The following publication H. Wu, Z. Yang, J. Cao and L. Lai, "TRiForm: Formation Control for Underwater Sensor Networks With Measurement Errors," in IEEE Transactions on Vehicular Technology, vol. 69, no. 7, pp. 7679-7691, July 2020 is available at https://doi.org/10.1109/TVT.2020.2989363.

TRiForm: Formation Control for Underwater Sensor Networks with Measurement Errors

Hejun Wu, *Member, IEEE,* Zhongheng Yang, Liqian Lai, and Jiannong Cao, *Fellow, IEEE*

***Abstract*—The formation control of mobile underwater wireless sensor networks (MUWSNs) is difficult due to the severe errors in distance and motion measurements. To address this problem, we propose a new scheme, TRiForm, for the distributed formation control of an MUWSN. TRiForm constructs a rigid graph virtual structure from triangles to improve the reliability and efficiency. TRiForm effectively utilizes anchor information to detect and compensate for measurement errors. We have proven the correctness of the rigidity construction by TRiForm. We have also performed extensive simulations to evaluate TRiForm in various application scenarios using the measured parameters from real underwater nodes. The results show that TRiForm can successfully maintain the formations of an MUWSN and control it to arrive at the destination under distance and motion measurement errors.**

***Index Terms*—Underwater Wireless Sensor Networks, Mobile Sensor Node, Formation Control, Localization, Graph Rigidity.**

I. Introduction

Mobile underwater wireless sensor networks (MUWSNs) are desired for the detection of dynamic events in aquaculture assistance, fault-movement monitoring, surveillance, pollution tracking, etc. MUWSNs are supposed to be capable of covering a large underwater space to track dynamics via patrolling [1], [22], [32]. Fig. 1 shows such an MUWSN. In this MUWSN, a mobile node is composed of underwater propellers, an underwater transducer and underwater sensing devices [28].

A formation control scheme is needed to organize and control a group of mobile nodes or robots to move as a whole entity. This scheme usually maintains a certain relation among the robots such as a virtual structure. Similar to a herd of animals, a team of connected underwater mobile nodes or robots is more robust to exceptions than a single node [33]. There have been many formation control schemes for unmanned aerial vehicle (UAVs) and terrestrial robot clusters [5], [13], [16], [33]. Those previous studies provide deep insights and helpful experience for the development of formation control schemes of various robots.

The major challenge of designing formation control schemes for MUWSNs is introduced by the distance and motion measurement errors. These errors are rooted in the nature of underwater environments: underwater distance measurements are not deterministic. The distance between two underwater nodes is usually measured using the time of arrival (TOA) of acoustic communication. The uncertainties come from the variable acoustic wave propagation speed at different depth levels and varying water qualities. Due to unknown water currents and turbulence, only using the direction, inclination and revolving speed to calculate the velocity of a mobile node is extremely inaccurate.

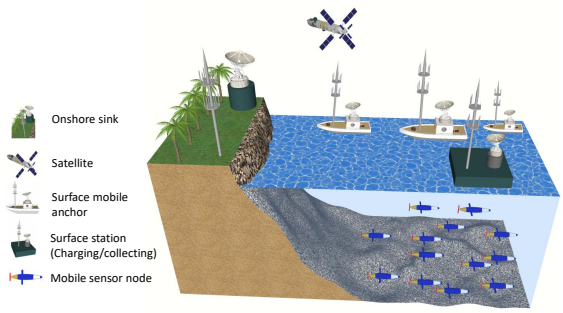

Fig. 1. Architecture of an MUWSN

In this paper, we propose a distributed scheme for MUWSN formation control: TRiForm. TRiForm employs graph rigidity theory to construct a rigid graph virtual structure. The rigid graph is composed of a number of triangle sub-network topologies of three mobile nodes for efficiency. Furthermore, TRiForm uses information about the anchors for cross-validation to minimize the effects of the errors. Simulations demonstrate that TRiForm achieves accurate motion control under realistic conditions with significant distance and motion measurement errors.

TRiForm makes the following contributions in the formation control study of MUWSN: (1) A simple triangle merging approach is proposed for fast rigid graph construction and maintenance. (2) A theorem is proposed and proved for the mobile nodes to ensure their convergence to the destinations. (3) A formation control scheme is proposed to work under significant motion and distance measurement errors. This is the first MUWSN formation control scheme that works under both the distance and motion measurement errors.

The remainder of this paper is organized as follows. Section 2 provides a brief review of the related schemes and systems. The algorithm details of TRiForm and the theorems are presented in Section 3. In Section 4, the simulation results from different formations of mobile nodes are reported. In addition,

Hejun Wu is the corresponding author: wuhejun@mail.sysu.edu.cn. Hejun Wu and Zhongheng Yang (yangzh@mail2.sysu.edu.cn) are from the Department of Computer Science, Sun Yat-sen University, Guangdong, China

Liqian Lai (201701041@jyu.edu.cn) is with the School of Computer Science, Jiaying University, Guangdong, China

Jiannong Cao (csjcao@comp.polyu.edu.hk) is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, China

Manuscript received June 25, 2019.

the components of a mobile node are separately tested to validate the simulations. Finally, the concluding remarks and future directions for improvement are summarized in Section 5.

II. Related Work

A distributed formation control scheme is more suitable for wireless networks because it requires much fewer message exchanges between the mobile nodes than centralized schemes. This distributed control is especially advantageous in underwater environments because the data communication delay is large [2], [13]. Hence, in the following, we mainly review the distributed formation schemes.

According to the formation pattern, there are two types of formations: rigid and flexible. A rigid formation control maintains a set of constant distances and angles between certain pairs of mobile nodes to ensure the graph rigidity of the formation [11]. In contrast, a flexible formation only defines the minimal and maximal ranges of distances between mobile nodes. The flexible formation control changes the distances freely as long as node collisions do not occur [16].

Specifically, the flexible formation control schemes do not maintain a fixed topology of a mobile network. Most flexible formation control schemes do not maintain the constant distances between the nodes that are connected via wireless links. A mobile node should know the starting times and trajectories of other mobile nodes in advance so that they can move together [9], [16], [20]. Usually, a virtual leader trajectory is defined as the predefined path for other nodes [19]. The advantage of a flexible control method is that it is relatively easy for control because nodes can move within much wider ranges of distances and angles. The major disadvantage is that most flexible formation control schemes are not adaptive to emergencies such as obstacles, intruders and lost nodes. Furthermore, a flexible formation scheme is complex due to the uncertain locations of agents.

In contrast, the rigid formation control schemes maintain fixed distances between certain specified nodes. Usually, a rigid formation control scheme applies the graph rigidity theory [14] to the network topology graph of a formation. Subsequently, the rigidity eigenvalue and the rigidity matrix can be calculated [33]. Distances and orientation angles are used to maintain the rigid formations [7], [29]. Infinitesimal, universal, or global rigidity theory are also employed in rigid formation control studies. However, strict rigidity causes high computation and communication traffic overhead.

In the process of formation control, the organization of a team of mobile nodes is performed by the following three types of mechanisms: leader-follower [10], [23], behavioral team [25], and virtual structure [3], [27]. A leader-follower organization mechanism uses a leader to control the entire group of nodes. A behavioral team mechanism partitions a network of mobile nodes based on the behaviors of the nodes. Nodes that are performing the same task will be organized into a single team. A virtual structure mechanism models the mobile nodes as particles embedded in a physical structure. The virtual physical structure is usually rigid, and a bar in the structure does not change its appearance or length.

In general, a formation control scheme employs a combination of the organization mechanisms to manage a team of mobile nodes. For instance, rigid formation control schemes maintain the formation rigidity using behavioral and virtual structure mechanisms [26], [33]. Flexible formation control schemes often use leader-follower and behavioral organization methods to keep a group of mobile nodes synchronized and to avoid the loss of nodes [19].

Special motion control methods have been proposed to address realistic problems such as obstacles, uncertain hydrodynamics, and long delays in underwater environments. For instance, underwater motion control scheme can partition the underwater areas and dynamically localize nodes to bypass dangerous areas [15]. Feedback and feedforward controllers have been combined to address long delays and severe packet dropout in underwater communication [21]. Using a distributed continuous control, a group of AUVs can achieve time-varying tracking control with uncertain hydrodynamics [12].

The autonomous ocean sampling network II (AOSN-II) was an earlier effort for realistic underwater studies with mobile underwater nodes [8]. The node on an AUV acts as a mobile sink with a powerful transceiver to collect sensory data from stationery sensor nodes, which are not mobile [6], [28]. The development of a man-portable hybrid autonomous underwater vehicle marks a recent attempt to make underwater experiments common for normal labs [4]. A vehicle prototype has also been proposed that uses acoustic signals to control an underwater biometric fish robot [24].

Few formation control schemes have been proposed specifically for MUWSNs. Existing control schemes for MUWSNs mostly target special motions. In addition, the errors in either the movement or distance measurements are not considered in most formation control schemes for MUWSNs. Several schemes have been proposed to improve the distance measurement accuracy by increasing the time synchronization accuracy. Nonetheless, due to the uneven signal propagation speed in underwater environments, accurate time synchronization alone is not sufficient to achieve the desired distance measurement accuracy.

In summary, state-of-the-art formation control techniques have been enriched with various mechanisms to organize UAVs and terrestrial robots. Due to the complex underwater environment conditions, most formation control schemes for planes and robots are not applicable to underwater mobile nodes. The key difference is that a mobile underwater node cannot simply use the rotation speed of the propeller to estimate the speed of the mobile node. The position information is also unavailable to the underwater mobile nodes. In contrast, UAVs and terrestrial robots can often use driver parameters and GPS devices to calculate their motion and positions with relatively high accuracy.

III. Graph Rigidity and Formation Control Problem

Before going into the details of the formation control process, we present the graph rigidity concepts and the problem formulation.

A. Graph Rigidity

Given a graph, $G = \{V, E, W\}$, where V is the vertex set, E is the edge set, and W is the weight of each edge, in the two-dimensional Euclidean space ($\mathbb{E}^2$) defined as follows: $V = \{A, B, C, D\}$; $E = \{(AB), (BC), (CD), (DA)\}$; and $W = \{w(AB) = w(CD) = 8,\ w(BC) = w(DA) = 10\}$. Fig. 2 shows three different shapes of G. Actually, G has an infinite number of shapes. Each shape of a graph is called a *framework*. A graph that has an infinite number of frameworks is called a *flexible* graph [17]. Hence, G is a flexible graph.

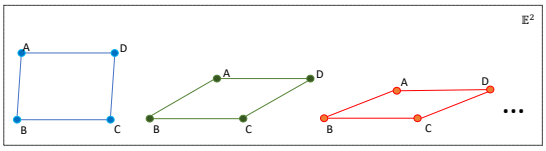

Fig. 2. A flexible graph has an infinite number of shapes in $\mathbb{E}^2$

Formation control algorithms face difficulties maintaining the network connection with the flexible graph shown in Fig.2. Since the distance between B and D and the distance between A and C are changeable, the network topology varies continuously. The link between B and D and the link between A and C break or recover frequently. Consequently, it is more costly to maintain the network topology. Furthermore, nodes are readily lost, as there are usually less than three stable links for each node.

In a given Euclidean space, a rigid graph has only a finite number of frameworks [17]. Fig.3 shows an example rigid graph in $\mathbb{E}^2$. This graph has only two frameworks in $\mathbb{E}^2$. Note that rotating an entire framework on a straight line or a point does not change the framework in $\mathbb{E}^2$, as long as the rotated result remains in the same $\mathbb{E}^2$ space. Hence, graph rigidity is an important property for mobile nodes in limiting network topology and maintaining network connectivity.

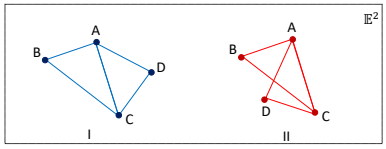

Fig. 3. Shapes of the same graph in $\mathbb{E}^2$

As shown in Fig.3, the shape of a general rigid graph is not uniquely determined. This is acceptable in most formation control scenarios since the graph rigidity itself ensures the limited distance candidate values among the mobile nodes. Framework changing, e.g., from I to II in Fig.3, still maintains the rigidity of the graph. In certain application scenarios, global rigidity is required. Fig.4 shows an example globally rigid graph in $\mathbb{E}^2$. Such a globally rigid graph has a unique framework; it does not allow D to move from the right side to the left side of the line AC. Hence, in many network localization schemes, a globally rigid graph is often used to ensure the unique position candidate for each node [30].

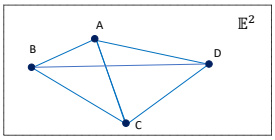

Fig. 4. A globally rigid graph in $\mathbb{E}^2$

B. Rigid Formation Control Problem for TRiForm

The rigid formation control process of TRiForm on a MUWSN is shown in Fig. 5. After network deployment, the rigid graph virtual structure of the MUWSN topology is constructed. Then the MUWSN starts to move. Since the MUWSN should periodically adjust the positions of each node to avoid collisions and lost nodes, the nodes are controlled to move and pause in cycles of fixed time period (epoch). On each node, the distance to adjust is calculated according to the rigidity and the distance measurement errors detected. This way, the graph rigidity is maintained after each epoch motion of the MUWSN.

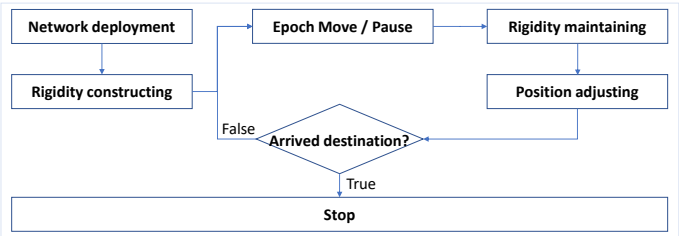

Fig. 5. The TRiForm rigid formation control process

The above design facilitates node motion control. Maintaining a rigid graph is relatively simple in that it avoids the frequent recalculation of appropriate distances between nodes. Accordingly, the problems facing the TRiForm formation control process are formulated as follows.

Graph Model: At time t, an MUWSN is given by a graph of $G(t) = \{V, E(t), W(t)\}$, where V is the set of mobile nodes. $E(t)$ is the set of the edges between two mobile nodes with a distance measured or calculated at time t, and $W(t)$ is the set of the lengths of each edge in $E(t)$.

Constraints:

1) Three dimensions: The MUWSNs are deployed in $\mathbb{E}^3$. Normally, the depths of the mobile nodes are different. The depth of each node should be kept at a specified level.
2) Passive motion: The MUWSN might be driven away from its original position when the nodes remain still.
3) Measurement errors: The speed, direction, inclination, and distance measurements are inaccurate. Using speed, direction and inclination to localize a mobile node is infeasible; the accumulated errors overwhelm the position estimation.
4) Multi-hop communication: There are quite a few nodes in an MUWSN that cannot communicate directly. Some nodes must rely on message forwarding via multi-hop communication to receive messages from the leader or other non-neighboring nodes. Two nodes are neighbors,

which means that they can communicate directly with each other.

Assumptions:

- Predefined leader: A node in the MUWSN is specified as the leader and is chosen as the node that is the nearest to the geometric center of the MUWSN topology. Having the leader node near the center is for ensuring communication efficiency since the leader node needs to coordinate and synchronize the other mobile nodes in the MUWSN.
- Time synchronization: Before performing formation controlling, the MUWSN nodes are assumed to be time-synchronized. The time synchronization of the MUWSNs is relatively simple in real MUWSNs. The mobile nodes move to areas that are extremely close to the specified leader. The leader then broadcasts a time message so that the other nodes can synchronize themselves with the leader. In this paper, we assume that the mobile sensor nodes have already been synchronized.
- Distance measurement: In this paper, we assume that the line-of-sight (LOS) distance between two neighboring nodes has been measured using the time of arrival (TOA). In the simulations, the errors are simulated and appended to the distance measurement.
- Accurate depth: We assume that the depth of each node can be accurately measured by the depth sensor on the node.In effect, the accuracy level of the current depth sensors is high enough.
- Surface anchors: The anchor nodes are specified as three nodes on the surface. They are also mobile, and the other nodes move together with these anchors. The anchors can obtain GPS signals and communicate with both underwater mobile nodes and the other surface anchors or sinks. Therefore, the positions of the anchor nodes are continuously updated.

Problem 1 (Rigidity construction): Find the distributed rigid graph constructer for N number of mobile nodes to collectively construct a virtual structure of rigid graph $G(t)$ at time t.

Problem 2 (Motion control): Find the local controller that moves the mobile nodes while preserving the virtual structure of the rigid graph.

Problem 3 (Error compensation): Find the adjuster that verifies and compensates the distance measurement errors between two nodes.

IV. TRiForm Formation Control

The three problems in the formation control processes of TRiForm are challenging due to the following facts: (1) It is difficult to use modest underwater communication to coordinate a group of mobile nodes. Underwater communication is notorious for its low transmission rate and high bit error rate. This requires TRiForm to be simple but effective. (2) The limitations of underwater communication also require a relatively stable structure so that the computation and communication costs for distance coordination can be reduced. However, this is difficult for MUWSNs undergoing passive or active motions. (3) Finally, large measurement errors, especially distance errors, make it difficult to maintain a stable virtual structure to avoid node loss and network disconnection.

A. Triangle Projection and Merging

To reduce the problem from three dimensions to two dimensions, TRiForm performs triangle projection and rigid graph construction. The two-dimensional plane is on the water surface, which is called the *reference plane*. Initially, all the mobile nodes of the MUWSN are deployed on the water surface. Their connections form triangles, and the triangles are projected onto the reference plane. To avoid collisions, two nodes cannot be deployed on the same point. Furthermore, to construct a rigid graph, three nodes cannot be on the same line. As a result, the connecting straight line of any three nodes can be used to construct a virtual triangle on the two-dimensional reference plane.

In the construction of a rigid graph, two triangles that share the same edge can be merged together. Two edges with different starting points and/or ending points, even the points on the same line, are not called the same edge. The merged triangles are called *neighboring triangles*. Fig. 6 shows the merging result of two triangles ΔABC and ΔABD. They share the same edge of AB. We use our previous method to localize the sensor nodes in the graph on the reference plane [31].

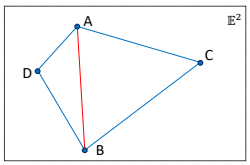

Fig. 6. Merging of two triangles

After the rigid graph virtual structure is constructed, the nodes dive vertically to the required depth levels. The nodes can reconstruct virtual triangles connecting themselves given that the new distances formed by the three edges in a triangle have been measured. The neighboring triangles on this reference plane are projected onto this reference plane.

The triangles can be projected onto the reference plane by projecting the three edges. The projection of the node position is not feasible since the node positions are not yet available. The new node position cannot be calculated simply using the motor information of the propeller due to the large accumulated motion measurement errors. The projection also suffers from distance measurement errors. Nevertheless, the distance measurement errors are not accumulated because they are measured and updated directly after the nodes move. In addition, TRiForm is able to correct parts of the distance measurement errors. The methods for correcting the errors will be discussed later in this paper.

The edge projection is more effective because the depth of each node can be accurately measured. With the distance measured and the depth of the two nodes, their edge projection can be simply calculated using the Pythagorean theorem. The projection of the neighboring triangles can be obtained by

calculating the projection of each edge of the neighboring triangles on the reference plane. The projected neighboring triangles can be further merged to a projected graph on the reference plane. This projected graph is a rigid graph, as specified in Theorem 1:

Theorem 1. *The resulting graph from merging the projection of the neighboring triangles is rigid in* $\mathbb{E}^2$.

Proof. The projection of ΔABC onto the reference plane is a triangle in $\mathbb{E}^2$. We call this projected graph (triangle) G. A triangle in $\mathbb{E}^2$ is a rigid graph [18]. Therefore, G is rigid.

Suppose that a new triangle ΔABD is merged into ΔABC, as they share the edge AB. The vertices A and B have three neighbors. Now, the two neighboring triangles include vertices A, B, C, D. They form a new graph G', and the projection of this graph also includes two triangles. Since G (ΔABC) is rigid, $|E(G)| = 2|V(G)| - 3$, where $|E(G)|$ is the number of edges and $|V(G)|$ is the number of vertices. Apparently, $|V(G')| = |V(G)| + 1$ and $|E(G')| = |E(G)| + 2$ (G' has two more edges than G, which are edges AD and BD). As a result, $|E(G')| = 2|V(G)| - 3 + 2 = 2|V(G')| - 2 - 1$. Finally, $|E(G')| = 2|V(G')| - 3$. According to Laman's Lemma [18], G' is rigid.

Similarly, when the above neighboring triangles merge with a new triangle, the newly merged graph is also rigid in $\mathbb{E}^2$. The deduction is similar to the above two triangles. □

Utilizing Theorem 1, the mobile nodes can merge the underwater triangles to construct and project a virtual rigid graph on the reference plane. The neighboring triangles are projected onto the reference plane to obtain a rigid graph. Then, they can start moving forward according to the predefined route of the team. Fig. 7 shows an example of the projection of 12 neighboring triangles onto the reference plane to obtain a rigid graph in $\mathbb{E}^2$.

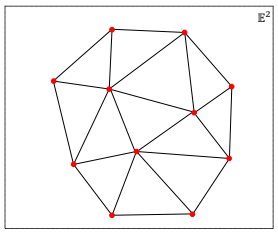

Fig. 7. An $\mathbb{E}^2$ rigid graph obtained by the projection of neighboring triangles

B. Virtual Structure Construction and Maintenance

In the construction of the rigid graph virtual structure, each node only maintains a subgraph of the entire graph of the network in TRiForm. Projecting the subgraphs of each node onto the reference plane and then merging the projected subgraphs produces a rigid graph on the reference plane. The next problem is to maintain this virtual structure.

Before starting the motion control, a leader is chosen in the initial deployment. The hop count of a leader node is 0. The other nodes follow the leader while maintaining their distances with their neighbors in the triangles. The leader and follower nodes move and stop periodically. Each moving and stopping period is called an epoch. The epoch length is not fixed, as the time needed for position and distance adjustment is not predictable. The leader determines when to start, adjust, and stop in an epoch according to the movement and adjustment progress of the follower nodes.

The specific operations of the virtual structure construction and maintenance in TRiForm are presented in Algorithm 1. After the network deployment on the water surface, the nodes first measure and broadcast their distances to other nodes. For example, n_i measures the distance between itself and the two neighbors n_1 and n_2. With these distances, n_i is able to build a triangle. Fig. 8(I) shows the triangle composed of n_i, n_1 and n_2.

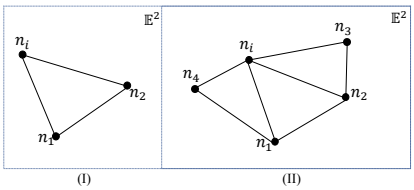

Fig. 8. Initial triangle (I) and the merged triangles (II) that all share node n_i

Algorithm 1 TRiForm: Rigid Graph Construction and Maintenance

```
Initialization() // deploy and construct triangle(s).
anchors ← anchor_table[], leader ← ChooseLeader()
n_i ← this node id
while state! = rigid do
    Measure and broadcast distances: d(n_i, n_1), d(n_i, n_2)
    Construct triangle T {(n_i, n_1), d(n_i, n_2), d(n_1, n_2)}
    G' ← G' + Projection(T) // project T onto the reference plane and the subgraph G' merges with the projection
    if G'! = NULL and timeout then
        state ← rigid
if this is a leader then
    hop = 0, epoch = 0
else
    hop = min(neighbor hops) + 1
while not getting to the destination do
    Dive to the required depth
    Steer one step toward its destination
    if n_i ∈ anchors then
        CheckLocation()
        PositionAdjust()
        call DistAdjust()
    else
        Broadcast moving_done
        call DistAdjust()
    if this is a leader then
        epoch ← epoch + 1
        Broadcast moving vector (step and epoch)
end
```

Suppose that the other neighbors n_1 and n_3 as well as the neighbors n_1 and n_4 can also construct triangles with n_i.

The three triangles are projected onto the reference plane and merged to the rigid sub-graph G' of n_i. These operations are shown in Line 6 and Line 7 of Algorithm 1. The resulting graph G' is illustrated in Fig. 8(II). According to Theorem 1, the subgraph G' is a rigid graph. In addition, the subgraphs of all the nodes can be merged into a larger rigid graph, as there are neighboring triangles among them.

With the rigid graph constructed, the nodes dive to the specified depth and start moving toward the destination (Line 14 - Line 28). After each moving operation in an epoch, the anchors first adjust their locations using the function CheckLocation() of Line 18 in Algorithm 1. If the location of an anchor deviates from its path, the anchors perform their position adjustment (Line 19).

The correct locations of the anchors are calculated using the paths toward the destination and the predefined step vector of each moving epoch. Although the motion measurement is not accurate, the anchors can manage to adjust to the correct position via tiny motion steps.

The follower nodes move one epoch along the same direction as the leader (Line 16) since the networked nodes are traveling to the same destination. However, due to the errors in the motion, they may also deviate from the desired location. The follower nodes need to perform the distance adjustment (Line 26). The detailed process of the distance adjustment is illustrated in the next subsection.

C. Distance Measurement Error and Compensation

We now explore the distance measurement errors. The distance measurement errors represent a great challenge to most motion control and localization schemes. Since the inter-node distance is the basis of most formation control and range-based localization schemes, quite a few studies assume that the distance measurement is accurate. There are also a number of joint localization and time synchronization studies that attempt to minimize the time and distance errors.

In effect, the time synchronization errors are actually negligible in MUWSNs because the mobile nodes can gather to be close enough to simultaneously receive accurate time messages. In addition, our experience with real-world underwater sensor nodes indicates that the distance errors have local characteristics, that is, distance errors measured by different nodes within a limited space are similar.

Therefore, utilizing the local property of distance errors, we design two cross-validation methods to compensate for the distance measurement errors. The first method is a simple method called anchor-distance validation. This method compares the calculated distances with the measured underwater distances between the three anchors. The calculated distances are assumed to be the ground truth. The errors in the underwater measured distances can be estimated by Eq. (1). Since distance errors in a local space are similar, this equation simply uses the average error of the anchor distance measurements to estimate the distance error of the whole network.

$$\epsilon_d = \frac{1}{3}((d_{n_1,n_2} - d'_{n_1,n_2}) + (d_{n_1,n_r} - d'_{n_1,n_r}) + (d_{n_2,n_r} - d'_{n_2,n_r})) \quad (1)$$

The second method uses localization to verify the distances. Fig. 9 shows an example MUWSN with distance errors. In this MUWSN, n_p, n_q, and n_r are the mobile anchors. The black dots denote the actual positions of the mobile nodes in the MUWSN. The white dots denote the estimated positions of the mobile nodes using the measured distances with errors. The localization details are omitted in this paper. We used our previous methods [31] to localize the nodes when they are in the pause phase (i.e., not moving). The solid line and the black dots form the correct rigid graph of the MUWSN. The inaccurate graph is composed of white dots and dashed lines. There are redundant dashed lines in the rigid graph and are recorded during the communication of the nodes. These lines may be used later in estimating the distance errors.

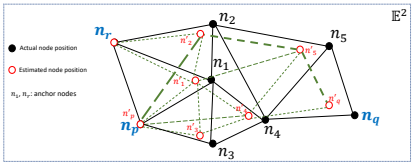

Fig. 9. Distance error estimation using position estimations

In the MUWSN of Fig. 9, the localization process starts from the two anchor nodes (n_r and n_p) using their known positions. Then, the localization proceeds toward the border of the network. Finally, the other anchor node n_q is localized. The estimated position and the actual position of n_q are different due to the accumulated distance measurement error. This distance error is accumulated from the distance errors of three triangles in this example. The average distance error is then used as the per-hop distance error in this MUWSN. Eq. (2) shows a naive calculation method of the average distance error using the hop h and the distance error from n_q to n'_q.

$$\epsilon_{d1} = \frac{1}{h}\epsilon_d = \frac{1}{h}\sqrt{(n'_q.x - n_q.x)^2 + (n'_q.y - n_q.y)^2} \quad (2)$$

Eq. (2) is a rough estimation of the distance error between a pair of nodes. We can further elaborate upon the distance error by using geometry constraints. In Fig. 9, the dashed lines from n'_p to n'_2, n'_2 to n'_5, and then n'_5 to n'_q constitute a path from n'_p to n'_q. Because the straight line from n'_p to n'_q is known, the angles from the three dashed lines to the straight line $n'_p n'_q$ can be calculated. Then, the contribution of the error of each dashed line is estimated according to the distance and the angle to $n'_p n'_q$ of the dashed line.

The error estimation process using $\boldsymbol{n'_p n'_q}$ and $\boldsymbol{n'_p n'_2}$ in Fig. 9 is illustrated as follows. The anchor node n'_p is defined as the start anchor, and n'_q is defined as the end anchor. One path, from n'_p to n'_q, is chosen, for instance, the path from n'_p, n'_2, n'_5 to n'_q. Given the estimated coordinates of n'_q and n'_2 (n'_p and n'_r are the same as n_p and n_r, respectively), the equations of the two lines of $\boldsymbol{n'_p n'_q}$ and $\boldsymbol{n'_p n'_2}$ are shown in Eq. (3). In

Eq. (3), $\omega_{11} = n'_q.y - n'_p.y$, $\omega_{12} = n'_q.x - n'_p.x$. Similarly, ω_{21} and ω_{22} can be obtained using the coordinates of n_2 and n_p.

$$\omega_{11}x + \omega_{12}y + \omega_{13} = 0$$
$$\omega_{21}x + \omega_{22}y + \omega_{23} = 0 \quad (3)$$

Next, given the cosine of the angle α between $\boldsymbol{n'_p n'_q}$ and $\boldsymbol{n'_p n'_2}$, the length of the projection of line $\boldsymbol{n'_p n'_2}$ onto $\boldsymbol{n'_p n'_q}$ is calculated via Eq. (4). $cos\alpha$ can be obtained in Eq. (5).

$$|\boldsymbol{Proj}_{\boldsymbol{n'_p n'_2}}| = |\boldsymbol{n'_p n'_2}| cos\alpha \quad (4)$$

$$cos\alpha = \frac{|\omega_{11}\omega_{21} + \omega_{12}\omega_{22}|}{\sqrt{\omega_{11}^2 + \omega_{12}^2}\sqrt{\omega_{21}^2 + \omega_{22}^2}} \quad (5)$$

The distance error of line $\boldsymbol{n'_p n'_2}$ is obtained using $\epsilon_{d1} = \epsilon_d \frac{cos\alpha|\boldsymbol{n'_p n'_2}|}{|\boldsymbol{n'_p n'_q}|}$. Similarly, the errors of $\boldsymbol{n'_2 n'_5}$ and $\boldsymbol{n'_5 n'_q}$ are calculated using their projection onto $\boldsymbol{n'_p n'_q}$. The vectors $\boldsymbol{n'_p n'_2}$, $\boldsymbol{n'_2 n'_5}$ and $\boldsymbol{n'_5 n'_q}$ constitute a path from n'_p to n'_q.

From the above process, it can be seen that the distance measurement errors on nodes that are on the path from one anchor to another anchor can be more accurately estimated in TRiForm. The path is called the compensation path.

A further location adjustment is needed for the nodes to obtain their correct position in each epoch. The adjustment is performed after the anchors have adjusted to their correct locations. As specified in the assumptions, the non-anchor nodes know the formation path and calculate their own moving vector in each epoch. With the errors in the distance measurements determined, the non-anchor nodes perform tiny moving steps to recover to these calculated correct positions.

Nevertheless, the remaining challenge for the non-anchor nodes is that the non-anchor nodes cannot move accurately. The nodes may not arrive at the correct location after several steps of moving. Therefore, the localization procedure is needed again after moving a specified number of steps. Given the distance measurement errors compensated during the localization, the error of the real position after the adjustment can be at the same level as that of a one-step motion measurement error.

Algorithm 2 presents the error checking and position adjustment procedure under both distance and motion measurement errors in TRiForm. Utilizing our previous localization method called GROLO [31], Algorithm 2 begins error checking from the node that has two anchors as its neighbors. Then, it follows the direction of the triangle extension. We still take the MUWSN shown in Fig. 9 as an example. The distance error checking is sequentially performed from n_1 to n_5, with the same sequence as the subscript of the five nodes. Finally, n_5 communicates with n_q and determines the distance measurement error using the Eq. (4) and Eq. (5).

Specifically, in Algorithm 2, Lines 1-13 show this process of error checking. Lines 15-21 show the nodes attempting to adjust from the incorrect locations to the correct locations. Line 18 calls our previous method of localization repeatedly. Nonetheless, the computational and communication load is not heavy since the nodes only perform tiny movements and since the network and rigid graph structures are not changed. The GROLO localization procedure only needs to use simple geometries to calculate the new position of each node.

Algorithm 2 DistAdjust() //Distance adjusting control

1: **if** $n_i \in anchors$ **then**
2: broadcast $\{anchor, n_i, adjusting\_done\, position\}$ to neighbors
3: **while** no $moving\_done$ from non-anchor neighbors **do**
4: wait until timeout
5: **if** this n_p has another anchor neighbor n_r **then**
6: call GROLOLocalization() // start from anchors
7: **else**
8: **while** not receiving $moving\_done$ from its triangle **do**
9: wait until timeout
10: call GROLOLocalization() // localization
11: **if** neighbor is the third anchor q **then**
12: $\epsilon_d = |\boldsymbol{n'_p n'_q}| - |\boldsymbol{n_p n_q}|$
13: obtain ϵ_{d1} from ϵ_d
14: broadcast the distance measurement error
15: **while** adjusting not done **do**
16: **if** received ϵ_{d1} from a node on its path **then**
17: adjust measured distance
18: call GROLOLocalization()
19: **if** $n_i \notin anchors$ **then**
20: perform tiny moving step toward the correct position
21: broadcast $adjusting\_done$ message to neighbors
22: end

V. Evaluation

To evaluate TRiForm, we performed simulations with distance and velocity measurement errors. To simulate the errors, we first measured the ranges of the real device errors using a wired remote-control underwater vehicle (RUV) and two underwater ultrasonic transducers, as shown in Fig. 10. We could control the RUV to dive to a specified depth, turn to certain directions, and move forward. The motion control snapshot is also shown in Fig. 10. We controlled the RUV to the desired speed (the maximum speed), direction, and depth 30 times each. The ranges of the device errors are listed in Table I. The two ultrasonic transducers were deployed in a swimming pool, which is clear enough to allow the use of a ruler and a timer to determine the distance and speed errors.

We used rulers and timers to record the moving distance and time interval of the RUV to evaluate its motion measurement device errors. We also verified the range of the distance measurement errors of the two transducers. The results are presented in Table I.

The formation control was evaluated using simulation. The simulation is able to apply controlled experimental conditions, whereas the real-world environment cannot be controlled. In the simulations, the mobile nodes were controlled to move from the source to the destination following certain shapes of paths.

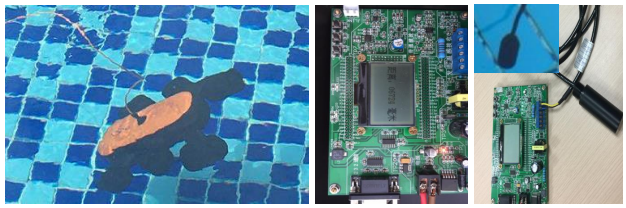

Fig. 10. The RUV and the underwater ultrasound transducer

TABLE I
RUV MOTION AND DISTANCE MEASUREMENT ERRORS

	Speed(m/s)	Direction (°)	Depth(m)	Distance (m)
Req	2.01(3.9knot)	45	1	10
Range	1.61-1.94	35 - 55	0.96-1.03	9-9.8
Error	-20%	±10%	-10%	-10%

We added random errors to the velocity of each mobile node in the simulations. The added moving errors conform to the experimental results in Table I. The distance measurement errors are also simulated in the distance measurement between two neighboring nodes in the simulations. Because TRiform depends on distances between nodes in the construction of the virtual structure of a rigid graph, distance measurement errors significantly affect the formation control results.

As the distance measurement errors have a local property, the simulations need to avoid the distance measurement errors becoming too random or too skewed at a certain level. In our simulations, the errors at each level are chosen using a Gaussian distribution. The random variable of the distance measurement error is denoted as $r_d = \epsilon_d/d$. The means and variances of the distance measurement errors are μ_{r_d} and σ_{r_d}, respectively.

We found from our experience with the RUV that the errors in the measurements of the motion are different from the distance measurement errors of each node. The errors of the motion measurement of the nodes fall into a certain range. Therefore, we use the random direction and speed errors within the measured range.

The errors in the distance and motion measurements are classified into two levels. The Gaussian distribution can simulate the distance measurement error differences of the nodes in an MUWSN. In the simulations, the parameters of the Gaussian distribution of the two levels of errors are set as $\mu 1_{r_d} = 0.1$, $\mu 2_{r_d} = 0.2$, $\sigma 1_{r_d} = 0.025$, and $\sigma 2_{r_d} = 0.05$. Similarly, the absolute error values of the motion measurements are set randomly within 0.1 and 0.2 in the first and second level, respectively. These error level parameters are listed in Table II. To ensure the repeatability of a simulation, we recorded the exact measurement error values of each mobile node and took the values as an input when we repeated the simulation.

TABLE II
ERROR LEVELS

levels	μ_{r_d}	σ_{r_d}	motion error
level1	0.1	0.025	0.1
level2	0.2	0.05	0.2

We performed a series of simulations on TRiForm with different paths under the two error levels. We also tested TRiForm with barriers. The results of the projected positions and paths on the reference plane are shown in Fig.11. Among the figures, the second 1 and 2 in the sub-figure caption denote the error level. For instance, a01-a02 are the trajectories of the network moving along a line under error levels 1 and 2. Similarly, a11-a12 are the trajectories of the network moving along the stairs under error levels 1 and 2.

The two figures of b01-b02 are the eclipse trajectories of the MUWSN moving as a whole, while b11-b12 are the eclipse trajectories of each mobile node in the MUWSN under the two distance measurement levels, respectively. The differences between b01 and b11 as well as the distance between b02 and b12 are as follows: The nodes in b11 and b12 follow the same size of eclipse. In contrast, the inner node follows the smallest eclipse trajectory, and the outer node follows the largest eclipse in b01 and b02.

Finally, c01 and c02 are the results of the MUWSN bypassing a barrier through a circular trajectory. c11 and c12 show the MUWSN bypassing a barrier by following a rectangular trajectory.

It can be seen from the figures that the mobile nodes adjust their positions periodically to avoid collisions and disconnections of the network. The formation is maintained via distance adjustment. The distance measurement errors are reduced using the validation method of Eq. (5) this paper.

We also evaluated the formation control accuracy under the two levels of device measurement errors in the above simulations. Table V shows the RMSE of the stopped position to the destination of each node. The RMSE of a single node is calculated by Eq. (6), where m is the number of simulation running times, (x_o, y_o) is the destination position and $(x_{r,i}, y_{r,i})$ is the real position of the node in the $i-th$ time running. d is the distance from the start position to the destination position. Since the results are evaluated on the reference plane, the positions are of two-dimensional coordinates.

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\frac{(x_o - x_{r,i})^2 + (y_o - y_{r,i})^2}{d^2}} \quad (6)$$

From the RMSE results, it can be seen that TRiForm is able to arrive at the positions around the specified destinations of the mobile nodes. The localization and error compensation methods are effective in maintaining rigid formation. The final arrived position is close to the destination, with low error rates (1%-18%). In comparison, the error rate is higher when the error level is high and when the node motion is complex. For instance, the formation of $b02$ requires the nodes to frequently change their directions to form the oval shape when observing the entire network path. The error rate of $b02$ is the highest on average.

We finally evaluated the energy consumption of a node in the 11-node MUWSN during the TRiForm formation control. The energy consumption parameters of the real hardware measurements are as follows: The working electric current and

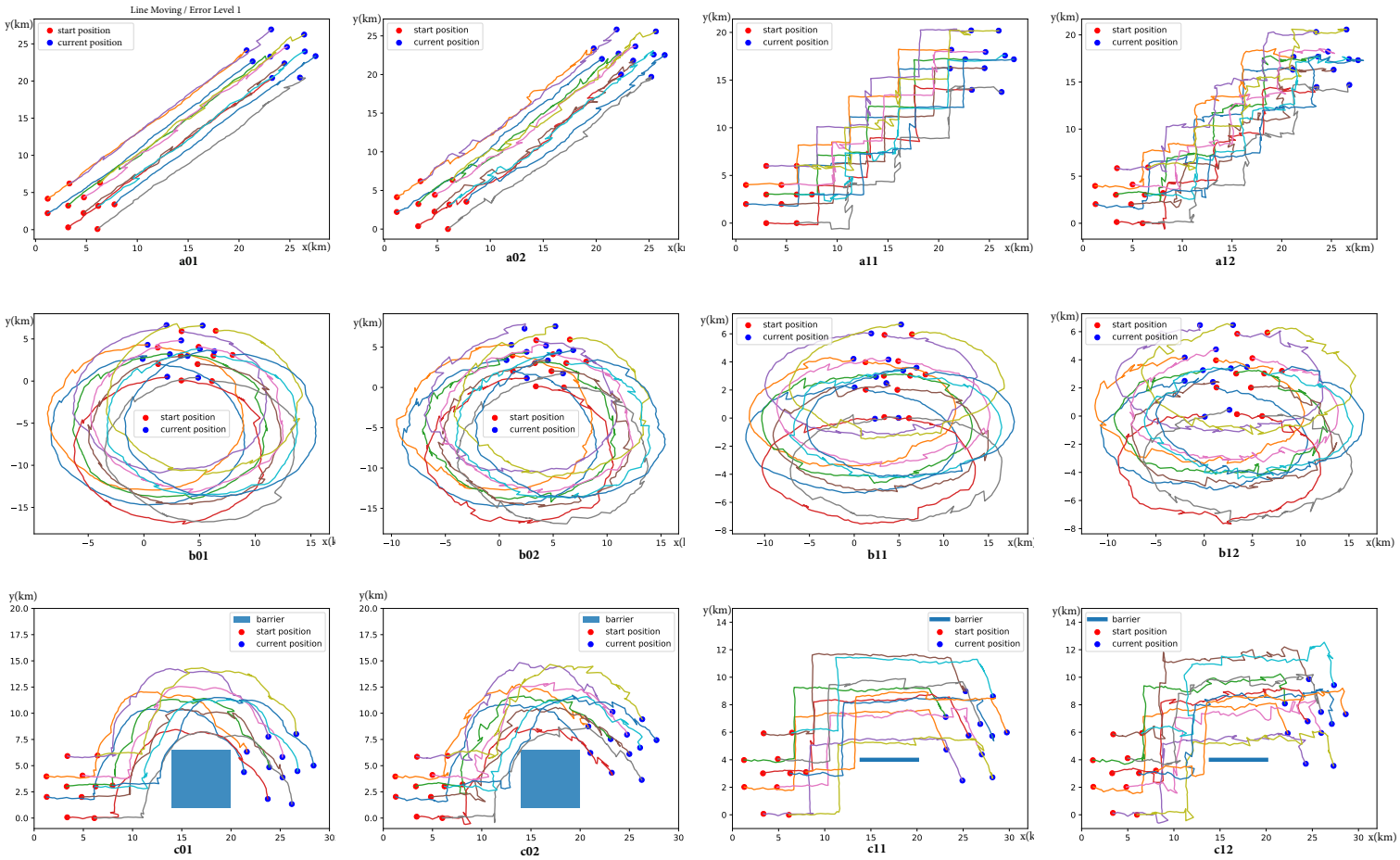

Fig. 11. The formation control of an MUWSN with different trajectories and under different conditions

TABLE III
RMSE OF THE NODE POSITION TO DESTINATIONS

Experiments	node 1	node 2	node 3	node 4	node 5	node 6	node 7	node 8	node 9	node 10	node 11
a01	0.0257	0.0100	0.0103	0.0169	0.0337	0.0131	0.0222	0.0165	0.0169	0.0385	0.0124
a02	0.0162	0.0486	0.0329	0.0229	0.0382	0.0362	0.0308	0.0300	0.0196	0.0248	0.0400
a11	0.0409	0.0513	0.0728	0.0463	0.0504	0.0428	0.0222	0.0553	0.0346	0.1347	0.0341
a12	0.0563	0.0663	0.1632	0.1188	0.0961	0.1355	0.0490	0.1832	0.1326	0.1528	0.0627
b01	0.0236	0.0550	0.0646	0.0144	0.0211	0.0903	0.0504	0.0221	0.0409	0.0437	0.0256
b02	0.1090	0.1630	0.0964	0.0807	0.0643	0.1504	0.1204	0.1660	0.0883	0.0985	0.1032
b11	0.0264	0.0323	0.0609	0.0450	0.0137	0.0692	0.0510	0.0076	0.0980	0.0848	0.1035
b12	0.0423	0.0054	0.0203	0.0347	0.0714	0.0891	0.0850	0.0632	0.0375	0.0420	0.0423

voltage of the mainboard are 0.1 A (transmission / receiving) and 24 V, respectively. The sleeping electric current of the mainboard is 0.01 A. The moving electric current and voltage of the RUV are 2 A and 12 V, respectively. The idling electric current of the RUV is 0.2 A. The energy consumption can be calculated by multiplying the voltage, current and average time intervals of a node in the network within a 300-second epoch. The energy consumption of the operations is listed in Table V. A 9000 mAh battery (output: 12 V and 24 V) can sustain approximately 6.4 hours of continuous formation control according to a rough estimation. This working period is estimated using the following equation:

$$\frac{9000 \times 3600 \times 12/1000}{(4800 + 200.6 + 57.6)/300}/3600 = 6.4\ (hours)$$

TABLE IV
AVERAGE ENERGY CONSUMPTION OF THE OPERATIONS IN AN EPOCH

Measurements	Communication	Moving	Sleep
t (seconds)	24	200	76
E(Joules)	57.6	4800	200.6

VI. Conclusions and Future Work

In this paper, we presented our initial study of distributed leader-follower formation control for a low-cost MUWSN. The algorithm controls the follower nodes to move toward the destination under large motion measurement errors. The mobile nodes are organized within a virtual rigid graph on the reference plane. The conditions for the mobile nodes to construct the projected rigid graph on the reference plane are theoretically analyzed and proven. The distance adjustment details are simulated and recorded. The simulations demonstrate that the formation control problem is efficiently reduced from 3D to 2D without sacrificing accuracy.

TRiForm requires three surface anchors, which may not be feasible for certain applications that possess narrow spaces, e.g., underwater mobile monitoring in a tube. We take the formation control scheme requiring fewer surface anchors as a future work direction.

References

[1] I. F. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: research challenges. *Ad hoc networks*, 3(3):257–279, 2005.

[2] J. Baillieul and P. J. Antsaklis. Control and communication challenges in networked real-time systems. *Proceedings of the IEEE*, 95(1):9–28, 2007.

[3] C. Belta and V. Kumar. Trajectory design for formations of robots by kinetic energy shaping. volume 3, pages 2593–2598, 05 2002.

[4] A. Cadena, P. Teran, G. Reyes, J. Lino, V. Yaselga, and S. Vera. Development of a hybrid autonomous underwater vehicle for benthic monitoring. In *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, pages 437–440, April 2018.

[5] M. Cao, C. Yu, and B. D. Anderson. Formation control using range-only measurements. *Automatica*, 47(4):776–781, 2011.

[6] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *Joint Conference of the IEEE Computer & Communications Societies IEEE*, 2002.

[7] L. Cheng and Y. Wang. Communication-based multiple mobile robots rigid formation control. In *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, volume 1, pages 729–734. IEEE, 2004.

[8] T. Curtin, J. G. Bellingham, J. Catipovic, and D. Webb. Autonomous oceanographic sampling networks. *Oceanography*, 6(3):86–94, 1993.

[9] S. Darbha, K. Rajagopal, et al. Information flow and its relation to the stability of the motion of vehicles in a rigid formation. In *American Control Conference, 2005. Proceedings of the 2005*, pages 1853–1858. IEEE, 2005.

[10] J. P. Desai, J. Ostrowski, and V. Kumar. Controlling formations of multiple mobile robots. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 4, pages 2864–2869 vol.4, May 1998.

[11] T. Eren, B. D. Anderson, A. S. Morse, W. Whiteley, P. N. Belhumeur, et al. Operations on rigid formations of autonomous agents. *Communications in Information & Systems*, 3(4):223–258, 2003.

[12] Z. Feng and G. Hu. Formation tracking control for a team of networked underwater robot systems with uncertain hydrodynamics. In *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 372–377, Aug 2018.

[13] Z. Hou, W. Wang, G. Zhang, and C. Han. A survey on the formation control of multiple quadrotors. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 219–225, June 2017.

[14] B. Jackson and T. Jordán. Connected rigidity matroids and unique realizations of graphs. *Journal of Combinatorial Theory, Series B*, 94(1):1–29, 2005.

[15] Q. Jia and G. Li. Formation control and obstacle avoidance algorithm of multiple autonomous underwater vehicles(auvs) based on potential function and behavior rules. In *2007 IEEE International Conference on Automation and Logistics*, pages 569–573, Aug 2007.

[16] J.-W. Kwon and D. Chwa. Hierarchical formation control based on a vector field method for wheeled mobile robots. *IEEE Transactions on Robotics*, 28(6):1335–1345, 2012.

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

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

[19] C. B. Low. A rapid incremental motion planner for flexible formation control of fixed-wing uavs. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 2427–2432. IEEE, 2012.

[20] C. B. Low. A flexible virtual structure formation keeping control design for nonholonomic mobile robots with low-level control systems, with experiments. In *Intelligent Control (ISIC), 2014 IEEE International Symposium on*, pages 1576–1582. IEEE, 2014.

[21] P. Millán, L. Orihuela, I. Jurado, and F. R. Rubio. Formation control of autonomous underwater vehicles subject to communication delays. *IEEE Transactions on Control Systems Technology*, 22(2):770–777, March 2014.

[22] D. A. M.Kiranmayi. Underwater wireless sensor networks: Applications, challenges and design issues of the network layer - a review. 3:05–11, 2015.

[23] S. Monteiro, M. Vaz, and E. Bicho. Attractor dynamics generates robot formations: From theory to implementation. volume 3, pages 2582 – 2586 Vol.3, 01 2004.

[24] S. Muminov, N. Yun, S. Shin, S. Park, J. Jeon, C. Hong, S. Park, C. Kim, G. Yang, and Y. Ryuh. Biomimetic fish robot controlling system by using underwater acoustic signal. In *2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 106–108, June 2012.

[25] H. Rezaee and F. Abdollahi. A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots. *IEEE Transactions on Industrial Electronics*, 61(1):347–354, 2014.

[26] F. Schiano and P. R. Giordano. Bearing rigidity maintenance for formations of quadrotor uavs. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1467–1474, May 2017.

[27] K. H. Tan and M. A. Lewis. Virtual structures for high-precision cooperative mobile robotic control. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, volume 1, pages 132–139 vol.1, Nov 1996.

[28] A. Walayat, N. Javaid, M. Akbar, and Z. A. Khan. Mees: Mobile energy efficient square routing for underwater wireless sensor networks. pages 292–297, 2017.

[29] Z. Wang, Y. Hirata, and K. Kosuge. Control a rigid caging formation for cooperative object transportation by multiple mobile robots. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1580–1585. IEEE, 2004.

[30] 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.

[31] H. Wu, Z. Ding, and J. Cao. Grolo: Realistic range-based localization for mobile iots through global rigidity. *IEEE Internet of Things Journal*, PP:1–1, 01 2019.

[32] C. H. Yu and J. W. Choi. Uwsns-based distributed trespasser tracking system for a fish farn. In *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pages 1003–1007, Oct 2018.

[33] 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.