Localized Algorithm for Precise Boundary Detection in 3D Wireless Networks

Hongyu Zhou, Su Xia, Miao Jin, and Hongyi Wu
The Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA, USA
E-mail: {hxz6029,sxx1110,mjin,wu}@cacs.louisiana.edu

***Abstract*—This research focuses on distributed and localized algorithms for precise boundary detection in 3D wireless networks. Our objectives are in two folds. First, we aim to identify the nodes on the boundaries of a 3D network, which serve as a key attribute that characterizes the network, especially in such geographic exploration tasks as terrain and underwater reconnaissance. Second, we construct locally planarized 2-manifold surfaces for inner and outer boundaries, in order to enable available graph theory tools to be applied on 3D surfaces, such as embedding, localization, partition, and greedy routing among many others. To achieve the first objective, we propose a Unit Ball Fitting (UBF) algorithm that discovers a set of potential boundary nodes, followed by a refinement algorithm, named Isolated Fragment Filtering (IFF), which removes isolated nodes that are misinterpreted as boundary nodes by UBF. Based on the identified boundary nodes, we develop an algorithm that constructs a locally planarized triangular mesh surface for each 3D boundary. Our proposed scheme is localized, requiring information within one-hop neighborhood only. Our simulation results demonstrate that the proposed algorithms can effectively identify boundary nodes and surfaces, even under high measurement errors. As far as we know, this is the first work for discovering boundary nodes and constructing boundary surfaces in 3D wireless networks.**

***Index Terms*—3D; boundary detection; triangulation; wireless sensor networks.**

I. Introduction

Many wireless networks exhibit substantial randomness, due to the lack of precise nodal deployment and the non-deterministic failures and channel dynamics. Therefore, the final formation of a wireless network heavily depends on its underlying environment. Consequently, there is a primary interest to discover the unknown geometry and topology of a wireless network formation (or a subnetwork formation), which provide salient information for understanding its environment and for efficient operation of the network itself. In particular, boundary is one of the key attributes that characterize the network in two or three-dimensional space, especially in such geographic exploration tasks as terrain and underwater reconnaissance.

A. Related Work

The quest for efficient boundary detection in wireless networks has led to two research thrusts outlined below:

Detection of Event Boundary

The investigation on boundary detection started from the estimation and localization of events in sensor networks. The spatially distributed sensors usually report different measurements in respond to an event. For example, upon a fire, the sensors located in the fire are likely destroyed (and thus resulting a void area of failed nodes), while the sensors close to the fire region measure higher temperature and smoke density than the faraway sensors do. Boundary detection is to delineate the regions of distinct behavior in a sensor network [1].

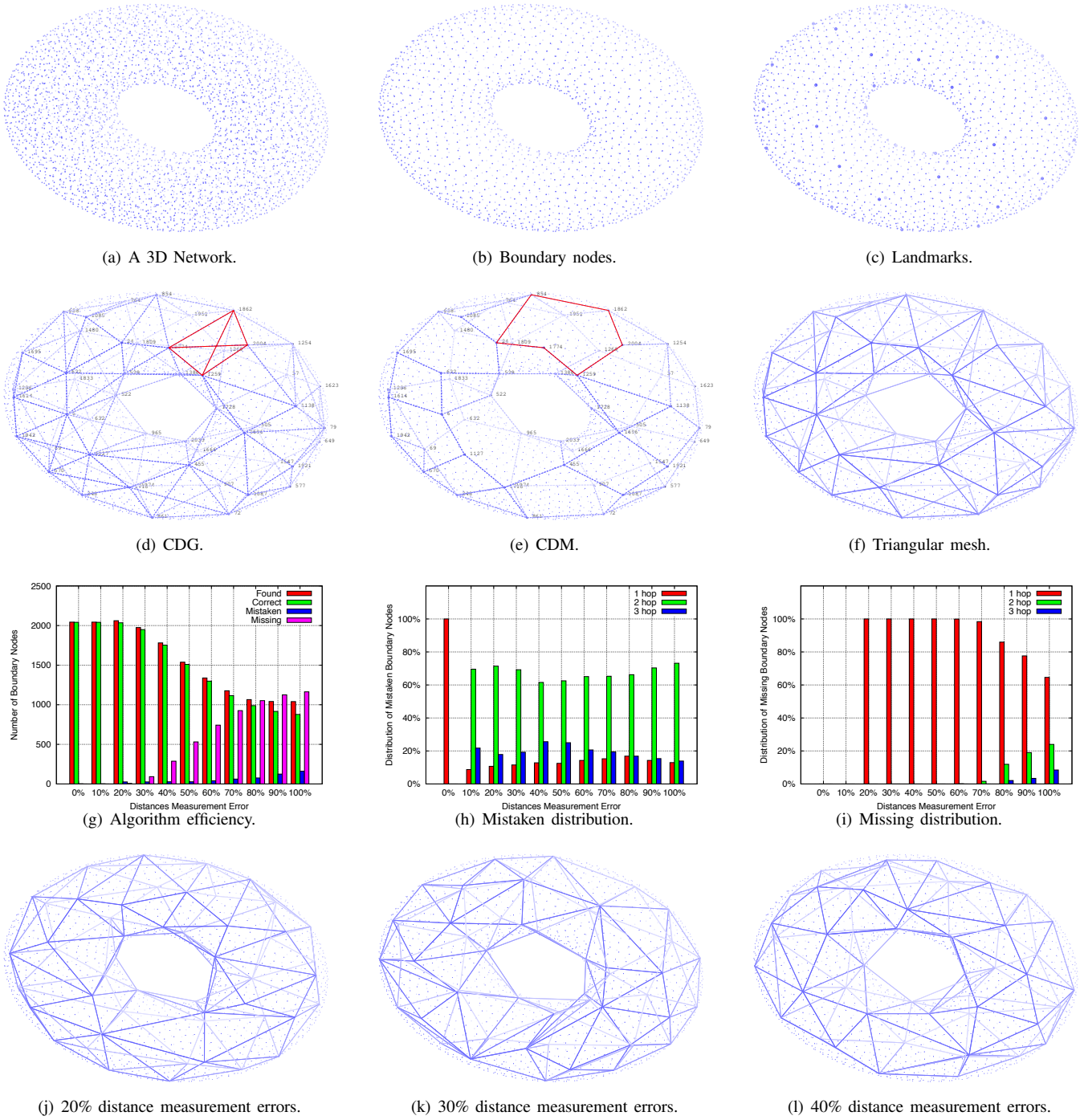
Achieving accurate detection of event boundary is challenging, because the sampling density is limited, the sensor readings are noisy, the delivery of sensor data is unreliable, and the computation power of individual sensors is extremely low [1], [2]. To this end, a series of studies have been carried out to explore efficient information processing and modeling techniques to analyze sensor data, in order to estimate the boundary of events [1]–[5].

Due to inevitable errors in raw sensor data, these approaches do not yield precise boundary. Instead, they aim at a close enough estimation that correctly identifies the events frontier, based on either global or local data collected from a set of sensors.

Detection of Network Boundary

Besides the researches discussed above that are mainly from the data processing perspective, interests are also developed to precisely locate the boundary of the network based on geometric or topology information of a wireless network. Noise in sensor data is no longer a concern here, because such boundary detection is not based on sensor measurement. However, new challenges arise due to the required accuracy of the identified boundary, especially in networks with complex inner boundary (i.e., "holes") or in high dimensional space.

Most proposed network boundary detection algorithms are based on 2D graphic tools. For example, Voronoi diagrams are employed in [6], [7] to discover coverage holes in sensor networks. Delaunay triangulation is adopted in [8] to identify communication voids. In contrast to [6]–[8] that exploit sensor positions, two distributed algorithms are proposed in [9] by utilizing distance and/or angle information between nodes to discover coverage boundary. In [10], an algebraic topological invariant called homology is computed to detect



(a) A 3D Network. (b) Boundary nodes. (c) Landmarks.

(d) CDG. (e) CDM. (f) Triangular mesh.

(g) Algorithm efficiency. (h) Mistaken distribution. (i) Missing distribution.

(j) 20% distance measurement errors. (k) 30% distance measurement errors. (l) 40% distance measurement errors.

Fig. 1. Illustration of the proposed boundary detection algorithm (based on a 3D wireless network of 4210 nodes with an average nodal degree of 18.8).

holes. The algorithm is generally applicable to networks in any dimensional space. However, it is a centralized approach and there is significant challenge to decentralize its computation [10]. In [11], the isosets (each of which consists of nodes with the same hop distance to a beacon node) are identified. The disconnection in an isoset indicates the boundary nodes of holes. Multiple beacons can be employed to locate the boundary nodes at different directions of a hole. This approach does not guarantee to discover the complete boundary of every hole. Higher accuracy can be achieved if more beacons are employed or when the network is denser. [12] introduces a deterministic algorithm for boundary detection. It searches for a special subgraph structure, called m-flower, which is bounded by a circle. Once a m-flower is identified, the algorithm can subsequently find the boundary nodes through a number of iterations of augmentation of the circle. But not every graph has an m-flower structure. Therefore, the algorithm may fail especially when the nodal density is low. In [13], a shortest

path tree is built to find the shortest circle, which is then refined to discover the tight boundaries of the inner holes.

All of the network boundary detection approaches discussed above are developed for networks in 2D space. Except [10] which is centralized, none of them can be readily applied to 3D networks since higher dimension space introduces significant complexity in searching for boundaries and many topological and geometrical tools cannot be extended from 2D to 3D. In addition, while boundary extraction has been extensively studied in 3D imaging, the algorithms developed therein always assume grid-like 3D pixels as inputs, which are in sharp contrast to network settings where nodes are randomly distributed, and thus are not applicable in 3D wireless networks.

B. Our Contribution

There are increasing interests in 3D wireless networks, with several areas such as routing [14]–[19], localization [20], nodal placement [21], [22], physical layer investigation [23] and applications [23], [24], being explored recently. This research aims to develop distributed and localized algorithms for precise boundary detection in 3D wireless networks. Our objectives are in two folds:

(1) First, we aim to identify the nodes on the boundaries of a 3D network (see Fig. 1(b) for example).
(2) Second, we construct locally planarized 2-manifold surfaces for inner and outer boundaries (as shown in Fig. 1(f)).

To achieve the first objective, we propose a Unit Ball Fitting (UBF) algorithm that discovers a set of potential boundary nodes, followed by a refinement algorithm, named Isolated Fragment Filtering (IFF), which removes isolated nodes that are misinterpreted as boundary nodes by UBF. Our proposed scheme is localized, requiring information within one-hop neighborhood only.

The boundary nodes are discrete. They serve as sample points that depict the network boundaries. However, many applications desire not only such discrete points, but also closed boundary surfaces, especially locally planarized 2-manifold in order to apply available graph theory tools on 3D surfaces, such as embedding, localization, partition, and greedy routing among many others. In this research we develop an algorithm that constructs locally planarized triangular meshes on the identified 3D boundaries. We adopt the method proposed in [25] that produces a planar subgraph in 2D, and extend it to 3D surfaces to achieve complete triangulation without degenerated edges. The algorithm is localized and based on connectivity only.

As far as we know, this is the first work for discovering boundary nodes and constructing boundary surfaces in 3D wireless networks. The rest of this paper is organized as follows: Secs. II and III introduce our proposed algorithms for boundary node identification and boundary surface construction, respectively. Sec. IV presents simulation results. Finally, Sec. V concludes the paper.

II. Boundary Node Identification

The proposed boundary node identification algorithm involves two phases. The first phase is the Unit Ball Fitting (UBF), which aims to discover most, if not all, boundary nodes. The second phase is Isolated Fragment Filtering (IFF), which removes isolated nodes that are misinterpreted as boundary nodes in Phases 1.

A. Phase 1: Unit Ball Fitting (UBF)

We present the Unit Ball Fitting (UBF) algorithm in this subsection. The related definitions, theories, and algorithm description are elaborated sequentially.

1) Definitions: To facilitate our exposition, we first introduce several basic definitions.

Definition 1: Without loss of generality, we consider an arbitrary radio transmission model with a maximum radio transmission range of 1.

Definition 2: The nodal density, denoted by ρ, is the average number of nodes in a unit volume.

Definition 3: We consider well connected networks only. A well connected network implies: (1) no nodes are isolated; and (2) there are no degenerated line segments. In other words, given a line segment between two nodes, e.g., Nodes i and j, there must be at least one node whose distances to Nodes i and j are less than $Max(1, d_{ij})$, where d_{ij} denotes the distance between Nodes i and j.

Definition 4: A unit ball is a ball with a radius of $r = 1 + \delta$, where δ is an arbitrarily small constant.

Definition 5: An empty unit ball is a unit ball with no nodes located inside.

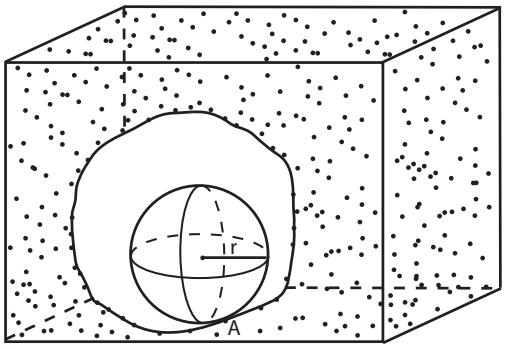
Definition 6: We say a unit ball *touches* a node if the node is on the surface of the ball.

Definition 7: A hole is an empty space that is greater than a unit ball. The space outside the network is treated as a special hole.

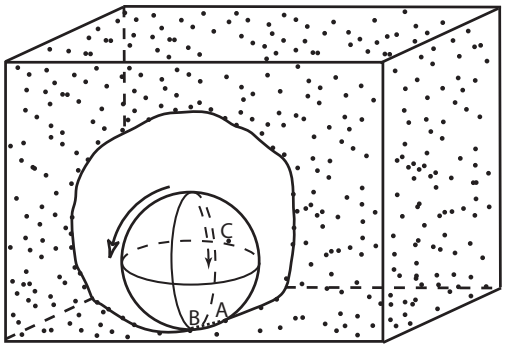
With the above definitions, we next discuss the motivations to develop the UBF algorithm and the theories that prove its correctness and computing complexity. Subsequently, we give the formal algorithm description.

2) Motivations and Theoretic Insights: The proposed UBF algorithm is motivated by the fact that a hole can always contain an empty unit ball. For example, the smallest hole is defined as a void space that can be filled by adding a node to connect with nearby nodes. Therefore, we can search for empty unit balls in order to identify holes and boundary nodes. More specifically, a node can test if it is on a boundary by constructing a unit ball with itself on the ball's surface. If at least one such ball can be found that no nodes are located inside, a hole is identified and the node is a boundary node (see Node A in Fig. 2(a) for example).

The above process is called unit ball fitting. It can be applied to identify both inner and outer boundaries. However, it is obviously infeasible for a node to perform a complete test of unit ball fitting via brute-force search, because there are infinite possible orientations to place the unit ball. Next, we will show that a localized algorithm with a polynomial



(a) An empty unit ball touching Node A.



(b) Ball rotation.

Fig. 2. Principles for Unit Ball Fitting (UBF).

computing complexity can be employed to test if such an empty unit ball exists.

Lemma 1: Node A can construct an empty unit ball that touches itself if and only if there exists an empty unit ball touching Node A and two neighbors of Node A (within $2r$).

Proof: We first show the sufficient condition, which is straightforward. If a unit ball touched by Node A and two neighbors of Node A is empty, i.e., there is an empty unit ball with Node A and two neighbors of Node A on its surface, Node A has constructed such an empty unit ball touching itself. Consequently, a hole is identified and Node A is a boundary node.

Now, we prove the necessary condition. If there exists an empty unit ball with Node A on its surface, we can always fix Node A and rotate the ball until it touches another node within $2r$, denoted by Node B (see Fig. 2(b)). Note that if Node B does not exist, Node A must be isolated, which conflicts with our assumption of well connected networks (see Definition 3). Then we can further rotate the ball with Line AB as an axis, until it touches another node, denoted by Node C. Similarly, Node C must exist, because otherwise Line AB is degenerated and thus against Definition 3. Therefore, if Node A can construct an empty unit ball that touches itself, we can always find an empty unit ball with Node A and two neighbors of Node A on its surface.

Based on the sufficient condition and the necessary condition discussed above, the lemma is thus proven. ■

According to Lemma 1, we can show that a node can determine if it can construct an empty unit ball that touches itself by a localized algorithm with a computing complexity of $\Theta(\rho^3)$. If such an empty unit ball can be constructed, the node must be a boundary node. Formally, we have the following theorem.

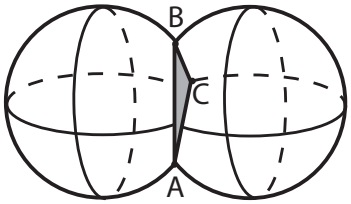


Fig. 3. Up to two unit balls determined by Node A and two of its neighbors.

Theorem 1: Node A can determine if it can construct an empty unit ball that touches itself by testing $\Theta(\rho^2)$ unit balls and $\Theta(\rho)$ nodes for each ball.

Proof: According to Lemma 1, Node A can exhaustively test all unit balls determined by Node A and its neighbors. Given Node A and any two neighbors (whose distances to Node A are less than $2r$), zero or one or two unit balls can be formed such that the three nodes are on the surface(s). Fig. 3 illustrates an example where two unit balls are determined by three nodes. Since Node A has about $\frac{4}{3}\pi(2r)^3\rho$, or $\Theta(\rho)$, neighboring nodes within the distance of $2r$, it needs to test up to $\Theta(2\times\binom{\rho}{2}) = \Theta(\rho^2)$ unit balls. For each unit ball, about $\frac{4}{3}\pi r^3\rho$, or $\Theta(\rho)$, nodes must be tested to judge if it is empty. Therefore, the overall computing complexity is $\Theta(\rho^3)$. Note that ρ is usually small and bounded. ■

3) Algorithm Description: Theorem 1 provides a clear guidance for our algorithm development. It suggests a distributed and localized algorithm where each node tests $\Theta(\rho^2)$ unit balls to judge if any one of them is empty. To this end, we propose the Unit Ball Fitting (UBF) algorithm as outlined in Algorithm 1 and elaborated below.

Algorithm 1: Unit Ball Fitting (UBF) Algorithm

Input: $N(i)$; //Neighbors of Node i
Output: $Boundary(i)$;
$Boundary(i) = FALSE$;
Establish a local coordinates system;
$\Omega_i = \{[j,(x_j,y_j,z_j)]\,|\,j \in N(i)\}$;
for $j,k \in \Omega_i$ *and* $j \neq k$ **do**
    Find the unit ball(s) determined by Nodes i,j,k;
    if *a unit ball is empty* **then**
        $Boundary(i) = TRUE$;
        Break;
    end
end

The proposed UBF algorithm largely follows the discussions in Sec. II-A2. The sole difference is that each node considers its one-hop neighbors only to realize a truly localized algorithm. It consists of the following three steps, and outputs

a boolean value $Boundary(i)$ indicating if Node i is on a boundary or not.

(I) *Local coordinates establishment (Lines 2-3):* If all nodes have known their coordinates, this step can be skipped. Otherwise each node employs a 3D embedding algorithm to establish a local coordinates system. More specifically, Node i collects the distances between all pairs of nodes within one hop. The distance between two nodes can be estimated by such ranging techniques as received signal strength indicator (RSSI) or time difference of arrival (TDOA) [26]. The measured distances are inaccurate in general and the errors will be discussed in Sec. IV. Based on the pair-wise distances, multiple schemes [27]–[31] are available to create a local coordinates system for Node i and its neighbors. Among them, [31] is adopted in our implementation. Once the coordinates system is established, Node i keeps a set of neighboring nodes and their coordinates, i.e., $\Omega_i = \{[j,(x_j,y_j,z_j)] \,|\, j \in N(i)\}$, where $N(i)$ denotes the set of nodes that include Node i itself and its one-hop neighbors.

(II) *Unit ball identification (Lines 4-5):* For every two distinct nodes, e.g., j and $k \in \Omega_i$, calculate the center(s) of the unit ball(s) determined by Nodes i, j and k. This is done by solving a set of standard equations as follows, where (x,y,z) are the coordinates of the center.

$$\begin{cases} (x-x_i)^2+(y-y_i)^2+(z-z_i)^2 = r^2, \\ (x-x_j)^2+(y-y_j)^2+(z-z_j)^2 = r^2, \\ (x-x_k)^2+(y-y_k)^2+(z-z_k)^2 = r^2. \end{cases} \quad (1)$$

Depending on the coordinates of Nodes i, j and k, Eq. (1) may yield no solution, or one solution, or two solutions for (x,y,z).

(III) *Empty unit ball check (Lines 6-9):* For each center point (x,y,z) identified above, check if any node in Ω_i is located inside the corresponding unit ball, i.e., if it is an empty unit ball. If an empty unit ball is found, Node i declares that it is on a boundary.

Steps (II) and (III) check all unit balls determined by Node i and its neighbors. If no empty unit ball is found, Node i reports that it is not a boundary node. As revealed by Theorem 1, only $\Theta(\rho^2)$ unit balls need to be examined by each node. Moreover, it requires local information only, i.e., merely the coordinates of the neighboring nodes are needed, and a local coordinates system (without global alignment) is sufficient.

In addition, the size of holes to be detected is adjustable by varying r (or δ). By default, one can set r close to 1, in order to identify the holes of any size. However, if one is interested in the boundary nodes of large holes only, a larger r can be chosen. As a result, a node on the boundary of a small hole cannot find an empty unit ball that can fit in according to Algorithm 1 and thus deems itself a non-boundary node.

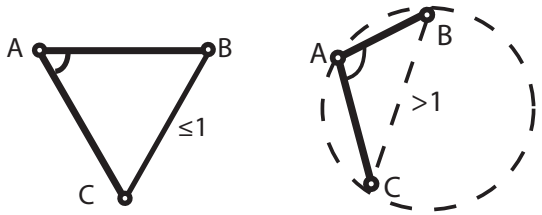
B. Phase 2: Isolated Fragment Filtering (IFF)

A small number of interior nodes may be interpreted by UBF as boundary nodes due to inaccurate nodal coordinates or unexpected low nodal density areas randomly distributed in the network, resulting in some isolated fragments that should be filtered out. Generally, the nodes on a boundary form a well connected closed surface. Therefore, we can set a threshold γ. Any fragment that consists of less than γ nodes is not considered as a boundary. To this end, each boundary node simply initiates a local flooding packet with a TTL of T, which will be forwarded by other boundary nodes but not non-boundary nodes. By counting the number of such flooding packets received, a boundary node learns the size of its fragment. If less than γ flooding packets are received, the node deems itself a non-boundary node. Appropriate γ and T are chosen according to the minimum size of the holes to be detected. For example, given the default value of r (i.e., $r = 1+\delta$ for an arbitrary small δ), a minimum hole will have at least 20 nodes on its surface, forming an icosahedron, where the maximum hop distance between two boundary nodes is 3. Thus we set $\gamma = 20$ and $T = 3$. Since IFF is based on a simple local flooding, it has a complexity of $O(1)$.

In addition, the boundary nodes can be easily grouped, when there are multiple boundaries. Note that, the nodes on a boundary are connected via boundary nodes. In other words, there must exist a path between two nodes on the same boundary, which involves boundary nodes only. For two nodes on different boundaries, such path does not exist, and their connection must go through at least one non-boundary node. Therefore a straightforward scheme (similar to the local flooding approach discussed above) can be employed to group the boundary nodes.

C. Summary of Boundary Node Identification

By following the two phases elaborated above, a node determines whether it is on the boundary based on local information only. Its performance depends on the accuracy in distance measurement, which is used to establish local coordinates. As to be discussed in Sec. IV, our simulations demonstrate that the proposed algorithms are effective, able to identify almost all boundary nodes with low miss and mistaken rate, when the distance measurement errors are moderate (as shown Fig. 1(g)). Under high distance measurement errors, the mistaken and missing rates naturally increase. But the mistakenly identified boundary nodes are all close to the true boundary, mostly within one or two hops (see Fig. 1(h)). At the same time, the missed boundary nodes are uniformly scattered.



(a) Non-boundary node. (b) Missed boundary node.

Fig. 4. Illustration of a missed boundary node. (a) Node A is not a boundary node. (b) Node A is a boundary node but cannot be identified by UBF.

Over 95% of such missed boundary nodes can always find at least one correctly identified boundary node within its one hop neighborhood (as illustrated in Fig. 1(i)). Therefore, the identified boundary nodes can well represent the network boundaries. More discussion will be presented in Sec. IV.

III. Triangular Boundary Surface Construction

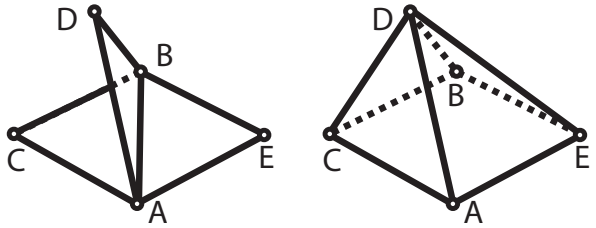
The boundary nodes identified so far are discrete. They largely depict the network boundaries. However, many applications require not only discrete boundary nodes, but also closed boundary surfaces. Moreover, it is highly desirable that such surfaces are locally planarized 2-manifold in order to apply available 2D graphic tools on 3D surfaces.

In this research we implement an algorithm that constructs locally planarized triangular meshes on the identified 3D boundaries. We adopt the method proposed in [25] that can produce a planar subgraph in a 2D network, and extend it to 3D surfaces to achieve complete triangulation without degenerated edges. The algorithm is localized and based on connectivity only. It consists the following five steps.

(I) *Landmark Selection:* The boundary nodes employ a distributed algorithm (e.g., [32]) to elect a subset of nodes as "landmarks". Any two landmarks must be k-hops apart. k determines the fineness of the mesh. It is usually set between 3 to 5 in our implementation. A non-landmark boundary node is associated with the closest landmark. If it has the same distance (in hop counts) to multiple landmarks, it chooses the one with the smallest ID as a tiebreaker. This step creates a set of approximate Voronoi cells on each boundary (as shown in Fig. 1(c)).

(II) *Construction of Combinatorial Delaunay Graph (CDG):* Each non-landmark boundary node checks if it has a neighboring boundary node that is associated with a different landmark. If it has, a message is sent to both landmarks to indicate that they are neighboring landmarks. If we simply connect all neighboring landmarks, we arrive at a Combinatorial Delaunay Graph (CDG) as illustrated in Fig 1(d), which is the respective dual of the Voronoi cells on a boundary found in Step I. However, such a CDG is not planar (see the crossing edges highlighted in Fig 1(d)).

(III) *Construction of Combinatorial Delaunay Map (CDM):* Each landmark node decides whether it connects to a neighboring landmark as follows. It sends a packet to a neighboring landmark through the shortest path (based on the identified boundary nodes only). The packet records the nodes along the path. The two landmarks are said to be connected if and only if the following two conditions are satisfied. First, all of nodes visited by the packet are associated to these two landmarks only. Second, assume the packet is sent from Landmark i to Landmark j. Then the packet must visit the nodes associated with Landmark i first, and then followed by the nodes associated with Landmark j, without interleaving. If the above two conditions are satisfied, Landmark j sends an ACK to Landmark i and a virtual edge is added between them. The boundary nodes that receive such ACK records that they are on the shortest path between two connected landmarks. This step yields a *Combinatorial Delaunay Map (CDM)*. It is proven that CDM is a planar graph [25].



(a) Before edge flip. (b) After edge flip.

Fig. 5. Illustration of edge flip. Edge AB has three faces before edge flip. It is removed and replaced by Edges CD and DE.

(IV) *Construction of Triangular Mesh:* The CDM obtained so far is planar, but not always a triangular mesh. Polygons with more than three edges may exist (see the polygon highlighted in Fig 1(e)). To achieve complete triangulation, appropriate edges should be added between some neighboring landmarks. If a landmark, e.g., Landmark i, has a non-connected neighboring landmark (e.g., Landmark j), it sends a *connection* packet to the latter (via the shortest path based on the identified boundary nodes). The packet will be dropped if it reaches an intermediate node that is already on the shortest path between two connected landmarks, in order to avoid crossing edges. If the *connection* packet arrives at Landmark j, a virtual edge can be safely added and an ACK is sent back to Landmark i. Similarly, the boundary nodes that receive the ACK records that they are on the shortest path between two connected landmarks. This step adds all possible virtual edges to divide polygons into triangles.

(V) *Edge Flip:* To ensure the mesh to be a 2-manifold, each virtual edge must be associated with two triangles. After the above step, there still possibly exist edges (like Edge AB in Fig. 5(a)) with three triangular faces, formed with three corresponding nodes (i.e., C, D, and E). Such edges can be detected by trivial local signaling. For each such edge, a transformation is done as follows. First, Edge AB is removed. Second, two shortest edges are added between the corresponding nodes, i.e., Nodes C, D, and E. For example, assume CE is longer than CD and DE. Then two virtual edges CD and DE are added, resulting in Fig. 5(b), where no edge has more than two faces. Note that the polygon $ACBE$ is not a face on the surface. Till now, we arrive at a planar triangular mesh for each 3D boundary surface, as illustrated in Fig. 1(f).

The above algorithm is able to form a closed triangular mesh surface for each boundary. The established triangular mesh is a locally planarized 2-manifold, although the whole 3D surface

is not planar. A virtual edge on a mesh surface has exactly two triangular faces. Such salient properties enable application of many useful graph theory tools on 3D boundary surfaces, including embedding, localization, partition, and greedy routing among many others.

Note that since the triangular mesh is established based on landmarks only, a small number of nodes (that are on or close to the boundaries) may be located outside the mesh surfaces. The number of such nodes is determined by k and the curvature of the boundary. The larger the k, the coarser the mesh surfaces, resulting in more nodes left outside. An appropriate k can be chosen according to the needs of specific applications. For example, Fig. 1(f) shows the results with $k = 3$.

In addition, we observed that the triangular mesh is not seriously deformed under distance measurement errors. As discussed in the previous subsection, the mistakenly identified boundary nodes are close to the true boundary and the missing boundary nodes are uniformly distributed. Therefore, the identified boundary nodes can still well represent the network boundaries, even under distance measurement errors. This is verified by our simulations. For example, Figs. 1(j)-1(l) show results under 20%, 30% and 40% distance measurement errors, respectively. They exhibit similar triangular meshs as Fig. 1(j), which is free of distance measurement errors.

IV. Simulations

To evaluate the effectiveness of our proposed boundary detection algorithms, we have carried out extensive simulations under various 3D wireless networks and studied the impact of a wide range of distance measurement errors. In this section, we will first introduce our simulation setup. Then we present the simulation results and discuss our observations.

A. Simulation Setup

The 3D networks used in our simulations are constructed by using a set of 3D graphic tools (including TetGen [33]). First, a 3D model is developed to represent a given network scenario (e.g., an underwater network, a 3D network in space, and general 3D networks with arbitrary shapes of our interest). A set of nodes are randomly uniformly distributed on the surface of the 3D model. They are marked as boundary nodes, serving as ground truth to evaluate our algorithm. A cloud of nodes are then deployed inside the 3D model. Again the nodes are randomly uniformly distributed. Once the nodes are determined, an appropriate radio transmission range is chosen according to nodal density, such that the network is connected. Each node connects to its neighbors within its radio transmission range. In our simulated networks, nodal degree ranges from 5 to 45, with an average of 18.5. A node also estimates its distance to each neighbor. While our simulations do not involve physical layer modeling, we introduce a wide range of random errors, from 0 to 100% of the radio transmission radius, in the distance measurement.

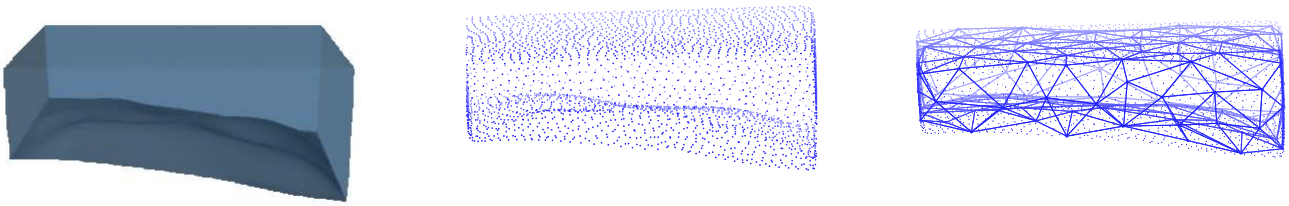
Till now, we have the input for our algorithm. For each simulated network, the input includes a set of the nodes (both interior and boundary nodes), the local 1-hop connectivity of each node, and the distance measurement (with various errors) within 1-hop neighborhood.

B. Simulation Results

We run our proposed distributed and localized algorithms for boundary node detection and surface construction. First, each node establishes a local coordinates system by using distributed multi-dimensional scaling [31] based on local distance measurement. Then boundary node identification is performed, followed by the triangular mesh algorithm.

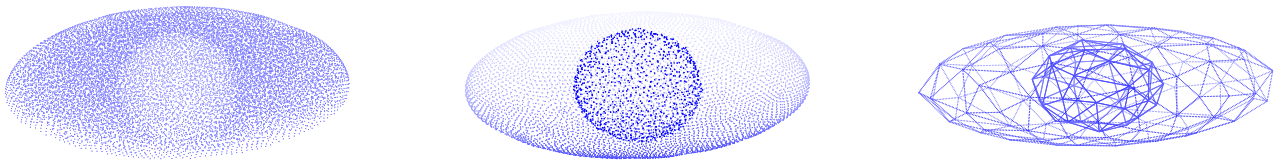
Several examples of our simulated networks are given in Figs. 6-10. Fig. 6 illustrates an underwater network, where nodes are distributed from the surface to the bottom of the ocean. As shown in Figs. 6(b) and 6(c), our algorithms effectively identify the boundaries of both smooth water surface and the bumpy bottom. Figs. 7 and 8 depicts a 3D network deployed in the space (e.g., for chemical dispersion sampling in 3D space). They have one and two internal holes, respectively, due to uncontrolled drift of sensor nodes. These examples demonstrate that our algorithm works for not only outer boundary but also the boundaries of interior holes. Figs. 9 and 10 show 3D networks deployed in a bended pipe and a sphere, respectively. As can be seen, boundary nodes are accurately identified and the triangular mesh surfaces are well constructed in the both networks.

Fig. 11 illustrates the performance statistics obtained from our simulations. The results are based on over 10,000 sample boundary nodes. As can be seen in Fig. 11(a), our algorithm performs almost perfectly to identify boundary nodes when the distance measurement error is less than 30%. With more errors introduced in distance measurement, noticeable errors are yielded in local coordinates establishment, which naturally lead to missing and mistaken boundary nodes. More specifically, when the coordinates errors exceed certain level, an original boundary node may become an interior node inside the network under the established coordinates and thus is missed by our boundary detection algorithm. At the same time, an original interior node may appear on the boundary due to the deformation of the coordinates of the node itself and its neighbors, leading to a mistakenly identified boundary node. However, as we have demonstrated in Fig. 1 and discussed in Sec. II, such missing and mistaken boundary nodes do not seriously affect our boundary identification, because they are well distributed. For example, Fig. 11(b) illustrates the distribution of mistaken boundary nodes. Specifically, we measure the shortest distance (in hops) from a mistaken boundary node to a correctly identified boundary node. As can be seen in Fig. 11(b), such distance is always less than 3 hops, with a majority of them in one (over 60%) and two hops (over 30%). These results clearly show that the mistakenly identified nodes are very close to the true boundary. Therefore, the triangular mesh surface does not deviate significantly from the true boundary surface. Similarly, the distribution of missing boundary nodes is given in Fig. 11(c). It is observed that almost 100% of the missing boundary nodes are within one-hop



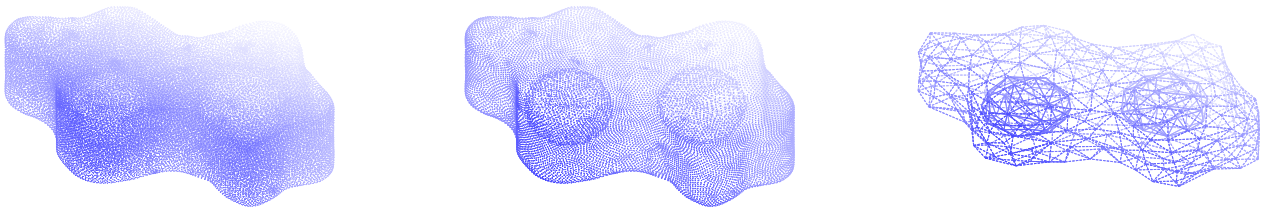
(a) Network model. (b) Boundary nodes. (c) Triangular mesh.

Fig. 6. An example of under water network. (In contrast to Figs. 7-10 where the network model, i.e., subfigure (a), shows a set of wireless nodes deployed, the network model in this figure gives the actual 3D model for better visualization.)



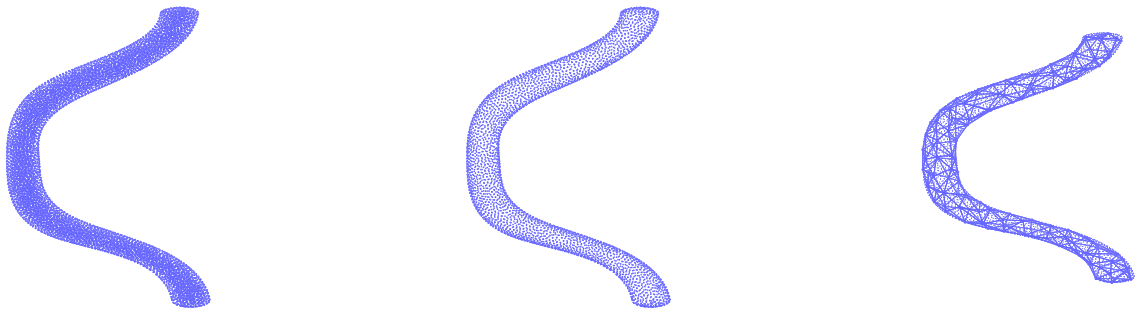
(a) Network model. (b) Boundary nodes. (c) Triangular mesh.

Fig. 7. An example of a 3D space network with an internal hole.



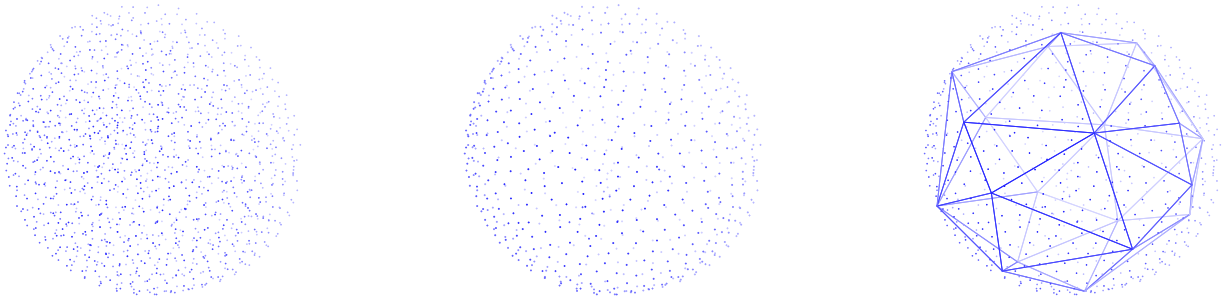
(a) Network model. (b) Boundary nodes. (c) Triangular mesh.

Fig. 8. An example of a 3D space network with two internal holes.



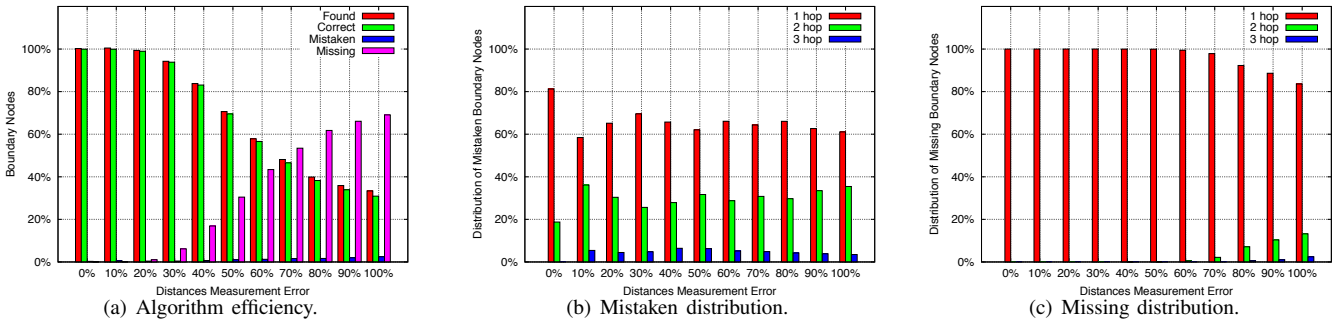
(a) Network model. (b) Boundary nodes. (c) Triangular mesh.

Fig. 9. An example of a 3D network in a bended pipe.



(a) Network model. (b) Boundary nodes. (c) Triangular mesh.

Fig. 10. An example of a 3D network in a sphere.



(a) Algorithm efficiency. (b) Mistaken distribution. (c) Missing distribution.

Fig. 11. Performance statistics.

neighborhood of correctly identified boundary nodes. In other words, the missing boundary nodes are uniformly distributed on the boundary surfaces (without forming large "holes") and thus do not affect the election of landmarks significantly. As a result, triangular mesh can be well constructed based on the set of landmark nodes sampled from identified boundary nodes.

V. Conclusion

We have proposed distributed and localized algorithms for precise boundary detection in 3D wireless networks. Our objectives have been in two folds. First, we have aimed to identify the nodes on the boundaries of a 3D network, which serve as a key attribute that characterizes the network, especially in such geographic exploration tasks as terrain and underwater reconnaissance. Second, we have intended to construct locally planarized 2-manifold surfaces for inner and outer boundaries, in order to enable available graph theory tools to be applied on 3D surfaces, such as embedding, localization, partition, and greedy routing among many others. To achieve the first objective, we have proposed a Unit Ball Fitting (UBF) algorithm that discovers a set of potential boundary nodes, followed by a refinement algorithm, named Isolated Fragment Filtering (IFF), which removes isolated nodes that are misinterpreted as boundary nodes by UBF. Based on the identified boundary nodes, we have developed an algorithm that constructs a locally planarized triangular mesh surface for each 3D boundary. Our proposed scheme is localized, requiring information within one-hop neighborhood only. Our simulation results have shown that the proposed algorithms can effectively identify boundary nodes and surfaces, even under high measurement errors. As far as we know, this is the first work for discovering boundary nodes and constructing boundary surfaces in 3D wireless networks.

References

[1] R. Nowak and U. Mitra, "Boundary Estimation in Sensor Networks: Theory And Methods," in *Proc. of The International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 80–95, 2003.

[2] M. Ding and X. Cheng, "Robust Event Boundary Detection and Event Tracking in Sensor Networks - a Mixture Model based Approach," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[3] K. Chintalapudi and R. Govindan, "Localized Edge Detection in Sensor Fields," in *Proc. of The First IEEE. 2003 IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 59–70, 2003.

[4] S. Duttagupta, K. Ramamritham, and P. Ramanathan, "Distributed Boundary Estimation using Sensor Networks," in *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pp. 316–325, 2006.

[5] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized Fault-Tolerant Event Boundary Detection in Sensor Networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 902–913, 2005.

[6] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted Sensor Deployment," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, 2004.

[7] A. Ghosh, "Estimating Coverage Holes and Enhancing Coverage in Mixed Sensor Networks," in *Proc. of The 29th Annual IEEE International Conference on Local Computer Networks*, pp. 68–76, 2004.

[8] Q. Fang, J. Gao, and L. J. Guibas, "Locating and Bypassing Routing Holes in Sensor Networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 2458–2468, 2004.

[9] C. Zhang, Y. Zhang, and Y. Fang, "Localized Algorithms for Coverage Boundary Detection in Wireless Sensor Networks," *Wireless Networks*, vol. 15, no. 1, pp. 3–20, 2009.

[10] R. Ghrist and A. Muhammad, "Coverage and Hole-Detection in Sensor Networks Via Homology," in *Proc. of The International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 254–260, 2005.
[11] S. Funke, "Topological Hole Detection in Wireless Sensor Networks and Its Applications," in *Proc. of Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pp. 44–53, 2005.
[12] A. Kroller, S. P. Fekete, D. Pfisterer, and S. Fischer, "Deterministic Boundary Recognition and Topology Extraction for Large Sensor Networks," in *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1000–1009, 2006.
[13] Y. Wang, J. Gao, and J. S. B. Mitchell, "Boundary Recognition in Sensor Networks by Topological Methods," in *Proc. of The ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pp. 122–133, 2006.
[14] C. Liu and J. Wu, "Efficient Geometric Routing in Three Dimensional Ad Hoc Networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, 2009.
[15] T. F. G. Kao and J. Opatmy, "Position-Based Routing on 3D Geometric Graphs in Mobile Ad Hoc Networks," in *Proc. of The 17th Canadian Conference on Computational Geometry*, pp. 88–91, 2005.
[16] J. Opatrny, A. Abdallah, and T. Fevens, "Randomized 3D Position-based Routing Algorithms for Ad-hoc Networks," in *Proc. of Third Annual International Conference onMobile and Ubiquitous Systems: Networking & Services*, pp. 1–8, 2006.
[17] R. Flury and R. Wattenhofer, "Randomized 3D Geographic Routing," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 834–842, 2008.
[18] F. Li, S. Chen, Y. Wang, and J. Chen, "Load Balancing Routing in Three Dimensional Wireless Networks," in *Proc. of IEEE International Conference on Communications (ICC)*, pp. 3073–3077, 2008.
[19] D. Pompili, T. Melodia, and I. F. Akyildiz, "Routing Algorithms for Delay-insensitive and Delay-sensitive Applications in Underwater Sensor Networks," in *Proc. of The ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pp. 298–309, 2006.
[20] W. Cheng, A. Y. Teymorian, L. Ma, X. Cheng, X. Lu, and Z. Lu, "Underwater localization in sparse 3d acoustic sensor networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 798–806, 2008.
[21] X. Bai, C. Zhang, D. Xuan, J. Teng, and W. Jia, "Low-Connectivity and Full-Coverage Three Dimensional Networks," in *Proc. of ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHOC)*, pp. 145–154, 2009.
[22] X. Bai, C. Zhang, D. Xuan, and W. Jia, "Full-Coverage and K-Connectivity (K=14, 6) Three Dimensional Networks," in *Proc. of IEEE Int'l Conference on Computer Communications (INFOCOM)*, 2009.
[23] J. Allred, A. B. Hasan, S. Panichsakul, W. Pisano, P. Gray, J. Huang, R. Han, D. Lawrence, and K. Mohseni, "SensorFlock: An Airborne Wireless Sensor Network of Micro-Air Vehicles," in *Proc. of The International Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 117–129, 2007.
[24] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou, "Challenges: Building Scalable Mobile Underwater Wireless Sensor Networks for Aquatic Applications," *IEEE Network, Special Issue on Wireless Sensor Networking*, vol. 20, no. 3, pp. 12–18, 2006.
[25] S. Funke and N. Milosavljevi, "How Much Geometry Hides in Connectivity? - Part II," in *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 958–967, 2007.
[26] Z. Zhong and T. He, "MSP: Multi-Sequence Positioning of Wireless Sensor Nodes," in *Proc. of The International Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 15–28, 2007.
[27] H. Wu, C. Wang, and N.-F. Tzeng, "Novel Self-Configurable Positioning Technique for Multi-hop Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 609–621, 2005.
[28] G. Giorgetti, S. Gupta, and G. Manes, "Wireless Localization Using Self-Organizing Maps," in *Proc. of The International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 293 – 302, 2007.
[29] L. Li and T. Kunz, "Localization Applying An Efficient Neural Network Mapping," in *Proc. of The 1st International Conference on Autonomic Computing and Communication Systems*, pp. 1–9, 2007.
[30] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from Mere Connectivity," in *Proc. of ACM Int'l Symposium on Mobile Ad hoc Networking and Computing (MobiHOC)*, pp. 201–212, 2003.
[31] Y. Shang and W. Ruml, "Improved MDS-based Localization," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 2640–2651, 2004.
[32] Q. Fang, J. Gao, L. J. Guibas, V. Silva, and L. Zhang, "GLIDER: Gradient Landmark-based Distributed Routing for Sensor Networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 339–350, 2005.
[33] http://tetgen.berlios.de/.