Flipping Free Conditions and Their Application in Sparse Network Localization

Haodi Ping, Yongcai Wang, Deying Li, and Tianyuan Sun

Abstract—Inferring network topology via inter-node distance measurements is an important problem. It is challenging when the distance measurements are sparse because the lack of edge constraints may lead to ambiguous realizations that differ greatly from the ground truth. The flipping ambiguities are caused by binary vertex cut sets in 2D and triple vertex cut sets in 3D, which are called *separators*. This paper investigates conditions on whether the flipping ambiguities caused by these separators can be disambiguated using neighborhood, full graph, and component-level conditions. Accordingly, local flipping-free condition (LFFC), global flipping-free condition (GFFC), and component-based flipping free condition (CFFC) are proposed. Then a disambiguating framework based on a combinatorial application of these conditions is proposed. It detects separators and first disambiguate separators locally by LFFC, which converts the graph to a binary tree, whose leaf nodes are flipping-free components and edges are LFFC unsolvable separators. Then the CFFC condition is further applied to disambiguate LFFC unsolvable separators between components. If k and g separators are disambiguated by LFFC and CFFC respectively, the number of ambiguous solutions for network localization will be reduced by 2^{k+g} times. Finally, the flipping-free components realize node coordinates in their local coordinate systems and a residue-based weighted component stitching algorithm (RWCS) is proposed to iteratively synchronize components' local coordinates to generate global coordinates of the network. Extensive simulations show the LFFC, CFFC and RWCS frameworks are efficient, which resolve a major portion of flipping ambiguities and greatly improve the localization accuracy than the state of art algorithms in various sparse network settings.

Index Terms—Network localization, negative edges, sparse networks, flipping free condition, tree of non-flipping components, component synchronization

1 Introduction

With the rapid development of information technologies such as fifth-generation (5G) mobile networks and the increasing popularity of smart things such as sensors, robots, and Unmanned Aerial Vehicles (UAVs), there is potential for networking vast numbers of heterogeneous devices to form Internet of Things (IoT)[1], [2]. *Network localization* is a crucial problem in many IoT applications, such as multi-agent formation control [3], [4], wireless sensor networks [5], [6], emergency services [7], [8], and structural biology [9]. It infers node coordinates by a given partially measured inter-node distance matrix and coordinates of some anchor nodes.

The problem is known as *graph realization*,[10] when there is no anchor, which focuses on inferring the geometric structure of the network that best satisfies the distance matrix (D). This paper studies the disambiguation problem, which is a key challenge in both *network localization* and *graph realization*. Localization and realization will be used interchangeably since the realized structure can be transformed into the global coordinate system by selecting not less than $d+1$ non-collinear nodes as anchors, where d is the space dimension.

Solution uniqueness is a key challenge in graph realization [11], which requires the graph hasn't other ambiguous realizations that also satisfy D. Note that rigid transformations, i.e., the global rotation, translation, and reflection don't change the inner structure of a realization. Two realizations $\mathbf{P}$ and $\mathbf{P}'$ of a graph are said ambiguous if their inter-node distances both satisfy D, but $\mathbf{P}'$ cannot be rigidly transformed to $\mathbf{P}$.

Existing studies show that being global rigid [10] is the necessary and sufficient condition for the graph to be uniquely realizable. The necessary condition for being global rigid requires the graph to be redundant rigid and $(d+1)$-connected [12], where d is the space dimension. Algorithms that check uniqueness of network localization are proposed in [6], [11], [13]. Common agreements of these studies are that: (1) the essence of being global rigid requires enough distance measurements in D to restrict the node coordinates' freedoms; (2) lacking enough distance measurements may cause ambiguous solutions, which generate large errors in formation calculation.

However, measurement sparsity is inevitable in practice. The main reasons are: (1) reducing node deployment cost is always required in practice, which generally results at sparse networks; (2) the limited scope of ranging techniques, such as signal round-trip time [14] or signal time of arrival (TOA) [15] all have limited ranging scope; (3) the uneven node distribution in random deployment may also cause some parts of the graph to be sparse, even if the overall density is fine. When a portion of nodes has ambiguous solutions, the errors may

- *Haodi Ping, Yongcai Wang, and Deying Li are with the School of Information, Renmin University of China, Beijing 100872, P.R.China. E-mail: {haodi.ping, ycw, deyingli}@ruc.edu.cn.*
- *Tianyuan Sun is with the HTC Research Beijng, Beijng 100084, P.R. China. E-mail: Tianyuan_Sun@htc.com.*

Manuscript received 27 Jan. 2020; revised 18 July 2020; accepted 30 July 2020. Date of publication 11 Aug. 2020; date of current version 3 Feb. 2022. (Corresponding author: Yongcai Wang.)
Digital Object Identifier no. 10.1109/TMC.2020.3015480

impact the realization results of the whole graph due to iterations in the graph optimization process [16], [17], [18].

To deal with sparsity, various theories and methods have been proposed. The detailed related works are in Section 2.3. A common idea is to exploit the knowledge provided by the *negative edges* to form additional constraints. An edge (i, j) is said *positive* if the distance between i, j is measured and is said *negative* if i, j are out of the ranging scope. Existing works add the negative edges' inequality constraints into the optimization problem to form a constrained optimization problem[19], or divide the graph into patches to infer the lengths of the negative edges in each patch by *Triangle Inequality* condition [9], [20]. Although these methods improve the localization performances, they haven't provided an in-deep investigation to the local and global structural properties of where ambiguities may happen. This paper proposes local, global, and component-based conditions on where ambiguity may happen, and methods to eliminate the ambiguities.

This paper provides two key observations: (1) $(d+1)$-connected components are flipping-free for generic networks in $\mathbb{R}^d$ ($d = \{2, 3\}$); (2) a local condition can be designed to infer the exact length of the negative edge in a *basic flipping graph (BFG)* with high probability, which disambiguates the flipping ambiguity in the BFG. These two conditions work in graph level and neighborhood level respectively and can be used in combination to greatly reduce the graph realization ambiguities. The key contributions are as follows:

1) We first propose *basic flipping graphs (BFG)* in 2D and 3D, which are the minimum size graphs in which flipping ambiguity may happen. Then a *local flipping free condition (LFFC)* in BFGs is proposed, which can infer the exact length of the negative edge in the BFG, so that the flipping ambiguity of the BFG can be disambiguated with high probability. LFFC is a local condition using one-hop neighborhood information. We show the advantages of LFFC than the *traditional triangle inequality (TI)* condition in resolving more flipping cases.
2) Then we prove a *global flipping free condition (GFFC)* in the graph level. It proves that the $d+1$ connected generic graphs in $\mathbb{R}^d$ are flipping-free for $d \in \{2, 3\}$. So in a rigid graph, each $(d+1)$-connected component is flipping free. Flipping may happen only at the d-vertex cut sets, which are called *flipping separators*. We further propose a *Component-based Flipping Free Condition (CFFC)* to determine whether flipping ambiguity may happen at a *flipping separator*.
3) By utilizing CFFC and LFFC in combination, a *flipping separator detection, LFFC checking, and non-flipping tree (NC-Tree) construction* method is proposed. The algorithm starts by finding a flipping separator and checks LFFC on it. If LFFC is TRUE, the flipping ambiguity is eliminated and another flipping separator will be found and be checked. If LFFC is FALSE, the graph is separated into two sub-graphs by the flipping separator. The result of the algorithm partitions the network into a binary tree, called NC-Tree, whose leaf nodes are $(d+1)$-connected non-flipping components and edges are LFFC unsolvable flipping separators.
4) CFFC condition is further applied on the NC-Tree to resolve LFFC unsolvable separators using the multi-hop geometrical condition. If k and g separators are resolved by LFFC and CFFC respectively, the number of ambiguous realizations of the network will be reduced by 2^{k+g} times. All feasible realizations can be inferred from the NC-Tree.
5) At last, to generate a realization with better tolerance to ranging noises, a *Residue-based Weighted Component Stitching (RWCS)* algorithm is developed. It first calculates the local coordinates of each non-flipping components in their local coordinate frames. Then the local coordinates are synchronized by iterative rotation and transition until convergence to produce the global coordinates of the graph. We show by extensive simulations that the proposed LFFC, CFFC and RWCS framework resolves flipping ambiguities efficiently and improves the network realization accuracy greatly in sparse networks than the state of art algorithms in various network settings.

The rest of the paper is organized as follows. In Section 2, the problem model and related works are introduced. The flipping free conditions are introduced in Section 3. The LFFC checking and NC-Tree construction algorithm is introduced in Section 4. The application of CFFC is introduced in Section 5. The RWCS algorithm is presented in Section 6. Algorithm analyses are given in Section 7. We present the experimental evaluation in Section 8 and conclude the paper with remarks in Section 9.

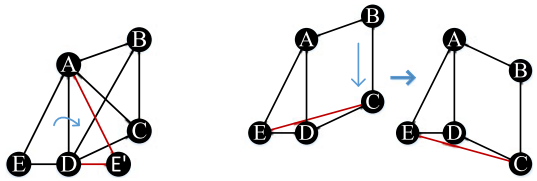
2 Preliminaries and Background

2.1 Notations

The measurement graph is denoted by $G = (V, E, D)$, where $n = |V|$ and $m = |E|$ are number of nodes and number of edges respectively. D is the measurement matrix. The true vertex coordinates are denoted by $\mathbf{P} = \{\mathbf{p_1}, \ldots, \mathbf{p_n}\}$ which are unknown. ($\mathbf{p}_i \in \mathbb{R}^d$) and d is the space dimension, which is 2 or 3. $(i, j) \in E$ and $d_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|_2 + \sigma_{ij}$ if $\|\mathbf{p}_i - \mathbf{p}_j\|_2 \leq R$ where R is a ranging scope and σ_{ij} is the ranging noise. (i, j) is called a *negative edge* if the true distance is beyond the ranging scope. The objective of the problem is given in (1):

$$\min_{\mathbf{p}_1, \cdots \mathbf{p}_n} \left\{ f(\mathbf{p}_1, \cdots \mathbf{p}_n) = \sum_{(i,j)\in \mathbf{E}} \left(\left\|\mathbf{p}_i - \mathbf{p}_j\right\|_2 - d_{ij} \right)^2 \right\} \quad s.t. \quad \left\|\mathbf{p}_i - \mathbf{p}_j\right\|_2^2 > R, \forall (i,j) \notin E. \tag{1}$$

Notations used throughout this paper are as follows. $\mathbf{V}(G)$ returns the vertex set in G. $\mathbf{E}(G)$ returns the edge set. $\mathbf{G}[E]$ returns the graph generated by the edges in E with their connected vertices. $\mathbf{G}[V]$ returns the graph generated by the vertices in V with their inter edges. $N[v]$ is the one-hop neighbor graph of a vertex v. $\setminus$ denote set minus. n is the number of nodes and m is the number of edges. $\mathbf{q}$ denotes the local coordinates of a node and $\mathbf{Q}$ denotes local coordinates of a sub-graph. $\mathbf{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_{n_c}\}$ denotes the graph components and $\mathbf{Q} = \{\mathbf{Q}_1, \ldots, \mathbf{Q}_{n_c}\}$ are local coordinates of each components; n_c is the number of the components. $\mathbf{s}$ denotes a separator; $\mathbf{R}$ denotes a rotation matrix and $\mathbf{T}$ is a transition matrix. The terms *graph realization* and *network*



(a) Flip ambiguity (b) Flex ambiguity

Fig. 1. (a) Flip ambiguity. Vertex E can be reflected across AD while satisfying all distance constraints. (b) Flex ambiguity. If edge CE is removed, then rejoined, the graph can flex along the arrow, obtaining a different topology but exactly preserving all distance constraints.

localization are used interchangeably. The term "a separator s is *eliminated/resolved/disambiguated* by LFFC" means the length of a negative edge is inferred by LFFC thus the ambiguity caused by s no longer exists. CFFC disambiguates s by finding one realization among the two possible realizations of two components $\mathbf{Q_l}$ and $\mathbf{Q_r}$ is unfeasible.

2.2 Assumptions

The considered graphs are assumed generic. Generic graphs are graphs where the node coordinates are algebraically independent over the rationals and are dense in space [21], which are widely assumed in network localization studies [12]. G is considered a simple graph without multiple edges between nodes nor self-loops.

G is also assumed rigid [22], because if G is not rigid, there will be one or more nodes who can change their coordinates continuously in a neighborhood without violating the distance constraints. Such a non-rigid graph will have an unlimited number of ambiguous realizations, which is not possible to find a unique realization. Note that a rigid graph in $\mathbb{R}^d$ is at least d-connected [22]. So the rigid graph assumption implies the considered graph is d-connected. Note that d-connected rigid graphs are still very sparse, e.g., a 2-connected rigid graph may still have many ambiguous realizations due to flipping and flex ambiguities as shown in Fig. 1.

2.3 Related Works

To tackle realization ambiguity and to improve localization accuracy, great attentions have been attracted. The literature related to this article is classified as follows.

2.3.1 Rigidity and Topology Based Methods

Whether a network can have unique realization solution has been widely investigated via rigidity theory by Jackson [21], Goldenberg [11], Aspnes [23], and Yang [13], [24] *et al.* A network can be uniquely realized if and only if the underlying graph is global rigid [12]. Global rigidity requires the graph to be $(d+1)$-connected and redundant rigid [13]. The necessary and sufficient condition for a 2-D graph being rigid is given by Laman [25] in 1979. Later in 1997, Jacobs proposed Pebble game [25] to test 2-D graph rigidity in polynomial time. Connelly [22] showed that the Laman condition is only necessary in $\mathbb{R}^d$ for $d \geq 3$. The rank of stress matrix can be used as necessary and sufficient conditions for judging graph rigidity in a higher dimension, which is proposed by Connelly [22], Gortler *et al.* [26]. They show that a graph in $\mathbb{R}^d, d \geq 3$ is rigid if and only if $rank(\mathbf{\Omega}) = n-1-d$, where $\mathbf{\Omega}$ is the associated stress matrix derived from the graph. Gortler *et al.* [26] proposes polynomial-time random algorithms for verifying graph rigidity in higher dimensions. The rigidity theory has also been extended to topology analysis to determine localizability. Yang *et al.* [13], [24] propose the necessary and sufficient condition for a node in the network to be uniquely localizable. The condition is based on the number of vertex disjoint paths to anchor nodes. Shamsi *et al.* [27] study the conditions for correct network localization by SDP relaxation and show the sparse triangulation graph can ensure the correctness of SDP relaxation.

2.3.2 Geometric-Based Ambiguity Reduction in Trilateration and Biliteration

Above rigidity analysis generally requires exact distance information. To consider noise impacts, geometric methods are investigated in the literature to avoid flipping ambiguities, mainly in trilateration methods. Kannan [28] *et al.* point that flipping may still happen even in global rigid graphs for the noise impacts, e.g., when three reference nodes are nearly collinear. They propose a robustness criterion to detect flip ambiguities in neighborhood geometries. Wang *et al.* [29] show that flipping detection equals to check whether there is a straight line intersecting with all range error circles, which is called the existence of intersecting line (EIL) problem and a convex hull algorithm is proposed to solve EIL. Liu *et al.* [30] improve the EIL detection by showing it equals to determine whether there is a straight line, which enables any two ranging circles to have an overlapping orthogonal projection on the line. Liu *et al.* [31] further extends the problem to 3D. Guo *et al.* [32] exploit the bounded errors of distance measurements and the constraints of motions. They propose flipping avoidance conditions for biliteration and trilateration for error bounded measurements. Other trilateration based algorithms reduce the probability of flipping happens by selecting "robust quadrilaterals" [33], [34] and "safe-triangle" [35]. Another direction is to expand node localizability scope in sparse networks. Biliteration [36] and shadow edge [37] methods are proposed, but ambiguity risks also increase in such methods since they relax the global rigid requirement in node localization. Error accumulation is also a big challenge to the sequential trilateration or biliteration methods in large networks.

2.3.3 Ambiguity Reduction in Iterative Optimization Methods

To avoid error accumulation, a large body of studies propose iterative optimization methods [16], [17]. They treat network localization as a graph optimization problem. In iterative optimization, ambiguity is mainly tackled by utilizing the negative edges as constraints. Saha *et al.* [19] covert the optimization problem with negative edge constraints to a Lagrangian optimization problem and propose a root-finding algorithm. [38] exploits the negative constraints to sharpen the probabilistic distributions of the target locations. [39] proposes a convex optimization model based on the euclidean Distance Matrix (EDM). The negative constraints are represented as lower and upper bounds on the elements in the distance matrix to reduce the localization flexibility. [40] proposes to find the set of non-locatable nodes and constructs an SDP formulation by adding negative constraints to the non-locatable nodes, so as to

avoid ambiguous solutions. [41] employs a global consistency check and a location correction phase in the localization process to deal with flip ambiguity. Shi *et al.* [42] formulate an optimization problem to minimize the worst-case estimation error and propose a distributed algorithm to solve the problem. SDP[18] approaches localization by relaxing the problem to a convex semidefinite programming problem. Xiao *et al.* [43] propose a matrix completion based optimization method for noise-tolerance.

2.3.4 Component Stitching Methods for Sparse Network Localization

But the iterative optimization methods are sensitive to the network sparsity. It is because the errors generated in the sparse sub-graph will affect other parts in the iterative optimization process. To overcome this, component stitching based methods are proposed to further deal with noises and sparsity. ETOC [44] and CALL [45] study component merging conditions and propose non-iterative algorithms to merge components. To better tolerate noises, iterative component stitching algorithms are proposed in ARAP [20], ASAP [9], WCS [47] and WCKF [4] . ARAP [20] and ASAP [9] divide component by one-hop neighborhood of each node. WCS [46] divides components according to node density. WCKF [4] divides components in 3D networks by finding two-center star graphs. After partition, components are realized in local coordinate systems and then synchronized by iterative registration and least square estimation. Experiments show that component stitching methods provide great accuracy and reliability improvement than trilateration methods and centralized iterative optimization methods in sparse and noisy networks.

2.3.5 Linear Barycentric Coordinate-Based Methods for Efficient Network Localization

Above models consider the localization problem as a nonlinear least square estimation problem [47]. Recent works have also proposed to utilize barycentric coordinates to convert the localization problem to a linear model, by iterations of a set of linear equations in a fully distributed manner, see [48] for comprehensive analysis. DILOC [50] calculates the barycentric coordinates using the Cayley–Menger determinants, then node coordinates are iteratively estimated by a linear function of reference nodes' coordinates with weights derived from barycentric coordinates. DLRE [49] and DILAND [50] extend DILOC to random environments and make DILOC robust to communication noise, communication failure, and measurement noise. ECHO [51] investigates signed barycentric coordinates on triangles, which does not require that each node should be located inside the convex hull of its neighbors (the main limitation of DILOC). The computation of barycentric coordinates for any possible network configuration is also discussed in [52], which further extends barycentric coordinates to any dimension. But these studies mainly study in dense networks, for a node needs enough neighbors to calculate the barycentric coordinates.

2.3.6 Applications and Further Optimization

Much literature has been reported for application and further optimization of network localization. Recent studies refer this problem as *network localization and navigation* (NLN)[53], [54], [55], [567], [57], [58], which concentrate on promoting practical localization in a cooperative manner. References [55], [56] provide theoretical foundation and practical issues. Win *et al.* [53], [54], [56] yield new information by introducing joint spatial and temporal cooperation. The reliability of localization results is enhanced since additional information is exploited. Network operation strategies such as node prioritization, node activation, and node deployment are considered in [57], [58] for better localization performance and prolonging the network lifetime. To tackle the reliability degradation caused by multi-path propagation and non-line-of-sight conditions, the ensemble of positional and environmental information is investigated as *soft information* to leverage measurements and contextual data [59], [60], [61]. The key idea is to rely on all probable range measurement values rather than on a single estimate of each measurement.

However, explicit local and component-based conditions to detect and to resolve flipping ambiguities are still lacked. This paper investigates the flipping conditions and proposes a jointly detecting, disambiguiting and localization framework.

3 Flipping Free Conditions

A rigid graph may still has *flip ambiguity* and *flex ambiguity*, which cause *discontinuously deformed realizations* as shown in Fig. 1. *Flip ambiguity* happens when a sub-graph can flip across an axis without violating the edge constraints. *Flex ambiguity* happens if we remove one edge to deform the graph continuously to another realization, and the removed edge can be added back without violating the length constraints. A generic framework is *global rigid* if any two realizations of it are congruent, i.e., any two realizations of it can be transformed to be a unique realization by rigid rotation, transition or reflection. A global rigid generic graph has a unique realization[12], [26].

3.1 BFG: Basic Flipping Graph

In practical applications, even if a graph is rigid, the composition of the discontinuous deformations may still cause a huge number of ambiguous realizations. Since flex ambiguity has much more rigorous conditions to happen than the flipping ambiguities and can be avoided in redundantly rigid components as discussed in our early work [46], we, therefore, focus on resolving flipping ambiguity in this paper. We first consider basic flipping components in $\mathbb{R}^2$ and $\mathbb{R}^3$ where flipping ambiguity may happen.

Proposition 1 (Basic Flipping Graph: BFG). *A four vertex, five edge component containing a flipping edge as shown in Fig. 2a, and five vertexes, nine-edge component containing a flipping face as shown in Fig. 2b are the simplest rigid graphs that may have flipping ambiguity, i.e., rigid graphs having the least number of vertices and edges that may have flipping ambiguity in $\mathbb{R}^2$ and $\mathbb{R}^3$ respectively.*

Proof. The proposition is proved from three aspects. (1) Graphs whose vertex number $n < 4$ in $\mathbb{R}^2$ and $n < 5$ in $\mathbb{R}^3$ can all be enumerated. These graphs are either global rigid (point, bar, triangle, and tetrahedron), in which flipping ambiguity cannot happen, or non-rigid. (2) From

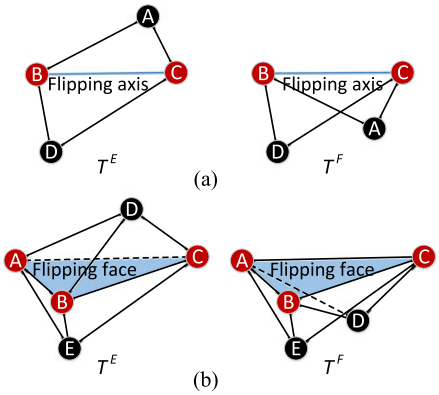


Fig. 2. Basic flipping graphs in $\mathbb{R}^2$ and $\mathbb{R}^3$.

rigidity theory [62], when a graph has n vertices it needs at least $nd - \binom{d+1}{2}$ edges to be rigid in $\mathbb{R}^d$. So when $n = 4$ in $\mathbb{R}^2$ and $n = 5$ in $\mathbb{R}^3$, at least 5 and 9 edges are required respectively for being rigid. (3) The condition for flipping happening is that the graph is disconnected by removing d vertices. When $n = 4, m = 5, d = 2$, the topology in Fig. 2a is the only topology and it satisfies the flipping happening condition. When $n = 5, m = 9, d = 3$, removing the three vertices on the flipping face in Fig. 2b will disconnect the graph and other typologies with $n = 5, m = 9, d = 3$ cannot be disconnected by removing three vertices. □

Note that each BFG has one negative edge. A BFG has two and only two possible ambiguous realizations. Fig. 2 shows the two realizations T^E and T^F for a BFG respectively. T^E is the *expanded topology* in which the two nodes with degree d are on the different sides of the flipping axis (face) and T^F is the *folding topology* in which the two nodes with degree d are on the same side of the flipping axis (face) in $\mathbb{R}^d$.

3.2 LFFC: Local Flipping-Free Condition in BFG

3.2.1 Local Flipping-Free Condition

In a BFG, by using the negative edge constraint, i.e., $d_{ij} > R$ if $(i, j) \notin E$, a condition can be designed to eliminate the flipping ambiguities in most of the BFGs, which confidently determines the unique realization from the two. The length of the negative edge is also inferred if the condition is true. For clarity, we present Local Flipping-Free Condition (LFFC) in BFG in $\mathbb{R}^2$. The condition for $\mathbb{R}^3$ is given in Appendix C, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2020.3015480.

Lemma 1. *In a BFG in $\mathbb{R}^2$, there must be x^E strictly larger than x^F, where x^E and x^F indicate the length of the negative edge in T^E and T^F respectively.*

Proof. As shown in Fig. 3, A' is the symmetric point of A about the axis BC. Thus AA' is perpendicular to BC. Let DH be perpendicular to AA' and intersect at H. The following equations can be obtained:

$$\begin{aligned} x^E &= \sqrt{h^2 + AH^2} \\ x^F &= \sqrt{h^2 + A'H^2}. \end{aligned} \tag{2}$$

It can be seen that $AH > A'H$. $AH = A'H$ only when B, C and D are collinear which is against the assumption that the r-Quad is generic. Thus $x^E > x^F$. □

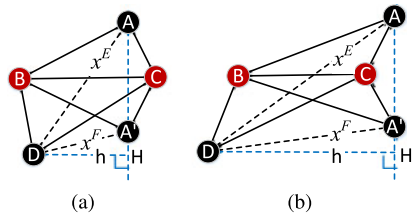


Fig. 3. The relationships of x^E and x^F.

Theorem 1 (LFFC: Local Flipping-Free Condition in BFG). *In a BFG with two ambiguous realizations, as T^E and T^F shown in Fig. 2, T^E is the unique realization, if*

$$x^E > R \text{ and } x^F \leq R, \tag{3}$$

where

$$\begin{aligned} x^E &= \sqrt{a^2 + c^2 - 2ac \times \cos(\alpha + \beta)} \\ x^F &= \sqrt{a^2 + c^2 - 2ac \times \cos(|\alpha - \beta|)} \\ \alpha &= \arccos\left(\frac{a^2 + e^2 - b^2}{2ae}\right), \beta = \arccos\left(\frac{c^2 + e^2 - d^2}{2ce}\right), \end{aligned} \tag{4}$$

e is the length of the flipping axis. a, d, b, c are lengths of two pairs of opposite edges, where a and d, b and c are the opposite edge of each other as shown in Fig. 4.

The proof of Theorem 1 is given in Appendix A, available in the online supplemental material. Using the same idea, the LFFC condition for BFG in $\mathbb{R}^3$ and its proof are given in Theorem 7 in Appendix C, available in the online supplemental material. Note that Theorem 1 and Theorem 7 not only give a clear condition of when T^E is the only feasible realization for $\mathbb{R}^2$ and $\mathbb{R}^3$, but also infer the length of the negative edge.

3.2.2 Reformulating the Triangle Inequality Condition

Triangle Inequality (TI) is one important method generally exploited for filtering out ranging outliers [24] in network localization. It can also be utilized to resolve flipping ambiguity. We reformulate TI to develop another local condition for BFG disambiguating:

Theorem 2 (Triangle Inequality (TI) Condition). *In $\mathbb{R}^2$, for a BFG with two ambiguous realizations, i.e., T^E and T^F as shown in Fig. 4, if $\max\{a + d - e, b + c - e\} < R$, then T^E is the unique feasible realization.*

The proof of Theorem 2 is given in Appendix B, available in the online supplemental material. But we find that comparing with LFFC, TI provides a very conservative condition in eliminating the flipping ambiguity.

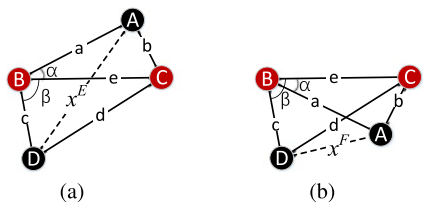


Fig. 4. (a) Calculation of x^E; (b) Calculation of x^F.

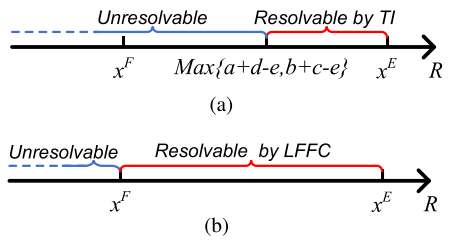


Fig. 5. The range of R for disambiguating by TI and LFFC, respectively.

3.2.3 The Advantages of LFFC than TI

Fig. 5 shows when flipping ambiguity can be eliminated as a function of R. $x^F < \max\{a+d-e, b+c-e\} < x^E$ can be derived from TI. Therefore, by TI, only when R is located in the range: $\max\{a+d-e, b+c-e\} < R < x^E$, can the ambiguity in the BFG be resolved. By LFFC, only if R is in the range: $x^F \leq R < x^E$, will the ambiguity be resolved. But note that both TI and LFFC cannot resolve the case when $R < x^E$ and $R < x^F$. In this case, the two topologies are both feasible. The ambiguity cannot be told unless other information is known. But such a case takes only a small proportion in practice.

We run simulations to generate 3000 BFGs in $\mathbb{R}^2$ randomly and apply LFFC and TI to resolve the flipping ambiguities respectively. The simulation is conducted 100 times and the average results are summarized in Table 1. By TI, the flipping ambiguities in 37.5 percent BFGs can be resolved, while LFFC can resolve ambiguities in 92.6 percent BFGs on average. The remaining 7.4 percent is the case when both $x^E > R$ and $x^F > R$, whose ambiguities cannot be determined by either TI nor LFFC.

3.2.4 LFFC Under Ranging Noises

The distance measurements are generally impacted by ranging noises. Assume $d_{i,j} = \|\mathbf{p}_i - \mathbf{p}_j\|_2^2 + \sigma_{i,j}$, where $\sigma_{i,j} \sim N(0, \sigma^2)$ is zero mean Gaussian noise. Then the ranging noises have more than 0.99 probability to be in the range of $[-3\sigma, 3\sigma]$. To tolerate ranging noises, the conditions in (3) can be modified as.

$$\begin{aligned}\sqrt{a^2+c^2-2ac\times\cos(\alpha+\beta)} &> R+3\sigma\\ \sqrt{a^2+c^2-2ac\times\cos(|\alpha-\beta|)} &\leq R-3\sigma.\end{aligned} \tag{5}$$

3.3 GFFC: Global Flipping-Free Condition on a Graph

Then we consider a condition from the graph level of when flipping ambiguity cannot happen.

Theorem 3 (Global Flipping-Free Condition: GFFC). *If a generic graph is $(d+1)$-connected in R^d ($d \in \{2,3\}$), then the graph is free of flipping ambiguity.*

Proof. We prove by contradiction. We consider the case when $d=2$ and the same process holds for $d=3$.

TABLE 1
Experiments to Test How QI and TI Can Resolve Flipping Ambiguity in Randomly Generated r-Quads

Number of BFGs	TI is TRUE	LFFC is TRUE	Both LFFC and TI fail
3000	1125	**2778**	222
100%	37.5%	**92.6%**	7.4%

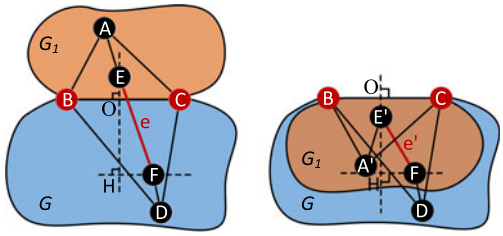


Fig. 6. If G_1 has two realizations, the length of $\mathbf{e}$ must be different in these two realizations, contradicting the known length of $\mathbf{e}$.

Consider a generic graph G, which is 3-connected, i.e., any two nodes in G have at least three node-disjoint paths. Without loss of generality, assume a subgraph $G_1 \in G$ may flip across a line formed by two vertices $\{B, C\} \in G$, as shown in Fig. 6. Then we select $A \in \mathbf{V}(G_1)$ and $D \in \mathbf{V}(G) \setminus (\mathbf{V}(G_1) \cup \{B, C\})$ who have two paths going through B and C. Since A and D are 3-connected, there must be a third path between A and D which has an edge e that penetrates the line formed by $\{B, C\}$. Since the graph is generic, the endpoint of e is not exactly on the line formed by BC; One endpoint of e is in G_1 and the other in $G \setminus G_1$. Considering the two flipping realizations of G_1, let A' and E' be the symmetric point of A and E about the axis BC respectively. Thus EE' is perpendicular to BC. Let FH be perpendicular to EE' and intersect at H. The following equations can be obtained:

$$\begin{aligned}e &= \sqrt{FH^2 + EH^2}\\ e' &= \sqrt{FH^2 + E'H^2}.\end{aligned} \tag{6}$$

Since $EH = (HO + OE) \neq E'H = (HO - OE')$, so $e \neq e'$ which contradicts to that the edge e is a positive edge with known length. □

So a $(d+1)$-connected component in $\mathbb{R}^d$ is flipping-free. From the proof of Theorem 3, if there is a d-vertex cut set in a graph, i.e., if a graph can be disconnected by removing d vertices, the graph is not $(d+1)$-connected. The sub-graph formed by the d-vertex in the cut set will form a flipping face, which is called a *flipping separator*. Flipping ambiguity may happen due to flipping across the flipping separator.

Definition 1 (Flipping Separator). *Given a rigid graph $G = (V, E)$ in $\mathbb{R}^d$, if there is a d-vertex cut set V', where $|V'| = d$ and $\mathbf{G}[V \setminus V']$ is disconnected, we call $\mathbf{G}[V']$ a* flipping separator.

In BFGs, the flipping axis in $\mathbb{R}^2$ and flipping face in $\mathbb{R}^3$ are flipping separators. A flipping separator is called resolved if only one of the two ambiguous realizations is feasible. The LFFC condition provides local methods to resolve the flipping separator in BFG.

3.4 CFFC: Component-Based Flipping-Free Condition

Then we consider a graph G containing a flipping separator $\mathbf{s} = \mathbf{G}[V']$. We denote the two disconnected sub-graphs $G_{l,\mathbf{s}}$, and $G_{r,\mathbf{s}}$ by removing the separator $\mathbf{s}$, i.e., $G_{l,\mathbf{s}} \cap G_{r,\mathbf{s}} = \emptyset$. The vertices in the two sub-graphs, and in the separator are denoted by V_l, V_r and $V_{\mathbf{s}}$ respectively. The graph have two

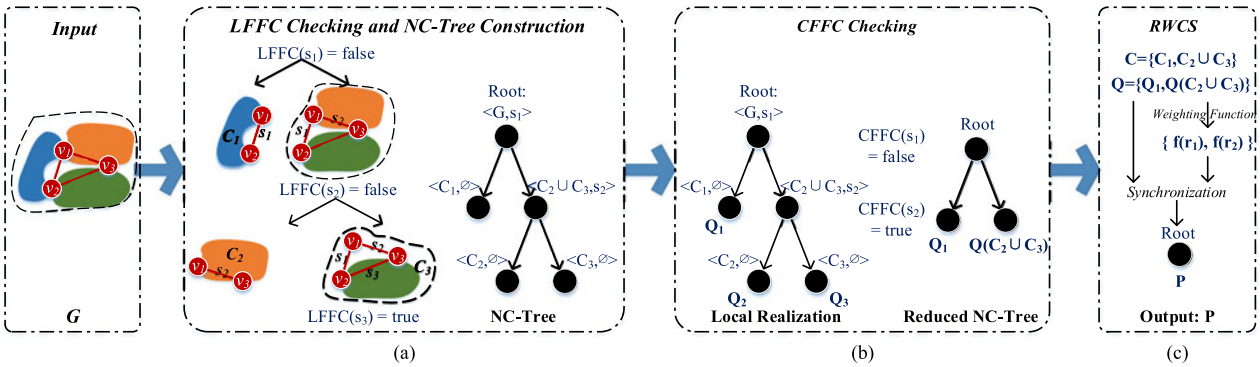


Fig. 7. An overview of the proposed scheme using a graph of three separators as an instance.

ambiguous realizations $\mathbf{P}^1$ and $\mathbf{P}^2$ due to flipping ambiguity across $\mathbf{s}$. The point coordinates are denoted by set $\mathbf{P}^1 = \{\mathbf{P}^1_l, \mathbf{P}^1_\mathbf{s}, \mathbf{P}^1_r\}$ and $\mathbf{P}^2 = \{\mathbf{P}^2_l, \mathbf{P}^2_\mathbf{s}, \mathbf{P}^2_r\}$ respectively.

Theorem 4 (Component-based Flipping Free Condition: CFFC). *For a graph G with a flipping separator $\mathbf{s}$ and two ambiguous realizations $\mathbf{P}^1$ and $\mathbf{P}^2$, for $k \in \{1, 2\}$, if*

$$\exists \mathbf{p}_i \in \mathbf{P}^k_l \text{ and } \mathbf{p}_j \in \mathbf{P}^k_r, \text{ s.t. } \|\mathbf{p}_i - \mathbf{p}_j\|_2 \leq R, \tag{7}$$

then the realization $\mathbf{P}^k$ is not feasible.

Proof. Since the two vertices $i \in V_l$ and $j \in V_r$ are disconnected, their inter-distance must be larger than R. If $\|\mathbf{p}_i - \mathbf{p}_j\|_2 \leq R$, the negative edge constraint is violated and the corresponding realization is not feasible. □

If (7) is satisfied in one of the two ambiguous realizations, this ambiguous realization is not feasible, so the other one is the unique solution. Note that these conditions only resolve flipping ambiguity. Flex ambiguity may still happen in flipping free components. But flex ambiguity has a much lower probability to happen and can be resolved by finding redundantly rigid components[46].

3.5 Overview of Disambiguating with Flipping Free Conditions

In the following sections, we exploit the proposed flipping free conditions to resolve flipping ambiguities and to generate reliable realization for sparse graphs. The key idea is to use LFFC and CFFC to check flipping separators until no flipping separators can be further resolved. For clarity, the following results are presented in $\mathbb{R}^2$. The main idea can be extended to $\mathbb{R}^3$. The overview of the combined utilization of LFFC and CFFC and the final graph realization is shown in Fig. 7.

1) At first a separator detection, LFFC checking and non-flipping component tree (NC-Tree) construction method is proposed. It starts by finding a separator and checking LFFC on BFGs containing the separator. If LFFC is true in any BFG, the separator is resolved and it finds another separator to check. Otherwise, the graph is separated into two components by the separator. This progress repeats until all separators in the offspring graphs are checked by LFFC. The result forms a binary tree whose leaf nodes are non-flipping 3-connected components and intermediate nodes are LFFC unsolvable separators.
2) Then each non-flipping leaf component on NC-Tree calculates local realizations of its nodes in a local coordinate system. This enables CFFC to be applied to further check the separators that cannot be resolved by LFFC. If CFFC can resolve the flipping ambiguity between two leaf nodes, the two nodes are merged into one component and the separator is resolved. The process starts from the leaf level to the root and the result is a reduced NC-Tree.
3) Finally, in each unresolvable leaf components on the reduced NC-Tree, node coordinates are in local coordinate systems. A *residue-based weighted component stitching* algorithm (RWCS) is proposed to synchronize their local coordinates to a common coordinate system to finally generate the realization of the graph, i.e, $\mathbf{P}$.

4 LFFC Disambiguating and NC-Tree Construction

At first, how to utilize LFFC to construct the NC-Tree is introduced.

4.1 LFFC Checking

Although binary vertex cut sets can be found by SPQR tree method in graph theory using $O(m+n)$ time [63], instead of finding all binary vertex cut sets, we detect an arbitrary binary vertex cut set $\{v_i, v_j\}$ at first. The separator formed by this vertex cut set is denoted by $\mathbf{s} = \mathbf{G}[v_i, v_j]$.

After a binary vertex cut set $\{v_i, v_j\}$ is detected, according to whether $(i, j) \in E$ or whether (i, j) exists implicitly (will be explained in Section 4.3), the procedure of checking LFFC for resolving the flipping ambiguity caused by $\{v_i, v_j\}$ is given in Fig. 8. The procedure can be separated into three cases.

CASE 1: If $(i, j) \in E$, and if v_i and v_j have two common neighbours u, v, the quadrangle (v_i, v_j, u, v) will be a BFG using (i, j) as the flipping axis. We enumerate all common neighbours of v_i and v_j to construct multiple BFGs using (i, j) as the flipping axis. The result BFG set is denoted by $\mathbf{B}_\mathbf{s}$. Then LFFC is checked for every BFG in $\mathbf{B}_\mathbf{s}$.

- If LFFC is true in any BFG in $\mathbf{B}_\mathbf{s}$, the length of a negative edge denoted by $l(u, v)$ is inferred by expression of x^E in (4). This added edge resolves the flipping ambiguity of $\mathbf{s}$. The inferred edge (u, v) is added into G, which turns G to be:

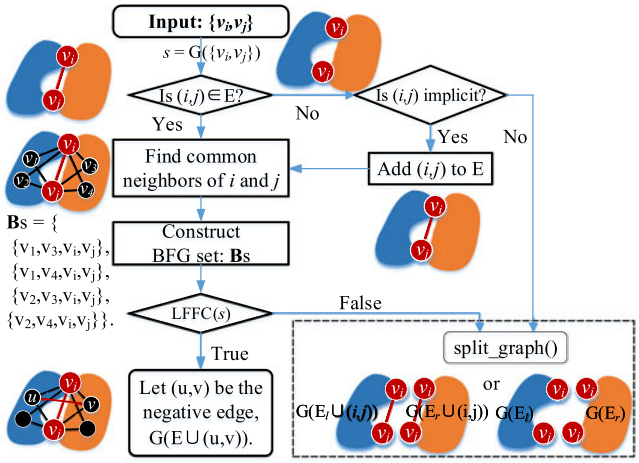


Fig. 8. LFFC checking procedure on a binary vertex cut set $\{v_i, v_j\}$.

$$G = \mathbf{G}[E \cup (u, v)], \tag{8}$$

and G will not be partitioned by $\mathbf{s}$.

- Otherwise, if LFFC is false for all the BFGs in $\mathbf{B_s}$, $\mathbf{s}$ is not resolved. It splits G into two subgraphs $G_{l,\mathbf{s}}$ and $G_{r,\mathbf{s}}$. Let's denote the edges separated in two subgraphs by E_l and E_r, where $E_l \cap E_r = \emptyset$ and $E_l \cup E_r = E \setminus (i, j)$.

$$G_{l,\mathbf{s}} = \mathbf{G}[E_l \cup (i, j)] \tag{9}$$

$$G_{r,\mathbf{s}} = \mathbf{G}[E_r \cup (i, j)]. \tag{10}$$

CASE 2: If $(i, j) \notin E$ but it is implicit, the length of (i, j) is inferred by implicit edge checking and calculation method as will be introduced in Section 4.3. We add (i, j) into E. $\mathbf{s}$ then includes v_i, v_j and (i, j). The same procedure as introduced in CASE 1 is applied to check $\mathbf{s}$ and partition the graph accordingly.

CASE 3: Let $\mathbf{C}$ be the graph to be divided. If $(i, j) \notin E$ and (i, j) is not implicit, the length of (i, j) is inferred by the local realization $\mathbf{Q^C}$, which can be obtained using ARAP. Let $\mathbf{q_i}$ and $\mathbf{q_j}$ be the coordinates of v_i and v_j in $\mathbf{Q^C}$, $d_{ij} = ||\mathbf{q_i} - \mathbf{q_i}||_2$. Then $\mathbf{s}$ includes v_i, v_j and (i, j). The same procedure is conducted as in CASE 1 to check $\mathbf{s}$.

Since multiple BFGs can be constructed in v_i and v_j's one-hop neighborhood, so LFFC can resolve the flipping ambiguity caused by $\{v_i, v_j\}$ with high probability since flipping is resolved if LFFC is true in any BFG.

4.2 NC-Tree Construction by LFFC Checking

To disambiguate separators of G network-wise, the LFFC checking will be applied recursively in each separated graph. Meanwhile, a non-flipping component tree (NC-Tree) is constructed. We use a structure $node = < graph,\ separator >$ to represent a node on the NC-Tree. The construction procedure is carried out efficiently by a queuing data structure, whose details are summarized in Algorithm 1.

The NC-Tree is initialized with $< G, \emptyset >$ as the root node. The *queue* stores the pending graph components and is initialized as the root node of NC-Tree. *Node* is explored by DEQUEUING *queue* while *queue* $\neq \emptyset$. Then the *separator detection*, *LFFC checking* routine is conducted for *Node* as in Section 4.1. If $\mathbf{s}$ is detected in *Node.graph* and $\mathbf{s}$ is not solved

Algorithm 1. LFFC Disambiguating and NC-Tree Construction

Input:
1: $G = (V, E)$: the original graph.
Output:
2: NC-Tree: non-flipping components tree.
3: n_s: separator number of G.
4: n_l: number of separator solved by LFFC.
5: $queue \leftarrow \emptyset$, $n_s \leftarrow 0$, $n_l \leftarrow 0$;
6: NC-Tree.root.graph $\leftarrow$ G;
7: Push NC-Tree.root in *queue*;
8: **while** *queue* $\neq \emptyset$ **do**
9: Node $\leftarrow$ Pop(*queue*);
10: **if find_bi_cut_set**(Node.graph) = $\emptyset$ **then** continue;
11: **end if**
12: $\{v_i, v_j\} \leftarrow$ **find_bi_cut_set**(Node.graph);(as in Section 4.1)
13: $s \leftarrow \mathbf{G}(\{v_i, v_j\})$, $n_s = n_s + 1$;
14: **if** more than two subgraphs are obtained **then**
15: $\mathbf{G}[E_l] \leftarrow$ the biggest subgraph, $\mathbf{G}[E_r] \leftarrow$ { Node.graph - $\mathbf{G}[E_l]$}
16: *queue*, NC-Tree $\leftarrow$ **split_graph**($\mathbf{G}[E_l]$, $\mathbf{G}[E_r]$,(i,j));
17: continue;
18: **else**
19: get two subgraphs $\mathbf{G}[E_l]$ and $\mathbf{G}[E_r]$;
20: **end if**
21: **if** $(i, j) \in E$ **then** /*CASE 1*/
22: goto LFFC Checking;
23: **else if** (i, j) is implicit **then** /*CASE 2*/
24: infer the length of (i, j) as in Section 4.3.3;
25: add the implicit edge to E;
26: **else** /*CASE 3*/
27: infer the length of (i, j) as in Section 4.1;
28: add the inferred edge to E;
29: **end if**
30: /*LFFC Checking*/
31: **if LFFC**(s)=true **then**
32: $n_l = n_l + 1$;
33: continue;
34: **else**
35: *queue*, NC-Tree $\leftarrow$ **split_graph**($\mathbf{G}[E_l]$, $\mathbf{G}[E_r]$,(i,j));
36: **end if**
37: **end while**
38: **return**NC-Tree, n_s, n_l.
LFFC(s)
1: construct BFG set $\mathbf{B_s}$ as described in Section 4.1;
2: **if** $\exists BFG \in \mathbf{B_s}$ satisfies LFFC **then**
3: the length of (u, v) in $BFG \leftarrow x^E$ in (4);
4: $E \leftarrow E \cup (\mathbf{u}, \mathbf{v})$;
5: **return**true.
6: **end if**
7: **return**false.
split_graph(s)
1: **if** (i,j)$\in E$ or (i,j) is implicit **then**
2: Node.LeftChild.graph $\leftarrow \mathbf{G}[E_l \cup (i, j)]$;
3: Node.RightChild.graph $\leftarrow \mathbf{G}[E_r \cup (i, j)]$;
4: **else**
5: Node.LeftChild.graph $\leftarrow \mathbf{G}[E_l]$;
6: Node.LeftChild.graph $\leftarrow \mathbf{G}[E_r]$;
7: **end if**
8: Push Node.LeftChild, Node.RightChild in *queue*;
9: **return**NC-Tree and *queue*.

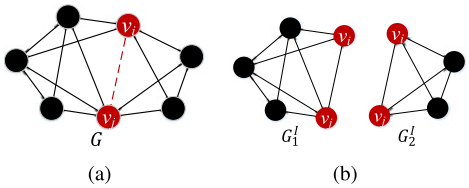


Fig. 9. (a) e_{ij} is implicit. (b) $e_{ij} \in$ redundantly rigid graphs in G_1^I and G_2^I.

by LFFC, the divided components ENQUEUE. Meanwhile they are put as the child nodes of $Node$ on the NC-Tree; If all separators on $Node.graph$ are resolved or if $Node.graph$ has no separator, this node becomes a leaf node.

Note that there are very low probability that a flipping separator divides the graph into more than two subgraphs. In this case, we select the subgraph with the most vertices as $G([E_l])$ and the union of other subgraphs as $G([E_r])$. If there are more than one biggest subgraphs, random choice is made. Then each graph is divided into at most two components, which provides facilitation for LFFC checking.

On the NC-Tree, an intermediate node has $node.graph$ equals to a graph component and $node.separator = \mathbf{s}$ is a LFFC unsolvable separator. The root node's $graph$ is G. Each leaf node has a non-flipping component as its $node.graph$ and $node.separator = \emptyset$. The number of intermediate nodes indicates the number of potential flipping ambiguities of G.

4.3 Check and Calculate the Implicit Edge

4.3.1 Implicit Edge

For a binary vertex cut set $\{v_i, v_j\}$, if edge $(i,j) \notin E$, from [13], if $\{v_i, v_j\}$ satisfies the condition in Definition 2, the edge (i,j) exists implicitly.

Definition 2 (Condition of Implicit Edge). *Consider a graph $G = (V, E)$ with two components separated by a binary vertex cut set $\{v_i, v_j\}$, i.e., $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $E_1 \cup E_2 = E$, $E_1 \cap E_2 = \emptyset$, and $V_1 \cap V_2 = \{v_i, v_j\}$. If both v_i and v_j belong to a rigid component in G_1 and a rigid component in G_2, then edge (i,j) exist implicitly if $(i,j) \notin E$.*

4.3.2 Implicit Edge Checking

From Definition 2, checking whether an implicit edge exists between v_i and v_j is reduced to check whether there are rigid graphs in G_l and G_r both containing $\{v_i, v_j\}$. It can be done efficiently by adding (i,j) into G_l and G_r to form extended subgraphs $G_l^I = \mathbf{G}(E_1 \cup (i,j))$ and $G_r^I = \mathbf{G}(E_2 \cup (i,j))$. Then we check whether there is a redundantly rigid graph in G_l^I that contains (i,j) by using Pebble Game[25] starting from (i,j). The same checking is conducted in G_r^I. As is shown in Fig. 9, if we can find redundantly rigid graphs containing (i,j) in both G_l^I and G_r^I, an implicit edge (i,j) exists between v_i and v_j. Otherwise, (i,j) is not implicit.

4.3.3 Implicit Edge Calculation

If (i,j) is implicit, even though the graph may have multiple realizations, the length of (i,j) remains the same in each of them[13]. The local coordinates of the initial graph $G_l \cup G_r$ can be realized by patch realization algorithms such as ARAP[20]. The estimated distance between v_i and v_j in the realized graph is used as the edge length of (i,j). E is then updated by $E = E \cup (i,j)$ and $G = G(V, E)$.

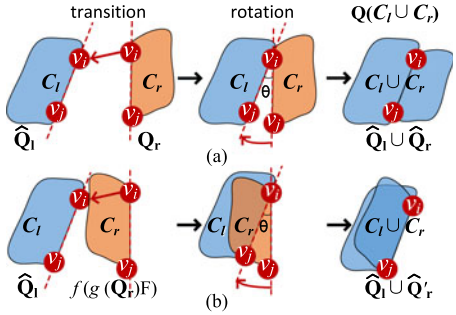


Fig. 10. Stitch $\mathbf{Q_r}$ to $\mathbf{Q_l}$. (a) $\hat{\mathbf{Q}}^1(\mathbf{C}_l \cup \mathbf{C}_r)$;(b) $\hat{\mathbf{Q}}^2(\mathbf{C}_l \cup \mathbf{C}_r)$.

5 CFFC-based Disambiguating on NC-Tree

On the constructed NC-Tree, all the leaf nodes are flipping free components. The intermediate nodes are separators that cannot be resolved by LFFC using one-hop information. But these separators may still be resolved by the Component-based Flipping Free Condition (CFFC) via multi-hop information.

Suppose the set of leaf components on the NC-Tree are $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_{n_c}\}$. Since all the leaf components are flipping free, we realize the local coordinates of each leaf component in its local coordinate system. Any realization method can be used, such as ARAP[20]. The local coordinates of $\mathbf{C}$ calculated in their local coordinate systems are denoted by $\{\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_{n_c}\}$, in which $\mathbf{Q}_i = \{\mathbf{q}_i^1, \ldots, \mathbf{q}_i^{n_i}\}$ are the local coordinates of nodes in the component $\mathbf{C}_i$.

Then we consider a bottom level separator $\mathbf{s}$ and its two connected leaf nodes $\mathbf{C}_l$ and $\mathbf{C}_r$. The local coordinates calculated for these two children are denoted by $\mathbf{Q}_l$ and $\mathbf{Q}_r$ respectively. Since these two components share two vertices v_i and v_j, there are two ways to stitch their realizations into one coordinate system as shown in Fig. 10. The first way is that $\mathbf{Q}_l$ and $\mathbf{Q}_r$ are stitched directly, and the second way is that $\mathbf{Q}_l$ is stitched with a flipping version of $\mathbf{Q}_r$ along $\mathbf{s}$. Their calculations are in (17). These two candidate stitching results are denoted by $\hat{\mathbf{Q}}^1(\mathbf{C}_l \cup \mathbf{C}_r)$ and $\hat{\mathbf{Q}}^2(\mathbf{C}_l \cup \mathbf{C}_r)$ respectively.

5.1 Calculation of $\hat{\mathbf{Q}}^1(\mathbf{C}_l \cup \mathbf{C}_r)$ and $\hat{\mathbf{Q}}^2(\mathbf{C}_l \cup \mathbf{C}_r)$

$\mathbf{C}_l$ and $\mathbf{C}_r$ share the cutting vertices v_i and v_j. Let $(q_i^{l,x}, q_i^{l,y})$, $(q_j^{l,x}, q_j^{l,y})$ and $(q_i^{r,x}, q_i^{r,y})$, $(q_j^{r,x}, q_j^{r,y})$ be local coordinates of $\{v_i, v_j\}$ in $\mathbf{Q}_l$ and $\mathbf{Q}_r$ respectively. We choose the coordinate system of $\mathbf{Q}_l$ as the coordinate system after stitching without loss of generality. Let $\hat{(\cdot)}$ be the stitched coordinate system. So the coordinates of $\mathbf{C}_l$ don't change:

$$\hat{\mathbf{Q}}_l = \mathbf{Q}_l. \tag{11}$$

Their coordinates for directly stitching $\mathbf{C}_r$ and stitching the flipping version are:

$$\hat{\mathbf{Q}}_r = \mathbf{R}\mathbf{Q}_r + \mathbf{T} \; or \tag{12}$$

$$\hat{\mathbf{Q}}'_r = \mathbf{R}f(g(\mathbf{Q}_r)\mathbf{F}) + \mathbf{T}, \tag{13}$$

respectively, where $\mathbf{R}$ is a rotation matrix and $\mathbf{T}$ is a transition matrix. $\mathbf{F}$ is a flip matrix that turnover $\mathbf{Q}_r$ to $\mathbf{Q}'_r$ along $\mathbf{s}$ as shown in Fig. 10. Note that to calculate the flipping realization

of $\mathbf{Q}_r$ along $\mathbf{s}$, we need to consider $\mathbf{P}_r$ is rotated along $\mathbf{s}$ in 3D space. So $\mathbf{F}$ is the 3D rotation matrix whose calculation can be inferred to Appendix D, available in the online supplemental material. $f(\cdot)$ means the operation of reducing dimension to $\mathbb{R}^2$ by abandoning the third dimension. $g(\mathbf{P}_r)$ adds a zero value at the third dimension to upgrade $\mathbf{P}_r$ to 3D.

Let $n_x = q_j^{r,x} - q_i^{r,x}$, $n_y = q_j^{r,y} - q_i^{r,y}$, then:

$$\mathbf{T} = \begin{pmatrix} q_i^{l,x} - q_i^{r,x} \\ q_i^{l,y} - q_i^{r,y} \end{pmatrix} \tag{14}$$

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{15}$$

and from Fig. 5 of Appendix D, available in the online supplemental material, we can calculate:

$$\mathbf{F} = \begin{bmatrix} 2n_x^2 - 1 & 2n_x n_y & 0 \\ 2n_x n_y & 2n_y^2 - 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \tag{16}$$

where $\theta = arctan\frac{q_j^{r,y} - q_i^{r,y}}{q_j^{r,x} - q_i^{r,x}} - \arctan\frac{q_j^{l,y} - q_i^{l,y}}{q_j^{l,x} - q_i^{l,x}}$ is the rotation angle to align $\{v_i^r, v_j^r\}$ to $\{v_i^l, v_j^l\}$. Then the two realizations for stitched $\mathbf{C}_l$ and $\mathbf{C}_r$ are denoted by

$$\begin{aligned} \hat{\mathbf{Q}}^1(\mathbf{C}_l \cup \mathbf{C}_r) &= \hat{\mathbf{Q}}_l \cup \hat{\mathbf{Q}}_r \\ \hat{\mathbf{Q}}^2(\mathbf{C}_l \cup \mathbf{C}_r) &= \hat{\mathbf{Q}}_l \cup \hat{\mathbf{Q}}'_r, \end{aligned} \tag{17}$$

respectively.

5.2 CFFC Checking

CFFC proposed in Theorem 4 is applied to disambiguate the two realizations $\hat{\mathbf{Q}}^1$ and $\hat{\mathbf{Q}}^2$ to determine which one is feasible. Without loss of generality, let's consider the feasibility of $\hat{\mathbf{Q}}^1$. Let u be a node in $\mathbf{C}_l$ and v be a node in $\mathbf{C}_r$. There realized coordinates are denoted $\hat{\mathbf{Q}}_u^1$ and $\hat{\mathbf{Q}}_v^1$ in $\hat{\mathbf{Q}}^1$ respectively. Because these two nodes are in two separated components separated by a binary vertex cut, the edge $(u, v) \notin E$ and their real distance should be larger than R.

CFFC checks the feasibility of $\hat{\mathbf{Q}}^1$ and $\hat{\mathbf{Q}}^2$ by above idea as in $CFFC(\hat{\mathbf{Q}}^k, \mathbf{C}_\mathbf{l}, \mathbf{C}_\mathbf{r})$ function in Algorithm 2. If $\exists \|\hat{\mathbf{Q}}_u^k - \hat{\mathbf{Q}}_v^k\| \leq R$, $CFFC(\hat{\mathbf{Q}}^k, \mathbf{C}_\mathbf{l}, \mathbf{C}_\mathbf{r})$ returns FALSE, which means the realization $\hat{\mathbf{Q}}^k$ is infeasible.

5.3 NC-Tree Simplification by CFFC Checking

$CFFC(\hat{\mathbf{Q}}^k, \mathbf{C}_\mathbf{l}, \mathbf{C}_\mathbf{r})$ is briefly written as $CFFC(\hat{\mathbf{Q}}^k)$. When only one realization is feasible, the separator is resolved by CFFC and its two leaf nodes can be merged into one to simplify the NC-tree. Therefore, CFFC checking is started from the leaf level on NC-Tree.

- For a separator $\mathbf{s}$, if both $CFFC(\hat{\mathbf{Q}}^1)$ and $CFFC(\hat{\mathbf{Q}}^2)$ are feasible, the separator $\mathbf{s}$ cannot be resolved and the children can not be merged.
- If only one realization is feasible, the two child nodes are merged into one component, as given by Line 16-19 in Algorithm 2.
- In noisy cases, if both two realizations are infeasible, the one with less violation of the negative edge constraints is chosen as the realization of the parent node and the two children are merged.

In Algorithm 2. $h\_mergeable$ records whether the hth level of NC-Tree can be merged. If all separators on the hth level cannot be merged by CFFC, $h\_mergeable$ is set FALSE. The process terminates when $h\_mergeable$ is FALSE or the root level of NC-Tree is reached.

Algorithm 2. CFFC Checking Algorithm

Input:
1: NC-Tree.
Output:
2: Simplified NC-Tree.
3: $h \leftarrow$ the height of NC-Tree;
4: $h\_mergeable \leftarrow$ TRUE;
5: **while** $h > 1$**AND**$h\_mergeable =$ TRUE **do**
6: $h\_mergeable \leftarrow$ FALSE, ;
7: **for** each connected pair N_l, N_r on h-level of NC-Tree **do**
8: $N \leftarrow$ father node of N_l and N_r, $s \leftarrow N.separator$;
9: $\mathbf{C}_l \leftarrow N_l.graph$, $\mathbf{C}_r \leftarrow N_r.graph$;
10: calculate $\hat{\mathbf{Q}}^1(\mathbf{C}_l \cup \mathbf{C}_r)$ and $\hat{\mathbf{Q}}^2(\mathbf{C}_l \cup \mathbf{C}_r)$ as in Section 5.1
11: /* denoted by $\hat{\mathbf{Q}}^1$ and $\hat{\mathbf{Q}}^2$ for brevity. */
12: **if** $CFFC(\hat{\mathbf{Q}}^1)$ **AND** $CFFC(\hat{\mathbf{Q}}^2)$ **then**
13: /* Both $\hat{\mathbf{Q}}^1$ and $\hat{\mathbf{Q}}^2$ are **feasible**, cannot disambiguate. */
14: continue;
15: **else**
16: /* At least one of $\hat{\mathbf{Q}}^1$ and $\hat{\mathbf{Q}}^2$ is **infeasible** */
17: /* The two children are merged into one node and the separator is resolved. */
18: $N.graph = \mathbf{C}_l \cup \mathbf{C}_r$, $N.separator = \emptyset$
19: $h\_mergeable \leftarrow$ TRUE;
20: **end if**
21: **end for**
22: $h \leftarrow h - 1$;
23: **end while**
24: **return**NC-Tree.
$CFFC$ ($\hat{\mathbf{Q}}^k, \mathbf{C}_l, \mathbf{C}_r$)
1: $feasible\_flag \leftarrow$ TRUE;
2: **for** each $u \in \mathbf{C}_l, v \in \mathbf{C}_r, e_{uv} \notin E$ **do**
3: **if** $\exists \|\hat{\mathbf{Q}}_u^k - \hat{\mathbf{Q}}_v^k\| \leq R$ **then**
4: $feasible\_flag \leftarrow$ FALSE;
5: break;
6: **end if**
7: **end for**
8: **return** $feasible\_flag$.

5.4 Generate All Feasible Realizations

After NC-Tree simplification, the remained separators are CFFC unresolvable. Each separator indicates two ambiguous realizations. We can generate all feasible, but ambiguous realizations with the NC-Tree. There are totally 2^K ambiguous realizations where K is the number of remained separators.

Let $\mathcal{Q}_i$ be a set to store the ambiguous realizations of node i of the NC-Tree. Initially $\mathcal{Q}_i$ is initialized as $\{\mathbf{Q}_i\}$ if it is a leaf node, $\emptyset$ otherwise. Then the NC-Tree is traversed in bottom-up level order and the following processes are conducted for each pair of brother nodes $Node_l$ and $Node_r$:

Step 1: Let $\mathcal{Q}_l$, $\mathcal{Q}_r$ and $\mathcal{Q}_f$ be the possible realizations of $Node_l$, $Node_r$ and their father node $Node_f$, respectively. For each $\mathbf{Q}_l \in \mathcal{Q}_l$ and $\mathbf{Q}_r \in \mathcal{Q}_r$, calculate $\hat{\mathbf{Q}}^1(\mathbf{C}_l \cup \mathbf{C}_r)$ and $\hat{\mathbf{Q}}^2(\mathbf{C}_l \cup \mathbf{C}_r)$ as in Section 5.1.

Step 2: Conduct CFFC checking for $\hat{\mathbf{Q}}^1(\mathbf{C}_l \cup \mathbf{C}_r)$ and $\hat{\mathbf{Q}}^2(\mathbf{C}_l \cup \mathbf{C}_r)$ and add $\hat{\mathbf{Q}}^k$ to $\mathcal{Q}_f$ when $CFFC(\hat{\mathbf{Q}}^k, \mathbf{C}_l, \mathbf{C}_r)$ is TRUE, where $k \in \{1, 2\}$. In noise scenarios, it may happen that both of them return FALSE, then the realization of $\mathbf{C}_l \cup \mathbf{C}_r$ by ARAP[20] will be assigned to $\mathcal{Q}_f$.

The process terminates when $Node_f$ is the root node of NC-Tree. Thus $\mathcal{Q}_{root}$ consists of all feasible realizations of G. $|\mathcal{Q}_{root}| = 2^K$. For simplicity, we use $\mathcal{P}$ for $\mathcal{Q}_{root}$. Fig. 15 in the experiment section plots the localization errors of the multiple feasible realizations.

6 Residual Based Weighted Component Stitching (RWCS)

Even though all the ambiguous realizations can be given, network localization requires only one accurate solution. Equation (17) calculates all feasible realizations by only stitching the components in different ways. However, it doesn't utilize iterative component synchronization, which is to solve an optimization problem to iteratively minimize the stitching error. It is reported in [4], [46], [64], which can greatly improve the synchronization accuracy against ranging noises.

This section proposes a residue-based iterative stitching method to generate a unique localization result by iteratively stitching of the local realizations of components.

6.1 Component Stitching for a Unique Localization

After CFFC checking, suppose the remained leaf components on the simplified NC-Tree are $\mathbf{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_{n_c}\}$ and unsolvable separators are $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_K\}$. The corresponding local coordinate systems are $\mathbf{Q} = \{\mathbf{Q}_1, \ldots, \mathbf{Q}_{n_c}\}$. A final synchronization process is conducted to generate a realization.

Each remained separator $\mathbf{s}_i$ contains two vertices. We denoted the vertex set of separators by V_S. Each node in V_S can form a patch which is the one-hop neighborhood of the node, and the patch is realized in a local coordinate system. Then these patches are synchronized with the components in $\mathbf{C}$.

The idea of weighted component stitching in [46] is exploited to realize the final graph. Let $\mathbf{P} = \{\mathbf{p_1}, \ldots, \mathbf{p_n}\}$ be the desired global coordinates of nodes. The objective of component stitching is as following:

$$F = \min_{\mathbf{P}, \mathbf{R_1}, \ldots, \mathbf{R_1^c}, \ldots,} \left\{ \sum_{k \in V_S} \sum_{m \in \mathbf{N_k}} \left\| (\mathbf{p}_k - \mathbf{p}_m) - \mathbf{R}_k (\mathbf{q}_k - \mathbf{q}_m) \right\|^2 + \sum_{l=1}^{n_c} \sum_{(i,j) \in \mathbf{C}_l} f(r_l) \left\| (\mathbf{p}_i - \mathbf{p}_j) - \mathbf{R}_l^c \left(\mathbf{q}_i - \mathbf{q}_j \right) \right\|^2 \right\}$$
$$s.t. \mathbf{R}_k^T \mathbf{R}_k = \mathbf{I}, \mathbf{R}_l^{cT} \mathbf{R}_l^c = \mathbf{I}, \tag{18}$$

which consists of two parts:

- *patches synchronization*: $\mathbf{N_k}$ is the one-hop neighborhood of node k. $\mathbf{R_k}$ is the rotation matrix of the kth patch G_k. $\mathbf{q}_k$ and $\mathbf{q}_m$ represent local coordinates of the kth and the mth node in patch G_k.
- *leaf components synchronization*: $\mathbf{R}_l^c$ is the rotation matrix of component $\mathbf{C}_l$. $\mathbf{q}_i$ and $\mathbf{q}_j$ represent local coordinates of the ith and the jth node in component $\mathbf{C}_l$.

The components are higher weighted by $f(r_l)$ since they have better realization quality [46] than patches of separator vertices. The weight is set negatively correlated with the residue error of local realization for insuring better local realization has larger impacts on overall network synchronization.

Definition 3 (Residual Error (RE)). *Considering a graph $G = (V, E)$ where $|V| = n$ and its realization result $\mathbf{Q} = \{\mathbf{q}^1, \ldots, \mathbf{q}^n\}$, the residual error (RE) is calculated as $r(G) = \sum_{(i,j) \in E} \left(|e_{ij}| - ||\mathbf{q}^i - \mathbf{q}^j|| \right)^2$.*

We use an exponential function as suggested in [46] to increase the impact of realizations with small residues. Let r_l be the residue of the component $\mathbf{C}_l$ and $r_{max} = \max\{r_1, \ldots, r_L\}$, then the weight of component $\mathbf{C}_l$ is:

$$f(r_l) = e^{r_{max}/r_l}. \tag{19}$$

Algorithm 3. RWCS Algorithm

Input:
1: $\{\mathbf{C}_1, \ldots, \mathbf{C}_{n_c}\}$: leaf components of simplified NC-Tree
2: $\{\mathbf{Q}_1, \ldots, \mathbf{Q}_{n_c}\}$: local coordinates of each component.
3: h: the maximum iteration times.
4: $thrd$: the convergence threshold.
Output:
5: $\mathbf{P} = \{\mathbf{p}_i, i = 1, \ldots, n\}$: global coordinates of all nodes
6: initialize $\mathbf{P}(0)$ by ARAP [20];
7: initialize $\mathbf{R}_k(0)$ and $\mathbf{R}_l^c(0)$ by (20) using ICP [65].
8: calculate $f(r_l)$ for each component by (19);
9: t $\leftarrow$ 1;
10: **while** $||\mathbf{P}(t) - \mathbf{P}(t-1)|| > thrd$ **AND** $t < h$ **do**
11: $\mathbf{R}_k(t)$, $\mathbf{R}_l^c(t) \leftarrow$ substitute $\mathbf{P}(t-1)$ to (18) (detailed as the local phase in Section 6.2);
12: $\mathbf{P}(t) \leftarrow$ substitute $\mathbf{R}_k(t)$ and $\mathbf{R}_l^c(t)$ to (18) (detailed as the global phase in Section 6.2);
13: t $\leftarrow$ t+1;
14: **end while**
15: **return** $\mathbf{P}(t)$

6.2 The Synchronization Algorithm

The objective function of RWCS in (18) can be solved through an Alternating Least-Squares (ALS) [46], which is an iteration of two phases:

- In the local phase, $\mathbf{P}$ is set to be fixed to solve the rotation matrix $\mathbf{R}_k$ and $\mathbf{R}_l^c$. $\mathbf{P}$ is initialized through solving the linear equations as in ARAP [20]. Then $\mathbf{R}_k$ and $\mathbf{R}_l^c$ can be solved by the following optimization problem:

$$\mathbf{R}_k = arg\min_{\mathbf{R}} \{ \|\mathbf{P}_k - \mathbf{R}\mathbf{Q}_k\|_F^2 : \mathbf{R}^\mathbf{T}\mathbf{R} = \mathbf{I} \}$$
$$\mathbf{R}_l^c = arg\min_{\mathbf{R}} \{ \left\|\mathbf{P}_l - \mathbf{R}\mathbf{Q}_l^c\right\|_F^2 : \mathbf{R}^\mathbf{T}\mathbf{R} = \mathbf{I} \}. \tag{20}$$

  The optimization problem in (20) is a typical point cloud matching problem that can be solved by Iterative Closet Point (ICP) algorithm [66]. The ICP algorithm calculates the rotation matrix by efficiently Singular Value Decomposition to align different point clouds, whose complexity is $O(u^2)$ for a component with u nodes.
- In the global phase, all $\mathbf{R}_k$ and $\mathbf{R}_l^c$ are fixed to solve $\mathbf{P}$. n linear equations can be set up by making the gradient $\frac{\partial \mathbf{F}}{\partial \mathbf{p}_i} = 0$ for $i \in \{1, \ldots, n\}$. Thus, the global coordinates

can be obtained by solving these n linear equations and then substituted into (18) to solve $\mathbf{R}_k$ and $\mathbf{R}_l^c$.

The procedure iterates until $\mathbf{P}$ is converged or reaching a maximum iteration time.

7 Algorithm Analyses

This section provides property analyses for the proposed algorithms, including the validity, the disambiguating performance, and the complexity.

7.1 Validity of Decomposition

The paper assumes that the graph is rigid (see Section 2.2). To iteratively construct the NC-Tree, the rigidness of the subgraphs in the decomposition process is verified as follows.

Theorem 5 (Rigidness of Subgraphs). *If a graph $G = (V, E)$ is rigid and is divided by a separator* s *using Algorithm 1 into two components G_l and G_r, then the divided components G_l and G_r are also rigid.*

Proof. Suppose $|V| = n$. Since G is rigid, it contains a spanning Laman graph, denoted by L. L has $2n-3$ edges and any k vertex sub-graph of L has no more than $2k-3$ edges [12]. The separator divides L into L_l and L_r. Suppose $|\mathbf{V}(L_l)| = k_1$; $|\mathbf{V}(L_r)| = k_2$ then $n = k_1 + k_2 - 2$ since the two vertexes of the separator are copied into two sub-graphs.

By Laman condition, without loss of generality, $2k_1 - 4 \leq |\mathbf{E}(L_l)| \leq 2k_1 - 3$ and $2k_2 - 4 \leq |\mathbf{E}(L_r)| \leq 2k_2 - 3$, since $2k_1 + 2k_2 - 7 = 2(n-2) - 3$. Otherwise L is not rigid. By dividing using Algorithm 1 , if the separator edge e exists, it will be divided into two edges and are added into G_l and G_r. If the separator edge originally doesn't exist, each subgraph adds a new edge. In either way, $|\mathbf{E}(L_l) + e| \geq 2k_1 - 3$ and $|\mathbf{E}(L_r) + e| \geq 2k_2 - 3$. They both contain a Laman graph and are both rigid. □

So each component divided by Algorithm1 remains rigid.

7.2 Performance of Disambiguating

We now provide the theoretical performance of disambiguating. Suppose the original rigid graph G has $c(G)$ number of potential ambiguous realizations, including the flipping and flex ambiguities.

Theorem 6. *In construction of NC-Tree, if LFFC resolves k separators, and CFFC resolves g separators, the number of ambiguous realizations is reduced from $c(G)$ to $\frac{c(G)}{2^{k+g}}$.*

Proof. If s_1 is resolved by LFFC or CFFC, the flipping ambiguity between G'_{l,s_1} and G'_{r,s_1} is eliminated. Originally, $c(G) = 2c(G'_{l,s_1})c(G'_{r,s_1})$. After s_1 is resolved, $c(G') = c(G'_{l,s_1})c(G'_{r,s_1}) = \frac{c(G)}{2}$. This process holds for all resolved separators $s_2, \ldots, s_{k+g}$, since LFFC and CFFC are applied sequentially and resolve different separators. The total number of ambiguities will be reduced to $\frac{c(G)}{2^{k+g}}$. □

7.3 Complexity Analysis

Theorem 7. *Given a sparse network with the underlying graph $G = (V, E)$, where $|V| = n$ and $|E| = m$. We consider the node degree is bounded by Δ for sparse networks. The complexity of the proposed scheme for localizing this network is $O(h \cdot n^3)$, where h is the maximum iteration time in the synchronization algorithm.*

TABLE 2
The Time Complexity for Selected Algorithms

Proposed	WCS	ARAP	SDP	SMACOF
$\mathbf{O(hn^3)}$	$O((\Delta n)^3 + hn^3)$	$O(hn^3)$	$O(m^{3.5})$	$O(hn^2)$

Proof.

- In LFFC disambiguating and NC-Tree construction, the time complexity of finding 3-connected components and separators is $O(n+m)$ [67]. If a detected separator divides the graph into more than two sub-components ($n-2$ components in the worst case), thus finding the one with most vertexes is an operation of $O(n)$. Suppose K separators are obtained, $0 \leq K \leq m$. Implicit edge checking is conducted by redundant rigidity checking using Pebble Game. Suppose c components are obtained by the K separators and the scale of each component is n/c. The worst-case performance of Pebble Game is $O((\frac{n}{c})^2)$ for any component [25]. The time complexity of implicit edge checking is $c \cdot O((\frac{n}{c})^2) = O(n^2)$. Then LFFC checking is conducted by constructing BFGs for each separator using $N[v_i] \cap N[v_j]$, whose time complexity is $O(K \cdot \Delta^2)$, and the worst case is $O(m \cdot \Delta^2)$, thus $O(n \cdot \Delta^3)$. So the total complexity for implicit edge checking and LFFC checking is $O(n^2) + O(n \cdot \Delta^3)$.
- In CFFC disambiguating, suppose w separators are unresolvable by LFFC, $0 \leq w \leq K \leq m$. First, we give a local realization of each component by ARAP, whose time complexity is $O(w \cdot h \cdot (\frac{n}{w})^3)$, where $\frac{n}{w}$ represents the average node number of each component. Then we check feasibility of $\hat{\mathbf{Q}}^1$ and $\hat{\mathbf{Q}}^2$ by CFFC, whose time complexity is $O(w \cdot (\frac{n}{w})^2)$. Thus, the time complexity of CFFC disambiguating is $O(w \cdot h \cdot (\frac{n}{w})^3 + w \cdot (\frac{n}{w})^2) = O(h \cdot \frac{n^3}{w^2})$, whose worst case is $O(h \cdot n^3)$.
- Finally, the proposed RWCS algorithm has the same time complexity with the synchronization step of WCS [46], which is $O(h \cdot n^3)$.

Therefore, CFFC disambiguating and RWCS are the most time-consuming steps. The total time complexity of the proposed scheme is $O(h \cdot n^3)$. □

Table 2 shows the time complexity comparison of the proposed scheme, WCS, ARAP, SDP and SMACOF.

8 Performance Evaluation

8.1 Simulation Settings and Performance Metric

Simulations are conducted in Matlab2018. n sensors are deployed randomly in an area of 100 $m \times 100$ m. Two variables are controlled for network settings: 1) average node degree; 2) ranging noises. The average degree is controlled by varying the maximum ranging radius R. The average node degree indicates the edge density of the network. The smaller the average node degree is, the sparser the network is.

TABLE 3
The Probability of Eliminating All Ambiguities

$\overline{\Delta}$	5	6	7	8	9
P	0.32	0.44	0.59	0.69	0.82

The ranging noises are assumed zero-mean Gaussian distributed, i.e., $\mathcal{N}(0, \sigma^2)$, where the noise level is controlled by varying σ [29], [46]. The average normalized localization error is adopted as the accuracy metric: $e = \frac{\sum_{i=1}^{n} \left\| \mathbf{p}_i - \mathbf{p}_i^{'} \right\|}{n}$ where $\mathbf{p}_i$ is the ground truth and $\mathbf{p}_i^{'}$ is the realized result of a method.

8.2 Validity of LFFC and CFFC

As mentioned in Theorem 6, if $k+g$ separators are resolved by LFFC and CFFC, the flipping ambiguity will be decreased at a scale of 2^{k+g}. We generate different network typologies to evaluate the disambiguation capability of LFFC and CFFC. The network configurations are: (i) Average degree changes from 5 to 9 and (ii) σ varies in the range of $\{0, 2, 5\}$. We run experiments for 100 times for each setting and calculate the average value. The generated networks are kept rigid by moving the flex node to a close-by node if the flex node exists in the randomly generated networks.

8.2.1 The flip ambiguity solved by LFFC and CFFC

Table 3 presents the probability of eliminating all ambiguities when ranging is noiseless. $\overline{\Delta}$ represents the average node degree in the network. We can see when the network becomes denser, the proposed methods have higher probability to resolve all the flipping ambiguities in the network.

Table 4 shows the average number of flip ambiguity solved by LFFC and CFFC in four different settings. The average number of potential flipping ambiguities are compared in the table, which is calculated by averaging $2^{|\mathbf{S}|}$, where $|\mathbf{S}|$ is the number of resolved separators in the graph. For example, in the first row, without using CFFC and LFFC, the average number of potential flipping ambiguities in 100 experiments is 3040.30; LFFC can on average resolve 3003.5 potential flipping ambiguities and CFFC can on average resolve 31.15. Only 5.65 potential flipping ambiguities cannot be resolved by LFFC and CFFC. So LFFC and CFFC can efficiently resolve a major portion of potential flipping ambiguities and LFFC contributes the major part. The LFFC and CFFC unresolvable flipping ambiguities become very limited. It can be concluded that most flipping ambiguities are eliminated by LFFC and CFFC in the proposed scheme, which leads to accurately final realization.

TABLE 4
The Number of Ambiguities Solved by LFFC or CFFC

Parameter Setting	Flip Ambiguity	Solved by LFFC	Solved by CFFC	After LFFC and CFFC
$\overline{\Delta} = 5, \sigma = 2$	3040.30	3003.50	31.15	5.65
$\overline{\Delta} = 5, \sigma = 5$	498.00	468.35	12.38	17.27
$\overline{\Delta} = 7, \sigma = 2$	215.27	205.95	5.06	4.16
$\overline{\Delta} = 7, \sigma = 5$	245.57	233.44	6.39	5.74

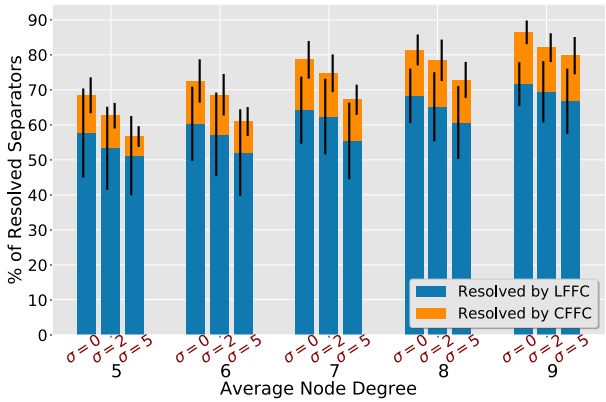


Fig. 11. The percentage of resolved separators by LFFC or CFFC.

Fig. 11 provides a more clearly comparison including the variance values. The average percentages of resolved separators by LFFC and CFFC are presented. The black lines are variances. The results show that LFFC and CFFC can resolve more than 50 percent separators even in the most sparse and noisy cases, where $\overline{\Delta} = 5$ and $\sigma = 5$.

8.2.2 LFFC and CFFC's Contribution on Reducing Localization Errors

The contribution of LFFC and CFFC on reducing the localization errors is detailed in Fig. 12. The proposed scheme is "LFFC + CFFC + RWCS". "LFFC + RWCS" means CFFC is not used; "RWCS" means both LFFC and CFFC are not used. The x-axis is the average node degree and the y-axis is the average localization error. It can be seen that the localization errors decrease as the average node degree increases, i.e., when the network becomes less sparse, which is consistent with our common sense. RWCS without LFFC and CFFC has the largest localization errors as expected and "LFFC + CFFC + RWCS" provides the smallest localization errors in all cases. Statistical results show that LFFC and CFFC help to reduce 33 percent localization errors on average, even in the most sparse and noisy cases.

8.2.3 Visualize the impacts of LFFC and CFFC

A sparse network of 100 nodes with $\overline{\Delta} = 6$ and $\sigma = 0$ is instantiated in Figs. 13 and 14, where black points represent

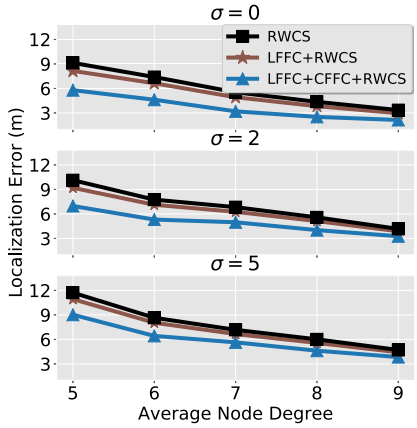


Fig. 12. The localization error with removing LFFC or CFFC from the proposed scheme.

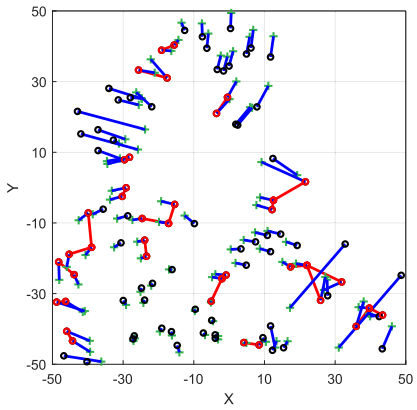


Fig. 13. Visualizing the separators and the localization result of WCS.

for the ground truth and the green cross markers show localization results. The blues lines show localization errors. The shorter are the blue lines, the more accurate are the localization results. Additionally, red lines represent separators of the network. Pink lines are resolved separators and red lines are unresolved separators. Yellow lines represent the inferred negative edges given by LFFC. Since $\sigma = 0$, the localization error is mainly caused by flipping ambiguity. Comparing Figs. 13 and 14, most of the flipping ambiguities are eliminated and the localization accuracy is improved greatly by LFFC and CFFC.

8.3 Validity of RWCS

This section verifies the accuracy and convergence efficiency of RWCS.

8.3.1 Errors of All Feasible Solutions and Errors of RWCS

From the simplified NC-tree, all feasible ambiguous realizations can be calculated as in Section 5.4. Fig. 15 shows the localization errors of all feasible realizations in 10 graph instances. The black bar shows the interval of the localization errors of the feasible solutions. The errors of each feasible solution are plotted on the bar. We also compare the errors of WCS [46] and our proposed method. Our proposed method is referred to LFFC+CFFC+RWCS. It can be seen clearly, our proposed method has the smallest

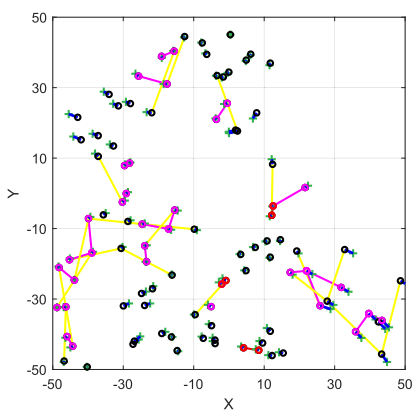


Fig. 14. Visualizing the resolved separators and the localization result of the proposed schema.

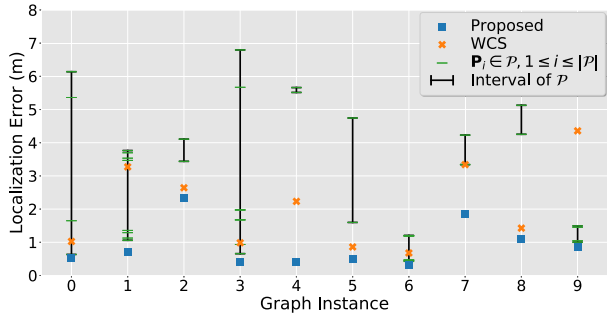


Fig. 15. The localization error of the proposed schema, WCS and all feasible realizations. 'Interval of $\mathcal{P}$' is the line segment between the maximum and the minimal localization error for all feasible realizations in $\mathcal{P}$.

localization errors comparing with WCS and those of all the feasible solutions. This is benefited by the ambiguity reduction and residue-based iterative stitching process proposed in RWCS.

8.3.2 Convergence of RWCS

How the nodes locations are converged from initial states to the final states in RWCS is shown in Fig. 16a, when $\overline{\Delta} = 6$ and $\sigma = 0$. The green diamonds represent the initial positions given by ARAP. Blue points and red cross markers show the intermediate results and the final positions, respectively. The localization errors are shown in Fig. 16b. It can be observed that RWCS converges very fast, and returns almost the true coordinates of the nodes in the noiseless ranging scenario.

8.4 VS. the State-of-the-Art Algorithms

The localization accuracy of the proposed scheme are compared with: the state-of-the-art component stitching methods: (1) WCS [46], (2) ARAP [20]; the state-of-the-art centralized localization methods (3) SDP [18], (4) SMACOF [17].

Fig. 17 shows the distribution of localization error in different parameters. The localization error of ARAP, SDP and SMACOF are evaluated, which are compared with the proposed scheme and WCS. It is shown that the proposed scheme greatly outperforms other algorithms. This shows the effectiveness of disambiguation through LFFC and CFFC.

8.4.1 The Cumulative Error Distribution

The cumulative distribution function (CDF) of the location errors are counted to evaluate the performances of different

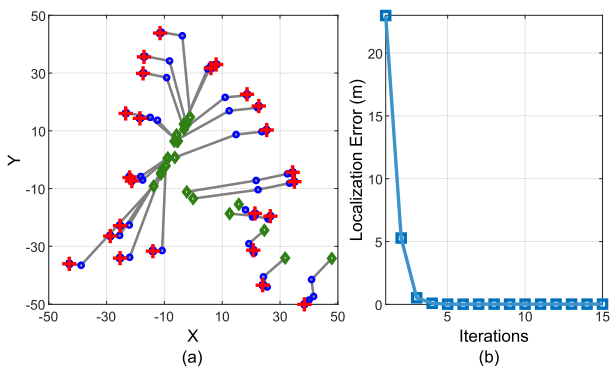


Fig. 16. (a) Trajectories of the coordinate estimates. (b) Localization errors w.r.t. the iteration count.

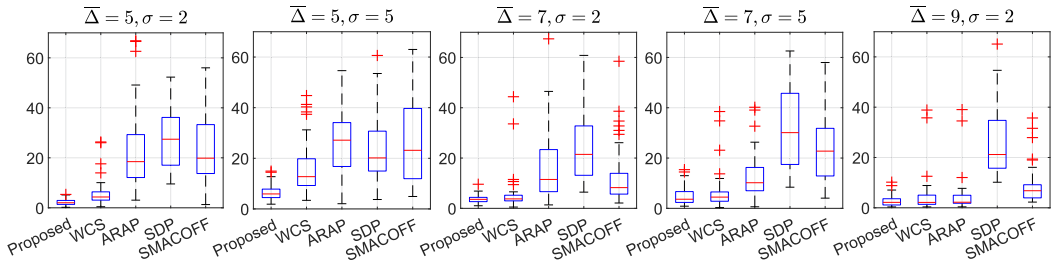


Fig. 17. The distribution of localization error of different algorithms.

algorithms. As shown in Fig. 18, we evaluate the location errors under six different ranging radius settings, so that the average node degree varies in $\{5, 6, 7, 8, 9, 10, 11, 12\}$ and σ varies in $\{0, 2, 5\}$.

The CDFs of the location errors in two sparse network settings when $\overline{D} = 5$ and $\overline{D} = 7$ are given in the first column, i.e., Figs. 18a, 18d, and 18g and the second column i.e., Figs. 18b, 18e, and 18h respectively. From the CDF curves, it can be seen more clearly that the proposed algorithm provides the best accuracy under different noise levels in sparse networks. The results show the importance of inferring negative edges to resolve flip ambiguities. The third column is the average locating errors of different algorithms as a function of the average node degree. The average location errors decrease as the network becomes less sparse (with higher node degree). When the network becomes dense, i.e., when $\overline{\Delta} \geq 10$, all the performances of the five algorithms become rather well.

From the results, we can see SDP and SMACOF are more sensitive to sparsity comparing with the component stitching methods, such as ARAP, WCS and our proposed method. Our proposed scheme provides the best accuracy under different sparsity and noise level settings.

8.4.2 *Evaluation under Multiplicative Noise Model*

We evaluate the impact of ranging noise of distance measurements. In previous sections, it is assumed that $d_{i,j} = \|\mathbf{p}_i - \mathbf{p}_j\|_2 + \sigma_{i,j}$, where $\sigma_{i,j} \sim N(0, \sigma^2)$. We call it *additive noise* . Another option for noise model is that larger noise

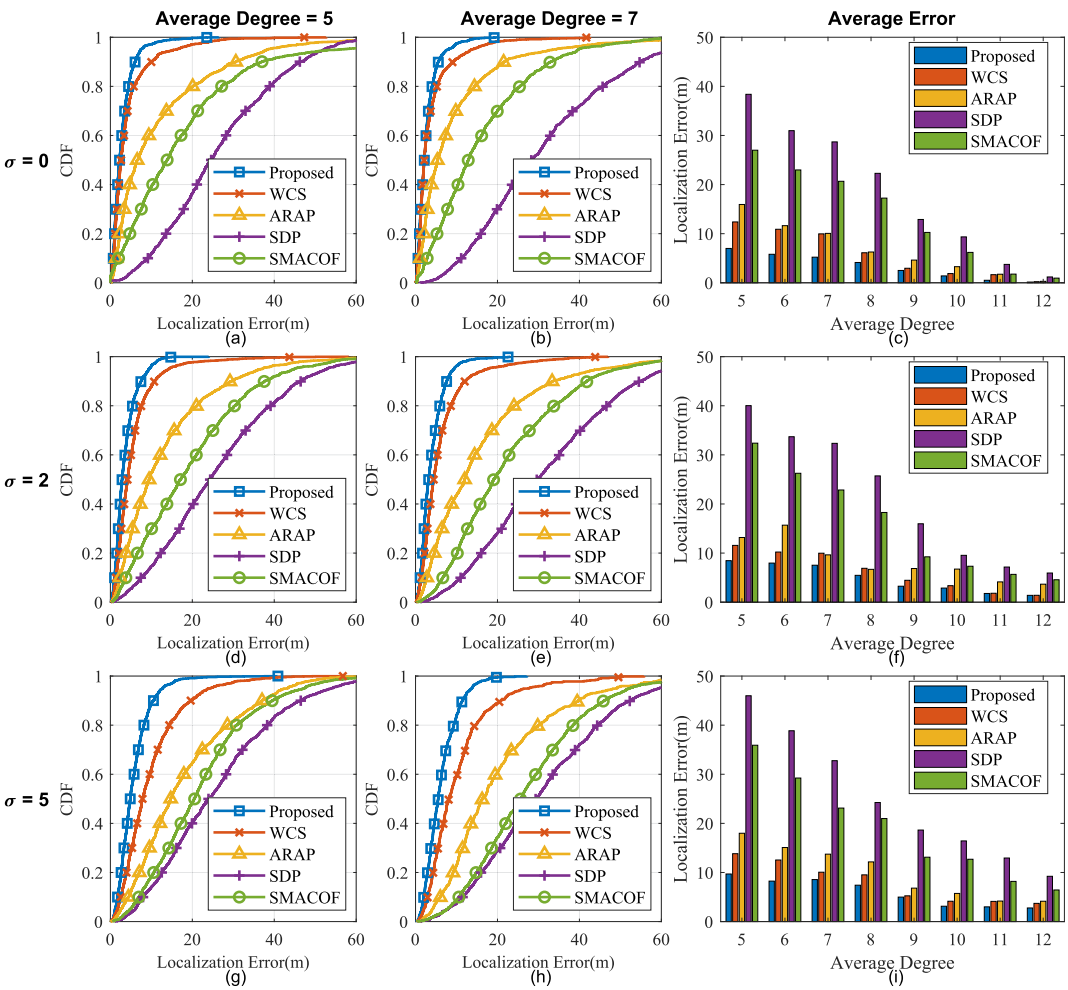
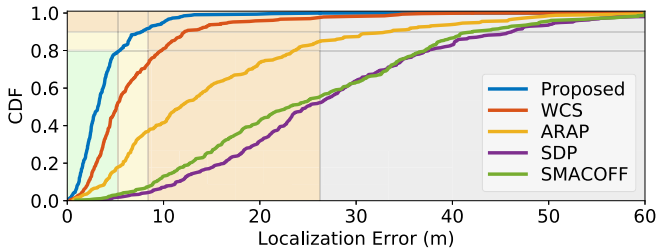
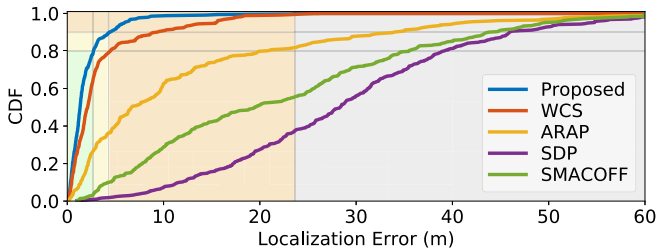


Fig. 18. Comparing the localization performances of different algorithms in different ranging radius, and noise level settings.



(a) CDF of localziation errors under additive noise model.



(b) CDF of localziation errors under multiplicative noise model.

Fig. 19. CDF of localization errors. Shaded regions show the cumulative distribution of the top 100, 90, and 80 percent localization errors of the proposed schema, which are shown in orange, yellow, and green, respectively.

variances exist for longer measurements and smaller for shorter measurements. Assume $d_{i,j} = (1 + \sigma_{i,j})\|\mathbf{p}_i - \mathbf{p}_j\|_2$, where $\sigma_{i,j} \sim N(0, \sigma^2)$. Similarly, we call it *multiplicative noise*.

Fig. 19 shows the impacts of the additive and multiplicative noises. The CDFs of localization errors in a sparse network where $\overline{D} = 5$ and $\sigma = 2$ are shown. From the CDF curves and orange shaded-area, it can be observed that multiplicative noise leads to less localization errors. The green- and yellow-shaded regions show, respectively, 80 and 90 percent of all localization errors. Comparing the top 100, 90 and 80 percent errors, we can see that the average errors under *multiplicative noise* are constantly smaller than errors under *additive noise* . This also suggests that the multiplicative noise model has less impact on localization accuracy.

8.5 The Localization Time Consumption

We conduct further experiments to compare the time consumption of different algorithms under various parameter settings. Table 5 shows that the proposed scheme conducts efficient inference of negative edges and requires even shorter running time than WCS. It is because that finding redundant rigid components in WCS by bipartite matching and component merging is time-consuming, while the proposed scheme detects 3-connected components by SPQR tree is more efficient. The other three algorithms are no doubt having better efficiency at a cost of worse accuracy. Overall, the proposed scheme provides an efficient and accurate state-of-the-art approach for sparse network localization.

9 Conclusion

This paper investigates conditions on resolving flipping ambiguity in rigid graphs and proposes LFFC and CFFC to resolve BFG-level and component-level ambiguity. A *LFFC, CFFC and RWCS synchronization* routine is proposed, where an NC-Tree structure is utilized throughout the scheme. LFFC needs only neighborhood information, which can be easily adopted to construct the NC-Tree. Then CFFC is applied on the NC-Tree from bottom to top to conduct further disambiguation. The final realization is obtained by Residual-based Weighted Component Stitching (RWCS), where the influence of high-quality local realizations are strengthened. Experiments have verified the good efficiency and accuracy of the proposed framework. In future work, the LFFC condition can be easily adopted in distributed localization algorithms because it needs only two-hop local information. The LFFC and CFFC conditions also provide hints on the optimization of node deployment and motion control of agents for better network localization, which will be studied in future work.

TABLE 5
The Localization Time Consumption (s)

Parameter Setting	Proposed	WCS	ARAP	SDP	SMACOF
$\overline{\Delta} = 5, \sigma = 2$	**11.82**	38.51	4.71	1.61	0.39
$\overline{\Delta} = 5, \sigma = 5$	**8.42**	38.24	3.17	1.82	0.46
$\overline{\Delta} = 7, \sigma = 2$	**11.27**	72.56	5.46	1.73	0.38
$\overline{\Delta} = 7, \sigma = 5$	**6.89**	61.96	2.83	1.70	0.39

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China Grant No. 61972404, 61672524, 11671400. The Fundamental Research Funds for the Central University, and the Research Funds of Renmin University of China, 2015030273.

References

[1] Y. Wang, Y. Wu, and Y. Shen, "Cooperative tracking by multi-agent systems using signals of opportunity," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 93–105, Jan. 2020.
[2] R. Mendrzik, F. Meyer, G. Bauch, and M. Z. Win, "Enabling situational awareness in millimeter wave massive MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 5, pp. 1196–1211, Sep. 2019.
[3] H. Zhao, J. Wei, S. Huang, L. Zhou, and Q. Tang, "Regular topology formation based on artificial forces for distributed mobile robotic networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 10, pp. 2415–2429, Oct. 2019.
[4] Y. Wang, T. Sun, G. Rao, and D. Li, "Formation tracking in sparse airborne networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2000–2014, Sep. 2018.
[5] L. Cheng, Y. Li, M. Xue, and Y. Wang, "An indoor localization algorithm based on modified joint probabilistic data association for wireless sensor network," *IEEE Trans. Ind. Inform.*, early access, Mar. 10, 2020, doi: 10.1109/TII.2020.2979690.
[6] S. Rai and S. Varma, "Localization in wireless sensor networks using rigid graphs: A review," *Wireless Pers. Commun.*, vol. 96, no. 3, pp. 4467–4484, 2017.
[7] M. Y. Arafat and S. Moh, "Localization and clustering based on swarm intelligence in UAV networks for emergency communications," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8958–8976, Oct. 2019.
[8] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009.
[9] M. Cucuringu, Y. Lipman, and A. Singer, "Sensor Network Localization by Eigenvector Synchronization over the euclidean Group," *ACM Trans. Sen. Netw.*, vol. 8, no. 3, pp. 19:1–19:42, 2012.
[10] B. Jackson and J. Owen, "Equivalent realisations of a rigid graph," *Discrete Appl. Math.*, vol. 256, pp. 42–58, 2019.
[11] D. K. Goldenberg *et al.*, "Network localization in partially localizable networks," in *Proc. 24th Annu. Joint Conf. IEEE Comput. Commun. Societies*, 2005, pp. 313–326.
[12] R. Connelly, T. Jordán, and W. Whiteley, "Generic global rigidity of body–bar frameworks," *J. Combinatorial Theory, Series B*, vol. 103, no. 6, pp. 689–705, 2013.
[13] Z. Yang and Y. Liu, "Understanding node localizability of wireless ad-hoc networks," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[14] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proc. 7th Annu. Int. Conf. Mobile Comput. Netw.*, 2001, pp. 166–179.

[15] H. Shen, Z. Ding, S. Dasgupta, and C. Zhao, "Multiple source localization in wireless sensor networks based on time of arrival measurement," *IEEE Trans. Signal Process.*, vol. 62, no. 8, pp. 1938–1949, Apr. 2014.

[16] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3607–3613.

[17] S. Korkmaz and A.-J. van der Veen, "Robust localization in sensor networkswith iterative majorization techniques," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2009, pp. 2049–2052.

[18] A. M.-C. So and Y. Ye, "Theory of semidefinite programming for sensor network localization," *Math. Program.*, vol. 109, no. 2–3, pp. 367–384, 2007.

[19] A. Saha and B. Sau, "Network localization by non-convex optimization," in *Proc. 7th ACM Int. Workshop Mobility Interference MiddleWare Manage. HetNets*, 2017, pp. 1–6.

[20] L. Zhang, L. Liu, C. Gotsman, and S. J. Gortler, "An as-rigid-as-possible approach to sensor network localization," *ACM Trans. Sensor Netw.*, vol. 6, no. 4, 2010, Art. no. 35.

[21] B. Jackson and A. Nixon, "Stress matrices and global rigidity of frameworks on surfaces," *Discrete Comput. Geometry*, vol. 54, no. 3, pp. 586–609, 2015.

[22] R. Connelly, "Generic global rigidity," *Discrete Comput. Geom.*, vol. 33, no. 4, pp. 549–563, 2005.

[23] J. Aspnes *et al.*, "A theory of network localization," *IEEE Trans. Mobile Comput.*, vol. 5, no. 12, pp. 1663–1678, Dec. 2006.

[24] Z. Yang, Y. Liu, and X. Y. Li, "Beyond trilateration: On the localizability of wireless ad-hoc networks," in *Proc. IEEE INFOCOM*, 2009, pp. 2392–2400.

[25] D. J. Jacobs and B. Hendrickson, "An algorithm for two-dimensional rigidity percolation: The pebble game," *J. Comput. Phys.*, vol. 137, no. 2, pp. 346–365, 1997.

[26] S. J. Gortler, A. D. Healy, Dylan, and P. Thurston, "Characterizing generic global rigidity," *Amer. J. Math.*, vol. 132, no. 4, pp. 897–939, 2010.

[27] D. Shamsi, N. Taheri, Z. Zhu, and Y. Ye, "Conditions for correct sensor network localization using SDP relaxation," in *Discrete geometry and optimization*. Berlin, Germany: Springer, 2013, pp. 279–301.

[28] A. A. Kannan, B. Fidan, and G. Mao, "Analysis of flip ambiguities for robust sensor network localization," *IEEE Trans. Veh. Technol.*, vol. 59, no. 4, pp. 2057–2070, May 2010.

[29] X. Wang, Z. Yang, J. Luo, and C. Shen, "Beyond rigidity: Obtain localisability with noisy ranging measurement," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 8, no. 1/2, pp. 114–124, 2011.

[30] W. Liu, E. Dong, Y. Song, and D. Zhang, "An improved flip ambiguity detection algorithm in wireless sensor networks node localization," in *Proc. 21st Int. Conf. Telecommun.*, 2014, pp. 206–212.

[31] W. Liu, E. Dong, and Y. Song, "Analysis of flip ambiguity for robust three-dimensional node localization in wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 97, pp. 57–68, 2016.

[32] Q. Guo, Y. Zhang, J. Lloret, B. Kantarci, and W. K. G. Seah, "A localization method avoiding flip ambiguities for micro-UAVs with bounded distance measurement errors," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1718–1730, Aug. 2019.

[33] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 50–61.

[34] F. Sottile and M. A. Spirito, "Robust localization for wireless sensor networks," in *Proc. 5th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, 2008, pp. 46–54.

[35] H. Akcan and C. Evrendilek, "Reducing the number of flips in trilateration with noisy range measurements," in *Proc. 12th Int. ACM Workshop Data Eng. Wireless Mobile Acess*, 2013, pp. 20–27.

[36] D. K. Goldenberg *et al.*, "Localization in sparse networks using sweeps," in *Proc. 12th Annu. Int. Conf. Mobile Comput. Netw.*, 2006, pp. 110–121.

[37] G. Oliva, S. Panzieri, F. Pascucci, and R. Setola, "Sensor networks localization: Extending trilateration via shadow edges," *IEEE Trans. Autom. Control*, vol. 60, no. 10, pp. 2752–2755, Oct. 2015.

[38] F. Wang, L. Qiu, and S. S. Lam, "Probabilistic region-based localization for wireless networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 11, no. 1, pp. 3–14, 2007.

[39] S. Bai and H. Qi, "Tackling the flip ambiguity in wireless sensor network localization and beyond," *Digit. Signal Process.*, vol. 55, pp. 85–97, 2016.

[40] S. Severi, G. Abreu, G. Destino, and D. Dardari, "Understanding and solving flip-ambiguity in network localization via semidefinite programming," in *Proc. IEEE Global Telecommun. Conf.*, 2009, pp. 1–6.

[41] X. Wang, Y. Liu, Z. Yang, K. Lu, and J. Luo, "OFA: An optimistic approach to conquer flip ambiguity in network localization," *Comput. Net.*, vol. 57, no. 6, pp. 1529–1544, 2013.

[42] X. Shi, G. Mao, B. D. O. Anderson, Z. Yang, and J. Chen, "Robust localization using range measurements with unknown and bounded errors," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 4065–4078, Jun. 2017.

[43] F. Xiao, W. Liu, Z. Li, L. Chen, and R. Wang, "Noise-tolerant wireless sensor networks localization via multinorms regularized matrix completion," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2409–2419, Mar. 2018.

[44] X. Wang, Y. Liu, Z. Yang, J. Liu, and J. Luo, "ETOC: Obtaining robustness in component-based localization," in *Proc. 18th IEEE Int. Conf. Netw. Protocols*, 2010, pp. 62–71.

[45] X. Wang, J. Luo, Y. Liu, S. Li, and D. Dong, "Component-based localization in sparse wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 2, pp. 540–548, Apr. 2011.

[46] T. Sun, Y. Wang, D. Li, Z. Gu, and J. Xu, "WCS: Weighted component stitching for sparse network localization," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2242–2253, Oct. 2018.

[47] U. A. Khan, S. Kar, and J. M. F. Moura, "Linear theory for self-localization: Convexity, barycentric coordinates, and cayley–menger determinants," *IEEE Access*, vol. 3, pp. 1326–1339, 2015.

[48] S. Safavi, U. A. Khan, S. Kar, and J. M. F. Moura, "Distributed localization: A linear theory," *Proc. IEEE*, vol. 106, no. 7, pp. 1204–1223, Jul. 2018.

[49] U. A. Khan, S. Kar, and J. M. F. Moura, "Distributed sensor localization in random environments using minimal number of anchor nodes," *IEEE Trans. Signal Process.*, vol. 57, no. 5, pp. 2000–2016, May 2009.

[50] U. A. Khan, S. Kar, and J. M. F. Moura, "DILAND: An algorithm for distributed sensor localization with noisy distance measurements," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1940–1947, Mar. 2010.

[51] Y. Diao, Z. Lin, and M. Fu, "A barycentric coordinate based distributed localization algorithm for sensor networks," *IEEE Trans. Signal Process.*, vol. 62, no. 18, pp. 4760–4771, Sep. 2014.

[52] P. P. V. Tecchio, "Range-only node localization: The arbitrary anchor case in d-dimensions," Publicly Accessible Penn Dissertations, no. 3519, 2019.

[53] M. Z. Win *et al.*, "Network localization and navigation via cooperation," *IEEE Commun. Magazine*, vol. 49, no. 5, pp. 56–62, May 2011.

[54] M. Z. Win, F. Meyer, Z. Liu, W. Dai, S. Bartoletti, and A. Conti, "Efficient multisensor localization for the internet of things: Exploring a new class of scalable localization algorithms," *IEEE Signal Process. Magazine*, vol. 35, no. 5, pp. 153–167, Sep. 2018.

[55] R. M. Buehrer, H. Wymeersch, and R. M. Vaghefi, "Collaborative sensor network localization: Algorithms and practical issues," *Proc. IEEE*, vol. 106, no. 6, pp. 1089–1114, Jun. 2018.

[56] M. Z. Win, Y. Shen, and W. Dai, "A theoretical foundation of network localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1136–1165, Jul. 2018.

[57] M. Z. Win, W. Dai, Y. Shen, G. Chrisikos, and H. V. Poor, "Network operation strategies for efficient localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1224–1254, Jul. 2018.

[58] B. Teague, Z. Liu, F. Meyer, A. Conti, and M. Z. Win, "Peregrine: Network localization and navigation with scalable inference and efficient operation," 2020, *arXiv: 2001.11494*.

[59] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, "Soft information for localization-of-things," *Proc. IEEE*, vol. 107, no. 11, pp. 2240–2264, Nov. 2019.

[60] S. Mazuelas, A. Conti, J. C. Allen, and M. Z. Win, "Soft range information for network localization," *IEEE Trans. Signal Process.*, vol. 66, no. 12, pp. 3155–3168, Jun. 2018.

[61] S. Bartoletti, W. Dai, A. Conti, and M. Z. Win, "A mathematical model for wideband ranging," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 2, pp. 216–228, Mar. 2015.

[62] B. Jackson, "Notes on the rigidity of graphs," in *Proc. Levico Conf. Notes*, 2007, vol. 4. [Online]. Available: www.science.unitn.it/cirm/JacksonLectures.pdf

[63] C. Gutwenger and P. Mutzel, "A linear time implementation of spqr-trees," in *Proc. Int. Symp. Graph Drawing*, 2000, pp. 77–90.

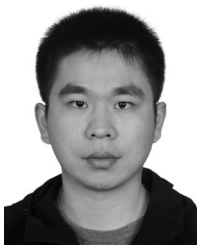
[64] R. Sanyal, M. Jaiswal, and K. N. Chaudhury, "On a registration-based approach to sensor network localization," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5357–5367, Oct. 2017.

[65] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.
[66] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.
[67] K.-P. Vo, "Finding triconnected components of graphs," *Linear Multilinear Algebra*, vol. 13, no. 2, pp. 143–165, 1983.

Haodi Ping received the BS and MS degrees from the Department of Computer Science and Technology, China University of Geosciences Beijing, in 2015 and 2018, respectively. He is currently working toward the PhD degree in the Department of Computer Sciences, Renmin University of China. His research interests include network localization algorithms, and graph optimization and applications.

Yongcai Wang received the BS and PhD degrees from the Department of Automation Sciences and Engineering, Tsinghua University, in 2001 and 2006, respectively. He worked as associated researcher with NEC Labs. China from 2007-2009. He was an research scientist with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University from 2009-2015. He was a visiting scholar with Cornell University in 2015. He is currently an associate professor at the Department of Computer Sciences, Renmin University of China. His research interests include network localization, and combinatorial optimization and applications.

Deying Li received the MS degree in mathematics from Huazhong Normal University, in 1988, and the PhD degree in computer science from the City University of Hong Kong, in 2004. She is currently a professor at the Department of Computer Science, Renmin University of China. Her research interests include wireless networks, mobile computing, social network, and algorithm design and analysis.



Tianyuan Sun received the BS and MS degrees from the Department of Computer Sciences, Renmin University of China, in 2015 and 2018, respectively. He is currently a research engineer with the HTC Research Beijng, China. His research interests include network localization algorithms, vision based perception algorithms for robot and virtual reality.