A Robust Boundary Detection Algorithm Based on Connectivity Only for 3D Wireless Sensor Networks

Hongyu Zhou, Hongyi Wu, and Miao Jin

***Abstract*—In this work we develop a distributed boundary detection algorithm, dubbed *Coconut*, for 3D wireless sensor networks. It first constructs a tetrahedral structure to delineate the approximate geometry of the 3D sensor network, producing a set of "sealed" triangular boundary surfaces for separating non-boundary 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. The proposed Coconut algorithm offers several salient features. First, it requires connectivity information only, with no need for localization or distance measurement. Second, it does not rely on particular communication models, but only assumes a constant maximum transmission range, which is generally known in practical wireless sensor networks. Third, it is robust to sensor distribution, effectively identifying boundaries in both uniformly and non-uniformly distributed sensor networks. We prove the correctness of the algorithm and quantitatively demonstrate its effectiveness via simulations under various network models.**

I. Introduction

A wireless sensor network is built upon a large number of low cost sensor nodes. Although a two-dimensional (2D) planar setting is assumed in most earlier studies on wireless sensor networks, there have been increasing interests in deploying sensors in three-dimensional (3D) space for such applications as underwater reconnaissance and atmospheric monitoring [1]–[12]. While the third dimension appears irrelevant to network communication and management protocols at the first glance, surprising challenges are observed in efforts to extend many 2D networking techniques to 3D space.

This work focuses on boundary detection in 3D wireless sensor networks. Boundary is a key attribute that characterizes a sensor network, providing salient information for understanding environmental data and for efficient operation of the network itself, especially in geographic exploration and monitoring tasks. Due to the lack of precise nodal deployment and the nondeterministic sensor failures and channel dynamics, many wireless sensor networks exhibit substantial randomness, with their final formations heavily dependent on underlying environment. Consequently, the boundaries are often unknown before network deployment, calling for distributed and autonomous algorithms for efficient boundary detection.

A. Challenges in Connectivity-Based 3D Boundary Detection

Let's first look back upon the development of boundary detection algorithms in wireless sensor networks, which offers a full-spectrum understanding of the boundary detection problem and the limitation of existing boundary detection solutions.

1) Boundary Detection in 2D Sensor Networks: The problem of boundary detection has been extensively studied in 2D wireless sensor networks, covering the detection of *event boundaries* and *network boundaries*.

Events are reported according to sensor readings. A sensor is called an event sensor if it detects the target event based on its measurement (e.g., high temperature and smoke density upon a fire). An event sensor declares itself on the event boundary if it has non-event sensors in its neighborhood. While the basic idea appears straightforward, event boundary detection is challenging, due to limited sampling density, noisy sensor readings, lossy data delivery, and low computation power of individual sensors [13], [14], calling for efficient information processing and modeling techniques to analyze sensor data, in order to estimate the boundary of events [13]–[17].

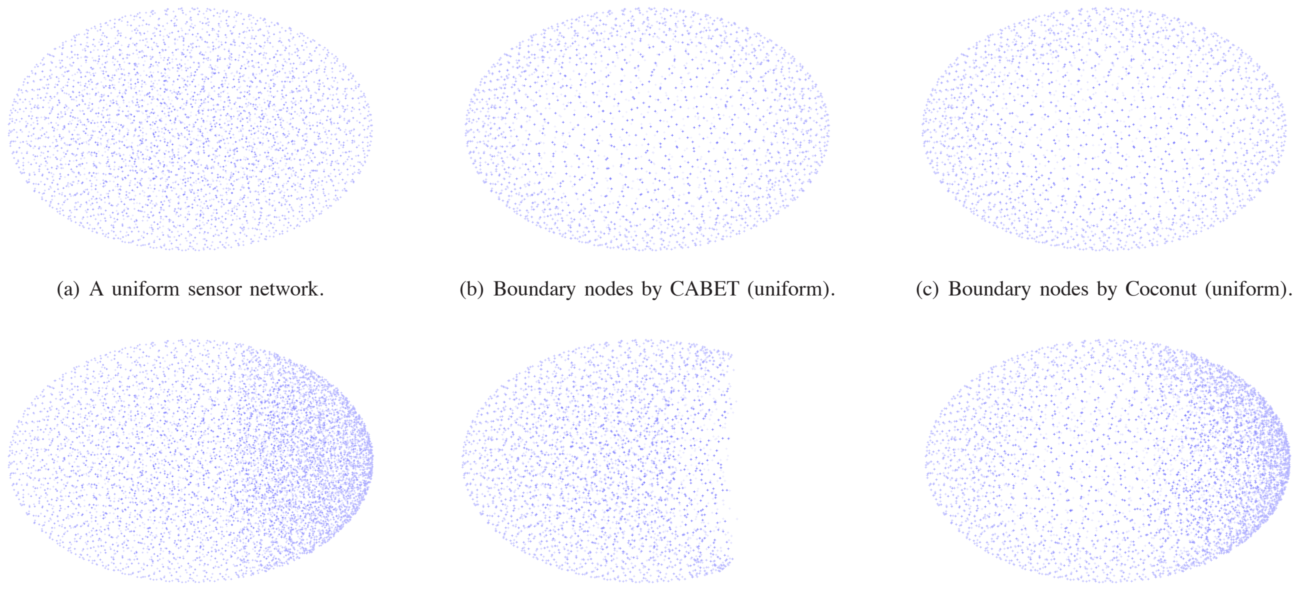
The detection of network boundary is to locate the outmost nodes in a sensor network, irrespective of sensor data or events. Without the facilitation of neighboring sensor readings, a sensor node depends on geometric or topological information to determine if it is on a boundary. The *geometry-based approaches* require the knowledge of location or distance for localized boundary detection [18], [19]. On the other hand, the *topology-based schemes* achieve location/distance-free by exploiting topological characteristics of the network [20]–[24]. This research is primarily interested in the latter.

2) Hurdles to Extending Topology-Based Schemes to 3D: Topology-based boundary detection is intrinsically challenging in 3D wireless sensor networks, because higher dimension space introduces significant complexity in searching for boundaries and many topological tools cannot be extended from 2D to 3D, rendering none of the available topology-based schemes [20]–[24] readily applicable for distributed and autonomous boundary detection in 3D sensor networks.

For example, the fundamental group persevering (FGP) transformation is adopted in [21] to produce a reduced topology graph with all holes preserved. It can effectively identify fine-grained boundaries, but the transformation and further refinement techniques are usable on 2D plane only.

The algorithm in [22] exploits the fact that, on a 2D plane with holes, the branches of a shortest path tree belong to different homotopy types, which cannot be continuously deformed from one to another. Thus two paths with distinct homotopy types are connected to form a circle around an inner hole, which is further refined to discover tight boundaries. However, similar concept no longer holds in 3D, where the shortest paths

The authors are with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, P.O. Box 44330, Lafayette, LA 70504. This work is supported in part by National Science Foundation under Award Number CNS-1018306.



(a) A uniform sensor network.

(b) Boundary nodes by CABET (uniform).

(c) Boundary nodes by Coconut (uniform).

(d) A nonuniform sensor network.

(e) Boundary nodes by CABET (nonuniform).

(f) Boundary nodes by Coconut (nonuniform).

Fig. 1. Comparison between CABET [28] and Coconut under uniform and nonuniform sensor distributions. Subfigures (a) and (d) depict a uniform and a nonuniform sensor network, respectively, where nodal density increases from left to right in the latter. CABET precisely captures the boundary of the uniform network, but misinterprets many internal sensors on the left side of the nonuniform network as boundary nodes and misses some true boundary nodes on the right side at the same time. Coconut exhibits robust performance under both uniform and nonuniform sensor distributions.

around a hole are homotopy equivalent. Similarly, the m-flower structure employed in [23] is effective in 2D only.

In [24], isosets are identified for boundary detection. An isoset consists of nodes with the same hop distance to a beacon node. The disconnection in an isoset indicates a boundary. While similar ideas can be applied in 3D, it becomes nontrivial to test disconnections in 3D isosets, and moreover the scheme does not guarantee to discover complete boundaries.

Finally, a hole detection algorithm based on homology is proposed in [20]. It is generally applicable to networks in any dimensional space, but it is a centralized approach and there exists significant challenge to decentralize its computation.

3) Boundary Detection in 3D Sensor Networks: With the emerging interests in 3D wireless sensor networks, a handful of boundary detection algorithms have been recently developed specifically for 3D settings [25]–[27]. A localized algorithm dubbed Unit Ball Fitting is proposed in [25] that enables individual nodes to test if they are on a boundary. It considers static sensor networks only. To timely track dynamic network boundaries, the UNiform Fast On-Line boundary Detection algorithm is introduced in [26]. It transforms "notched" surface into a convex one, thus reducing computational complexity in support of fast online boundary detection. Both [25] and [26] require local coordinates or distance information, i.e., belong to the category of geometry-based approaches.

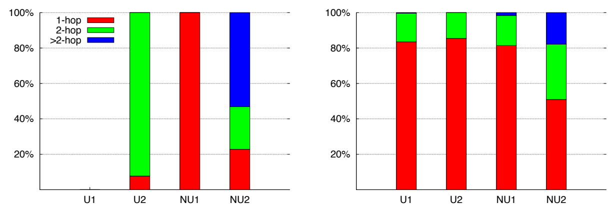
The only connectivity-based 3D boundary detection algorithm (named CABET) is discussed in [27]. It first identifies a set of boundary nodes based on the assumption that a boundary node has less neighbors than its internal counterpart. Then three types of critical boundary nodes (i.e., convex, concave and saddle nodes) are selected to depict the geometric features of the 3D 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 the sensor network is nonuniform, the above boundary node identification scheme often becomes error-prone. For example, an internal node may be located at a low-density area with a small number of neighbors only, while the boundary regions may have a high concentration of sensors. Fig. 1 illustrates the impact of nodal density on CABET. Fig. 1(a) shows a sensor network with uniformly distributed nodes, whose boundary is precisely captured by CABET (see Fig. 1(b)). Fig. 1(d) depicts a nonuniform sensor network, where nodal density linearly increases from left to right. As can be seen in Fig. 1(e), many internal sensors on the left side (with low density) are misinterpreted as boundary nodes (i.e., false-positives or mistaken boundary nodes), and at the same time some true boundary nodes on the right side (with high density) are left out (i.e., missing boundary nodes). Quantitative results are given in Table I and Fig. 2. CABET achieves excellent boundary detection under uniform sensor deployment. When the network is nonuniform, it suffers high missing and mistaken rates, where many missing and mistaken boundary nodes are far away from true boundaries.

B. Contributions of This Work

This research aims to develop a distributed topology-based boundary detection algorithm for 3D wireless sensor networks that is robust to sensor distribution. It first constructs a tetrahedral structure to delineate the approximate geometry of

TABLE I
COMPARISON OF BOUNDARY DETECTION.

	Algorithm	Correct	Missing	Mistaken
Uniform	Coconut	100%	0%	57.5%
	CABET	98.4%	1.6%	59.3%
Nonuniform	Coconut	99.4%	0.6%	124.8%
	CABET	64%	36%	105%



(a) Missing node distribution. (b) Mistaken node distribution.

Fig. 2. Distributions of hop distances from missing and mistaken boundary nodes to nearest true boundary nodes. U1 and U2: Coconut and CABET under uniform sensor deployment, respectively; NU1 and NU2: Coconut and CABET under non-uniform settings.

the 3D sensor network, consequently producing a set of coarse triangular boundary surfaces. Then, the triangular boundary surfaces are "sealed" to separate absolutely non-boundary (i.e., internal) nodes and boundary node candidates, with the former hollowed out immediately and the latter further refined to yield the final boundary nodes and fine-grained boundary surfaces. The proposed scheme is an analogy of hollowing out a coconut, and thus we name it the *Coconut* algorithm.

The proposed Coconut algorithm offers several salient features. First, as a topology-based scheme, it requires connectivity information only, with no need for localization or distance measurement that often incurs increased device cost and energy consumption. Second, it does not rely on any particular communication model (such as the unit ball graph model or quasi-unit ball graph model). It only assumes a constant maximum transmission range, which is generally known in practical wireless sensor networks. Third, it is robust to sensor distribution, effectively identifying boundaries in both uniformly and non-uniformly distributed sensor networks. We prove the correctness of the algorithm and quantitatively demonstrate its effectiveness via simulations under various network models. The rest of this paper is organized as follows: Sec. II introduces the proposed Coconut algorithm for 3D boundary detection. Sec. III presents simulation results. Finally, Sec. IV concludes the paper.

II. Proposed Coconut Algorithm

With connectivity information only, a local view is often insufficient to determine whether a node is on a boundary or not. More specifically, due to the lack of precise geometry information, local connectivity cannot differentiate boundary nodes and internal nodes in a general sensor network, as evident by above discussions on CABET [27] that results in high missing and false-positive rates under nonuniform sensor distribution. On the other hand, given the assumption of a constant maximum radio transmission range which is generally known in practical wireless sensor networks, topology indeed reflects rough geometric characteristics of a sensor network. To this end, we propose a distributed algorithm, i.e., *Coconut*, which builds a big picture based on connectivity to extract and refine boundaries of the sensor network.

The proposed Coconut algorithm follows three steps for boundary detection. First, it constructs a tetrahedral structure to approximate the geometry of the 3D sensor network and to identify coarse triangular boundary surfaces. Then, it builds sealed boundary surfaces to separate absolutely non-boundary nodes and boundary node candidates. Finally, the boundary node candidates are refined to yield the final boundary nodes and fine-grained boundary surfaces.

A. Coarse Boundary Surface Construction

It is fundamentally nontrivial to determine whether a node is on a boundary or not based on connectivity only. The first step of the algorithm aims to construct a skeletal structure to depict the geometric property of a 3D wireless sensor network and to delineate the rough network boundaries.

A trivial distributed scheme can be adopted to randomly select a subset of sensor nodes as "landmarks" such that any two landmarks must be at least k hops apart. For example, an arbitrary node is designated as the first landmark. It marks all nodes within k hops by a k-hop flooding. The remaining nodes (excluding the landmark and the marked nodes) repeat a similar process until all nodes are either landmarks or marked by landmarks. A non-landmark node is associated with the closest landmark. If it has the same hop distance to multiple landmarks, it chooses the one with the smallest ID as a tiebreaker. This process creates a set of approximate Voronoi cells in a 3D wireless sensor network (as shown in Fig. 3(a)).

Definition 1: If there exists a shortest path between two landmarks where all nodes on the path are associated with one of the two landmarks only, the two landmarks are called *neighboring landmarks* and the corresponding Voronoi cells are called *neighboring cells*.

Virtual edges are added to connect neighboring landmarks, forming a 3D Voronoi mesh structure. When k is sufficiently large (e.g., $k = 6$ in average in our implementation) and the landmarks are uniformly randomly distributed, the 3D Voronoi mesh becomes a tetrahedral mesh that consists of a set of tetrahedra with no crossing edges or faces (as shown in Fig. 3(b))[1]. Based on the tetrahedral mesh, it is straightforward to identify boundary faces and landmarks. More specifically, an internal triangular face is always shared by two neighboring tetrahedra. Thus, a boundary face can be recognized as follows.

Definition 2: A triangular face in a tetrahedral mesh is a *boundary face* if and only if it is associated with one and only one tetrahedron.

Definition 3: A landmark on a boundary face is a *boundary landmark*. Non-boundary are called *internal landmarks*.

[1]The method introduced here is a practically viable approach for constructing coarse-grained tetrahedral mesh in 3D wireless sensor networks. Formal discussions on general tetrahedral mesh is beyond the scope of this paper.

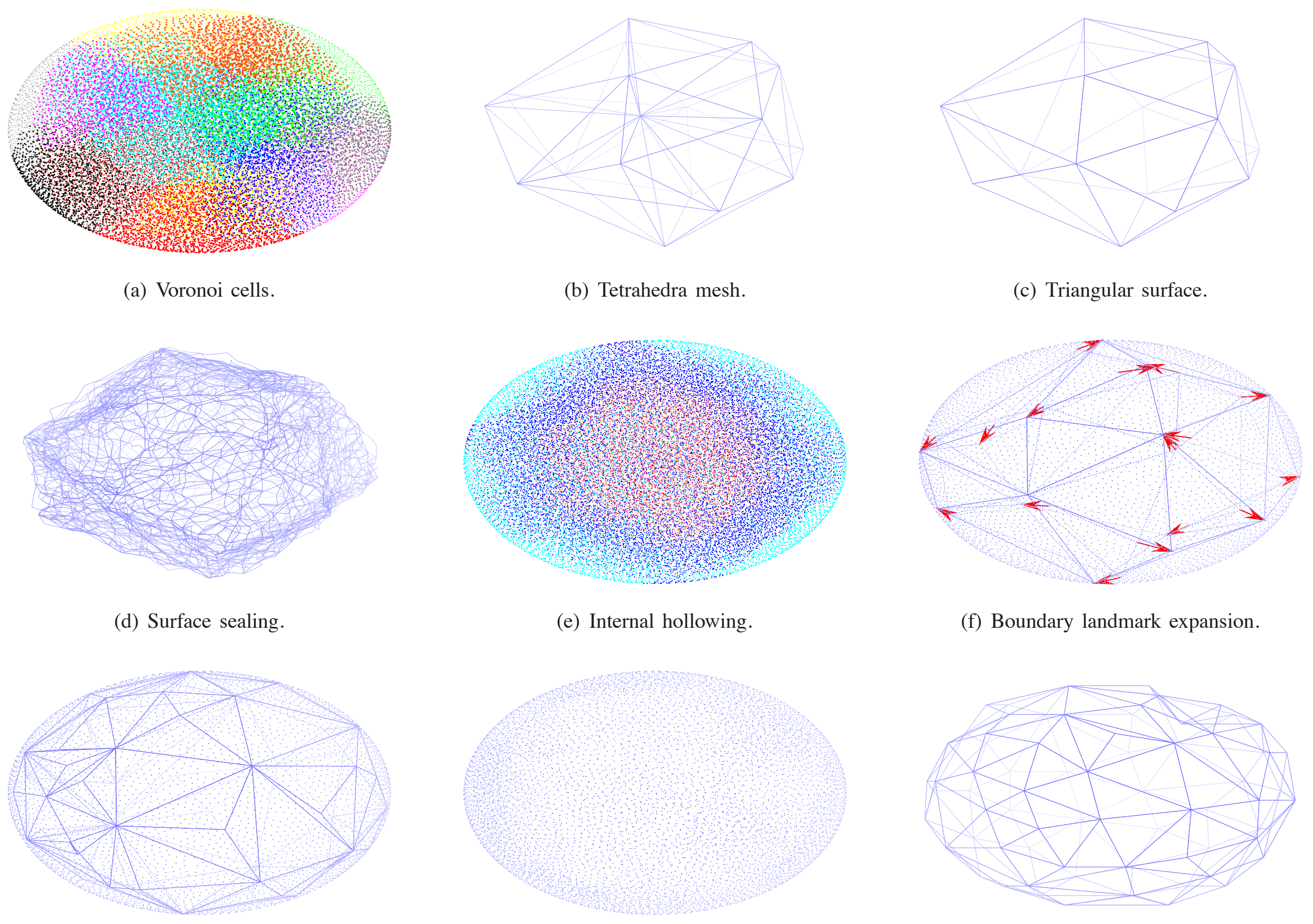

(a) Voronoi cells. (b) Tetrahedra mesh. (c) Triangular surface. (d) Surface sealing. (e) Internal hollowing. (f) Boundary landmark expansion. (g) Boundary face splitting. (h) Final boundary nodes. (i) Fine-grained triangular boundary surface.

Fig. 3. An example of boundary detection by using Coconut. In Subfigure (e), O-nodes, S-nodes and I-nodes are highlighted in cyan, blue and red, respectively.

Clearly, boundary faces and landmarks can be identified based on local information. Assume the network is connected and there are no degenerated edges or faces in the tetrahedral mesh, a boundary surface is defined below.

Definition 4: The *triangular boundary surfaces* of a tetrahedral mesh are a set of closed surfaces formed by triangular faces that together enclose all tetrahedra (including landmarks and faces on the tetrahedra).

An example of boundary surface is illustrated in Fig. 3(c). Moreover, we observe the following results regarding boundary faces and surfaces.

Lemma 1: A triangular boundary surface consists of boundary faces only.

Proof: We prove the lemma by contradiction. Assume a boundary surface includes an internal face. Since an internal face must be shared by two neighboring tetrahedra, the two tetrahedra must be separated on two sides of the boundary. In other words, one of the tetrahedra must be located outside the boundary surface, contradicting Definition 4 that requires boundary surfaces to enclose all tetrahedra. Therefore the lemma is proven. ■

Lemma 2: A boundary face must belong to at least one boundary surface.

Proof: This is obvious from the definition of boundary face (i.e., Definition 2). Since all tetrahedra are enclosed by the boundary surfaces, a face must either be shared by two tetrahedra (i.e., inside) or reside on a boundary surface. Therefore, a boundary face (that is not shared by two tetrahedra) must belong to a boundary surface. ■

As a rigid structure, the tetrahedral mesh effectively depicts the geometric property of the 3D sensor network, and the triangular boundary surfaces serve as a delineation of the rough network boundaries. The construction of the tetrahedral and triangular boundary surfaces is based on landmarks, which is not affected by nodal density and thus robust to nonuniform sensor distribution. However, note that, a triangular boundary surface is merely a "skeleton", consisting of landmarks and virtual edges only, where a virtual edge is mapped to a shortest path between two corresponding neighboring landmarks. Given an arbitrary node in the network, it still remains nontrivial to determine whether it is enclosed by the skeleton or not based on connectivity information only. To this end, we next introduce an efficient scheme to build a thin layer to "seal" the triangular boundary surfaces, which serve as the basis for further boundary refinement.

B. Surface Sealing and Internal Hollowing

Based on the skeletal triangular boundary surfaces obtained above, the second step of the proposed algorithm intends to build sealed boundary surfaces, which effectively separate internal nodes (that are absolutely non-boundary) and boundary node candidates (i.e., the nodes on or outside the sealed boundary surfaces). The former can then be easily hollowed out, while the latter are subject to further refinement.

Let us consider a boundary triangular face, e.g., $\triangle ABC$ shown in Fig. 4(a). An edge of the triangular face is formed by the shortest path between corresponding landmark nodes. For example, Edge AB is the shortest path between Nodes A and B, denoted by $\Gamma(A,B)$, including a sequence of intermediate nodes (see the black dots between A and B in Fig. 4(a)).

The basic idea of sealing is to choose multiple pairs of nodes on two edges of a triangular face and build a thin layer based on the shortest paths between them. For example, let $\langle p_0, p_1, .., p_m \rangle$ and $\langle q_0, q_1, .., q_n \rangle$ denote the intermediate nodes on the shortest paths $\Gamma(A,B)$ and $\Gamma(A,C)$, respectively. Without loss of generality, let us assume $m \leq n$. The algorithm intends to establish shortest paths between p_i and q_i $\forall\ 0 \leq i \leq m$ and between p_m and q_i $\forall\ m < i \leq n$, in order to construct a *surface layer* to seal the triangle (see Fig. 4(a) where the dashed lines represent shortest paths). Note that multiple shortest paths often exist between two nodes, forming a spindle shape in 3D space. One can of course include all such paths in the sealed surface, but it will result in an undesired thick layer (as shown in Fig. 4(b)), rendering difficulties in later refinement. On the other hand, if only one of the shortest paths is randomly chosen for the formation of the sealed surface, it fails frequently because the shortest paths between two adjacent pairs of intermediate nodes (e.g., $\Gamma(p_i,q_i)$ and $\Gamma(p_{i+1},q_{i+1})$) are not always well connected, leaving holes on the surface layer (illustrated in Fig. 4(c)). To this end, a simple and effective approach is proposed as follows.

The shortest paths are established sequentially, starting from $\Gamma(p_0,q_0)$. It follows the classic Dijkstra's algorithm to search for the shortest path from p_0 to q_0, but with a constraint that only the neighboring nodes of $\Gamma(B,C)$ are involved in the algorithm. Therefore the established shortest path, $\Gamma(p_0,q_0)$, is tightly connected with $\Gamma(B,C)$. The process repeats, where $\Gamma(p_i,q_i)$ is established based on the neighboring nodes of $\Gamma(p_{i-1},q_{i-1})$ if $i \leq m$, or of $\Gamma(p_m,q_{i-1})$ if $m < i \leq n$. All boundary triangular faces run the same algorithm to establish such shortest paths in a distributed manner. Finally, when the process terminates, the sealed boundary faces and sealed boundary surfaces are defined as follows.

Definition 5: For a given triangular boundary face $\triangle ABC$, the nodes on the shortest paths $\Gamma(p_i,q_i)$ $\forall\ 0 \leq i \leq m$ and $\Gamma(p_m,q_{i-1})$ $\forall\ m < i \leq n$, plus the nodes on $\Gamma(A,B)$, $\Gamma(A,C)$, and $\Gamma(B,C)$, are called *Γ-nodes*.

Definition 6: The Γ-nodes and their one-hop neighbors form the *sealed boundary face*.

Definition 7: For a given triangular boundary surface, the union of all corresponding sealed boundary faces form the *sealed boundary surface*. The nodes on sealed boundary surfaces are called *S-nodes*.

For example, Fig. 4(d) illustrates the sealed boundary face of $\triangle ABC$, and Fig. 3(d) depicts the shortest paths formed by Γ-nodes on the boundary surface. Note that the sealed boundary surfaces are built based on triangular boundary surfaces, which are approximation but not true boundaries. While they ensure to contain every landmark, they do not enclose all sensors. Based on the sealed boundary surfaces, the sensors can be grouped into internal nodes (or I-nodes), surface nodes (or S-nodes), and outside notes (or O-nodes), as evident by the following theorem.

Theorem 1: The sealed boundary surfaces can separate internal nodes and outside nodes.

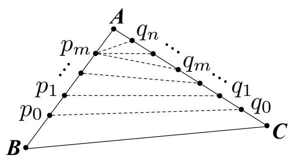
Proof: To prove the theorem, we show that, if the sealed boundary surfaces are removed, the network is partitioned into disconnected subnetworks, located inside and outside the sealed boundary surfaces. According to the method of constructing sealed boundary surfaces, any two adjacent Γ-nodes are separated by at most R from each other, where R is the maximum transmission range of sensors. Since the neighboring nodes with a distance no greater than R to the Γ-nodes are included in the sealed boundary surface, the thinnest part of a sealed boundary surface is $\sqrt{3}R$ (as illustrated in Fig. 4(e)[2]), ensuring that a node above the sealed boundary surface cannot communicate with any nodes beneath it. Thus the sealed boundary surfaces can always separate internal nodes and outside nodes. ■

The motivation to build the sealed boundary surfaces is to hollowed out internal nodes because they are absolutely non-boundary, while nodes on or outside the sealed boundary surfaces are boundary node candidates that will be further refined as to be discussed in the next subsection. With known sealed boundary surfaces, it is straightforward to achieve such internal hollowing. More specifically, any internal landmark initiates a *hollowing request*, broadcasted to its neighbors. Upon receiving the hollowing request, a node rebroadcasts the request and marks itself as an *I-node* (for internal node) if it is not on a sealed boundary surface; or simply drops the request otherwise. According to Theorem 1, the hollowing request cannot be passed across the sealed boundary surfaces, from a node inside to a node outside. Therefore, a node marks itself as an *O-node* (for outside node) if it neither locates on a sealed boundary surface nor receives a hollowing request.

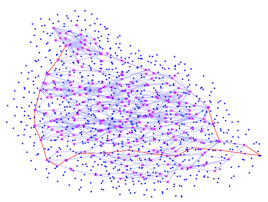
Lemma 3: An I-node is not a boundary node.

Proof: I-nodes are enclosed by the sealed boundary surfaces. Assume high nodal density and let us draw an arbitrary line from an I-node toward the outside of the network. The line must intersect a sealed boundary surface, with a corresponding S-node closer to the outside than the I-node. This is true for

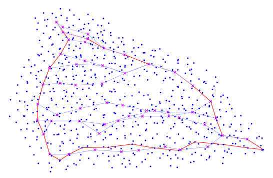
[2]Note that although balls are used for illustration in Fig. 4(e), this work is not limited to Unit Ball Graph (UBG) communication model. Only a maximum transmission range (i.e., R) is assumed, indicating that two nodes cannot communicate if they are separated by a distance more than R. However, two nodes may or may not be able to communicate if their distance is less than or equal to R, in contrast to UBG which assumes a communication link must exist if the distance between two nodes is no greater than R.



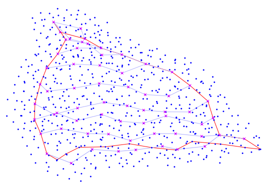
(a) Boundary triangular face.



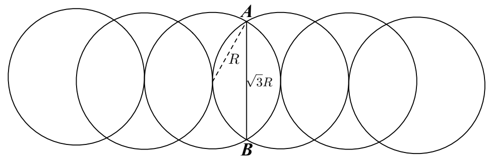
(b) Surface by all shortest paths.



(c) Surface by random paths.



(d) Sealed boundary surface.



(e) The skeleton layer.

Fig. 4. Construction of sealed boundary surfaces.

any line drawn in any direction. Therefore, the I-node is not a boundary node. ■

The hollowing process hollows out the internal nodes and yields a set of boundary node candidates defined below.

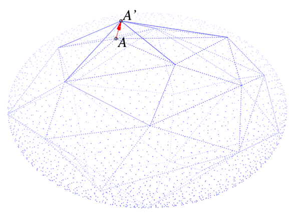
Definition 8: O-nodes and S-nodes create a set of *boundary node candidates*.

The boundary node candidates obtained so far often form a thick layer (as shown in Fig. 3(e)) that includes many non-boundary nodes, i.e., false-positives. To this end, a series of refinement are discussed below to produce a thin layer of boundary nodes with low missing and false-positive rates.

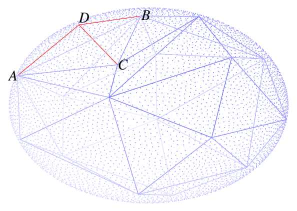
C. Boundary Refinement

The strategies for boundary surface refinement follow two streams of thought, i.e., to expand the sealed coarse boundary surfaces such that they become as close to the true boundary surfaces as possible, and to thin the sealed boundary surfaces. The former can decrease both missing rate and false-positive rate, while the latter effectively slashes false-positives.

1) Boundary Landmark Expansion: Since the radius of Voronoi cells (introduced in Sec. II-A) is k hops, landmarks can be up to k-hop away from real boundary where k ranges from 4 to 8 in our implementation. Therefore, many true boundary nodes are not included in the sealed coarse boundary surfaces, and at the same time, most nodes in the sealed coarse boundary surfaces are in fact false-positives. To this end, a local iterative process is proposed to push boundary landmarks outward. More specifically, if there exist one or multiple O-nodes in a cell, the O-node closest to the current landmark is chosen to serve as the new landmark of the cell. The boundary



(a) Boundary landmark expansion.



(b) Boundary face splitting.

Fig. 5. Boundary surface expansion.

surfaces are updated (i.e., expanded) accordingly based on the new boundary landmarks. Fig. 5(a) illustrates an example of boundary landmark expansion, where Landmarks A is replaced by O-node A'. An effective expansion should include previous O-nodes into the boundary surfaces and reduce the number of remaining O-nodes. Clearly, only a boundary landmark could find O-nodes in its cell. The process repeats until no more effective expansion is possible. Boundary landmark expansion effectively dilates the boundary surface as exemplified in Fig. 3(f).

2) Boundary Face Splitting: After boundary landmark expansion, the boundary landmarks are now near to true boundaries. However, since the edges of boundary triangular faces are long (between k to $2k$-hops), the triangular boundary surfaces are often not kept perfectly close to the true boundaries (see boundary face $\triangle ABC$ in Fig. 5(b) for example). This problem motivates us to split a triangular boundary face into smaller faces to approach the true boundary. More specifically, if there exists an O-node that has equal hop distance (or differed by one) to three landmarks of a triangular boundary face, the boundary face is replaced by three new (and smaller) triangular faces formed by the O-node and the three landmarks. For example, assume Node D (shown in Fig. 5(b)) is such an O-node. $\triangle ABC$ is thus replaced by $\triangle ACD$, $\triangle ABD$ and $\triangle BCD$, which are better fit to the true boundary. Fig. 3(g) illustrates the result after boundary face splitting.

3) Boundary Surface Thinning: The boundary landmark expansion and boundary face splitting can effectively expand the boundary surfaces outward, approaching as close as possible to the true boundaries. However, the boundary node candidates (with the majority of S-nodes and a small number of O-nodes if there are any remaining) still form a relatively thick layer. Note that the S-nodes alone include Γ-nodes and their 1-hop neighbors (in both inward and outward directions) as described in Definition 7, resulting in 2-hops of nodes likely false-positives. To this end, each I-node that has a S-node or O-node in its one-hop neighborhood broadcasts a *hollowing request* in two hops to mark the inmost two-hop boundary node candidates as I-nodes, yielding a much thinner layer of boundary nodes as illustrated in Fig. 3(h).

4) Fine-Grained Triangulation on Boundary Surfaces: As to be discussed in Sec. III, the proposed algorithm can identify virtually all boundary nodes, with extremely low missing rate. At the same time, although false-positives are inevitable since the algorithm is based on nodal connectivity only, nearly all such false-positives are one-hop neighbors of true boundary nodes. In short, the proposed algorithm is highly accurate in boundary node identification. However, there are a range of applications (such as network segmentation [28] and boundary mapping and routing [29]) require not only boundary nodes but also fine-grained triangular boundary surfaces, i.e., triangular boundary surfaces consisting of small triangular faces. The boundary surfaces obtained above may have an edge length up to $2k$ hops. While boundary face splitting helps achieve finer triangulation, it does not ensure to split every triangular face. To address this problem, a set of new landmarks are elected among the boundary nodes such that any two landmarks are δ-hops apart, where δ is the desired fineness of the triangular boundary surfaces. Then the planar triangulation algorithm developed in our previous work [30] is adopted here to produce desired fine-grained triangular boundary surfaces, as demonstrated in Fig. 3(i).

D. Computational and Communication Complexity

The proposed Coconut algorithm has a linear time complexity and communication cost (measured by messages sent) with respect to the size of the network. More specifically, surface construction introduced in Sec. II-A has a computational complexity of $O(n)$ and communication cost of $O(n)$, where n is the total number of nodes in the network. Surface sealing and Internal Hollowing has a complexity of $O(1)$ and communication cost of $O(n)$. Note that this process can be carried out in a parallel manner. The last step has a complexity of $O(m)$ and communication cost of $O(m)$, where m is the number of boundary nodes. Given $m \ll n$, the overall time complexity and communication cost of the Coconut algorithm are dominated by $O(n)$ in practice.

III. Simulations

To evaluate the effectiveness of the proposed Coconut boundary detection algorithm, we have carried out extensive simulations under various 3D network models. In this section, we first introduce our simulation setup, and then present the simulation results and discuss our observations.

A. Simulation Setup

A set of 3D graphic tools (including Autodesk Maya and TetGen [31]) are employed to construct 3D networks for our simulations. First, we build 3D models that represent practical sensor network scenarios (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 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 each 3D model uniformly or non-uniformly. Once the nodes are placed, an appropriate maximum radio transmission range R is chosen according to nodal density, such that the network is connected. In our simulated network scenarios, nodal degree ranges from 3 to 140, with an average of 25.

B. Simulation Results

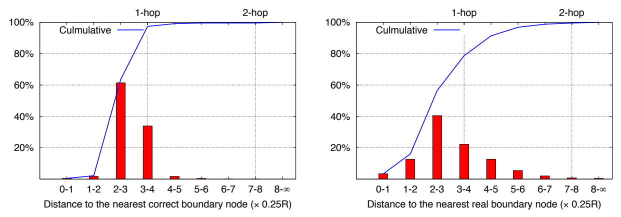
Several network models and their boundary detection results are illustrated in Figs. 6-9. Fig. 6(a) depicts an underwater network, where nodes are uniformly deployed from top to bottom of the lake, e.g., for water pollution monitoring. As shown in Fig. 6(b), the proposed Coconut algorithm effectively identifies the boundaries at both the flat top surface and the bumpy bottom. We linearly increase nodal density from top to bottom, as shown in Fig. 7. Despite much more nodes are resided in the bottom area, the Coconut algorithm successfully detects the network boundary, yielding a boundary surface largely identical to the result under uniform sensor deployment (compare Fig. 6(c) and Fig. 7(c)). It is noticed that more nodes at the bottom (i.e., the high-density area) are considered as boundary nodes by the proposed algorithm, thus increasing false-positive rate. This a is natural outcome because the connectivity-based approach cannot differentiate nodes within one hop. Fig. 8 depicts a 3D network deployed in the space (e.g., for chemical dispersion sampling). It contains an internal hole due to uncontrolled drift of sensor nodes. As shown in Fig. 8(b) and 8(c), the Coconut algorithm effectively identifies both outer and inner boundaries. Fig. 9 shows a 3D network deployed in a S-shape pipe, where the boundary nodes are identified and the triangular mesh surfaces are well constructed despite large surface curvatures.

Table II reports quantitative data and performance statistics obtained from our simulations. Among the boundary nodes identified by the Coconut algorithm, a node is called a *correct boundary node* if it is in the ground truth boundary node set, which is known from network model creation discussed in Sec. III-A; otherwise it is a *mistaken boundary node*. If a node is a real boundary node, however the boundary detection algorithm fails to detect it, it is called a *missing boundary node*. The correct (or missing or mistaken) rate is the ratio of the number of correct (or missing or mistaken) boundary nodes to the number of ground truth boundary nodes.

As can be seen in the table, the proposed Coconut algorithm can always identify boundary nodes with high correct rate ($> 95\%$) under various network settings with both inner and outer boundaries and uniform or nonuniform sensor deployment. Only a small percentage of boundary nodes are left out ($< 3\%$) due to the boundary thinning process applied to some part with less than 2-hop thickness. Moreover, as illustrated in Fig. 10(a), all of the missing nodes are located within 1 hop of correctly identified boundary nodes. In other words, the missing boundary nodes do not form a large "hole" and thus do not affect the formation of boundary surfaces. On the other hand, some interior nodes are falsely marked as boundary nodes. Compared to the missing rate, the mistaken rate is generally high. However, the mistaken boundary nodes are well distributed. We calculate the distance between every mistaken boundary node and its nearest real boundary node, which shows how far the mistaken node is away from the ground truth boundary. As can be seen in Fig. 10(b), most ($> 80\%$) of the mistaken nodes are located within 1 hop of the real boundary nodes. Therefore the boundary nodes identified by the Coconut algorithm effectively depict the shape of network boundary, based on which the triangular surfaces can be well constructed.

TABLE II
BOUNDARY DETECTION RESULTS.

Model	Correct	Missing	Mistaken
Model 1	99.9%	0.1%	47.9%
Model 2	99.9%	0.1%	147.1%
Model 3	98.1%	1.9%	174.9%
Model 4	97.0%	3.0%	54.1%



(a) Missing node distribution. (b) Mistaken node distribution

Fig. 10. Distribution of missing and mistaken boundary nodes.

IV. Conclusion

We have proposed a distributed boundary detection algorithm, dubbed *Coconut*, for 3D wireless sensor networks. Its basic idea is to construct a tetrahedral structure to delineate the approximate geometry of the 3D sensor network, which consequently yields a set of "sealed" triangular boundary surfaces for separating non-boundary nodes and boundary node candidates. While the former are hollowed out immediately, the latter are further refined to identify the final boundary nodes and fine-grained boundary surfaces. We have proven the correctness of the algorithm and quantitatively demonstrated its effectiveness via simulations under various network models. The proposed Coconut algorithm is a connectivity-based approach, with no need for localization or distance measurement. It has not constraint on communication models and only assumes a constant maximum transmission range, which is generally known in practical wireless sensor networks. Moreover, it can effectively identify boundaries in both uniformly and non-uniformly distributed sensor networks, exhibiting excellent robustness to sensor distribution.

References

[1] 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 SenSys*, pp. 117–129, 2007.

[2] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou, "Challenges: Building Scalable Mobile Underwater Wireless Sensor Networks for Aquatic Applications," *IEEE Network*, vol. 20, no. 3, pp. 12–18, 2006.

[3] X. Bai, C. Zhang, D. Xuan, J. Teng, and W. Jia, "Low-Connectivity and Full-Coverage Three Dimensional Networks," in *Proc. of MobiHOC*, pp. 145–154, 2009.

[4] X. Bai, C. Zhang, D. Xuan, and W. Jia, "Full-Coverage and K-Connectivity (K=14, 6) Three Dimensional Networks," in *Proc. of INFOCOM*, pp. 388–396, 2009.

[5] C. Liu and J. Wu, "Efficient Geometric Routing in Three Dimensional Ad Hoc Networks," in *Proc. of INFOCOM*, pp. 2751–2755, 2009.

[6] 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.

[7] J. Opatrny, A. Abdallah, and T. Fevens, "Randomized 3D Position-based Routing Algorithms for Ad-hoc Networks," in *Proc. of Mobiquitous*, pp. 1–8, 2006.

[8] R. Flury and R. Wattenhofer, "Randomized 3D Geographic Routing," in *Proc. of INFOCOM*, pp. 834–842, 2008.

[9] F. Li, S. Chen, Y. Wang, and J. Chen, "Load Balancing Routing in Three Dimensional Wireless Networks," in *Proc. of ICC*, pp. 3073–3077, 2008.

[10] J. Zhou, Y. Chen, B. Leong, and P. S. Sundaramoorthy, "Practical 3D Geographic Routing for Wireless Sensor Networks," in *Proc. of SenSys*, pp. 337–350, 2010.

[11] D. Pompili, T. Melodia, and I. F. Akyildiz, "Routing Algorithms for Delay-insensitive and Delay-sensitive Applications in Underwater Sensor Networks," in *Proc. of MobiCom*, pp. 298–309, 2006.

[12] 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 INFOCOM*, pp. 798–806, 2008.

[13] R. Nowak and U. Mitra, "Boundary Estimation in Sensor Networks: Theory And Methods," in *Proc. of IPSN*, pp. 80–95, 2003.

[14] M. Ding and X. Cheng, "Robust Event Boundary Detection and Event Tracking in Sensor Networks - a Mixture Model based Approach," in *Proc. of INFOCOM*, pp. 2991–2995, 2009.

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

[16] S. Duttagupta, K. Ramamritham, and P. Ramanathan, "Distributed Boundary Estimation using Sensor Networks," in *Proc. of MASS*, pp. 316–325, 2006.

[17] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized Fault-Tolerant Event Boundary Detection in Sensor Networks," in *Proc. of INFOCOM*, pp. 902–913, 2005.

[18] M. Fayed and H. Mouftah, "Localized Alpha-Shape Computations for Boundary Recognition in Sensor Networks," *Elsevier Ad Hoc Networks*, vol. 7, no. 6, pp. 1259–1269, 2009.

[19] 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.

[20] R. Ghrist and A. Muhammad, "Coverage and Hole-detection in Sensor Networks via Homology," in *Proc. of IPSN*, pp. 34–40, 2005.

[21] D. Dong, Y. Liu, and X. Liao, "Fine-grained Boundary Recognition in Wireless Ad hoc and Sensor Networks by Topological Methods," in *Proc. of MobiHoc*, pp. 135–144, 2009.

[22] Y. Wang, J. Gao, and J. Mitchell, "Boundary Recognition in Sensor Networks by Topological Methods," in *Proc. of MOBICOM*, pp. 122–133, 2006.

[23] A. Kröller, S. Fekete, D. Pfisterer, and S. Fischer, "Deterministic Boundary Recognition and Topology Extraction for Large Sensor Networks," in *Proc. of SODA*, pp. 1000–1009, 2006.

[24] 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.

[25] H. Zhou, S. Xia, M. Jin, and H. Wu, "Localized Algorithm for Precise Boundary Detection in 3D Wireless Networks," in *Proc. of ICDCS*, pp. 744–753, 2010.

[26] 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. of MobiHoc*, pp. 141–152, 2011.

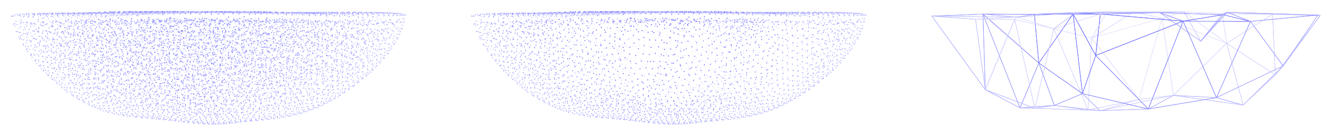
[27] H. Jiang, S. Zhang, G. Tan, and C. Wang, "CABET: Connectivity-based Boundary Extraction of Large-Scale 3D Sensor Networks," in *Proc. of INFOCOM*, pp. 784–792, 2011.

[28] H. Zhou, N. Ding, M. Jin, S. Xia, and H. Wu, "Distributed Algorithms for Bottleneck Identification and Segmentation in 3D Wireless Sensor Networks," in *Proc. of SECON*, 2011.

[29] S. Xia, X. Yin, H. Wu, M. Jin, and X. D. Gu, "Deterministic Greedy Routing with Guaranteed Delivery in 3D Wireless Sensor Networks," in *Proc. of MobiHoc*, pp. 1–10, 2011.

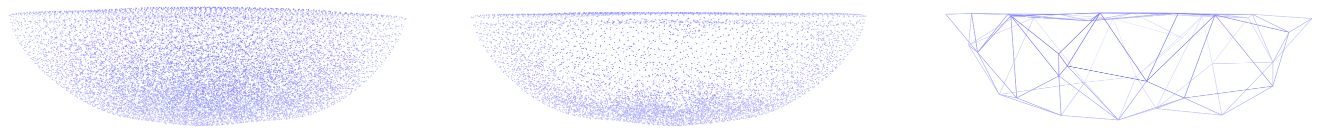
[30] H. Zhou, H. Wu, S. Xia, M. Jin, and N. Ding, "A Distributed Triangulation Algorithm for Wireless Sensor Networks on 2D and 3D Surface," in *Proc. of INFOCOM*, 2011.

[31] http://tetgen.berlios.de/.



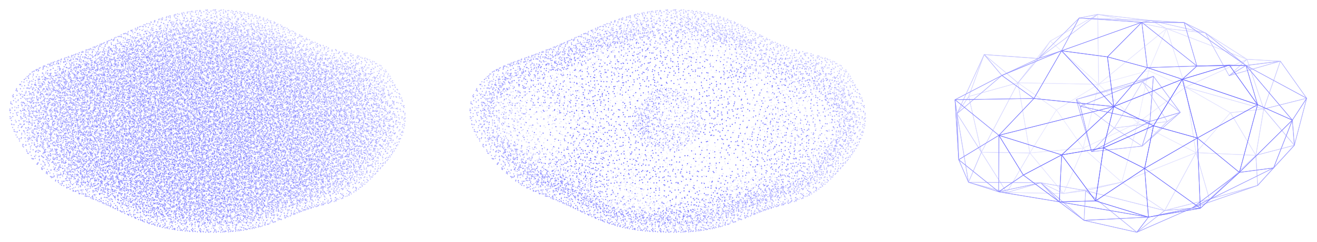
(a) Network model. (b) Boundary nodes. (c) Triangular mesh.

Fig. 6. Model 1: an example of under water network (with uniform sensor deployment).



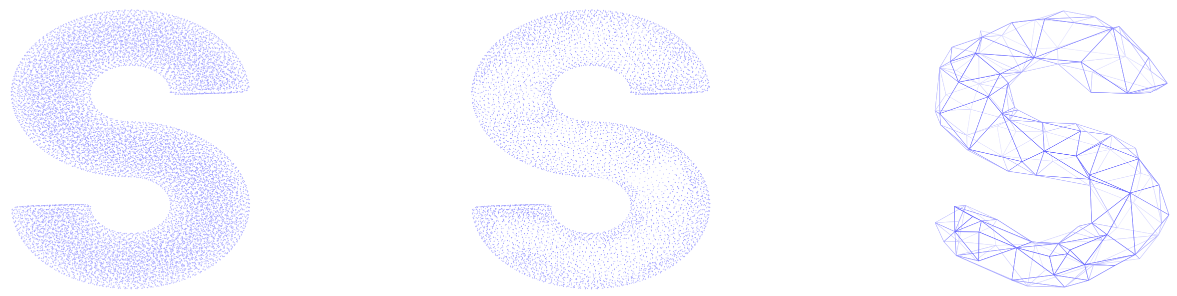
(a) Network model. (b) Boundary nodes. (c) Triangular mesh.

Fig. 7. Model 2: an example of under water network (with non-uniform sensor deployment).



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

Fig. 8. Model 3: an example of a 3D space sensor network with an internal hole.



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

Fig. 9. Model 3: an example of a 3D sensor network deployed in a S-shape pipe.