The following publication H. Wu, Z. Ding and J. Cao, "GROLO: Realistic Range-Based Localization for Mobile IoTs Through Global Rigidity," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5048-5057, June 2019 is available at https://doi.org/10.1109/JIOT.2019.2895127.

GROLO: Realistic Range-based Localization for Mobile IoTs through Global Rigidity

Hejun Wu, *Member, IEEE,* Zhimin Ding, and Jiannong Cao, *Fellow, IEEE*

***Abstract*—We study the realistic problem of range-based localization for mobile Internet of Things (IoTs) without using GPS. This problem arises from the real-world applications and is characterized by the following three challenges: (1) Inaccurate devices, such as the low cost motion sensors and moving parts in the mobile IoT nodes, make it infeasible to localize the nodes using their speed and direction. (2) When a team of IoT nodes keep moving, some nodes may get lost and be disconnected from the network due to the large accumulated errors in the distances. (3) Although the theory of global rigidity ensures the position uniqueness of nodes, there are still non-localizable nodes in a globally rigid graph. To address these challenges, we propose a distributed localization protocol, called GROLO. GROLO is able to perform efficient distributed localization through an adaptive global rigidity formation maintenance mechanism especially designed for the resource limited IoT nodes. GROLO is able to localize all of the nodes periodically in a mobile IoT by short distance adjustment. Furthermore, GROLO requires only necessary additional neighbor distance measurements. We evaluated GROLO and other localization protocols using both simulated nodes and ten real mobile nodes in an IoT. The results show that GROLO is promising for the localization and formation control with inaccurate mobile IoTs in realistic environments.**

I. Introduction

Many real-world mobile sensing or monitoring applications require sensors moving with devices under in-door environment conditions such as stations or airports where GPS devices fail. A cluster of mobile IoT nodes are desired because of their coordinated but distributed nature and higher robustness than a single robot. A variety of tasks in the in-door environments, including underwater sensing, building structure health monitoring, disaster rescue assistance, moving object tracking and other operations need mobile IoTs [5]. In comparison with stationery wireless sensor networks (WSNs), mobile IoTs have the following advantages: (1) moving sensing points; (2) self-charging; (3) low cost and (4) multi-purpose [1].

The current robots are not suitable for the above applications. Firstly, the robots are too expensive to be deployed as a team to cooperatively perform a task. The price of an accurately moving robot is about tens of thousands times to that of a moving IoT node using the moving parts like toys. Secondly, the expensive parts of robots may not work in the realistic applications of IoTs. The IoT application environments are usually not ideal, e.g., slippery grounds or rough floors due to incidents or disasters. As a result, the motion information would be inaccurate and the expensive Inertial Navigation Systems (INSs) cannot work. Finally, the deployment areas are too wide or the walls / backgrounds are too similar for a robot / a human to differentiate two locations using vision matching. Consequently, laser-radars or video cameras are not quite useful in these applications.

In the mobile sensing applications, dynamic localization and the corresponding moving monitoring are the desired tasks for IoTs. However, the motion sensors, such as directional and velocity monitors, are not accurate due to the device errors, noises and environment unevenness. The inaccurate moving information results in large accumulated errors in calculating the positions of the mobile IoT nodes. The size, cost and power constraints represent further challenges for distributed protocols of localization, since the limited communication and sensing ranges often require multi-hop communication and connectivity maintenance [11].

Our main contributions in this work are threefold: (1) We propose the minimal necessary conditions for a node to be localized in a globally rigid graph. These conditions significantly reduce the complexity of localization compared with the previous complex conditions. (2) We design a new formation control strategy that is capable of maintaining the connectivity of the edges in the globally rigid graph. (3) We evaluate our proposed GROLO protocol using both simulated and real-world IoT nodes. The experimental results demonstrate the superior accuracy of GROLO.

The organization of this paper is as follows. Section 2 provides a brief overview of the concepts of rigidity theory and the related studies as the preliminaries of this work. In Section 3, we introduce our extension of rigidity theory and summarize it into a theorem. The strategy for a distributed global rigidity construction and maintenance protocol using our proved theorem is presented in Section 4. The simulation and experimental results are reported in Section 5. Finally, the detailed reviews of the related studies and concluding remarks are presented in Section 6 and Section 7.

II. Problem Formulation and Preliminaries

A. Problem Formulation

First, the notations used in this paper are listed in Table I:

Conditions: A mobile IoT is deployed in a space without GPS signals, and the mobile nodes connect via multi-hop wireless communication. Initially, the nodes are deployed around the entrance to the space. The entrance is denoted as (0,0) and the exit point, as (x, y). At least three non-collinear

Hejun Wu is the corresponding author: wuhejun@mail.sysu.edu.cn. Hejun Wu and Zhimin Ding are with Guangdong Key Laboratory of Big Data Analysis and Processing, Department of Computer Science, Sun Yat-sen University, Guangdong, China

Jiannong Cao is with Department of Computing, Hong Kong Polytechnic University, Hong Kong, China

TABLE I
NOTATIONS

Notation	Description
$\mathcal{G}(t)$	weighted graph at time t
$\mathcal{F}$	*framework* of a graph
$\mathcal{N}$	set of graph's nodes
$\mathcal{E}$	set of graph's edge
$\mathcal{Q}$	configuration of framework
$\mathbb{E}^d$	d dimensional space
$\boldsymbol{p'_i}$	estimated coordinate vector of node i
$\boldsymbol{p'_j}$	estimated coordinate vector of node j
d'_{ij}	estimated distance between node i and node j
$J_G(x,\ y)$	Jacobin matrix
$H_F(x,\ y)$	Hessian matrix
$\nabla F(x,\ y)$	gradient of the function F
θ	deviates in angle when moving
Δv	deviates in velocity vector
Δd	current position deviation
Δd_M	upper bound in position deviation
Δd_m	lower bound in position deviation
ΔH	threshold to adjust the formation

nodes (i.e., the nodes are not on the same line) are deployed with known positions. These nodes are called the *beacons*. The remaining nodes do not know their own positions, but they can receive the information of positions of the beacons from the network broadcast. Each node can measure the distances to its single-hop neighbors using wireless signals but does not have accurate measurements of its velocity (speed and direction). To be realistic, the environment is not equipped with positioning devices or mapping points.

Problem: move the mobile IoT nodes from the entrance to the exit, following a predefined route.

Limitations: In applications requiring a mobile IoT, the network deployment area is usually large. As a result, the node communication range d is far smaller than the width x and length y of the area. The direction sensors and motion sensors are not accurate, as real complex environments often causes oscillation and skidding of the node moving parts. Furthermore, the nodes cannot use the walls and stationery objects that are too far away for localization. Finally, due to the large area, the nodes cannot be densely deployed due to the high cost and low efficiency.

B. Global Rigidity and Localization

We model a cluster of mobile IoT nodes as a weighted graph, $\mathcal{G}(t)$, at time t. In $\mathcal{G}(t)$, the weight of each edge is the distance between the two nodes connected by the edge.

Framework and Rigidity: Given the nodes, edges, and weights, a weighted graph can be realized in different shapes in a space. A *framework* is such a shape realization of a graph. For instance, Fig.1 shows two different frameworks for a weighted graph, with four nodes and four edges weighted 3, 4, 3.16, and 5, respectively.

A *framework* of a graph $\mathcal{G}(t)$ is denoted as $\mathcal{F} = (\mathcal{N},\ E,\ \mathcal{P})$, where $\mathcal{N}$ is the set of graph nodes, and E is the edge set. An

Fig. 1. Frameworks of a weighted graph

edge e_{ij} indicates that nodes i and j are connected ($i, j \in \mathcal{N}$, e_ij in E). $\mathcal{P}$ is called the configuration of framework $\mathcal{F}$. $\mathcal{P} = (p_1, p_2, p_3, \cdots, p_n)$ in $\mathbb{R}^d$. Here, $\mathcal{P}_i$ is the vector from the point of node i ($i \in \mathcal{N}$) to the center of $\mathcal{F}$, which is designated as vector $\mathbf{0}$.

Two frameworks $F_1 = (\mathcal{N},\ E,\ \mathcal{P})$ and $F_2 = (\mathcal{N},\ E,\ \mathcal{Q})$ are equivalent if the following conditions are true: (1) F_1 and F_2 are formed by the same node set and edge set: $\mathcal{N}$, E. (2) $||p_i - p_j|| = ||q_i - q_j||$, where $i, j \in \mathcal{N}$ and $e_{i,j} \in E$, $\mathcal{P} = (p_1, p_2, p_3, \cdots, p_n)$, $\mathcal{Q} = (q_1, q_2, q_3, \cdots, q_n)$. F_1 and F_2 are the equivalent frameworks of the same graph $\mathcal{G}(t)$ since their node set, edge set, and edge weights are the same as those of $\mathcal{G}$. Note that $\mathcal{P}$ and $\mathcal{Q}$ are not the same.

A connected graph $\mathcal{G}(t)$ is *rigid* in $\mathbb{E}^d$ if $\mathcal{G}(t)$ has a finite number of equivalent frameworks in $\mathbb{E}^d$, where $\mathbb{E}^d$ is the d-dimensional Euclidean space. The graph shown in Fig.2(a) is a rigid graph in $\mathbb{E}^2$ (two-dimensional plane), as it has a finite number of frameworks (two possible frameworks in total) in a $\mathbb{E}^2$ plane. By contrast, Fig.2(b) shows a *flexible* graph, as it has an infinite number of frameworks.

(a)rigid graph (b)flexible graph

Fig. 2. Frameworks of a rigid graph and a flexible graph

As shown in Fig.2(a), the above generic rigidity is not sufficient to localize a node in a network since a rigid graph has several frameworks with different configurations. These different configurations specify a number of candidate coordinate vectors (positions) for a single node in the rigid graph, even if the positions of the other nodes are all fixed. For instance, in the graph of Fig.2(a), moving the right corner node from the current position to the upper candidate position does not change the lengths of the existing edges of the graph. Hence, we need to further confine the rigidity by the *global rigidity*.

Global Rigidity: Before illustrating the concept of global rigidity, we need to define *congruent*: Two equivalent frameworks $F_1 = (\mathcal{N},\ E,\ \mathcal{P})$ and $F_2 = (\mathcal{N},\ E,\ \mathcal{Q})$ are congruent if their configurations $\mathcal{P}$ and $\mathcal{Q}$ are *congruent*. Two configurations $\mathcal{P}$ and $\mathcal{Q}$ are congruent if $|\mathcal{P}| = |\mathcal{Q}|$ and $||p_k - p_m|| = ||q_k - qm||$, $k, m \in \mathcal{N}$ and $e_{k,m} \notin E$.

Given the above graph and framework models, the global rigidity of a node graph instance $\mathcal{G}(t)$ is the following property: any two frameworks of $\mathcal{G}(t)$ are congruent. In a globally rigid graph, the configurations of all frameworks are the same. Hence, in the globally rigid graph $\mathcal{G}(t)$, a node has a unique coordinate vector candidate satisfying the distance constraints between it and other nodes, when the coordinates of the other

nodes are fixed. The distance constraints are specified by the weights of graph $\mathcal{G}(t)$.

Our previous studies have proven that in E^2, if a graph satisfies: (1) globally rigid and (2) three non-collinear beacons, then all nodes in the graph are localizable [17]. As shown in Fig. 3, a unique pair of coordinates satisfy the constraints in E^2, given the coordinates of three nodes.

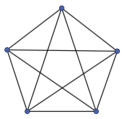

Fig. 3. Global rigidity

The global rigidity ensures unique coordinate vector (position) for each node (localizablity). Next, we present our approach to localization.

III. Range-based Localization via Global Rigidity

In accordance with the problem formulation in the previous section, we assume that three beacon nodes exist in a network. The beacons are initially deployed to form a triangle so that they are not on the same line. The nodes construct a fully connected graph through wireless communication.

Determining whether an arbitrary graph within the network is globally rigid is an NP-hard problem. We use our previous Triangle Extension (TE) approach to construct a globally rigid graph, to reduce the problem complexity [17]. In TE, initially two of the three beacons (which may not be single-hop neighbors) start a triangle extension operation by informing their neighbors to *extend* them. Then, the single-hop neighbors of the two beacons can perform the triangle extension operation. In Fig. 4, the extension operations of TE are shown in steps (1)-(4), where $B1$, $B2$, and $B3$ are beacons:

(1) a extends $B1$ and $B2$ ($B1$ and $B2$ become a's parents);
(2) b extends a and $B2$ (a and $B2$ become b's parents);
(3) c extends a and b (a and b become c's parents);
(4) $B3$ extends f and e, after extensions by d, e, and f.

The theorem of globally rigid graph determination in our previous study [17] can be used to prove that the constructed graph in Fig. 4 is globally rigid.

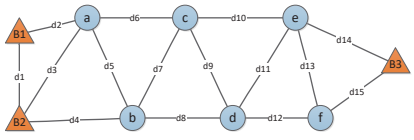

Fig. 4. Globally rigid graph constructed using TE

Upon the completion of the extensions by TE, a node initially deployed too far away from the others may not be included in the globally rigid graph. This problem can be addressed by *contracting*: The lost nodes move towards the center of the graph framework until they find at least two neighbors in the globally rigid graph. Subsequently, a lost node can extend these two neighbors as its parents and join the globally rigid graph. This contracting operation is performed in our GROLO protocol as presented next.

Nonetheless, as has been demonstrated before, the pair of coordinates of each node is only theoretically unique even in a globally rigid graph [26]. Next, we show our attempts to localize the nodes in a globally rigid graph. We prove the theorem of minimal necessary conditions for accurate node localization. Our GROLO protocol then integrates this theory to accurately localize and to control the inaccurately moving nodes as a team to their destination.

IV. GROLO Protocol

For the sake of successful mobile IoT localization, the localizability of the IoT nodes should be maintained while the nodes are moving without GPS; otherwise, the nodes might be no longer localizable after a while due to the device inaccuracy and environment noises. Also, because of the accumulated distance errors during moving, the coordinate vector of each node should be calculated periodically. Therefore, our GROLO is composed of two parts to perform two tasks: localization processing (GROLO-LP) for periodic localization and formation control (GROLO-FC) for moving towards the destination and distance adjustment for localizability.

A. GROLO-LP

GROLO-LP (Localization processing) in a mobile IoT is essentially an optimization algorithm, since its objective is to minimize the calculated position errors for each IoT node. Hence, GROLO-LP includes the following three stages for position estimation and optimization:

(1) Estimation: Estimate the coordinate vector of a node using a *distance vector* (DV);
(2) Optimization: Cross-validate the coordinate vector of a node with the information from all its neighbors using the gradient decent method;
(3) Localization: Calculate the exact position on the basis of the edge lengths of the triangles in a globally rigid graph using the Gauss-Newton method.

In the first stage of GROLO-LP, the coordinate vector of each node in a graph is estimated using the hop distance (distance vector) between the node and a beacon node. The simple equation $d'_{ij} = ||\boldsymbol{p'_i} - \boldsymbol{p'_j}||$ calculates the estimated distance d'_{ij} based on the estimated coordinates $\boldsymbol{p'_i}$ and $\boldsymbol{p'_j}$ of node i and its neighbor j. Fig.5 shows an example network topology and the coordinate vector estimated in this stage. As shown in the figure, this step is unable to obtain satisfactory accuracy.

As specified in the assumption, the distance between single-hop neighbors can be measured; hence, the estimated coordinate vector is cross-validated using the measured distance. As the DV method is not accurate [8], we use the gradient descent method to minimize $\mathcal{L}_i = \Sigma_{j\in\mathcal{N}(i)}(d'_{ij} - d_{ij})^2$, where $\mathcal{N}(i)$ is the neighbor set of i, and d_{ij} is the measured distance between nodes i and j.

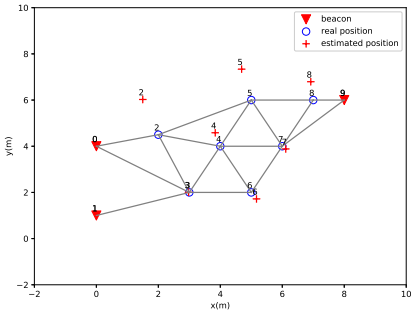

Fig. 5. DV-distance

However, the gradient descent method sometimes falls into a local minima, which causes errors. Therefore, in the third stage, we further reduce the errors using the property of global rigidity. In a globally rigid graph, a node and its parents form a triangle, as shown in Fig.4. In this stage, a node uses the estimated coordinate vector from the last stage and the distances between itself and the beacons to obtain the global optimal solution. For instance, in Fig. 4, the coordinate vector of node a can be solved by the vectors in $\triangle B_1B_2a$, where $\boldsymbol{v} = (x_v, y_v)^T$ is a vector, $\boldsymbol{v} \in \{B_1B_2, \boldsymbol{a}\}$, $d_1 = ||B_1 - \boldsymbol{a}||$ and $d_2 = ||B_2 - \boldsymbol{a}||$:

$$\begin{cases} (x_a - x_{B1})^2 + (y_a - y_{B1})^2 = {d_1}^2 \\ (x_a - x_{B2})^2 + (y_a - y_{B2})^2 = {d_2}^2 \end{cases} \tag{1}$$

Since d_1 and d_2 can be measured, Eq.(1) can be solved via the Gauss-Newton method. Consider the system of equations in Eq.(2), and let $F(x,\ y) = \frac{1}{2} f_1^2(x,\ y) + \frac{1}{2} f_2^2(x,\ y)$. The problem is then transformed into finding $(x^*,\ y^*)$ that minimizes $F(x,\ y)$, as shown in Eq.(3).

$$\begin{cases} f_1(x,\ y) = (x - x_1)^2 + (y - y_1)^2 - d_1^2 \\ f_2(x,\ y) = (x - x_2)^2 + (y - y_2)^2 - d_2^2 \end{cases} \tag{2}$$

$$(x^*,\ y^*) = \underset{(x,\ y)}{\arg\min}\ F(x,\ y) \tag{3}$$

The Gauss-Newton approach is used to solve Eq.(3) via the following steps: Given the initial value $(x^{(0)},\ y^{(0)})$ the optimal solution $(x^*,\ y^*)$ can be iteratively calculated as:

$$(x^{(k+1)},\ y^{(k+1)}) = (x^{(k)},\ y^{(k)}) + h_k \tag{4}$$

until $|\ F(x^{(k+1)},\ y^{(k+1)}) - F(x^{(k)},\ y^{(k)})\ | < \varepsilon$, where ε is a predefined constant, and h_k is calculated using the following equations (5)-(7):

$$h_k = -H_F(x^{(k)},\ y^{(k)})^{-1}\ \nabla F(x^{(k)},\ y^{(k)}) \tag{5}$$

$H_F(x,\ y)$ is the Hessian of $F(x,\ y)$, which is estimated as:

$$H_F(x,\ y) \approx J_G^T(x,\ y)\ J_G(x,\ y) \tag{6}$$

Finally, $\nabla F(x,\ y)$ is obtained using Eq.(7), where $J_G^T(x,\ y)$ is the Jacobin.

$$\nabla F(x,\ y) = J_G(x,\ y) \begin{bmatrix} f_1(x,\ y) \\ f_2(x,\ y) \end{bmatrix} \tag{7}$$

From the above localization calculation steps, we can generalize a rule to accurately localize an IoT node. This rule is given in Theorem 1.

Theorem 1. *Given a node with an unknown position in E^2 and the node extended two position-known nodes as its parents using TE, the coordinate vector of this node can be accurately calculated when the following conditions hold: (1) the node has a position-known neighbor; (2) the neighbor is not on the same line as its two parent nodes.*

Proof. As shown in Fig. 6, the coordinate vector of node A are unknown. Nodes P and Q are extended by A using TE. According to the extension rule of TE, the nodes are direct communication neighbors of A and are also the two parents of A. Hence, their distances can be measured as shown in Fig. 6. We can use the Gauss-Newton method to obtain two possible solutions for the coordinate vector of A, denoted as S_1 and S_2.

Now, suppose that A has another neighbor R, whose position is also known. R is not on the connecting or extension line of P and Q. The distance RA can be measured by nodes R and A. The correct coordinate vector must satisfy the constraint of distance RA. Sine there are only two possible coordinate vectors for A, the coordinate vector of node A can be determined by using the coordinate vector of R to perform cross-validation.

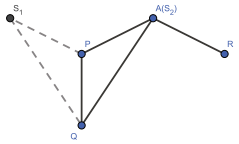

Fig. 6. The position of node A, calculated using the positions of its parents P and Q, can be cross-validated by its neighbour R

□

In practice, we use the gradient descent method to obtain the possible coordinate vector for A. Then, we apply the Gauss-Newton method to A with the coordinates of its neighbors. When a node cannot be localized, GROLO checks whether Theorem 1 is satisfied. If not, a *formation contracting* operation is conducted to enable the node to be localized. This operation is presented later in the section of $GROLO-FC$.

The above three localization stages are formulated in Algorithm 1. Initially, before localization, the globally rigid graph should be constructed such that the nodes are theoretically localizable. The globally rigid graph is maintained by the GROLO-FC so that the IoT nodes can be localized continuously. Then, in the first stage, Line 8 estimates the coordinate vector of each node based on the coordinates of the beacons and the hop distances between the IoT node and the beacons. In the second stage, Lines 9-13 run the gradient descent method. Finally, Line 15 uses the Gauss-Newton method to calculate the exact values of the coordinates.

Algorithm 1 GROLO-LP Distributed

this is node i:
type Node: {id, state, $\boldsymbol{p}\{x,y\}$, pt1, pt2, Neighbors }
//$\boldsymbol{p}\{x,y\}$ is the coordinate vector of the node
//state is beacon, flexible, rigid, localizable
type Neighbors[]:{id, state, d, $\boldsymbol{p}\{x,y\}$ }
//d: distance between this node and the neighbor
while not receiving the beacon position messages **do**
  wait
//p_i: the coordinate vector of the current node
//Use TE to construct the globally rigid graph
$\boldsymbol{r_i} \leftarrow newNode$ //on node i
run TE to determine node localizability and estimate $\boldsymbol{r_i}.\boldsymbol{p}$
if $\boldsymbol{r_i}$.state is localizable **then**
  for j in Neighbors[] and optimization not converge **do**
    $\boldsymbol{b_j}$ gets Neighbors[j]
    $\mathcal{L}_i \leftarrow \frac{1}{2}\sum_{j=1}^{m}(\boldsymbol{r_i}.d - dist(r_i.\boldsymbol{p}, \boldsymbol{b_j}.\boldsymbol{p}))^2$
    $\boldsymbol{r_i}.\boldsymbol{p} \leftarrow$ GD$(i, j, \mathcal{L}_i)$ //gradient descent
  for parents $\boldsymbol{r_i}.pt1$ and $\boldsymbol{r_i}.pt2$ not NULL **do**
    $\boldsymbol{r_i}.\boldsymbol{p} \leftarrow$ Gauss-Newton($\boldsymbol{r_i}.\boldsymbol{p}$, $\boldsymbol{r_i}.pt1.\boldsymbol{p}$, $\boldsymbol{r_i}.pt2.\boldsymbol{p}$)
output $r_i.p$

In summary, our GROLO-LP uses global rigidity to ensure localizability. The gradient decent is to minimize the estimation errors. The global rigidity and neighbor information are used for further validation using the Gauss-Newton method.

B. GROLO-FC

This subsection presents the motion control strategy in our GROLO to maintain the global rigidity formation and to adjust the node distances for dynamic localization in a moving mobile IoT. The controlling operation still uses the velocity of a node, but its inaccuracy is considered.

GROLO-FC includes two phases to satisfy the above requirements: (1) epoch moving phase and (2) position adjustment phase. During the first phase, as long as the global rigidity is maintained, the nodes can be localized continuously because the GROLO-LP algorithm has constructed the globally rigid graph and obtained the initial accurate positions for the mobile nodes. However, as the motion control is inaccurate, the distances between nodes may become too far or too close to maintain the links between single-hop neighbors. Furthermore, obstacles often cause formation changes. Hence, the second phase adjusts the positions of the nodes so that the links are not broken.

Specifically, GROLO-FC creates a virtual center (O), which is defined as the center of the beacons. The coordinate vector of (O) are defined as the arithmetic mean of the coordinates of the n_B number of beacons:

$$(x_O, y_O)^T = (\frac{1}{n_B}\sum_{j=1}^{n_B} x_{B_j},\ \frac{1}{n_B}\sum_{j=1}^{n_B} y_{B_j})^T \tag{8}$$

In an epoch moving phase, each node moves towards the destination with its normal speed. Before moving, a node r_i records the initial vector P_iO, where P_i is the position of r_i. When the virtual center moves from O to O' and, subsequently, P_i changes to P_i', node i has the new vector $P_i'O'$. The ideal moving pace is $P_i'P_i = O'O$, which means that every node moves with the same velocity vector $\boldsymbol{v}$. However, this condition is not realistic due to the inaccurate motion control of nodes.

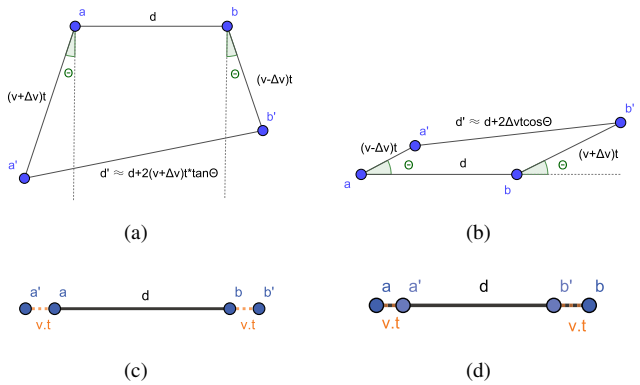

Fig. 7. Distance and direction deviations

In realistic environments, the velocity vector may deviate to a certain degree, and the speed may be different from the desired value. Suppose that a node deviates at most Θ ($\Theta < 45°$) from the specified direction and that its speed range is $[v-\Delta v, v+\Delta v]$. Fig. 7 shows the possible deviation scenarios in such circumstances. In Fig. 7, the dashed lines denote the specified node moving directions and routes. The solid lines marked with velocities are the real routes followed by the nodes. d' is the real distance between two nodes after they have moved.

Fig.7(a) illustrates the scenario when the movement direction of two nodes differs from the line direction of d. From time 0 to time t, the upper bound for the change in distance Δd of two nodes is $\Delta d_M^{(1)} = 2t(v+\Delta v)tan(\Theta)$, and the lower bound of $\Delta d_m^{(1)} = -2t(v+\Delta v)tan(\Theta)$. The scenario of moving in the line direction is illustrated in Fig.7(b). From time 0 to time t, the upper bound for the change in distance Δd of two nodes is $\Delta d_M^{(1)} = 2t(\Delta v)cos(\Theta)$, and the lower bound of $\Delta d_m^{(1)} = -2t(\Delta v)cos(\Theta)$.

Note that the above analysis is based on the assumptions that Θ is relatively small and that the deviation degrees of the two nodes are similar. Suppose that the deviations cause two nodes to move in exactly opposite directions, as shown in Fig.7(c) and Fig.7(d). Then, the upper bound of the change in distance is $\Delta d_M^{(2)} = 2(v+\Delta v)t$, and the lower bound $\Delta d_m^{(2)} = -2(v+\Delta v)t$.

As $\Theta < 45°$, the extreme scenarios in Fig.7(c) and Fig.7(d) usually do not occur. If such a scenario does occur, the node moving in the opposite direction should leave the node team since it causes network failures. Therefore, we consider only the two moving scenarios shown in Fig.7(a) and Fig.7(b). The change in distance bounds can then be summarized as follows. Upper bound:

$$\Delta d_M = \max_{0 \leqslant \theta \leqslant \Theta} \{2tan(\theta)(v+\Delta v)t, 2\Delta v t cos(\theta)\}$$

Lower bound:

$$\Delta d_m = \min_{0 \leqslant \theta \leqslant \Theta} \{-2tan(\theta)(v+\Delta v)t, -2\Delta v t cos(\theta)\}$$

The maximum and minimum values for d_M and d_m can be calculated after v, Δv and Θ are given. These Δd_M and Δd_m values are used to predict whether a node may collide or disconnect from the globally rigid graph within the next epoch. When this is probably happening by prediction, it performs the second phase: position adjustment.

For efficiency, we convert the upper and lower bounds to a timing for each node to start the position adjustment phase. To obtain the time at which to perform the adjustment, we need to set a threshold that specifies the maximum allowable distance change. Eq.(9) shows the definition of the threshold. In Eq.(9), d_{col} is the shortest allowable distance between two nodes. A distance shorter than d_{col} leads to a collision. d_{com} is the longest allowable distance between two nodes i and j, and (i, j) is an edge in E_{pc}. E_{pc} is the set of all links in the globally rigid graph by triangle extension.

$$\Delta H = \min\{\frac{1}{2}\min_{i,j \in \mathcal{N}}(d_{ij} - d_{col}), \frac{1}{2}\min_{(i,j) \in E_{pc}}(d_{com} - d_{ij})\} \quad (9)$$

Algorithm 2 shows the operations following the above rule. In this algorithm, the time Synchronization method in Line 7 is adopted from the previous study [18]. When the destination point of a node is about to exceed the ΔH diameter circle: $|P_i O| - |P_i' O| > \Delta H$, the node enters the position adjustment phase. The time of ΔH being exceeded can be predicted based on the threshold, current velocity, position and inaccuracy of the velocity. $\frac{1}{2}\Delta d_M \geq \Delta H$ is used to calculate the safe moving time period Δt, where v, Δv, Θ, and ΔH are given and Δd_M is the upper bound of distance deviations. The nodes can thus move within an epoch of time length t. After that time, the nodes must further localize themselves and check whether the distances require further adjustment.

Node failure can be addressed similarly via position adjustment. When a node failure breaks the global rigidity, the neighbors of the node start a *contracting* adjustment as shown in Line 11 of Algorithm 2. From the border line of the network, the nodes move one step towards the virtual center, hop by hop, until the global rigidity is recovered.

In normal scenarios, all of the nodes running Algorithm 2 should follow the same route as they are in a team. However, if there is an obstacle that forces some nodes to deviate from the route of their team, these nodes should determine a new route towards the destination after the deviation. When the obstacle is big enough, then all of the nodes in the network will deviate from the obstacle using the same route so that they still maintain the formation. If unfortunately the obstacle force several nodes disconnect from the team, these lost nodes can later recover to the original route after the obstacle is bypassed.

Finally, the time complexity is analyzed as follows. The graph construction in GROLO-LP is a one-time operation during initialization. Its time complexity is $O(n)$ [17]. The time complexity of localization is $O(m^2)$, where m is the number of direct communication neighbors. In total, the time complexity of GROLO-LP makes it applicable to real-world scenarios.

Algorithm 2 GROLO-FC Distributed

```
1: In an epoch:
2: r_i.moved ← false
3: while r_i.p ≠ destination do
4:   if r_i.p ==NULL then
5:     r_i.p ← GROLO-LP() // call Algorithm 1
6:     //synchronization after localization t ← 0
7:     timeSync(&t)
8:     broadcast(GROLO-LP-DONE-MSG)
9:   Δd ← error distance
10:  if Δd + tan(Θ)(v+Δv)Δt > ΔH then
11:    contracting : (r_i.p ← obj p)
12:  Δt = (ΔH − Δd)/tan(Θ)(v+Δv)
13:  r_i.velocity ← (v, α) // v is speed, α is the direction
     towards the destination
14:  if r_i.state is beacon then
15:    r_i.move(r_i.velocity, Δt) // move vΔt
16:  else
17:    while no GROLO-LP-DONE-MSG from beacons do
18:      r_i.wait_for_msg()
19:    r_i.respond_to_beacons()
20:    r_i.move(r_i.velocity, Δt)
21:  r_i.p ← NULL
```

GROLO-FC performs the node control to maintain the global rigidity formation dynamically. The time complexity of the moving control operation is $O(1)$. At the end of each epoch (Δt), GROLO-FC calls GROLO-LP to perform localization. Suppose that in the worst case every node needs to perform the above *contraction* operation. Therefore, the worst case complexity of GROLO-FC is $O(n)$.

V. Evaluation

As illustrated in the previous sections, GROLO is composed of two parts: GROLO-LP for localization and GROLO-FC for moving. The evaluation of GROLO should be separated accordingly to clarify the factors involved in each operation. In this section, we first describe the simulations to evaluate these two parts first. We then present the experiments using real mobile nodes to verify GROLO.

In simulation, we compare our GROLO-LP algorithm with three distributed localization algorithms for stationery wireless networks: gradient descent algorithm (GDA) [4][13], distance vector distance (DV-distance)[10], and multi-dimensional scaling (MDS-MAP) [16]. We do not compare the recent centralized localization algorithms that run deep learning or matrix operations on super central nodes, as they are not applicable to IoT. The simulation setup is in Table II.

TABLE II
Experimental Setup

Num of nodes	Num of beacons	Area width	Communication range
100	5	100 m	25 m

The distance measurement between the nodes is simulated using a low-cost method with RSSI. To simulate RSSI, we used a signal propagation model as in Eq. (10):

$$P = P_0 - 10 \cdot \beta \cdot log(\frac{d}{d_0}) \quad (10)$$

where P is the average received power at distance d; P_0 is the received power at reference distance d_0; d_0 is normally 1 meter. d is the distance between the receiver and the sender. β is a constant path loss exponent depending on the environment. We assume that the measured distance by RSSI is precise.

The ground-truth positions of the 100 nodes and its globally rigid graph constructed by GROLO-LP is shown in Fig.8. The positions calculated by GROLO-LP and the real node positions are shown in Fig.9. We use the root-mean-square error (RMSE) to quantitatively evaluate the accuracy of the algorithm. In the equation of $RMSE$, n is the total number of nodes being localized in an IoT; $p_i - \hat{p}_i$ is the difference between the ground truth position p_i of the i-th node:

$$RMSE = \sqrt{(\|p_i - \hat{p}_i\|)^2/n}.$$

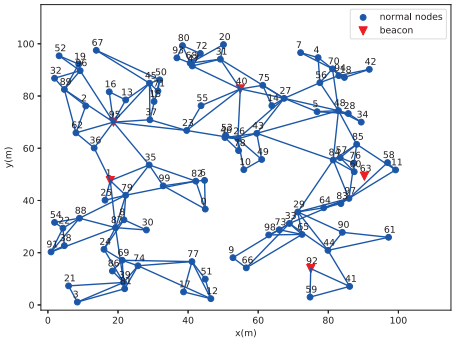

Fig. 8. The globally rigid graph of 100 nodes

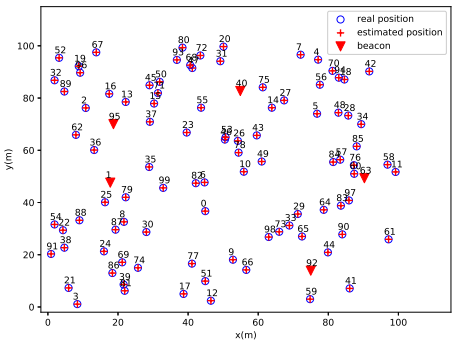

Fig. 9. Calculated positions and ground-truth positions

We compared the localizations of GROLO-LP with those of GDA, DV-distance, and MDS-MAP under a series of network configurations. In the networks, if not specially pointed out, the normal communication range is set to $2.5 \times \sqrt{(S/n)}$, where S is the deployment area, and n is the number of nodes. The communication range is adjusted to keep the mobile IoT connected; otherwise, frequent node disconnections would occur due to the sparse deployment in the large area and the short communication range. In addition, the normal beacon ratio is 5% in a network if not specified.

We evaluated the algorihtms with different communication ranges, beacon densities (beacon ratio), and network scales (node number). All the simulations are run ten times with different random. The results shown in this section are the average of ten simulations. We first conducted simulations of networks with different communication ranges. The localization RMSE results of a 100-node network running the four algorithms with different communication ranges are shown in Fig. 10. The results indicate that the accuracy of GROLO-LP outperforms the other algorithms in the network.

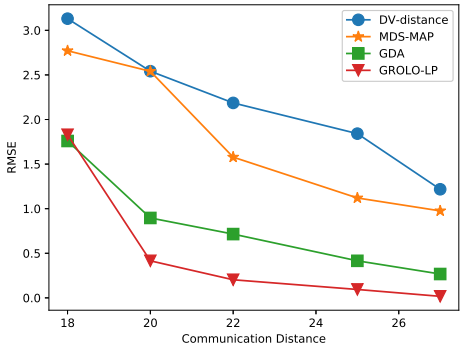

Fig. 10. RMSE Comparison of GROLO-LP with the others under different communication ranges

We then compared the localization errors of the algorithms in a 400-node network with different ratio of beacons. Fig.11 shows the results. It can be seen that MDS-MAP performs better on average. In the end, when beacons become denser, GROLO-LP performs the best. This is because MDS-MAP uses local maps and requires less beacons. When beacon number is less than 5%, the performance of GROLO-LP is worse than MDS-MAP, as constructing globally rigid graph needs at least two beacons to start locally. Nevertheless, 5% is a normal level of beacon densities in a hundred-node level scale network. Furthermore, MDS-MAP is not suitable for mobile IoTs as it assumes the stationery networks.

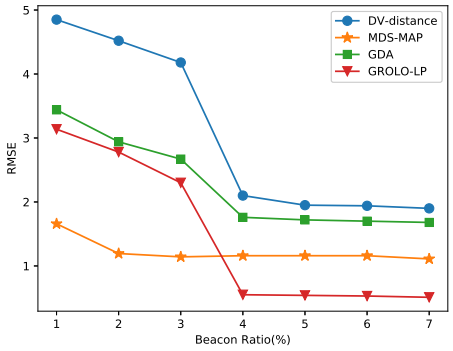

Fig. 11. RMSE Comparison of GROLO-LP with the others under different beacon densities

As the last evaluation of GROLO-LP, the algorithms were run in networks with different scales. Fig.12 shows the results of RMSE of the four algorithms. Among them, GROLO-LP performs the best on average. In some networks MDS-MAP performs better than GROLO-LP. This is because MDS-MAP localizes the nodes through building a local map on each node, which can include nodes from multi-hops away. In contrast, GROLO-LP uses less information within one hop, which causes errors in some sparsely deployed local areas.

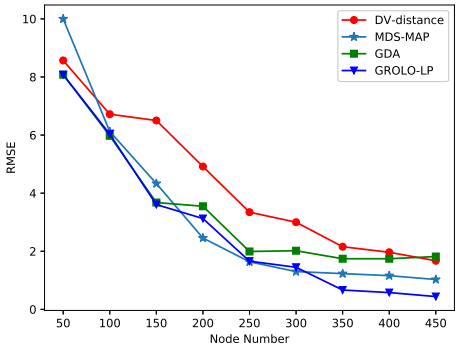

Fig. 12. RMSE Comparison of GROLO-LP with the others under different network scales

To evaluate continuous localization after node movements, we simulated random errors of speed and directions within 20% and 30°. The initial node deployment is shown as the red dots in Fig. 13. Then, the nodes were controlled to maintain a globally rigid formation while moving. The trajectory of the nodes from the entrance to the destination is shown in Fig. 13. Apparently, nodes managed to reach the destination with GROLO-FC. Moreover, Fig. 13 shows that the nodes adjusted their positions to maintain their formation.

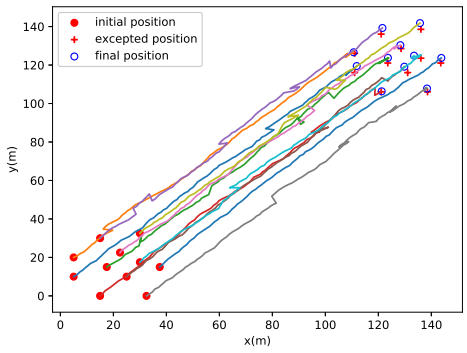

Fig. 13. Moving localization with moving errors

As our method is based on the range measurement, the range measuring error would inevitably affect the localization accuracy. We performed a series of simulations to test the relation between the distance measuring error. We found that when the number of hops is large and the node density is low, the distance measurement errors tend to accumulate hop by hop. These errors cause large localization errors. Nonetheless, if the network density is high enough, there will be more neighbors for the cross-validation in localization. These neighbours can provide extra information to correct the distance errors using probabilities. Fig. 14 shows a simulation test of mobile IoT localization with +/-1% range measuring errors. It can be seen that the distance errors can be compensated by adjustments during localization and moving. Although in comparison with Fig. 13, the localization error is a little larger, finally the nodes can approach their destinations.

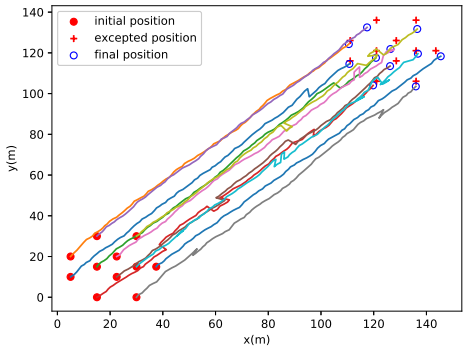

Fig. 14. Moving localization with additional 1% range measurement error

We then used 10 real-world mobile IoT nodes to verify our GROLO protocol in reality. Fig. 15 shows a snap-shot when the nodes are moving. In the experiments, we used the hexapod and four-leg toys as the moving devices. Such moving devices cannot be called robots as they lack the capabilities of synchronous localization and mapping (SLAM). They do not have radars and cameras. Each IoT node uses an embedded CPU board with Wi-Fi and Blue-tooth connectivity (RASPBERRY PI 3) as its main board. The IoT nodes communicate with each other using Wi-Fi.

Fig. 15. Experiment snap-short of 10 real-world mobile IoT nodes

The initial deployment is the same as the one shown in Fig. 5. The actual graph generated by the 10 IoT nodes is the same as that in Fig. 5, as can be seen from the messages of the neighbour information of each node. The localization error of each node is below the level of 10^{-5}, with an assumption that the distance measured is correct on the node. Therefore, the final localization error is dependent on the distance measuring error. Considering that the current range-based measurements can reach the error level of 10^{-3}, this result is quite satisfactory for realist applications. The source code can be downloaded from: "https://github.com/mylofty/DistributeGROLO".

Finally, we evaluated the energy consumption of the nodes in one time localization in a 10-node mobile IoT. We averaged

the operations of the sensor nodes in an IoT. We then sum up the time intervals of each type operation such as transmission (denoted as Tx), receiving (Rx), computation for localization (Loc), distance measuring (Dist) and sleeping (Sleep). The time out is set as five seconds to allow the sensor nodes to finish the localization. The electric current of transmission is about 0.22A on the IoT device we used. The currents of receiving, computing, distance measuring and sleeping are 0.2A, 0.2A, 0.22A, and 0.001A, respectively. The voltage of a node is 3V. Subsequently, the energy consumption can be calculated by multiplying the voltage, current and time intervals. The results are shown in Table V. Considering that a mobile IoT can charge itself, the energy consumption for each time localization is not high. A 5000 mAh battery can sustain about 455 hours, since the average current is about 11mA.

TABLE III
AVERAGE OPERATIONS ENERGY CONSUMPTION

Measurements	Tx	Rx	Loc	Dist	Sleep
t (seconds)	0.086	0.344	0.7	0.25	3.62
E(Joules)	0.057	0.206	0.420	0.165	0.011

VI. RELATED WORK

Rigidity theory [6] has been studied from various perspectives and for numerous purposes with regard to determining localizability, estimating relative or absolute positions, etc. There have been studies to construct rigid graphs for localizability of nodes in IoTs [17] [21][22]. There are also efforts devoted to maintaining the rigidity property using the rigidity eigenvalue and the rigidity matrix for robot formation control [3] [23]. However, it usually requires a super central node to coordinate the other nodes in a team and thus not applicable to IoTs. In addition to the global rigidity, Zhu et al.[26] proposed a universal rigidity method to estimate position. However, such an approach is centralized and cannot be used in distributed environments, such as mobile node formation control.

The centralized range-based localization algorithms can be mainly represented by three approaches: multi-dimensional scaling (MDS)[15], semi-definite programming (SDP) [2] and stochastic optimization approach (SA) [4]. The major problem with these centralized methods is that they depend on the global information of a mobile IoT. Due to the complexity, these methods often work in a snapshot manner, i.e., perform the calculation once and obtain the results. Such centralized snapshot algorithms cannot handle the dynamics of distributed mobile IoTs. Furthermore, the global information is often not available or outdated due to network mobility dynamics.

Decentralized approaches have been proposed for formation rigidity to address the problems of centralized schemes. For instance, using distributed method to estimate the rigidity eigenvectors [11]. Nonetheless, the estimation is still of high time complexity for IoT nodes. DV-hop and DV-distance are two classical representative localization algorithms for stationery WSNs using distance vectors (DV) [10]. The DV-hop algorithm uses the hop number of a message travelled to estimate the distances between nodes. Apparently, it cannot reach high localization accuracy [8]. The DV-distance algorithm performs better as it uses the real distance measured rather than using the average hop distance. Our algorithm adopts the idea of distance vector to estimate the initial location of nodes only for efficiency.

In comparison with these previous mobile node localization, three critical differences exist between our work and previously reported work. First, the key condition of the previous mobile localization method is that the proposed method must be started from some super nodes that can measure the positions and directions between a node and a super node. Our approach does not require such an assumption. Second, as shown by the rigidity concepts, theoretically only global rigidity can be used to uniquely localize nodes, whereas infinitesimal rigidity cannot. Third, the symmetric rigidity matrix is still used in their approach, which is expensive in terms of computation and maintenance.

Since DV alone are not enough, researchers have proposed various methods to refine the estimations from distance vectors. Savvides et al. proposed an algorithm to estimate the nodes' positions by means of DV-distance and least-squares trilateration [14]. Xiao et al. proposed a weighted DV-hop algorithm [19] that computes the average hop distance using RSSI. Zhou et al.[25] proposed an optimized method using a back propagation neural network based on DV-hop. In comparison with these enhanced DV methods for stationery WSN localization, our algorithm uses the rigidity together with distance vectors and measurements to achieve high accuracy. The accurate distance measurement can be obtained by the recent novel methods[9][12].

Recently, researchers have proposed new localization methods for other different challenging scenarios. For instance, Liu et al. proposed a localization algorithm for sparse 3D Sensor Networks [7]. This algorithm uses the common nodes to obtain higher localization accuracy than before. Zhang et al. proposed a localization algorithm for anisotropic wireless sensor networks [24]. Yan et al. proposed a novel asynchronous localization method for underwater sensor networks [20]. These methods can be adopted in our further study of 3D mobile heterogenous network localization.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a distributed localization protocol, GROLO for mobile IoTs with modest resources. GROLO does not depend on the inaccurate moving parts and motion sensors of nodes. For the sake of efficiency, we do not use universal rigidity in the localization of GROLO. Instead, we prove the minimal necessary conditions that ensure not only the position uniqueness but also the position calculability. The experimental results show that our proposed protocol is both accurate and efficient for real-world applications. Future research will consider three-dimensional localization for underwater sensing and multi-storey structure monitoring applications.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (NSFC) (Grant No. 61672552 and

U1611461), National Key R&D Program of China - 2018 YFB1004801, and Shenzhen Basic Research Funding Scheme JCYJ20170818104222072.

REFERENCES

[1] M. Bertanha and R. W. Pazzi. Jlpr: Joint range-based localization using trilateration and packet routing in wireless sensor networks with mobile sinks. In *Computers and Communications (ISCC), 2017 IEEE Symposium on*, pages 645–650. IEEE, 2017.

[2] P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 46–54. ACM, 2004.

[3] C. Godsil and G. Royle. Strongly regular graphs. In *Algebraic graph theory*, pages 217–247. Springer, 2001.

[4] A. A. Kannan, G. Mao, and B. Vucetic. Simulated annealing based localization in wireless sensor network. In *The IEEE Conference on Local Computer Networks Anniversary*, pages 513–514, 2005.

[5] C. Kuo, T. Chen, and S. Syu. Robust mechanism of trap coverage and target tracking in mobile sensor networks. *IEEE Internet of Things Journal*, 5(4):3019–3030, Aug 2018.

[6] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4(4):331–340, Oct 1970.

[7] X. Liu, Q. Yang, J. Luo, B. Ding, and S. Zhang. An energy-aware offloading framework for edge-augmented mobile rfid systems. *IEEE Internet of Things Journal*, 2018.

[8] G. Mao, B. Fidan, and B. D. O. Anderson. Wireless sensor network localization techniques. *Computer Networks*, 51(10):2529–2553, 2007.

[9] S. G. Nagarajan, P. Zhang, and I. Nevat. Geo-spatial location estimation for internet of things (iot) networks with one-way time-of-arrival via stochastic censoring. *IEEE Internet of Things Journal*, 4(1):205–214, Feb 2017.

[10] D. Niculescu and B. Nath. Dv based positioning in ad hoc networks. *Telecommunication Systems*, 22(1-4):267–280, 2003.

[11] P. Robuffo Giordano, A. Franchi, C. Secchi, and H. H. Bülthoff. A passivity-based decentralized strategy for generalized connectivity maintenance. *The International Journal of Robotics Research*, 32(3):299–323, 2013.

[12] S. Sadowski and P. Spachos. Rssi-based indoor localization with the internet of things. *IEEE Access*, 6:30149–30161, 2018.

[13] C. Savarese, J. M. Rabaey, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *General Track of the Conference on Usenix Technical Conference*, pages 317–327, 2002.

[14] A. Savvides, H. Park, and M. B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 112–121. ACM, 2002.

[15] Y. Shang, W. Rumi, Y. Zhang, and M. Fromherz. Localization from connectivity in sensor networks. *IEEE Transactions on parallel and distributed systems*, 15(11):961–974, 2004.

[16] Y. Shang and W. Ruml. Improved mds-based localization. In *Joint Conference of the IEEE Computer and Communications Societies*, pages 2640–2651 vol.4, 2004.

[17] H. Wu, A. Ding, W. Liu, L. Li, and Z. Yang. Triangle extension: Efficient localizability detection in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 16(11):7419–7431, 2017.

[18] J. Wu, L. Jiao, and R. Ding. Average time synchronization in wireless sensor networks by pairwise messages. *Computer Communications*, 35(2):221–233, 2012.

[19] H. Xiao, H. Zhang, Z. Wang, and T. A. Gulliver. An rssi based dv-hop algorithm for wireless sensor networks. In *Communications, Computers and Signal Processing (PACRIM), 2017 IEEE Pacific Rim Conference on*, pages 1–6. IEEE, 2017.

[20] J. Yan, X. Zhang, X. Luo, Y. Wang, C. Chen, and X. Guan. Asynchronous localization with mobility prediction for underwater acoustic sensor networks. *IEEE Transactions on Vehicular Technology*, 67(3):2543–2556, March 2018.

[21] Z. Yang and Y. Liu. Understanding node localizability of wireless ad hoc and sensor networks. *IEEE Transactions on Mobile Computing*, 11(8):1249–1260, 2012.

[22] Z. Yang, Y. Liu, and X.-Y. Li. Beyond trilateration: On the localizability of wireless ad hoc networks. *IEEE/ACM Transactions on Networking (ToN)*, 18(6):1806–1814, 2010.

[23] D. Zelazo, A. Franchi, F. Allgöwer, H. H. Bülthoff, and P. R. Giordano. Rigidity maintenance control for multi-robot systems. In *Robotics: Science and Systems*, pages 473–480, 2012.

[24] S. Zhang, X. Liu, J. Wang, J. Cao, and G. Min. Accurate range-free localization for anisotropic wireless sensor networks. *TOSN*, 11(3):51:1–51:28, 2015.

[25] C. Zhou, L. Wang, and Z. Lu. The study of wsn node localization method based on back propagation neural network. In *International Conference on Applications and Techniques in Cyber Security and Intelligence*, pages 458–466, 2018.

[26] Z. Zhu, A. M.-C. So, and Y. Ye. Universal rigidity: Towards accurate and efficient localization of wireless networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.