Localized and Precise Boundary Detection in 3-D Wireless Sensor Networks

Hongyu Zhou, *Student Member, IEEE*, Su Xia, *Member, IEEE*, Miao Jin, and Hongyi Wu, *Member, IEEE*

***Abstract*—This research focuses on distributed and localized algorithms for precise boundary detection in 3-D wireless networks. Our objectives are twofold. First, we aim to identify the nodes on the boundaries of a 3-D 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 3-D 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 majority of boundary nodes, followed by a refinement algorithm, named Isolated Fragment Filtering (IFF), to remove isolated nodes that are misinterpreted as boundary nodes. Based on the identified boundary nodes, we develop an algorithm that constructs a locally planarized triangular mesh surface for each 3-D boundary. Our proposed scheme is localized, requiring information within 1-hop neighborhood only. We further extend the schemes for online boundary detection in mobile sensor networks aiming to achieve low overhead. Our simulation and experimental results demonstrate that the proposed algorithms can effectively identify boundary nodes and surfaces, even under high measurement errors.**

***Index Terms*—Boundary detection, triangulation, wireless sensor networks.**

I. Introduction

MANY wireless networks exhibit substantial randomness, due to the lack of precise nodal deployment and the nondeterministic 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- (2-D) or three-dimensional (3-D) 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 here.

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 has 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 research discussed above that is 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 2-D graphic tools. For example, Voronoi diagrams are employed in [6] and [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.

Manuscript received February 25, 2014; revised June 10, 2014; accepted July 11, 2014; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor X. Wang. Part of this work was presented at the IEEE International Conference on Distributed Computing Systems (ICDCS), Genova, Italy, June 21–25, 2010.

H. Zhou was with the Center for Advanced Computer Studies (CACS), University of Louisiana at Lafayette, Lafayette, LA 70503 USA. He is now with Epic Systems, Verona, WI 53593 USA (e-mail: hongyu.ull@gmail.com).

S. Xia was with the Center for Advanced Computer Studies (CACS), University of Louisiana at Lafayette, Lafayette, LA 70503 USA. He is now with Cisco Systems, Inc., San Jose, CA 95134 USA (e-mail: suxia.ull@gmail.com).

M. Jin and H. Wu are with the Center for Advanced Computer Studies (CACS), University of Louisiana at Lafayette, Lafayette, LA 70503 USA (e-mail: mjin@cacs.louisiana.edu; wu@cacs.louisiana.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2014.2344663

In [10], an algebraic topological invariant called homology is computed to detect 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 as pointed out in [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. Reference [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 an m-flower is identified, the algorithm can subsequently find the boundary nodes through a number of iterations of augmentation of the circle. However, 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 2-D space. Except for [10], which is centralized, none of them can be readily applied to 3-D networks since higher-dimension space introduces significant complexity in searching for boundaries, and many topological and geometrical tools cannot be extended from 2-D to 3-D. Note that if global coordinates are available, boundary detection would become straightforward. However, this approach is often overkilling because the process of establishing global coordinates itself results in significant computation and communication overhead [14]. In addition, while boundary extraction has been extensively studied in 3-D imaging, the algorithms developed therein always assume grid-like 3-D pixels as inputs, which are in sharp contrast to network settings where nodes are randomly distributed, and thus are not applicable in 3-D wireless networks.

This work (partially presented in [15]) proposes the first algorithms for efficiently discovering boundary nodes and constructing boundary surface in 3-D wireless sensor networks. Following [15], several relevant research works have been carried out recently. For example, an effective algorithm is proposed in [16] to timely track dynamic network boundaries. It transforms a notched surface into a convex one to support of fast online boundary detection. However, the performance of the algorithm is determined by two important parameters required by the transformation. Unfortunately, both of them are model-dependent. Different models have different optimal parameters for achieving the best results. More discussion and comparison will be presented in Section V. It deals with boundary nodes only, but not surface. References [17] and [18] aim to address the 3-D wireless boundary detection problem based on connectivity only. In [17], an algorithm called CABET is proposed. It first identifies a set of boundary nodes based on the assumption that a boundary node has less neighbors than its internal counterpart. Then, three types of critical boundary nodes (i.e., convex, concave, and saddle nodes) are selected to depict the geometric features of the 3-D sensor network, based on which closed boundary surfaces are constructed. CABET is effective when the sensors are uniformly distributed, yielding accurate boundary nodes and boundary surfaces. When it comes to nonuniform networks, CABET often becomes error-prone. An algorithm dubbed "Cococut" is introduced in [18]. It overcomes this problem by constructing a tetrahedral structure to delineate the approximate geometry of the 3-D sensor network, which is independent to nodal distribution. A set of sealed triangular boundary surfaces is produced based on this structure to separate nonboundary nodes and boundary node candidates. The former are hollowed out immediately, while the latter are further refined to yield the final boundary nodes and fine-grained boundary surfaces. However, the connectivity-based approaches cannot differentiate nodes within 1 hop, and thus are less accurate compared to the performance of methods based on distance or coordinates.

B. Our Contribution

There are increasing interests in 3-D wireless networks, with several areas such as routing [19]–[24], localization [14], [25], nodal placement [26], [27], physical-layer investigation [28], and applications [28], [29] being explored recently. This research aims to develop distributed and localized algorithms for precise boundary detection in 3-D wireless networks. Our objectives are twofold.

1) First, we aim to identify the nodes on the boundaries of a 3-D network based on local information [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 boundary nodes, followed by a refinement algorithms, 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 1-hop neighborhood only. This quality is highly desired to enable fast and low-cost boundary detection.

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 3-D 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 3-D boundaries. We adopt the method proposed in [30] that produces a planar subgraph in 2-D, and extend it to 3-D surfaces to achieve complete triangulation without degenerated edges. The algorithm is localized and based on connectivity only.

We further extend the proposed methods for dynamic sensor networks where sensors are mobile. We propose an efficient updating scheme to reduces the number of nodes running UBF without degrading the performance of the online boundary detection and to maintain the surface triangle mesh after it is constructed initially, with high efficiency in terms of time and energy.

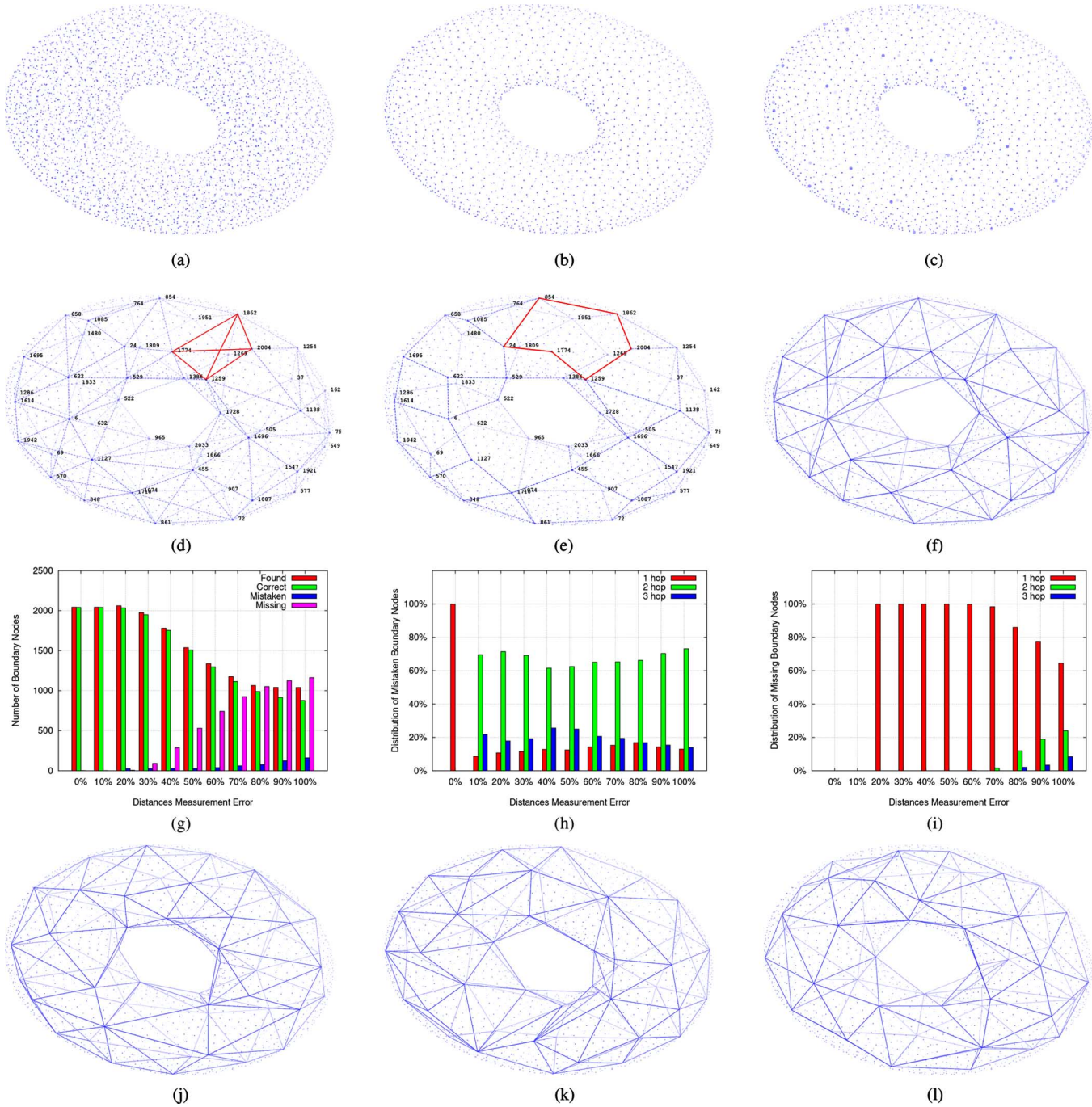


Fig. 1. Illustration of the proposed boundary detection algorithm (based on a 3-D wireless network of 4210 nodes with an average nodal degree of 18.8). (a) 3-D network. (b) Boundary nodes. (c) Landmarks. (d) CDG. (e) CDM. (f) Triangular mesh. (g) Algorithm efficiency. (h) Mistaken distribution. (i) Missing distribution. (j) 20% distance measurement errors. (k) 30% distance measurement errors. (l) 40% distance measurement errors.

The rest of this paper is organized as follows. Sections II and III introduce our proposed algorithms for boundary node identification and boundary surface updating, respectively. Section IV discusses online boundary detection and triangle mesh updating in mobile wireless sensor networks. Section V presents simulation results. Finally, Section VI concludes the paper.

II. Boundary Node Identification

The proposed boundary node identification algorithm involves two phases. The first phase is the Unit Ball Fitting, which aims to discover a set of boundary nodes. The second phase is Isolated Fragment Filtering, which removes isolated nodes that are misinterpreted as boundary nodes in Phase 1.

A. Phase 1: UBF

We present the UBF algorithm in this section. The related definitions, theories, and algorithm description are elaborated sequentially.

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

Definition 1: The nodal radio transmission range is assumed a constant. Without loss of generality, we normalize it to be 1.

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

Definition 3: A well-connected network is a network where: 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 $\mathrm{Max}(1, d_{ij})$, where d_{ij} denotes the distance between Nodes i and j.

We consider well-connected networks only in this work because the isolated nodes and degenerated line segments are swingable, causing ambiguity in boundary definition and detection.

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. 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 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.

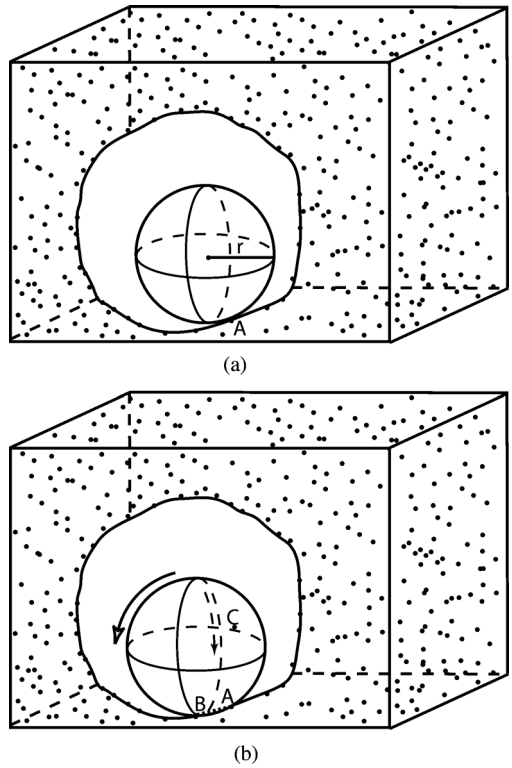


Fig. 2. Principles for UBF. (a) Empty unit ball touching Node A. (b) Ball rotation.

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.

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 $\Theta(\rho^3)$.

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

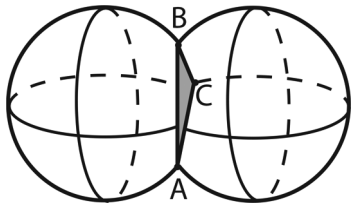


Fig. 3. Up to two unit balls determined by Node A and two of its neighbors.

Algorithm 1: Unit Ball Fitting (UBF) Algorithm

```
Output: Boundary(i);
1  Boundary(i) = FALSE;
2  Establish a local coordinates system;
3  Ω_i = {[j,(x_j,y_j,z_j)] | j ∈ N(i)};
4  for j,k ∈ Ω_i and j ≠ k do
5      Find the unit ball(s) determined by Nodes i,j,k;
6      if a unit ball is empty then
7          Boundary(i) = TRUE;
8          Break;
9      end
10 end
```

to judge if any one of them is empty. To this end, we propose the UBF algorithm as outlined in Algorithm 1 and elaborated as follows.

The proposed UBF algorithm largely follows the discussions in Section II-A.2. The sole difference is that each node considers its 1-hop neighbors only to realize a truly localized algorithm. It consists of the following three steps and outputs a boolean value $\mathrm{Boundary}(i)$ indicating if Node i is on a boundary or not.

I) *Local coordinates establishment (Lines 2 and 3)*: If all nodes have known their coordinates, this step can be skipped. Otherwise, each node employs a 3-D embedding algorithm to establish a local coordinates system. More specifically, Node i collects the distances between all pairs of nodes within 1 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) [31]. The measured distances are inaccurate in general, and the errors will be discussed in Section V. Based on the pairwise distances, multiple schemes [32]–[36] are available to create a local coordinates system for Node i and its neighbors. Among them, [36] 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 includes Node i itself and its 1-hop neighbors.

II) *Unit ball identification (Lines 4 and 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, (1) may yield no solution, or one solution, or two solutions for (x, y, z).

III) *Empty unit ball check (Lines 6–9):* If there is no solution, i.e., Nodes i, j, and k do not determine a unit ball, no action is taken. Otherwise, for each solution of (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 nonboundary node.

B. Phase 2: 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 δ''. Any fragment that consists of less than δ'' nodes is not considered as a boundary. To this end, each boundary node simply initiates a local flooding packet with a time-to-live (TTL) of T, which will be forwarded by other boundary nodes but not nonboundary nodes. By counting the number of such flooding packets received, a boundary node learns the size of its fragment. If less than δ'' flooding packets are received, the node deems itself a nonboundary node. Appropriate δ'' 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 $\delta'' = 20$ 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 a path does not exist, and their connection must go through at least one nonboundary 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 outlined above, a node determines whether it is on a boundary based on local information only. Its performance depends on the accuracy of distance measurement. As will be discussed in Section V, our simulations demonstrate that the proposed algorithms are effective, able to identify almost all boundary nodes with low missing and mistaken rates, when the distance errors are moderate [as shown in Fig. 1(g)]. Under high measurement errors, the mistaken and missing rates naturally increase. However, the mistakenly identified boundary nodes are all close to the true boundary, mostly within 1 or 2 hops [see Fig. 1(h)]. At the same time, the missed boundary nodes are uniformly scattered. Over 95% of such missed boundary nodes can find at least one correctly identified boundary node within their 1-hop neighborhood [as illustrated in Fig. 1(i)]. Therefore, the identified boundary nodes well represent the network boundary shapes. The complexity of the algorithm is dominated by Phase 1, i.e., UBF. Therefore, the overall computing complexity for boundary node detection is $\Theta(\rho^3)$. While the nodal density ρ may vary arbitrarily in theory, it is often bounded by a small constant in practical sensor network settings, where the radio range is chosen properly by individual nodes to maintain the desired nodal degree. Numeric results of computing complexity will be presented in Section V.

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 2-D graphic tools on 3-D surfaces.

In this research, we implement an algorithm that constructs locally planarized triangular meshes on the identified 3-D boundaries. We adopt the method proposed in [30] that can produce a 2-D planar subgraph (which, however, is not a triangular mesh) and extend it to 3-D surfaces to achieve complete triangulation without degenerated edges. The algorithm is localized and based on connectivity only. It consists of the following five steps.

I) *Landmark selection*: The boundary nodes employ a distributed algorithm (e.g., [37]) 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 3–5 in our implementation. A nonlandmark 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 nonlandmark 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 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 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 CDM. It is proven that CDM is a planar graph [30].

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 nonconnected 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. 4(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. 4(b), where no edge has more than two faces. Note that the polygon $ACBE$ is not a face on the surface. Until now, we arrive at a planar triangular mesh for each 3-D boundary surface, as illustrated in Fig. 1(f).

The above algorithm ensures to form a closed triangular mesh surface for each boundary. The established triangular mesh is a

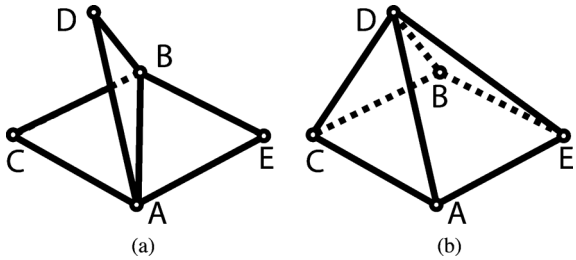


Fig. 4. Illustration of edge flip. Edge AB has three faces before edge flip. It is removed and replaced by Edges CD and DE. (a) Before edge flip. (b) After edge flip.

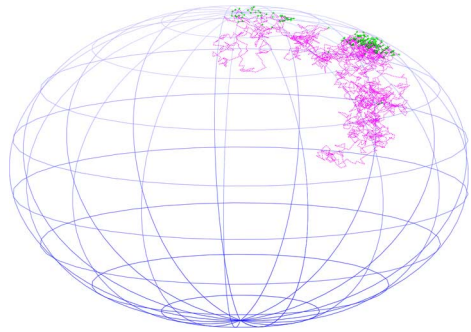


Fig. 5. Example of the random mobility model. The node is highlighted by green crosses, when it moves to the boundary.

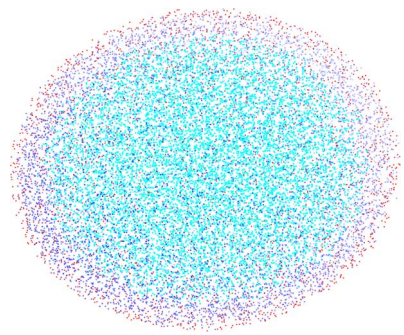


Fig. 6. Three layers of nodes in online boundary detection. Red nodes are the boundary nodes $B(t-1)$, blue nodes are the candidates nodes $BC(t)$, and the rest of the nodes with aqua color will not be involved in the UBF algorithm.

locally planarized 2-manifold, although the whole 3-D 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 3-D 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 Section II-C, 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, Fig. 1(j)–(l) shows results under 20%, 30%, and 40% distance measurement errors, respectively. Fig. 1(j)–(l) exhibits similar triangular mesh as Fig. 1(j), which is free of distance measurement errors.

IV. Boundary Detection in Dynamic Wireless Sensor Networks

In many application scenarios, the network topology changes over time due to environment dynamics (such as, water flow, wind, and animal movement) or the evolvement of the network itself (as links break or nodes run out of batteries). Topology dynamics often cause the change of network boundary, calling for an effective online boundary detection algorithm, with low overhead and high energy efficiency.

In this section, we will introduce effective algorithms that not only identify boundary nodes, but also maintain a closed triangle mesh surface in dynamic wireless sensor networks.

A. Online Boundary Nodes Detection

In this work, we consider a general random mobility model, where a node moves for a distance of d that is (less than transmission range R) to any direction in a time unit. Note that we do not consider any special radio model constraints here, except the maximum radio transmission range R. The sensors' mobility is constrained by their container (e.g., seabed and shore). Fig. 5 illustrates how a node moves according to the above mobility model. As can be seen, sometimes it moves to the surface, becoming a boundary node (highlighted as green crosses), and sometimes it moves inside.

The UBF algorithm is localized only involving 1-hop neighbor nodes. Although sensors are mobile, the current boundary nodes give valuable clues to find the boundary nodes in the near future. More specifically, even though some old boundary nodes might have moved inside and not be boundary nodes anymore, they should still be located near the boundary and thus help to narrow down the boundary candidates. As a matter of fact, the previous boundary nodes and their neighbor nodes are good candidates for identifying the new boundary nodes. The UBF algorithm can be applied only to these candidate nodes instead of all nodes in the network in order to reduce energy consumption. The online boundary detection algorithm is outlined as follows.

Since there is no clue about the boundary nodes at the beginning, we have no choice but to let all sensor nodes to run the UBF algorithm to identify the boundary nodes $B(0)$; see Lines 1–6 in Algorithm 2. After that, in each iteration t, we can always find a Boundary Candidate $BC(t)$, which includes the previous boundary nodes set $B(t-1)$ and their neighbors $N(B(t-1))$, as shown in Lines 8–11. New boundary nodes $B(t)$ will be found by applying the UBF algorithm to the candidate set $BC(t)$ as shown in Lines 12–16. Fig. 6 shows the distribution of $B(t-1)$, $BC(t)$, and the rest of the nodes.

Algorithm 2: On-line Boundary Nodes Detection Algorithm

```
for each node i in network N do
    Boundary(i) = UBF(i) ;
    if Boundary(i) == TRUE then
        B(0) = B(0) ∪ i ;
    end
end
for each iteration t, t > 0 do
    BC(t) = B(t − 1) ;
    for each node i ∈ B(t − 1) do
        BC(t) = BC(t) ∪ N(i) ;
    end
    for each node i ∈ BC(t) do
        Boundary(i) = UBF(i) ;
        B(t) = B(t) ∪ i ;
    end
end
```

B. Online Triangle Mesh Updating

While we can simply reconstruct the triangle mesh based on the newly identified boundary nodes, a more effective approach is to maintain the triangle mesh constructed previously and update it according to the new boundary. We observe that the Voronoi cells are more stable than boundary nodes, and the triangle mesh is only determined by the Voronoi cells, more specifically the landmarks of the cells. How to update the landmarks forming the triangle mesh is the key to solve this problem efficiently.

However, the selection of new landmark is not trivial because new triangle mesh must satisfy two properties. First, all of the landmarks must be boundary nodes. Second, there are no crossing edges between landmarks. Some landmarks in the previous round might not be boundary nodes in this round due to mobility, and thus are not eligible for new landmarks. Therefore, we have to find a new boundary node as a new landmark to replace the old one, if the old landmark is not on the boundary any more. Picking the boundary node closest to the old landmark is a straightforward way to fulfill the mission.

It is much more challenging to preserve the second property, i.e., the planarized mesh. Simply replacing an old landmark with the closest boundary node does not ensure planar mesh. However, we notice that if we can keep every landmark at the center of the polygon formed by its adjacent landmarks and edges between them (see Fig. 7), this property often holds. A landmark can build a set of shortest paths from its neighbor landmarks to itself. The length difference between these paths gives us an idea about the landmark's position. If the landmark is located in the center of the polygon, the length difference of paths should be very small. A better estimation can be made if the shortest paths between a landmark to surrounding landmarks edges (e.g., middle node of the edges) are taken into account as well. After the triangle mesh is updated, the method introduced in Section III is employed to test if it is planarized.

Fig. 7 illustrates an example of the adjustment of landmark v (marked as a black circle). Its neighbors, $N(v)$, which are also boundary nodes, are marked as green squares. The adjacent landmarks $L(v)$ of v are marked as red dots. Every node $i \in N(v)$ will try to established shortest paths $E(i, L(v))$ to these

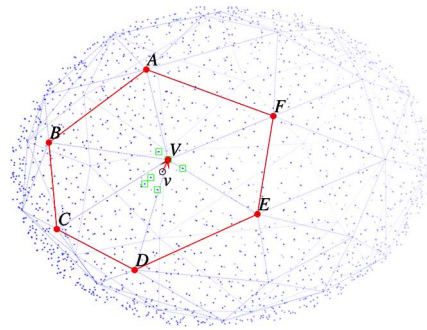


Fig. 7. Example of landmark adjustment. A new boundary node V replaces the previous landmark node v because $\mathrm{Max}(E(V, L(v))) - \mathrm{Min}(E(V, L(v)))$ is the smallest among the neighbor boundary nodes of v.

adjacent landmarks. Among them, V (marked as a red dot) is the best choice to replace landmark v as a new landmark because the difference between the longest path $\mathrm{Max}(E(V, L(v)))$ and the shortest path $\mathrm{Min}(E(V, L(v)))$ is the smallest.

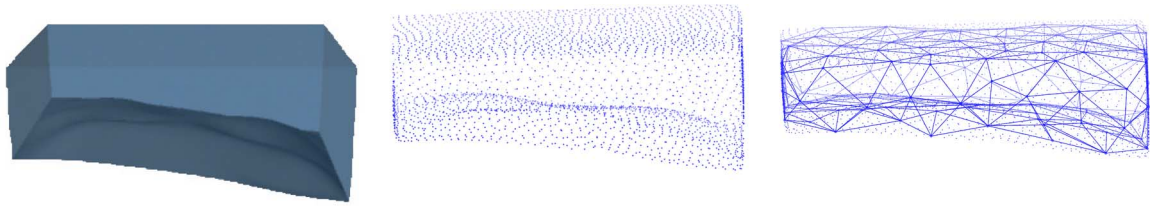
Note that the online algorithms for boundary node detection and triangular mesh construction discussed above are developed to reduce (on a best-effort basis) the computation and communication overhead for boundary detection in dynamic wireless sensor networks. They naturally become less effective when the network experiences more dramatic changes that require most nodes to be involved in the detection of boundaries and the reconstruction of triangle meshes.

V. Simulations and Experiments

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

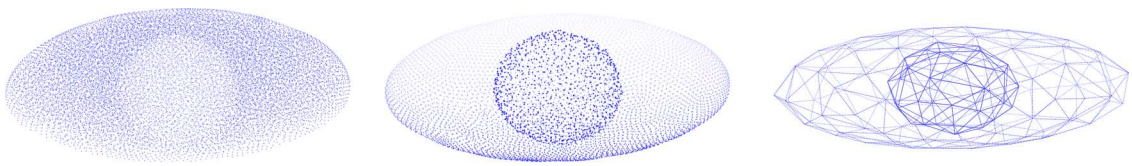
A. Simulation Setup

The 3-D networks used in our simulations are constructed by using a set of 3-D graphic tools (including TetGen [38]). First, a 3-D model is developed to represent a given network scenario (e.g., an underwater network, a 3-D network in space, and general 3-D networks with arbitrary shapes of our interest). A set of nodes are randomly uniformly distributed on the surface of the 3-D model. They are marked as boundary nodes, serving as ground truth to evaluate our algorithm. A cloud of nodes is then deployed inside the 3-D 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.



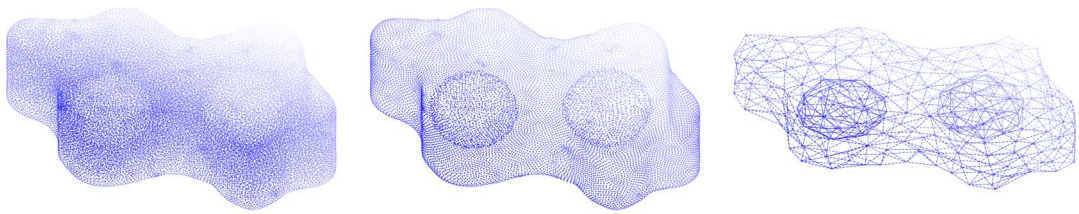
(a) (b) (c)

Fig. 8. Example of underwater network. (a) Network model. In contrast to Figs. 9(a), 10(a), 11(a), and 12(a), where the network model shows a set of wireless nodes deployed, the network model in this figure gives the actual 3-D model for better visualization. (b) Boundary nodes. (c) Triangular mesh.



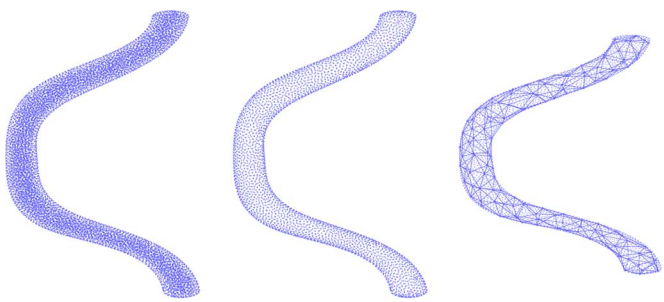
(a) (b) (c)

Fig. 9. Example of a 3-D space network with an internal hole. (a) Network model. (b) Boundary nodes. (c) Triangular mesh.



(a) (b) (c)

Fig. 10. Example of a 3-D space network with two internal holes. (a) Network model. (b) Boundary nodes. (c) Triangular mesh.



(a) (b) (c)

Fig. 11. Example of a 3-D network in a bended pipe. (a) Network model. (b) Boundary nodes. (c) Triangular mesh.

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 multidimensional scaling [36] 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. 8–12. Figs. 8 illustrates an underwater network, where nodes are distributed from the surface to the bottom of the ocean. As shown in Fig. 8(b) and (c), our algorithms effectively identify the boundaries of both smooth water surface and the bumpy bottom. Figs. 9 and 10 depict a 3-D network deployed in the space (e.g., for chemical dispersion sampling in 3-D 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. 11 and 12 show 3-D networks deployed in a bended pipe and a sphere, respectively. Fig. 13 demonstrates a nonuniform distributed 3-D network. The node density is increasing from top to bottom, and node degree ranges from 7 up to 92. As can be seen, boundary nodes are accurately identified, and the triangular mesh surfaces are well constructed in the all networks.

We have also simulated nodal mobility and run the online boundary detection and triangle mesh updating algorithms introduced in Section IV. Two examples are given in Figs. 14 and 15. As can be seen, landmarks change overtime, but the triangle mesh remains nicely distributed. They clearly demonstrate that the online boundary detection algorithm and boundary mesh updating algorithm are adaptive to the dynamics of network. Only 43% and 57% of nodes on average involve UBF in Figs. 14 and 15, respectively. All the boundary nodes are successfully identified. Larger-scale networks will benefit more from our algorithm because only nodes within one hop layer from the surface run the algorithms.

Fig. 16 illustrates the performance statistics obtained from our simulations. The results are based on over 10 000 sample

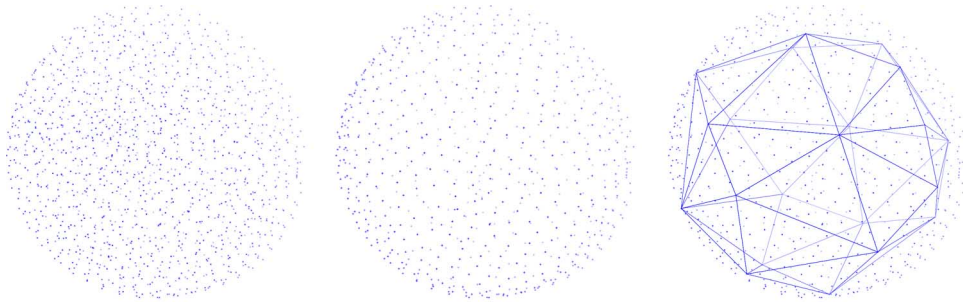


Fig. 12. Example of a 3-D network in a sphere. (a) Network model. (b) Boundary nodes. (c) Triangular mesh.

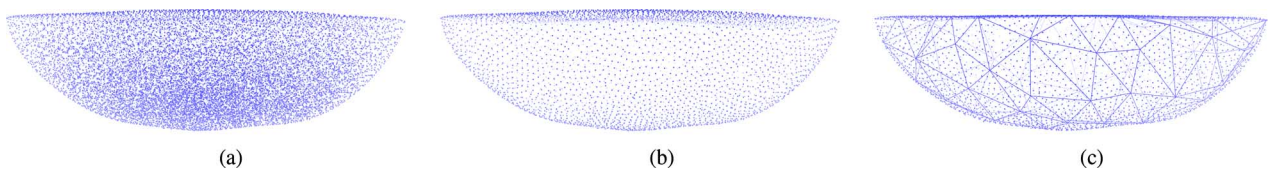


Fig. 13. Example of a nonuniform distributed 3-D network in a lake. Node degree is increasing from top to bottom and ranges from 7 to 92. (a) Network model. (b) Boundary nodes. (c) Triangular mesh.

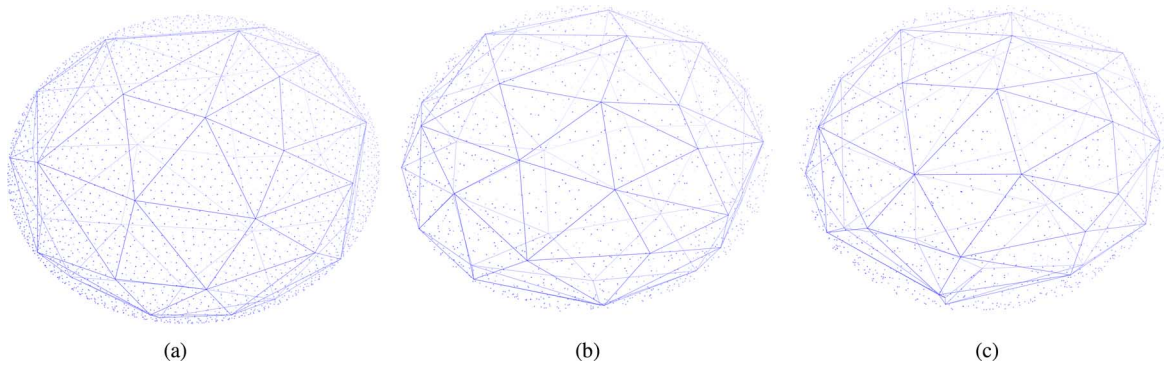


Fig. 14. Example of a dynamic 3-D sensor network in atmosphere. Only about 43% of nodes involved UBF algorithm in each iteration. (a) Initial boundary nodes and triangle mesh. (b) After 20 iterations. (c) After 40 iterations.

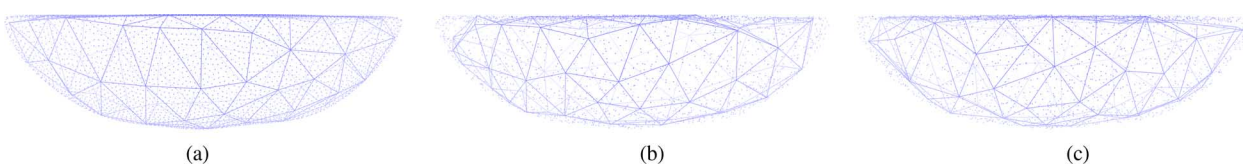


Fig. 15. Example of a dynamic 3-D sensor network deployment in lake. Only about 57% of nodes involved UBF algorithm in each iteration. (a) Initial boundary nodes and triangle mesh. (b) After 20 iterations. (c) After 40 iterations.

boundary nodes. As can be seen in Fig. 16(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 a 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 Section II, such missing and mistaken boundary nodes do not seriously affect our boundary identification because they are well distributed. For example, Fig. 16(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. 16(b), such distance is always less than 3 hops, with a majority of them in 1 (over 60%) and 2 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. 16(c). It is observed

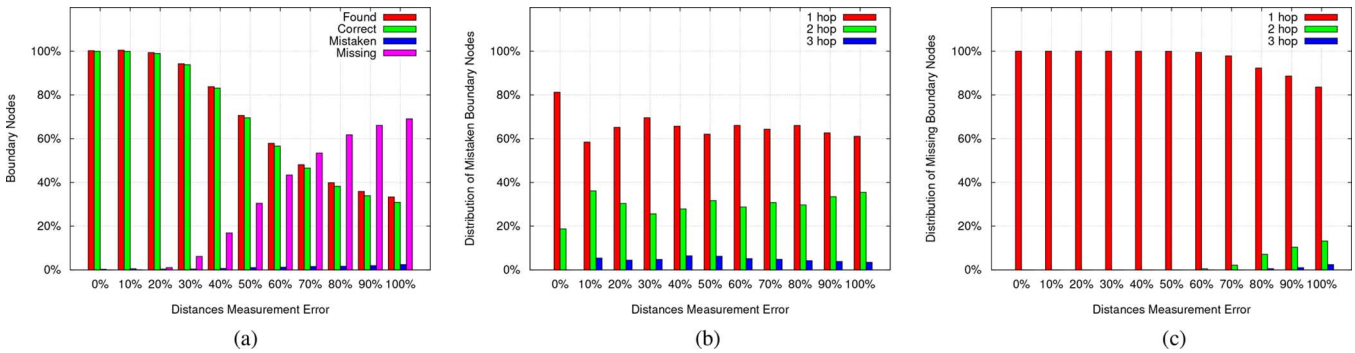


Fig. 16. Performance statistics. (a) Algorithm efficiency. (b) Mistaken distribution. (c) Missing distribution.

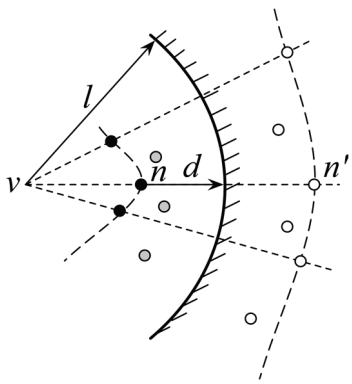


Fig. 17. Spherical mirror mapping used in UNFOLD.

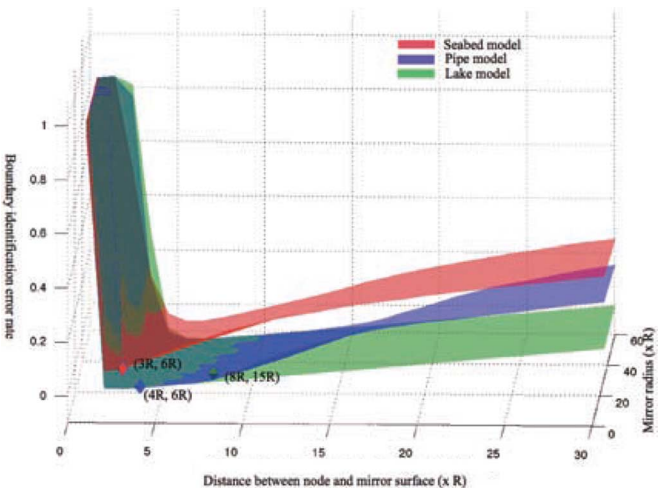


Fig. 18. UNFOLD performance with different parameters for three models.

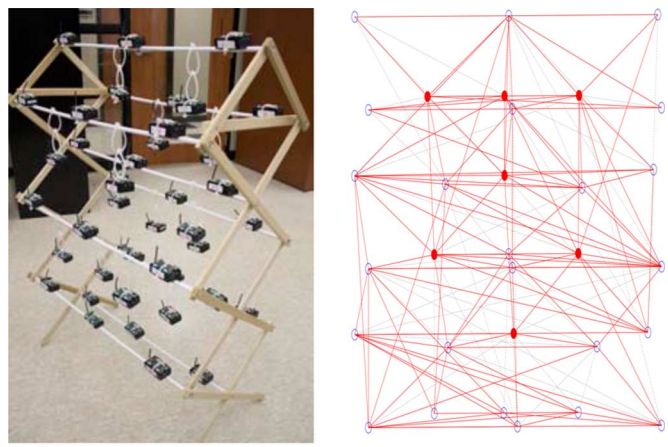


Fig. 19. Experiment setting. (a) Small 3-D sensor network with 44 sensor notes. (b) Snapshot of network topology. Dashed lines denote asymmetric links. Blue circles are identified boundary nodes.

that almost 100% of the missing boundary nodes are within 1-hop neighborhood of correctly identified boundary nodes. In other words, the missing boundary nodes are uniformly distributed on the boundary surfaces (without forming "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.

Among the related work discussed in Section I, the UNFOLD algorithm [16] is most relevant. Therefore, we have quantitatively studied it for comparison. The basic idea behind UNFOLD is to use a spherical mirror to map a Node n and its 1-hop neighbors to a convex image. Since the image nodes are convex, a simple boundary detection algorithm can be applied to identify the boundary nodes. It has the same complexity as fit-ball algorithm, however it relies on two important parameters in mapping that are model-dependent, i.e., the spherical mirror radius l and the distance between the mirror surface to a Node d. The two parameters together determine the view point v and the image position of Node n and its 1-hop neighbors as illustrated in Fig. 17. The viewpoint v is also the center of the spherical mirror.

We applied UNFOLD algorithm to three different models with different parameters, l and d, ranging from $2R$ to $30R$, where R is the maximum radio range of nodes. As shown in Fig. 18, boundary identification errors (mistaken rate plus missing rate) vary widely when the parameters change. There are optimal parameters for each individual model to yield the lowest error, but they vary from one model to another. In a sharp contrast, our proposed algorithm employs a fixed parameter r (i.e., the radius of the ball) for all models to yield results as good as or better than the UNFOLD optimal results, as shown in Table I.

C. Experiments

To further evaluate the proposed algorithm, we implement it in Tiny OS and run on 44 Crossbow sensor nodes. All nodes are

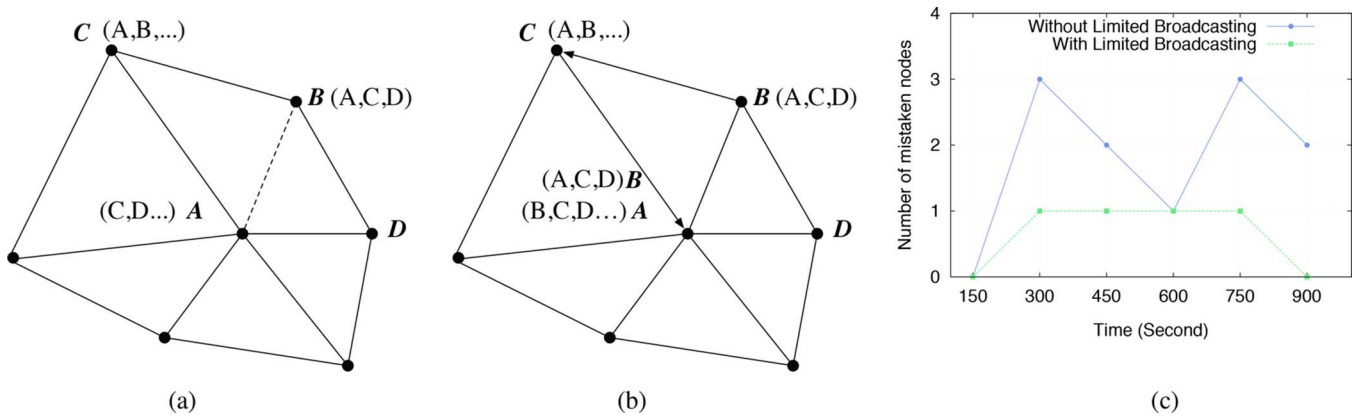


Fig. 20. Asymmetric links problem and solution. (a) Asymmetric Link AB. (b) Limited broadcast. (c) Number of mistaken nodes comparison between with and without limited broadcasting.

TABLE I
BOUNDARY IDENTIFICATION ERRORS COMPARISON

Models	UNFOLD optimal results	UBF results
Seabed	6.6%	6.0%
Pipe	0.02%	0%
Lake	0%	0%

configured to use close to minimum radio transmission power (Level 2), with a communication range between 25–25 cm. They are tied onto a rack to form a small 3-D sensor network shown in Fig. 19(a).

Every sensor periodically broadcasts a beacon message with its node ID. Based on the beacon message, a node builds a neighbor list with the RSSI of corresponding links. RSSI is used to estimate the length of links by lookup up an RSSI-distance table established by experimental training data. The preliminary test shows that, under low transmission power, such estimation has an error rate about 20%. With neighbor nodes and their estimated distance, every node can determine itself as a boundary node or not based on the proposed algorithm individually.

The asymmetric links are common in real 3-D wireless sensor networks [see the dashed line in Fig. 19(b) for example]. They often cause false positive boundary nodes. For instance, Link AB is asymmetric in Fig. 20(a) because Node A cannot receive messages from Node B, but Node B can receive Node A's messages. In Node A's point of view, there is no Node B at all. Therefore, it will mark itself as a boundary node. In this work, we adopt a simple approach based on limited broadcasting to solve the problem. For example, if Node B has established its neighbor list, it will broadcast a message containing its neighbor list and set TTL of the message as three (neighbor nodes number). A node will discard this message if the TTL turns to zero or there is no common node between its neighbor lists and the sender's list. Finally, Node A will receive this message from its neighbors, e.g., Node C, and learn that Node B should be its neighbor as well, shown in Fig. 20(b). The limited broadcasting helps to reduce the false positive as shown in Fig. 20(c).

VI. CONCLUSION

We have proposed distributed and localized algorithms for precise boundary detection in 3-D wireless networks. Our objectives have been twofold. First, we have aimed to identify the nodes on the boundaries of a 3-D 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 3-D surfaces, such as embedding, localization, partition, and greedy routing among many others. To achieve the first objective, we have proposed a Unit Ball Fitting algorithm that discovers a set of potential boundary nodes, followed by a refinement algorithm, named Isolated Fragment Filtering, 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 3-D boundary. Our proposed scheme is localized, requiring information within 1-hop neighborhood only. We have further extended the schemes for online boundary detection in mobile sensor networks aiming to achieve low overhead. Our simulation and experimental results have shown that the proposed algorithms can effectively identify boundary nodes and surfaces, even under high measurement errors.

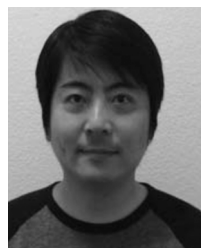
REFERENCES

[1] R. Nowak and U. Mitra, "Boundary estimation in sensor networks: Theory and methods," in *Proc. IPSN*, 2003, pp. 80–95.
[2] M. Ding and X. Cheng, "Robust event boundary detection and event tracking in sensor networks—A mixture model based approach," in *Proc. IEEE INFOCOM*, 2009, pp. 2991–2995.
[3] K. Chintalapudi and R. Govindan, "Localized edge detection in sensor fields," in *Proc. 1st IEEE Int. Workshop Sensor Netw. Protocols Appl.*, 2003, pp. 59–70.
[4] S. Duttagupta, K. Ramamritham, and P. Ramanathan, "Distributed boundary estimation using sensor networks," in *Proc. IEEE MASS*, 2006, pp. 316–325.
[5] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks," in *Proc. IEEE INFOCOM*, 2005, pp. 902–913.
[6] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted sensor deployment," in *Proc. IEEE INFOCOM*, 2004, pp. 2469–2479.
[7] A. Ghosh, "Estimating coverage holes and enhancing coverage in mixed sensor networks," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, 2004, pp. 68–76.
[8] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing routing holes in sensor networks," in *Proc. IEEE INFOCOM*, 2004, pp. 2458–2468.

[9] C. Zhang, Y. Zhang, and Y. Fang, "Localized algorithms for coverage boundary detection in wireless sensor networks," *Wireless Netw.*, 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. IPSN*, 2005, pp. 254–260.
[11] S. Funke, "Topological hole detection in wireless sensor networks and its applications," in *Proc. DIALM-POMC*, 2005, pp. 44–53.
[12] A. Kroller, S. P. Fekete, D. Pfisterer, and S. Fischer, "Deterministic boundary recognition and topology extraction for large sensor networks," in *Proc. ACM-SIAM SODA*, 2006, pp. 1000–1009.
[13] Y. Wang, J. Gao, and J. S. B. Mitchell, "Boundary recognition in sensor networks by topological methods," in *Proc. ACM/IEEE MobiCom*, 2006, pp. 122–133.
[14] W. Cheng *et al.*, "Underwater localization in sparse 3D acoustic sensor networks," in *Proc. IEEE INFOCOM*, 2008, pp. 798–806.
[15] H. Zhou, S. Xia, M. Jin, and H. Wu, "Localized algorithm for precise boundary detection in 3D wireless networks," in *Proc. ICDCS*, 2010, pp. 744–753.
[16] F. Li, J. Luo, C. Zhang, S. Xin, and Y. He, "UNFOLD: Uniform fast on-line boundary detection for dynamic 3D wireless sensor networks," in *Proc. MobiHoc*, 2011, pp. 141–152.
[17] H. Jiang, S. Zhang, G. Tan, and C. Wang, "CABET: Connectivity-based boundary extraction of large-scale 3D sensor networks," in *Proc. IEEE INFOCOM*, 2011, pp. 784–792.
[18] H. Zhou, H. Wu, and M. Jin, "A robust boundary detection algorithm based on connectivity only for 3D wireless sensor networks," in *Proc. IEEE INFOCOM*, 2012, pp. 1602–1610.
[19] C. Liu and J. Wu, "Efficient geometric routing in three dimensional ad hoc networks," in *Proc. IEEE INFOCOM*, 2009, pp. 2751–2755.
[20] T. F. G. Kao and J. Opatmy, "Position-based routing on 3D geometric graphs in mobile ad hoc networks," in *Proc. 17th Can. Conf. Comput. Geometry*, 2005, pp. 88–91.
[21] J. Opatrny, A. Abdallah, and T. Fevens, "Randomized 3D position-based routing algorithms for Ad-hoc networks," in *Proc. 3rd Annu. Int. Conf. Mobile Ubiquitous Syst., Netw. Services*, 2006, pp. 1–8.
[22] R. Flury and R. Wattenhofer, "Randomized 3D geographic routing," in *Proc. IEEE INFOCOM*, 2008, pp. 834–842.
[23] F. Li, S. Chen, Y. Wang, and J. Chen, "Load balancing routing in three dimensional wireless networks," in *Proc. IEEE ICC*, 2008, pp. 3073–3077.
[24] D. Pompili, T. Melodia, and I. F. Akyildiz, "Routing algorithms for delay-insensitive and delay-sensitive applications in underwater sensor networks," in *Proc. ACM/IEEE MobiCom*, 2006, pp. 298–309.
[25] X. Cheng, A. Thaeler, G. Xue, and D. Chen, "TPS: A time-based positioning scheme for outdoor wireless sensor networks," in *Proc. IEEE INFOCOM*, 2004, vol. 4, pp. 2685–2696.
[26] X. Bai, C. Zhang, D. Xuan, J. Teng, and W. Jia, "Low-connectivity and full-coverage three dimensional networks," in *Proc. ACM MobiHoc*, 2009, pp. 145–154.
[27] X. Bai, C. Zhang, D. Xuan, and W. Jia, "Full-coverage and K-connectivity $(\mathrm{K} = 14, 6)$ three dimensional networks," in *Proc. IEEE INFOCOM*, 2009, pp. 388–396.
[28] J. Allred *et al.*, "SensorFlock: An airborne wireless sensor network of micro-air vehicles," in *Proc. SenSys*, 2007, pp. 117–129.
[29] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou, "The challenges of building mobile underwater wireless networks for aquatic applications," *IEEE Netw.*, vol. 20, no. 3, pp. 12–18, May–Jun. 2006.
[30] S. Funke and N. Milosavljevi, "How much geometry hides in connectivity?—Part II," in *Proc. ACM-SIAM SODA*, 2007, pp. 958–967.
[31] Z. Zhong and T. He, "MSP: Multi-sequence positioning of wireless sensor nodes," in *Proc. SenSys*, 2007, pp. 15–28.
[32] H. Wu, C. Wang, and N.-F. Tzeng, "Novel self-configurable positioning technique for multi-hop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 609–621, Jun. 2005.
[33] G. Giorgetti, S. Gupta, and G. Manes, "Wireless localization using self-organizing maps," in *Proc. IPSN*, 2007, pp. 293–302.
[34] L. Li and T. Kunz, "Localization applying an efficient neural network mapping," in *Proc. 1st Int. Conf. Auton. Comput. Commun. Syst.*, 2007, pp. 1–9.
[35] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," in *Proc. ACM MobiHoc*, 2003, pp. 201–212.
[36] Y. Shang and W. Ruml, "Improved MDS-based localization," in *Proc. IEEE INFOCOM*, 2004, pp. 2640–2651.
[37] Q. Fang, J. Gao, L. J. Guibas, V. Silva, and L. Zhang, "GLIDER: Gradient landmark-based distributed routing for sensor networks," in *Proc. IEEE INFOCOM*, 2005, pp. 339–350.
[38] Fraunhofer-Institut FOKUS, Berlin, Germany, "berliOS," [Online]. Available: http://tetgen.berlios.de/

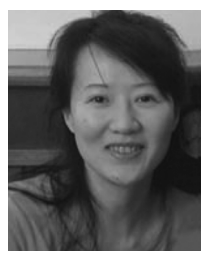
Hongyu Zhou (S'10) received the B.S. and M.S. degrees from Sichuan University, Chengdu, China, in 2006 and 2009, respectively, and the Ph.D. degree in computer science from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 2013, all in computer science.

He is working with Epic, Madison, WI, USA.



Su Xia (S'12–M'13) received the B.S. and M.S. degrees in radio engineering and computer science from Southeast University, Nanjing, China, in 1998 and 2001, respectively, and the Ph.D. degree in computer science from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 2012.

He is currently working with the Internet of Things (IoT) Group, Cisco Systems, Inc., San Jose, CA, USA.



Miao Jin received the B.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2000, and the M.S. and Ph.D. degrees from the State University of New York at Stony Brook, Stony Brook, NY, USA, in 2006 and 2008, respectively, all in computer science.

She is an Associate Professor with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA, USA. Her research results have been used as cover images of mathematics books and licensed by the Siemens Healthcare Sector of Germany for virtual colonoscopy. Her research interests are computational geometric and topological algorithms with applications in wireless sensor networks, computer graphics, computer vision, geometric modeling, and medical imaging.

Prof. Jin received the NSF CAREER Award in 2011.

Hongyi Wu (M'02) received the B.S. degree in scientific instruments from Zhejiang University, Hangzhou, China, in 1996, and the M.S. degree in electrical engineering and Ph.D. degree in computer science from the State University of New York (SUNY) at Buffalo, Buffalo, NY, USA, in 2000 and 2002, respectively.

Since then, he has been with the Center for Advanced Computer Studies (CACS), University of Louisiana at Lafayette (UL Lafayette), Lafayette, LA, USA, where he is now a Professor and holds the Alfred and Helen Lamson Endowed Professorship in Computer Science. His research spans delay-tolerant networks, radio frequency identification (RFID) systems, wireless sensor networks, and integrated heterogeneous wireless systems.

Prof. Wu received the NSF CAREER Award in 2004 and the UL Lafayette Distinguished Professor Award in 2011.