of flip ambiguity probabilities in robust sensor network localization. *Wireless Networks* 17, 5 (2011), 1157–1171.
- [22] Anushya A. Kannan, Baris Fidan, and Guoqiang Mao. 2010. Analysis of flip ambiguities for robust sensor network localization. *IEEE Transactions on Vehicular Technology* 59, 4 (2010), 2057–2070.
- [23] David Moore, John Leonard, Daniela Rus, and Seth Teller. 2004. Robust distributed network localization with noisy range measurements. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*. 50–61.
- [24] Bill Jackson and Tibor Jordán. 2005. Connected rigidity matroids and unique realizations of graphs. *Journal of Combinatorial Theory, Series B* 94, 1 (2005), 1–29. DOI : <http://dx.doi.org/10.1016/j.jctb.2004.11.002>
- [25] Qingbei Guo, Yuan Zhang, Jaime Lloret, Burak Kantarci, and Winston K. G. Seah. 2018. A localization method avoiding flip ambiguities for micro-UAVs with bounded distance measurement errors. *IEEE Transactions on Mobile Computing* 18, 8 (2018), 1718–1730.
- [26] Kevin M. Lillis. 2008. *Improved Robustness of Topology Control and Routing Algorithms for Ad-Hoc Wireless Sensor Networks*. The University of Iowa.

- [27] Yunhao Liu, Zheng Yang, Xiaoping Wang, and Lirong Jian. 2010. Location, localization, and localizability. *Journal of Computer Science and Technology* 25, 2 (2010), 274–297.
- [28] Francesco Sottile and Maurizio A. Spirito. 2008. Robust localization for wireless sensor networks. In *Proceedings of the 2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks*. IEEE, Los Alamitos, CA, 46–54.
- [29] Tianyuan Sun, Yongcai Wang, Deying Li, Wenping Chen, and Zhaoquan Gu. 2018. Robust component-based network localization with noisy range measurements. In *Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN'18)*. IEEE, Los Alamitos, CA, 1–9.
- [30] Xiaoping Wang, Zheng Yang, Jun Luo, and Changxiang Shen. 2011. Beyond rigidity: Obtain localisability with noisy ranging measurement. *International Journal of Ad Hoc and Ubiquitous Computing* 8, 1-2 (2011), 114–124.
- [31] Wei Liu, Enqing Dong, Yang Song, and Dejing Zhang. 2014. An improved flip ambiguity detection algorithm in wireless sensor networks node localization. In *Proceedings of the 2014 21st International Conference on Telecommunications (ICT'14)*. IEEE, Los Alamitos, CA, 206–212.
- [32] Wei Liu, Enqing Dong, and Yang Song. 2016. Analysis of flip ambiguity for robust three-dimensional node localization in wireless sensor networks. *Journal of Parallel and Distributed Computing* 97 (2016), 57–68.
- [33] Donald J. Jacobs. 1998. Generic rigidity in three-dimensional bond-bending networks. *Journal of Physics A: Mathematical and General* 31, 31 (1998), 6653.
- [34] Robert Connelly. 2005. Generic global rigidity. *Discrete & Computational Geometry* 33, 4 (2005), 549–563.
- [35] Tolga Eren, O. K. Goldenberg, Walter Whiteley, Yang Richard Yang, A. Stephen Morse, Brian D. O. Anderson, and Peter N. Belhumeur. 2004. Rigidity, computation, and randomization in network localization. In *Proceedings of IEEE INFOCOM 2004*, Vol. 4. IEEE, Los Alamitos, CA, 2673–2684.
- [36] David Kiyoshi Goldenberg, Arvind Krishnamurthy, Wesley C. Maness, Yang Richard Yang, Anthony Young, A. Stephen Morse, and Andreas Savvides. 2005. Network localization in partially localizable networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 1. IEEE, Los Alamitos, CA, 313–326.
- [37] Donald J. Jacobs and Bruce Hendrickson. 1997. An algorithm for two-dimensional rigidity percolation: The pebble game. *Journal of Computational Physics* 137, 2 (1997), 346–365.
- [38] John E. Hopcroft and Robert Endre Tarjan. 1973. Dividing a graph into triconnected components. *SIAM Journal on Computing* 2, 3 (1973), 135–158.
- [39] Dong Wen, Lu Qin, Ying Zhang, Lijun Chang, and Ling Chen. 2019. Enumerating k -vertex connected components in large graphs. In *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE'19)*. IEEE, Los Alamitos, CA, 52–63.
- [40] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2018. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*. Springer Proceedings in Advanced Robotics, Vol. 5. Springer, 621–635.
- [41] Jan De Leeuw. 2005. *Applications of Convex Analysis to Multidimensional Scaling*. University of California, Los Angeles.

Received 6 November 2022; revised 17 April 2023; accepted 20 June 2023