LBDP: Localized Boundary Detection and Parametrization for 3-D Sensor Networks

Feng Li, Chi Zhang, Jun Luo, *Member, IEEE*, Shi-Qing Xin, and Ying He, *Member, IEEE*

***Abstract*—Many applications of wireless sensor networks involve monitoring a time-variant event (e.g., radiation pollution in the air). In such applications, fast boundary detection is a crucial function, as it allows us to track the event variation in a timely fashion. However, the problem becomes very challenging as it demands a highly efficient algorithm to cope with the dynamics introduced by the evolving event. Moreover, as many physical events occupy volumes rather than surfaces (e.g., pollution again), the algorithm has to work for 3-D cases. Finally, as boundaries of a 3-D network can be complicated 2-manifolds, many network functionalities (e.g., routing) may fail in the face of such boundaries. To this end, we propose *Localized Boundary Detection and Parametrization* (LBDP) to tackle these challenges. The first component of LBDP is *UNiform Fast On-Line boundary Detection* (UNFOLD). It applies an inversion to node coordinates such that a "notched" surface is "unfolded" into a convex one, which in turn reduces boundary detection to a localized convexity test. We prove the correctness and efficiency of UNFOLD; we also use simulations and implementations to evaluate its performance, which demonstrates that UNFOLD is two orders of magnitude more time- and energy-efficient than the most up-to-date proposal. Another component of LBDP is *Localized Boundary Sphericalization* (LBS). Through purely localized operations, LBS maps an arbitrary genus-0 boundary to a unit sphere, which in turn supports functionalities such as distinguishing interboundaries from external ones and distributed coordinations on a boundary. We implement LBS in TOSSIM and use simulations to show its effectiveness.**

***Index Terms*—3-D wireless sensor networks, boundary detection, convexity test, inversion, localized algorithm, surface parametrization.**

I. Introduction

ONE OF the main applications of *wireless sensor networks* (WSNs) is to constantly monitor physical events (or phenomena) either too widely spread or too remote to be accessed through conventional techniques. Boundary detection, as an enabling technique to such applications, becomes very crucial to the functionality of WSNs. It allows the network users to be aware of the geometry of the network, which infers either the sensing coverage (if the network only partially covers the targeted event [17]) or the boundary of the targeted event (if the network fully covers it [23]). Moreover, boundary detection is also an important component to support geographic data routing in WSNs [10], [26], [31].

While most of the existing boundary detection approaches are designed just for a "one-time shot," the detection actually has to be constantly conducted, given the time-varying nature of the events under surveillance. Such events, for example, can be bio-geo-chemical processes, streams/currents, or pollution in atmosphere or waterbodies (e.g., ocean). Depending on different deployments, a WSN can be either static (e.g., in smart buildings) to observe the event passing through or stuck to an event to keep monitoring it (e.g., for water monitoring). In both cases, boundary detection has to be performed online to keep tracking either the event or the network boundary. Unfortunately, the existing approaches have too high message or time complexity to be performed in an online manner.

Another feature of the events under consideration is that they often span a 3-D volume rather than a 2-D surface. Given the fact that very few existing proposals deal with 3-D boundary detection and that extending the approaches designed for 2-D surfaces to 3-D volumes is highly nontrivial in geometry,[1] a clean-slate boundary detection algorithm needs to be designed for 3-D WSNs. Note that, should a 2-D boundary detection be ever needed, it would be really trivial to reduce a 3-D detection approach to 2-D.

Furthermore, 3-D WSNs may have arbitrarily complicated but orientable 2-manifolds as their boundaries. This, on one hand, makes it difficult to tell internal boundaries from the external one from a local point of view. On the other hand, routing protocols may fail when facing or on such boundaries [8]. Therefore, if we do not regularize the boundary geometry, the boundary detection may be performed in vain without delivering necessary supports to networking functionalities.

In this paper, we tackle the aforementioned challenges by proposing *Localized Boundary Detection and Parametrization* (LBDP). LBDP comes with two components: *UNiform Fast On-Line boundary Detection* (UNFOLD) for boundary detection and *Localized Boundary Sphericalization* (LBS) for boundary regularization. The underlying principle of UNFOLD is to apply a spherical inversion to the local coordinates of every node, such that a (locally) concave surface can be "unfolded" into a convex one. As a result, the painful procedure of

Manuscript received May 16, 2012; revised March 07, 2013; accepted March 10, 2013; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Ramasubramanian. Date of publication April 08, 2013; date of current version April 14, 2014. This work was supported in part by AcRF Tier 2 Grant ARC15/11, NTU under the Start-up Grant, AcRF Tier 1 Grant RG 32/09, and AcRF Tier 1 Grant RG40/12. Preliminary results were presented in the Proceedings of the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Paris, France, May 16–19, 2011.

The authors are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: fli3@ntu.edu.sg; czhang8@ntu.edu.sg; junluo@ntu.edu.sg; sqxin@ntu.edu.sg; yhe@ntu.edu.sg).

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.2013.2253561

[1]For example, the well-known *edge flip* algorithm for Delaunay triangulation in 2-D does not converge in 3-D [19].

identifying a boundary node on a "notched" surface is reduced to convexity test (which can be tackled with simple geometric tools). Though the idea of inversion is borrowed from [20], UNFOLD has substantially improved on it by implementing it in a distributed networking scenario. As UNFOLD entails only simple and uniform computation for every node, it can be performed super fast and hence enable online boundary detection. Building upon UNFOLD, LBS employs a diffusion process to regularize an arbitrary boundary into a unit sphere. The algorithm is purely localized and involves only arithmetic operations, and the resulting virtual coordinates on the sphere may be exploited by many networking functionalities, such as distinguishing internal boundaries from the external one. Our main contributions in LBDP are the following:

- the idea of 3-D WSN boundary detection in a transformed domain;
- the online algorithm, UNFOLD, that entails only localized communications and computations;
- a real implementation of UNFOLD in MICAz Motes for time and energy efficiency evaluations;
- the parametrization algorithm, LBS, that efficiently regularizes an arbitrary genus-0 boundary into a unit sphere;
- distinguishing internal boundaries from the external one (from a local point of view) based on the virtual coordinates delivered by LBS;
- a full implementation of LBS in TOSSIM for realistic simulations.

In the following, we first discuss backgrounds and related literature in Section II. We focus on UNFOLD in Section III: We first discuss boundary definitions and properties, then we present UNFOLD in detail along with the corresponding analysis. We then present LBS in Section IV, along with the descriptions of two crucial applications of LBS. We finally report the simulation and experiment results in Section V, before concluding our paper in Section VI. To maintain fluency, we postpone the proof details or sketches to the Appendix.

II. Background and Motivations

In this section, we briefly discuss the existing proposals for boundary detection, which in turn serves as the motivations for our proposal. To the best of our knowledge, boundary regularization through distributed parametrization has never been done in the literature. Note that, in the computational geometry community, *surface reconstruction* from sample points has been extensively studied. Representative proposals include CRUST [2] and Cocones [3], which are based on Voronoi diagrams generated by the sample points. These approaches assume that the input points are densely sampled from some surface, so they are not suitable for boundary detection, whose objective is to identify boundary points from a set of points that form a 3-D volume.

A. Geometric or Topological

The existing boundary detection approaches can be roughly classified into two categories, namely geometric (e.g., [11], [29], [34], and [35]) and topological (e.g., [7], [10], [12]–[14], [21], and [32]). While the former always requires the knowledge of nodes location or distance, the latter is often claimed as a location/range-free approach. For example, Ghrist and Muhammad [14] compute homology groups, algebraic topological invariants, to recognize "holes" within WSN coverage. Kröller *et al.* [21] define boundary based on chordless cycles and propose a series of fairly sophisticated algorithms to identify subgraphs that satisfy the boundary criterions. References [7], [10], and [32] are similar in the sense that they rely on global connectivity information (e.g., shortest path tree or primary boundary circle) to "guide" further boundary refinements.

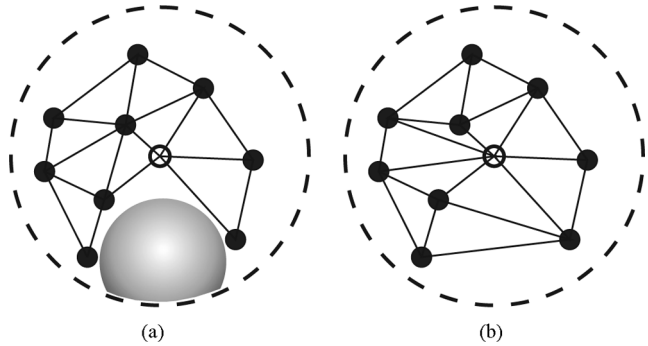


Fig. 1. A node (white) and all its one-hop neighbors (black) are shown. Whether the white one is identified as on the boundary or not depends on the specific geometric interpretation. Precisely, the triangulation on the left indicates the node as on the boundary, but the answer is negative for the case on the right. The shaded ball is used to illustrate the idea of α-shape [9].

Generally, the price a topological approach pays to avoid relying on location/range information is a highly complicated procedure that often requires a large scale coordination among a WSN (in particular, the algorithm proposed in [14] is actually centralized). Therefore, while the topological approaches do offer a one-time boundary detection for static WSNs, they are not adequate to online detection. Moreover, whether these topological approaches can be extended to distributed 3-D boundary detection is still open.

B. "Pain" of Geometric Approaches

Equipped with location or range information (precisely, each node knows either its own location that may come with a certain error or the distances between itself and close-by nodes), geometric approaches often involve fairly localized computations and hence have the potential to be performed online. However, the discrete nature of a WSN "volume" may lead to complicated local detections. On one hand, there is no commonly agreed definition for the boundary of a point cloud (the geometric representation of a WSN). For example, Fig. 1 shows that whether a node is on the boundary or not heavily depends on the specific geometric interpretation. On the other hand, the algorithms for boundary detection, albeit localized, can still incur a high time and/or message complexity.

Zhang *et al.* [34] propose to use two local geometric structures, namely *localized Voronoi polygon* and *neighbor embracing polygon*, for boundary characterization only in 2-D. It is shown in the paper that the detection procedure involves several rounds of interactions between (at least) all one-hop neighbors. Both [11] and [35] use a concept called α-shape [9] for boundary detection. In a nutshell, α-shape results from "erasing" the convex hull of a point cloud using a spherical

"eraser" with a certain radius α: While 0-shape is the original point cloud, ∞-shape is the convex hull. In Fig. 1, the left triangulation is actually an α-shape with α (roughly) equal to half of the transmission range. Although the α-shape construction leads to localized boundary detection, its computation cost is still nonnegligible.

Remark: Although we are concerned with boundary detection in 3-D WSNs, we have to provide examples in 2-D to facilitate visual illustration. However, our simulations will be performed for real 3-D WSNs.

C. Event Boundary Versus Network Boundary

Some existing proposals tend to distinguish between event boundary and network boundary [35]. As we discussed in Section I, we are concerned with both boundaries for the WSN applications under consideration. In fact, network boundary can be considered as a particular event boundary, with the "event" being the WSN itself. It is true that, whereas the network boundary is a clear-cut concept, other event boundaries can be rather fuzzy due to sensing errors or smooth changes (in terms of certain physical quantities indicating the event) around the boundary. Fortunately, relying on statistic approaches such as [6], each node can arrive at an (binary) indicator on whether it covers a certain event or not [36]. Moreover, using an event indicator may suggest a simple way to detect event boundaries: A boundary node has at least one of its neighbors carrying an indicator value different from its own value. However, this method often leads to rather "thick" boundaries if the transmission range of nodes is relatively large. Therefore, we do not make a distinction between these two types of boundaries in our paper.

D. Our Approach

In summary, a geometric approach that relies on the location or range information appears to be the right way toward a fast online boundary detection algorithm. The demand of location/range information is not too much a constraint, given recent proposals for node localization in 3-D WSNs (e.g., [27] and [31]), as well as the fact that many events to be monitored are in open spaces and thus amenable to GPS localization.

To cope with the notched boundaries of a 3-D WSN, our approach is inspired by the principle that a transformed domain may offer features absent in the original domain. For example, a 3-D (implicit) surface $xyz = \mathrm{c}$ is concave, but a logarithmic transformation makes it convex in the transformed domain: $x'+y'+z' = \mathrm{c}'$ with $(\cdot)' = \log(\cdot)$. Therefore, the essence of our proposal is to find a simple transformation that "unfolds" the notched boundaries into convex ones, such that we can avoid the troublesome α-shape construction and rely on a simple convexity test to locally detect boundary instead.

III. UNFOLD: Detecting Boundary in a "Mirror" Image

In this section, we first introduce our network model and give a brief overview of UNFOLD. Then we describe the transformation used by UNFOLD to transform a point cloud (geometrically representing a WSN), as well as our formal definitions of the boundaries for the point cloud. Finally, we present in detail the localized algorithm for boundary detection, along with its performance analysis.

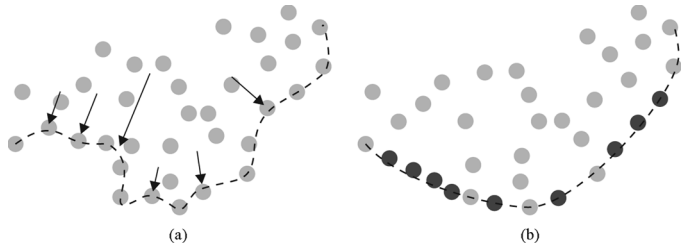


Fig. 2. Transformation that "blows up" a boundary. The length of a certain arrow in the left figure shows the "force" applied to that part of the boundary. The darker nodes in the right figure are those being blown up to the new convex boundary.

A. UNFOLD in a Nutshell

We model a 3-D WSN as a *point cloud* $\mathcal{P} = \{p_i | 1 \leq i \leq n\} \subset \mathbb{R}^3$, with each point p_i representing a sensor node. We are not concerned with network topology, so the boundaries that we aim at detecting are purely geometric and only concern local network connectivity. We assume that each node has a convex *transmission volume*, such that a bidirectional communication link exists between this node and any other node within this volume. We denote by $\mathcal{N}_i$ the set of nodes within the transmission volume of node p_i, or p_i's *one-hop neighbor set*. We also assume that node p_i is either aware of its geographic location or can measure the distance between itself and another node in $\mathcal{N}_i$. In this paper, we follow the definition of orienting a boundary surface in [18, pp. 611–615] and draw a distinction between external and internal boundaries. Intuitively, the (unique) external boundary encloses the whole WSN, whereas an internal boundary encloses a "hole" inside the WSN (so it is also termed *hole boundary* [32]). Similar to the existing work [31], [32], [35], we assume that these boundaries are all of genus-0.

As explained in Section II-B, the difficulty of determining whether a node is on the boundary stems from the existence of notches and, more importantly, from the *absence of convex boundary* due to consecutive notches; one would need to rely on the fairly complicated α-shape construction to identify boundary nodes. Our UNFOLD applies a special transformation to "blow up" a boundary such that it becomes almost convex, and we define a boundary node based on its local convexity. We illustrate the idea by Fig. 2.

Given such a transformation, the localized algorithm for UNFOLD to perform boundary detection becomes straightforward. Each node periodically exchanges location or range information with its one-hop neighbors to construct a local coordinate system. Then, the transformation is applied to the coordinates of all nodes in $\mathcal{N}_i \cup \{p_i\}$, possibly from different directions. In the transformed domain, node p_i performs a convexity test to check if it is on or out of the convex hull of $\mathcal{N}_i$, and it indicates itself as a boundary node if this test succeeds. Relevant questions we need to address are the following.

- Q1) What are the properties of the boundaries resulting from UNFOLD?
- Q2) How many transformations need to be applied?

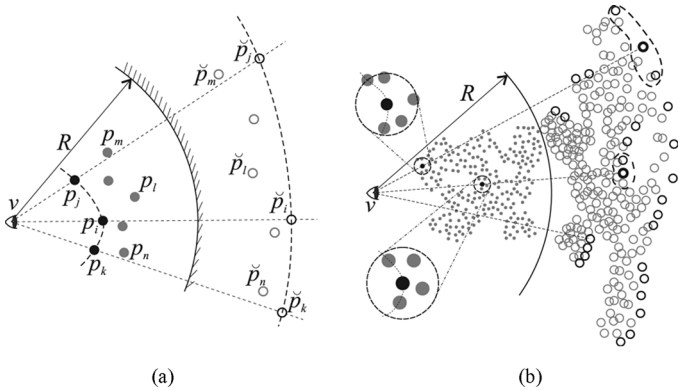


Fig. 3. Illustrating the effect of $f(\cdot)$ (defined by the viewpoint v and the "mirror" with radius R) on point clouds. For $\mathcal{N}_i$ of an arbitrary point p_i, the transformation effectively "bends" the locally concave boundary to a convex one in the transform domain (a). Given a point cloud (representing a WSN) $\mathcal{P}$ in (b) and let $\mathcal{P}' = \mathcal{P}$, we show the image $f(\mathcal{P})$ in the transform domain, along with part of the detectable boundary. As demonstrated by the two amplified one-hop neighborhoods, points on nonconvex surfaces (of the original 2-D area) are also detected as boundary points. (a) Local effect. (b) Global effect.

Q3) What if the location/range information for individual nodes comes with errors?

In the following, we present detailed principles and algorithms for UNFOLD while addressing these questions.

B. Transformation and Boundary Definition

As we mentioned in Section II-B, there is no commonly agreed definition for the boundary of a point cloud. Our definition is based on the assumption that a point cloud results from sampling a certain hypothetical 3-D volume. Therefore, a point is on the *boundary* of the point cloud if it lies on the hypothetical *surface* of that volume. As such a surface is unknown, we borrow the idea of *direct visibility* from [20]: A surface (hence the points lying on it) is what we can see from a certain viewpoint. We first present a transformation that enables the recognition of points on such a surface, then we define boundary and its properties.

1) Definitions: Given any subset $\mathcal{P}' \subseteq \mathcal{P}$, we associate with $\mathcal{P}'$ a local coordinate system, and we place a *viewpoint* v that does not belong to the convex hull of $\mathcal{P}'$ at the origin and set a *spherical* "*mirror*" centered at v with radius R (where $R > \|p_i\|, \forall\ p_i \in \mathcal{P}'$). The transformation for a point $p_i \in \mathcal{P}'$ is given by

$$\breve{p}_i = f(p_i) = p_i + 2(R - \|p_i\|)\frac{p_i}{\|p_i\|} \tag{1}$$

where $\|\cdot\|$ can be any norm, but we take Euclidean norm $\|\cdot\|_2$ in this paper. We illustrate the effect of this "spherical reflection" in Fig. 3.

Our definition of point cloud boundary is based on the transformation $f(\cdot)$.

Definition 1: For any point $p_i \in \mathcal{P}' \subseteq \mathcal{P}$, p_i is said to be *on the boundary* of $\mathcal{P}'$ with respect to a *common* transformation $f(\cdot)$ (defined by v and R) if $\mathcal{N}_i \subseteq \mathcal{P}'$ and $\breve{p}_i$ lies on the convex hull of $f(\mathcal{N}_i) \bigcup \{v\}$.

We refer to Fig. 3 for the two extreme cases of $\mathcal{P}' = \mathcal{N}_i$ and $\mathcal{P}' = \mathcal{P}$. Note that as the definition concerns the image of $\mathcal{N}_i$ in the transform domain, it focuses only on the local property of a point cloud. Actually, we have another definition of boundary for which even the transformation is made local:

Definition 2: For any point $p_i \in \mathcal{P}' \subseteq \mathcal{P}$, p_i is said to be *on the boundary* of $\mathcal{P}'$ with respect to a *particular* transformation $f_{p_i}(\cdot)$ (defined by v_{p_i} and R_{p_i} that depend on p_i) if $\mathcal{N}_i \subseteq \mathcal{P}'$ and $\breve{p}_i$ lies on the convex hull of $f_{p_i}(\mathcal{N}_i) \bigcup \{v_{p_i}\}$.

While the latter definition allows the transformation to be adapted to local geometry (hence requires only local range information), the former definition (which requires location information) may have certain practical significance.[2] In general, these two definitions, on one hand, are both amenable to the design of localized algorithm. On the other hand, the local properties stated in both definitions do have a global implication to some extent, according to the following result.

Proposition 1: If p_i is a boundary point of $\mathcal{P}$ according to either definition, there exists a *nontrivial* $\mathcal{P}'$, i.e., $\mathcal{N}_i \subset \mathcal{P}' \subseteq \mathcal{P}$, and a transformation $f'(\cdot)$, such that $f'(p_i)$ lies on the convex hull of $f'(\mathcal{P}') \bigcup \{v'\}$. In particular, p_i is an *external* boundary point of $\mathcal{P}$, iff $f'(p_i)$ lies on the convex hull of $f'(\mathcal{P}) \bigcup \{v'\}$.

The transformation $f(\cdot)$ shown in (1) is inspired by an inversion applied in a quite different context [20]. Katz *et al.* [20] applies this inversion to identify *visible* points on part of the external boundary of a point cloud. Our extension involving only local convexity test, however, allows boundary detection for both internal and external boundaries of a point cloud from all directions.

2) Properties: First, it is straightforward to show that the boundary definitions preserve convexity, a safety property.

Proposition 2: If p_i is on the convex hull of $\mathcal{P}' \subseteq \mathcal{P}$ and $\mathcal{N}_i \subseteq \mathcal{P}'$, then p_i is on the boundary of $\mathcal{P}'$ with respect to a certain transformation $f(\cdot)$.

In other words, if a point is on a (internal or external) surface of the hypothetical volume represented by $\mathcal{P}$ and is definitely "visible" (due to locally convexity) from some viewpoint, it will be recognized as a boundary node.

Second, our definition assures that a node that is hidden behind a boundary node from the viewpoint v will not be recognized as a boundary node, i.e., the following proposition.

Proposition 3: If p_i is recognized as a boundary node for $\mathcal{P}' \subseteq \mathcal{P}$ with respect to a certain transformation $f(\cdot)$, $p_i' \in \mathcal{N}_i$ will not be recognized as a boundary node if v, p_i, p_i' are collinear. This is another safety property.

The third property quantifies the impact of the transformation on a notched surface to "blow it up." We first define quantities to measure to what extent a point is *notched* locally.

Definition 3: Given $p_i \in \mathcal{P}$ and $\mathcal{N}_i$, if p_i lies within the convex hull, $\mathbf{conv}(\mathcal{N}_i)$, of $\mathcal{N}_i$, we define the *Depth of Notch* (DoN) d of p_i as the distance from p_i to the nearest plane on $\mathbf{conv}(\mathcal{N}_i)$, and the *inner radius* ρ of $\mathcal{N}_i$ as the distance from p_i to the nearest vertex of $\mathbf{conv}(\mathcal{N}_i)$.

We illustrate the definition of DoN in Fig. 4(a), and we show that, given $f(\cdot)$ (with certain v and R), a node can be "blown" to

[2]Taking the recent BP oil spill as an example, should a WSN be deployed to monitor the spill coverage, a question some Miami tourism authority might ask would be: How far is the spill frontline toward Key West? As the concerned boundary is only the section facing Key West, it makes more sense to perform a boundary detection based on a common viewpoint at Key West.

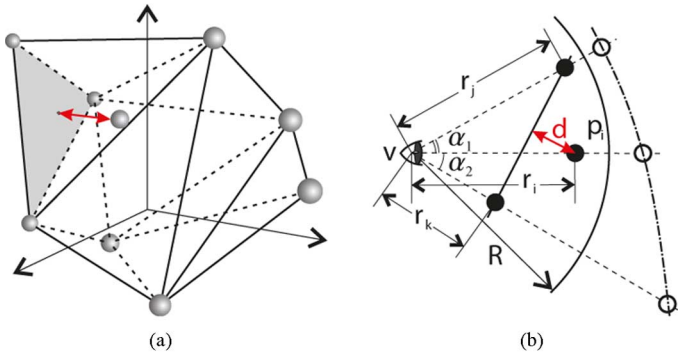


Fig. 4. (a) Definition of DoN is shown in 3-D. However, to simplify the interpretation, we analyze the 2-D case in (b), which can be considered as a projection of the 3-D case on a certain plane.

the convex hull only if its DoN is below a certain threshold. The proposition, illustrated by Fig. 4(b), is stated in 2-D for brevity, but it can be readily extended to 3-D, where the 2-D conditions need to be satisfied for three nonidentical planes containing v and p_i, and we sketch the idea in the Appendix.

Proposition 4: For a given $f(\cdot)$, p_i is recognized as a boundary node for $\mathcal{P}' \subseteq \mathcal{P}$ iff its DoN d satisfies

$$d \leq \frac{r_i\left(r_j \sin\alpha_1 + r_k \sin\alpha_2\right) - r_j r_k \sin(\alpha_1+\alpha_2)}{\sqrt{r_j^2 + r_k^2 - 2r_j r_k \cos(\alpha_1+\alpha_2)}}$$

where $r_i \leq 2R - \frac{(2R-r_j)(2R-r_k)\sin(\alpha_1+\alpha_2)}{(2R-r_j)\sin\alpha_1 + (2R-r_k)\sin\alpha_2}$. If $\alpha_1 = \alpha_2 = \alpha, r_j = r_k = r$ in particular, $d \leq 2R(1-\cos\alpha)$.

Given a common viewpoint transformation (*Definition 1*), (r_i, r_j, r_k) and (α_1, α_2) are fixed to each point, but R can be tuned for different requirements on compensating d. If a transformation is chosen for individual points (*Definition 2*), all the parameters of the transformation can be tuned.

Finally, we answer Q2 raised in Section III-A by showing that no matter which direction a boundary surface is facing, only a constant number of transformations are needed to recognize points on this surface.

Proposition 5: If p_i is on a boundary (internal or external) surface of $\mathcal{P}$ and it satisfies $\frac{d}{\rho} < \frac{1}{2}$, at most *four* transformations (hence four viewpoints) are needed to recognize p_i as a boundary point. Here, ρ is the inner radius of $\mathcal{N}_i$ defined in *Definition 3*.

Intuitively, these four viewpoints are the vertices of a tetrahedron that encloses $\mathbf{conv}(\mathcal{N}_i)$.

Summarizing *Propositions 2–5*, our boundary definitions based on $f(\cdot)$ have the following three properties:

1) *Safety I:* Points that are surely on the boundary defined by the convex hull of $\mathcal{P}$ will be identified as boundary points.
2) *Safety II:* Points that are surely not on any boundary defined by the already identified boundary points and visible from v will not be identified as boundary points.
3) *Tunable liveness:* Points on a nonconvex surface but with reasonable DoN can be identified with a small number of viewpoints (hence transformations).

Remark: It is important to note that for a hypothetic surface passing through p_i, the ratio $\frac{d}{\rho}$ is actually a discrete indicator of its curvature at p_i. As we set an upper bound for $\frac{d}{\rho}$, the identified boundary points form a boundary surface that is low-curvature filtered from the real boundary surface. Moreover, it is possible that many nodes are identified as boundary nodes in extremely sparse 3-D WSNs because the condition $\frac{d}{\rho} < \frac{1}{2}$ may often be satisfied. Therefore, our approach performs well in relatively dense WSNs. This is actually a common feature for localized detection mechanisms, as they are all not concerned with global topology.

Algorithm 1: CVX–TEST

Input: $f(\mathcal{N}_i)$, $\breve{p}_i$, a coord. system with v as the origin
Output: Binary indicator $boundary(\breve{p}_i)$

1 **forall the** *distinct node pair* $m_1, m_2 \in f(\mathcal{N}_i)$ **do**
2 $\vec{n} \leftarrow (\vec{p}_i - \vec{m}_1) \times (\vec{p}_i - \vec{m}_2)$; $\vec{n} \leftarrow \frac{\vec{n}}{\|\vec{n}\|}$; $b = \vec{p}_i \cdot \vec{n}$
3 $boundary(\breve{p}_i) \leftarrow$ **true**
4 **forall the** $m \in f(\mathcal{N}_i) \backslash \{m_1, m_2\}$ **do**
5 **if** $\vec{m} \cdot \vec{n} > b$ **then** $boundary(\breve{p}_i) \leftarrow$ **false**; **break**
6 **if** $boundary(\breve{p}_i) =$ **true then break**

C. Online Boundary Detection Algorithm

Given the definitions of transformation and boundary, we are ready to present our UNFOLD for online boundary detection. UNFOLD involves mainly *three steps for each node* in a WSN.

1) Local Interactions: Each node p_i exchanges location or range information with its neighbors in $\mathcal{N}_i$ to construct a local coordinate system. This step is trivial if the location information is available (through, for example, [27] and [31]); otherwise, a certain 3-D embedding algorithm (e.g., [30]) is used to create the coordinate system using mutual distances. The origin of the coordinate system is the viewpoint v.

1) If the location information is available, we could afford to have a common viewpoint v (*Definition 1*), which can be required by certain applications (such as the BP oil spill example we gave in Footnote 2).
2) Viewpoint v_{p_i} (*Definition 2*) specific to every p_i can always be applied. This is preferred if only range information is available, as otherwise we have to perform a costly procedure for constructing a global coordinate system with local range information (e.g., [15]).

2) Transformation: Having the coordinates for all nodes in $\mathcal{N}_i \bigcup \{p_i\}$ with respect to a origin v (or v_{p_i}), node p_i applies the transformation given in (1) to these coordinates and obtains their images. This step involves only simple computations.

3) Convexity Test: Each node p_i performs a convexity test to decide whether or not its image $\breve{p}_i$ is on the convex hull of $f(\mathcal{N}_i) \bigcup \{v\}$ (or $f_{p_i}(\mathcal{N}_i) \bigcup \{v\}$). A basic algorithm is shown in *Algorithm 1*. We use $\vec{x}$ to represent the vector form of a point $\breve{x}$. The idea is to try all possible planes determined by $\breve{p}_i$ and another two points in $f(\mathcal{N}_i)$ (lines 1–3), and to check whether one of them is a *supporting plane* for $f(\mathcal{N}_i) \bigcup \{v\}$ (lines 4–5), i.e., if $f(\mathcal{N}_i) \bigcup \{v\}$ lies on one side of that plane.

To improve the efficiency, we apply the *divide-and-conquer* strategy. Observe that, after each inner loop (lines 4–5), points in $f(\mathcal{N}_i)$ can be totally ordered under $\geq$, according to their dot products with the normal $\vec{n}$. Therefore, instead of arbitrarily choosing a (m_1, m_2) pair, we replace only one of the current

two points with what is ordered first in $f(\mathcal{N}_i)$ (ties broken arbitrarily), such that the replaced point lies on the same side as v with respect to the new plane. In fact, no sorting is needed; the maximum point is naturally obtained at the end of each inner loop (lines 4–5). The algorithm terminates if neither points can be replaced: either because replacing either of them separates another from v, or because the node ordered first in $f(\mathcal{N}_i)$ lies on the current plane (i.e., the plane is a supporting plane). The algorithm returns $boundary(p_i) = \mathbf{true}$ if the current plane is a supporting plane, and returns $boundary(p_i) = \mathbf{false}$ otherwise. We call this enhanced algorithm CVX-TEST-DC. Note that the algorithm is conducted by individual nodes without the need for time synchronization. Therefore, UNFOLD is a *localized* algorithm requires only asynchronous operations; this makes UNFOLD extremely efficient.

D. Performance Analysis

We analyze the performance of UNFOLD on two aspects: namely time complexity of UNFOLD that also represents the energy efficiency of the algorithm and the robustness of UNFOLD against location or range errors.

1) Complexity Analysis: Our analysis on UNFOLD focuses on the transformation and convexity test steps, as the first step either has a negligible complexity if location information is available or otherwise involves well-known procedures that are commonly applied in other proposals (e.g., [35]).

Assuming $\eta = |\mathcal{N}_i|$, the complexity of the transformation is obviously $\Theta(\eta)$, as we basically apply the transformation (1) to all nodes in $\mathcal{N}_i \bigcup \{p_i\}$. The complexity of the basic convexity test, CVX-TEST, is also obvious: As the outer iteration has $\eta(\eta-1)$ loops and the inner iteration has $\eta-2$ loops, the complexity is $\Theta(\eta^3)$. Consequently, the complexity (both average and worst-case) [5] of UNFOLD with CVX-TEST is $\Theta(\eta^3)$. Although this is same as that of the α-shape based boundary detection [35], the actual CPU time (hence energy consumption) of UNFOLD is much less (as shown in Section V-B). This is because the convexity test involves only vector operations (which are basically arithmetics), whereas α-shape construction entails complicated operations/procedures such as square root and solving equation systems. Moreover, we may further reduce the complexity of UNFOLD by applying the enhanced convexity test: CVX-TEST-DC.

Proposition 6: The average-case complexity of CVX-TEST-DC is $\mathcal{O}(\eta \log \eta)$.

This complexity holds under the condition that the input is introduced in a random order, and it effectively reduces the average-case complexity of UNFOLD to $\mathcal{O}(\eta \log \eta)$. Though the worst-case complexity is $\Theta(\eta^2)$, those cases rarely happen based on our experience.

2) Error Analysis: Given the fact that the geometric boundary of a point cloud is not well defined, it is impossible to perform error analysis rigorously, as there is no *ground truth* to which to be compared. One may be able to create a set of artificial "ground truth" boundary points in simulations by deliberately sampling on the surface of the volume from which a point cloud is derived (which is the method that we will apply in Section V-A to evaluate the robustness of UNFOLD). However, unless those points are sampled extremely dense, there are still chances that certain "under the surface" points are detected as boundary points, regardless of which boundary detection mechanism is used. It is definitely unreasonable to categorize these points as detection errors, as they may well be on the surface of another volume that results in the same point cloud. Consequently, the error analysis we discuss here is rather *qualitative*.

If location information is available to every node, the error can be characterized by a small ball with radius ε around the expected location of the node, where ε can be the *mean square error* resulting from certain localization mechanism (e.g., GPS). If only range information between neighboring nodes is available, the initial errors come from the given ranging technique. However, this error will eventually be translated into location errors through a 3-D embedding algorithm (e.g., [30]). In either case, ε cannot be too large compared to the radius of $\mathbf{conv}(\mathcal{N}_i)$, as otherwise it could be corrected (thus reduced) using local connectivity relations. Therefore, we may safely assume that ε is bounded by the radius of $\mathbf{conv}(\mathcal{N}_i)$.

For a boundary node (based on our definitions), the location errors may either decrease or increase DoN, if we consider a node p_i on or out of its local convex hull $\mathbf{conv}(\mathcal{N}_i)$ as having a nonpositive DoN. Apparently, a decreased DoN has no impact on boundary detection, whereas an increased DoN is somewhat compensated by our transformation, according to *Proposition 4*. For a nonboundary node, there are two cases: either it is very close to a boundary node, or it indeed lies in the very interior of the point cloud. The former case, compared to a boundary node, is inversely affected by location errors. As the node is anyway close to a boundary node, a false positive does not really compromise boundary detection. The latter case can hardly be affected by location errors, given the boundedness of these errors. In summary, UNFOLD is very robust against location errors; we will demonstrate this in Section V-A.

IV. LBS and Its Applications

The outcome of our boundary detection algorithm is that every node may tell whether it is on some boundary or not. However, such individual boundary awareness is not sufficient in supporting some WSN functionalities. In this section, we propose LBS as a localized boundary parametrization technique that maps each boundary surface to a sphere. This allows three immediate applications: distinguishing internal boundaries from the external one, enabling 3-D greedy geographic routing, and supporting surface routing (hence distributed coordinations) on a certain boundary.

One major limitation of LBS is that it only works for boundary surfaces of genus-0. As a surface of higher genus always has negative Gaussian curvature for part of it, it is not topologically equivalent to a sphere. Coping with boundaries of arbitrary topology is still an open problem that remains to be tackled in our future work.

A. Boundary Sphericalization

Let $\mathcal{B}$ be a boundary of $\mathcal{P}$. By our assumption, it is of genus-0, i.e., topologically equivalent to a unit sphere $\mathcal{S}^2$. Our goal is to parameterize $\mathcal{B}$ to $\mathcal{S}^2$, for which we need to construct a bijective

Algorithm 2: Localized Boundary Sphericalization

Input: For each $p \in \mathcal{P}$, the stopping tolerance ϵ
Output: A harmonic function $\phi : \mathcal{B} \to \mathcal{S}^2$

1 For every boundary node p, compute its normal vector $\mathbf{n}(p)$, and initialize $\phi(p) = \mathbf{n}(p)$
2 For every boundary node p periodically:
3 **if** $\| \triangle \phi(p)_{\|} \| > \epsilon$ **then**
4 $\quad \triangle\phi(p) = \phi(p) - \sum_{(p,q)\in E_{\mathcal{B}}} \omega_{pq}\phi(q)$
5 $\quad \triangle\phi(p)_{\|} = \triangle\phi(p) - \langle \triangle\phi(p), \mathbf{n}(p)\rangle \mathbf{n}(p)$
6 $\quad \phi(p) \leftarrow \phi(p) - \triangle\phi(p)_{\|}$
7 $\quad$ Normalize $\phi(p)$
8 $\quad$ Broadcast $\phi(p)$ to all $q : (p, q) \in E_{\mathcal{B}}$

map between $\mathcal{B}$ and $\mathcal{S}^2$. This can be done by solving a harmonic function $\phi : \mathcal{B} \to \mathcal{S}^2$

$$\frac{\partial \phi}{\partial t} = - \triangle \phi_{\|} \tag{2}$$

where $\triangle\phi(p)_{\|} = \triangle\phi(p) - \langle \triangle\phi(p), \mathbf{n}(p)\rangle \mathbf{n}(p)$ is the tangent component of $\triangle\phi$, $\mathbf{n}(p)$ is the normal vector of vertex p and $\langle , \rangle$ is the dot product. Intuitively speaking, the Laplace $\triangle\phi \in \mathbb{R}^3$ is a 3-D vector that, in general, is not on the tangent plane of the sphere, $\triangle\phi_{\|}$ is the projection of $\triangle\phi$ to the tangent plane. Thus, the image of vertex $\phi(p)$ is only allowed to diffuse along the tangential direction of the sphere. The function ϕ is harmonic if and only if its tangential component is zero, i.e., $\triangle\phi_{\|} = 0$. In differential geometry, it is known that the above harmonic function $\phi : \mathcal{B} \to \mathcal{S}^2$ is a conformal mapping, i.e., bijective, C^{∞} smooth and angle preserving [16].

In our case, the boundary is sampled by a set of points (the boundary nodes being detected). Therefore, we first reconstruct the surface by a localized triangulation [35]; it leads to an edge set $E_{\mathcal{B}}$. Then, we propose a localized algorithm to perform a diffusion process that solves (2), as shown by *Algorithm 2*. Line 1 in *Algorithm 2* initializes the map $\phi : \mathcal{B} \to \mathcal{S}^2$ by the Gauss Map [4], which provides a mapping from every vertex on $\mathcal{B}$ to a corresponding point on a unit sphere $\mathcal{S}^2$. Note that this initial map is not bijective in general since $\mathcal{B}$ may be concave. Line 4 is to compute the Laplacian of $\phi(p)$, where $\triangle$ is the discrete Laplace–Beltrami operator and ω_{pq} is the weight defined for edge $(p, q) \in E_{\mathcal{B}}$. A commonly used scheme is the uniform weight, i.e., $\omega_{pq} = \frac{1}{deg(p)}$, where $\deg(p)$ is the degree of p. Then, line 5 is to compute the tangential component of $\triangle\phi(p)$ followed by the diffusion along the tangential direction in line 6. Next, line 7 normalizes $\phi(p)$ to ensure that the image is on the unit sphere. The above diffusion process is repeated until the tangential component of $\triangle\phi(p)$ is less than the user-specified threshold ϵ.

B. Applications

1) Distinguishing Internal From External: Telling internal boundaries from external one usually requires global information on a 3-D mesh. As localization in WSNs often needs a boundary detection mechanism that can distinguish the two types of boundaries [31], it benefits from having a localized algorithm that achieves this goal. Our UNFOLD + LBS makes it fairly easy to distinguish internal and external boundaries on the parametric domain by using only local information.

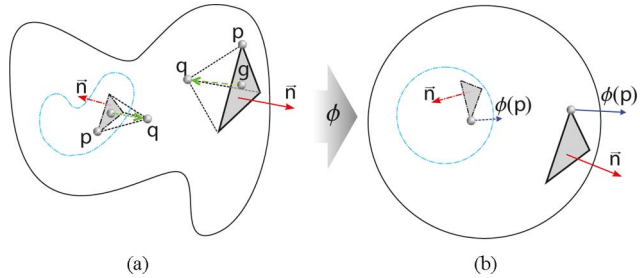


Fig. 5. Rationale of distinguishing internal boundaries from the external one for 3-D WSNs. Though both boundaries should be unit spheres under ϕ, we abuse the size a bit to differentiate them. (a) Original boundaries. (b) Parameterized boundaries.

Let $p \in \mathcal{P}$ be an arbitrary boundary node with spherical coordinates $\phi(p) \in \mathcal{S}^2$. Find a boundary triangle $\mathbf{t}$ such that $p \in \mathbf{t}$. Without loss of generality, $\mathbf{t}$ is determined by the tuple (p, m_1, m_2) that terminates Algorithm 1. Let $\mathbf{h}$ be the unique tetrahedron[3] with $\mathbf{t}$ as the boundary face. We denote $q \in \mathbf{h}$, $q \notin \mathbf{t}$ the opposite vertex of face $\mathbf{t}$, as shown by Fig. 5(a). Let g be the geometric center of $\mathbf{t}$ and $\overrightarrow{n}$ be the normal of $\mathbf{t}$. The direction of the normal $\overrightarrow{n}$ is defined according to the order of the three vertices of $\mathbf{t}$: $\overrightarrow{n}$ points to the outward of the tetrahedron $\mathbf{h}$, i.e., $\overrightarrow{gq} \cdot \overrightarrow{n} < 0$.

Note that each vertex of $\mathbf{t}$ has been mapped to the unit sphere $\mathcal{S}^2$ by ϕ resulting from LBS (see Algorithm 2). Let $\overrightarrow{n}_{\phi(\mathbf{t})}$ be the normal of the $\phi(\mathbf{t}) \in \mathcal{S}^2$, where the order of the three vertices of $\phi(\mathbf{t})$ is the same as $\mathbf{t}$. Then, the node p is on the external boundary if and only if $\phi(p) \cdot n_{\phi(\mathbf{t})} > 0$, as shown in Fig. 5(b); otherwise, p is on the internal boundary. Intuitively speaking, this means that a boundary is external (resp. internal) if there exists a node inside (resp. outside) of it. Given an arbitrary boundary, this inside or outside relation cannot be recognized in a localized manner. It is our LBS that makes such an intuitive recognition procedure possible.

2) Supporting Geographic Routing: LBS may serve as a building block for performing greedy geographic routing with a guaranteed delivery. In a nutshell, greedy routing may fail when facing an internal boundary (the boundary of a hole) due to the so-called "local minimum phenomena" [10]. Our solution relies on LBS to regularize the internal boundary and hence to eliminate the local minimum phenomena. As this aspect is presented in a companion work, we refrain from providing details here. Instead, we briefly discuss a geographic routing on the boundary surface. As boundaries (especially event boundaries) are important from data collection point of view (because those are where information is produced), data routing as well as distributed coordinations (e.g., constructing quorum systems for data sharing [25], [33]) among nodes that share the same boundary are crucial to WSN functionalities.

Given the virtual coordinates of the boundary nodes on a sphere, the shortest path routing between any two nodes follows the great circle that is uniquely determined by the locations of these nodes and the sphere center. Assuming the coordinate of the sphere center is $[0, 0, 0]^T$ and those of the two nodes are $\vec{p}_1 = [x_1, y_1, z_1]^T$ and $\vec{p}_2 = [x_2, y_2, z_2]^T$, such a great circle

[3]It stems from the uniqueness of Delaunay triangulation for general point set $\mathcal{P}$.

has a very simple representation by the following parametric equations:

$$\begin{cases} x = x_1 \cos t + \tilde{x}_2 \sin t \\ y = y_1 \cos t + \tilde{y}_2 \sin t \\ z = z_1 \cos t + \tilde{z}_2 \sin t \end{cases}$$

where $[\tilde{x}_2, \tilde{y}_2, \tilde{z}_2]^T = (\vec{p}_1 \times \vec{p}_2) \times \vec{p}_1$ and $\times$ is the cross product between two vectors. The routing path can be specified by the source node, and a trajectory-based forwarding [28] is then applied to follow the parametric curve (the great circle) until reaching the destination.

V. Simulations and Experiments

We first evaluate UNFOLD through both simulations and experiments. With simulations, we demonstrate the efficacy of UNFOLD in large-scale WSNs. Through experiments based on an implementation in MICAz Motes, we confirm UNFOLD's superiority in efficiency over the most up-to-date proposal [35]. We have also implemented LBS in TinyOS [1], and we illustrate the convergence of LBS and also evaluate the performance of LBS in terms of convergence time through simulations with TOSSIM [22].

A. Simulations for UNFOLD

We first construct several 3-D volumes to represent the physical events to be monitored. Then, we randomly deploy sensor nodes in each volume and on the volume surface. As explained in Section II-C, we may treat network and event boundary equivalently without loss of generality. Therefore, the goal of our simulations is to verify if UNFOLD can correctly identify those points that we have deliberately put on the volume boundaries (which, according to Section III-D.3, is an inevitable but artificial setting). Although UNFOLD applies to arbitrary convex transmission volumes, we assume a regular volume, a ball with radius r, for every node in order to simplify our presentation. We choose the value of r such that the average size of $\mathcal{N}_i$ is about 40.

The implementation of UNFOLD in our simulator follows the protocol description in Section III-C. Specifically, each node first exchanges location information with its neighbors. As our high-level simulator neglects the MAC effect, this information exchange is considered to be reliable. In practice, the reliability can be achieved through, for example, ARQ. After collecting the neighbor information, all the computations left for a node are strictly localized: transformation and convexity test.

In Fig. 6, four examples of the simulated WSNs are shown. The WSNs, shown in the left column, are designed to exhibit various 3-D shapes we may face in atmospheric or ocean monitoring. We apply UNFOLD with either a common viewpoint or local viewpoints to perform boundary detection, and the results are shown in the central and right columns, respectively. It is clearly shown that, while a single common viewpoint only detects the boundaries that face the viewpoint, a few local viewpoints detect the whole boundary. Note that, though the inversion used in [20] only detects an external boundary, our extension also detects the internal boundary, as shown by the second and third WSNs.

We also evaluate the robustness of UNFOLD under location errors. We assume each node has an error radius ε, such that the location information available to a node may be uniformly distributed within a ball with radius ε and centered at the real location. We then vary ε to different fractions of r and apply UNFOLD to detect the boundary. In general, a node that is either a real boundary node or detected as a boundary node may have four states:

- Found: detected as a boundary node;
- Correct: a boundary node and also detected as one;
- Mistaken: not a boundary node but detected as one;
- Missing: a boundary node but not detected as one.

In Fig. 7(a), we report the statistics in terms of these four states based on all the WSNs that we have simulated. We also report the statistics on the distance (in hop) from a mistaken/missing node to the closest correct boundary node in Fig. 7(b) and (c). The same metrics are used in [35]; we reuse them to facilitate comparisons.

As shown in Fig. 7(a), both found and correct nodes decrease with an increasing error radius. However, even with the worst-case error $\varepsilon = r$, correct nodes still account from around 50% the total boundary nodes. Also, the distributions in Fig. 7(b) and (c) show that both mistaken and missing nodes are not far from the real boundary (at most two hops, but mostly within one hop). We finally provide one example, corresponding the first WSN in Fig. 6, in Fig. 7(d)–(f). The message conveyed by all these figures is clear: Although increasing location errors may marginally affect the detected boundaries (which is characterized by the found nodes), these boundaries still well characterize the geometry of the network volume. Compared to the statistics reported in [35], UNFOLD appears to be more robust against the location errors: higher correct nodes percentage and shorter distances (from the mistaken and missing nodes) toward the real boundary.

B. Implementation and Experiments for UNFOLD

We implement UNFOLD in MICAz Motes and compare UNFOLD to the Unit Ball Fitting (UBF) algorithm proposed in [35] in terms of CPU time for boundary detection. Our experiments focus only on local computation steps, as the local communications to exchange location/range information are common to both approaches. Therefore, we use the location information sampled from our simulated WSNs and inject these data to a MICAz Mote, so that we can directly start the actual boundary identification steps: transformation/convexity tests for UNFOLD, and ball tests for UBF. As the microcontroller of our MICAz (ATMEL ATMega 128L) is fully occupied (i.e., never in idle mode or being interrupted) during both computations, we may roughly use the CPU time to also represent the energy consumption spent for computations. Consequently, our experiments also compare UNFOLD to UBF in terms of energy efficiency.

We show two sets of results, internal and boundary nodes, in Figs. 8 and 9, respectively. The comparisons are made between UBF and two versions of UNFOLD that differ in the convexity test algorithms. As an internal node is often the worst case for all the three schemes, Fig. 8 serves as a confirmation of the complexity analysis (Section III-D.1). Because the complexity of

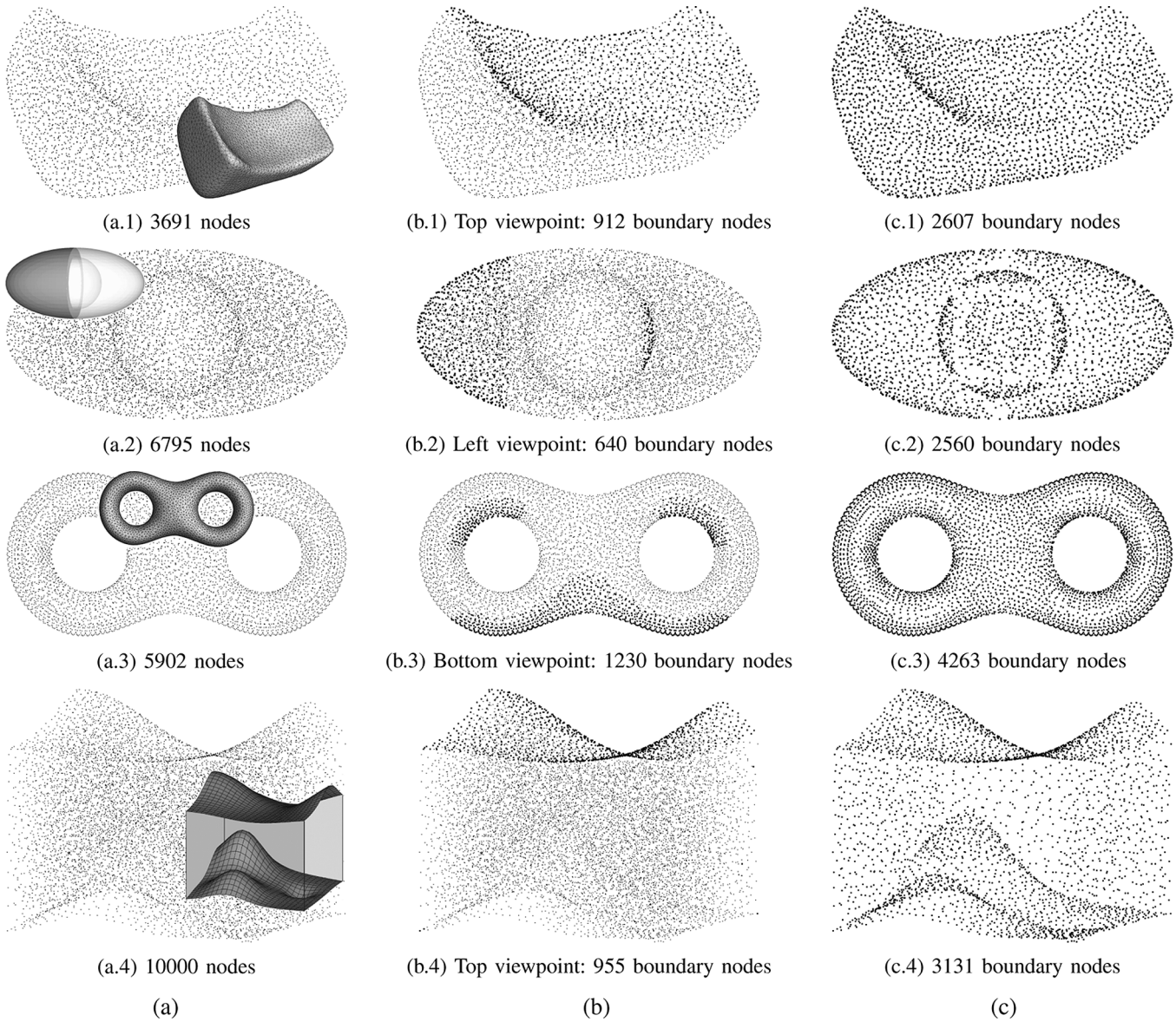


Fig. 6. Boundary detection through UNFOLD. (a) As the "shapes" of the WSNs may not be easily recognizable from their point clouds, we attach their original 3-D volumes to respective point clouds. (b) For boundary detection with a common viewpoint, boundary nodes are marked on top of the original clouds. (c) We, however, remove the internal nodes when local viewpoints are applied to detecting the whole boundary. (a) Original networks. (b) Boundary from a common viewpoint. (c) Boundary with local viewpoints.

both UBF and CVX-TEST are $\Theta(\eta^3)$, the more-than-10-times difference between them stems from the simple computations incurred by CVX-TEST. Another more-than-10-times improvement brought by CVX-TEST-DC follows from its $\mathcal{O}(\eta \log \eta)$ complexity.

For boundary nodes (Fig. 9), all the three schemes may face cases varying from the best to the worst. Therefore, the CPU times are rather dispersed, but the roughly 100-time advantage of CVX-TEST-DC over UBF still holds. Note that the fairly constant (and almost always the smallest) CPU times for CVX-TEST-DC mostly indicate the cost of the transformation $f(\mathcal{N}_i)$. In summary, UNFOLD with CVX-TEST-DC is far more time- and energy-efficient than UBF. In fact, with a CPU time of tens of seconds, UBF may not even be eligible for an online boundary detection.

C. Simulations for LBS

We have also implemented LBS in TinyOS. However, as the effect of LBS can only be observed collectively for relatively large-scale WSNs and our current MICAz-based testbed only has tens of nodes, we use simulations in TOSSIM (rather than experiments with our testbed) to evaluate the performance of LBS. Let π denote the period specified in Algorithm 2. We will evaluate the convergence time of LBS against different values of π.

We first illustrate the evolution of the diffusion process in Fig. 10. Given the original boundary (obtained by performing triangulation on the outcome of UNFOLD) shown in Fig. 10(a), the initial map on a unit sphere, Fig. 10(b), is rather messy due to the many nontriangular edges (those induced by the nonbijectivity, i.e., foldover, of the Gauss map). The number of such

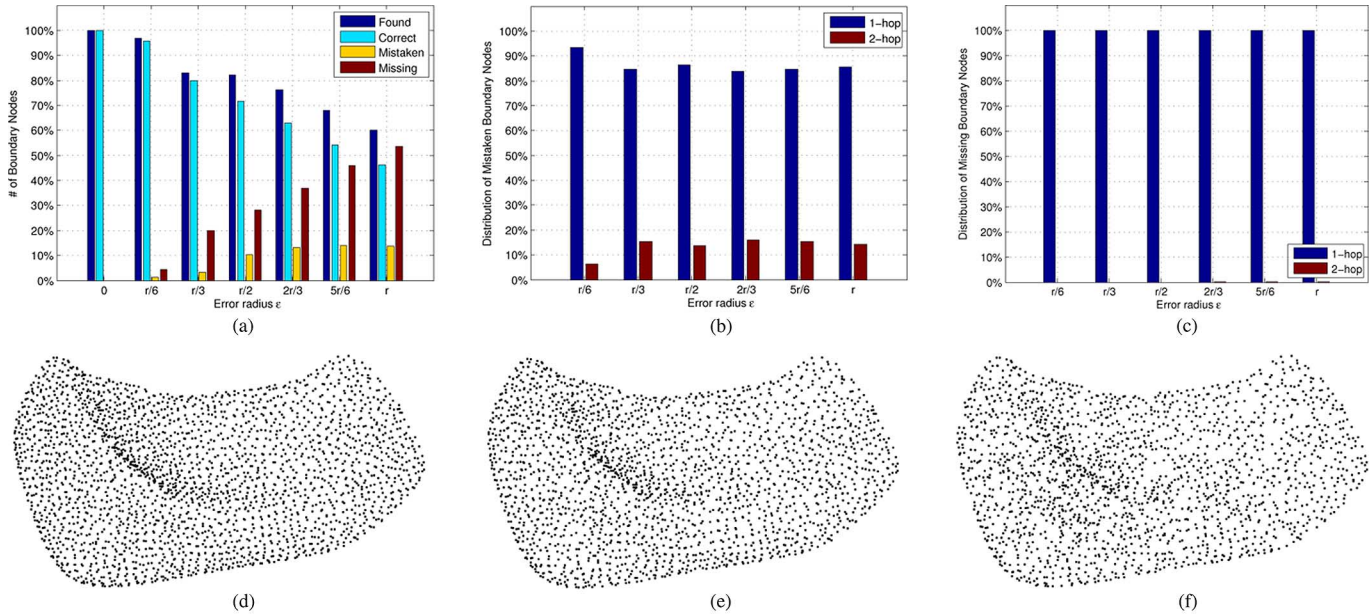


Fig. 7. Boundary detection under location errors. We vary the error radius ε as different fractions of the transmission range r, and we use both (a)–(c) statistics and (d)–(f) an example. (a) Algorithm efficiency. (b) Mistaken distribution. (c) Missing distribution. (d) $\epsilon = \frac{1}{6}r$: 2554 boundary nodes. (e) $\epsilon = \frac{1}{3}r$: 2189 boundary nodes. (f) $\epsilon = \frac{2}{3}r$: 1902 boundary nodes.

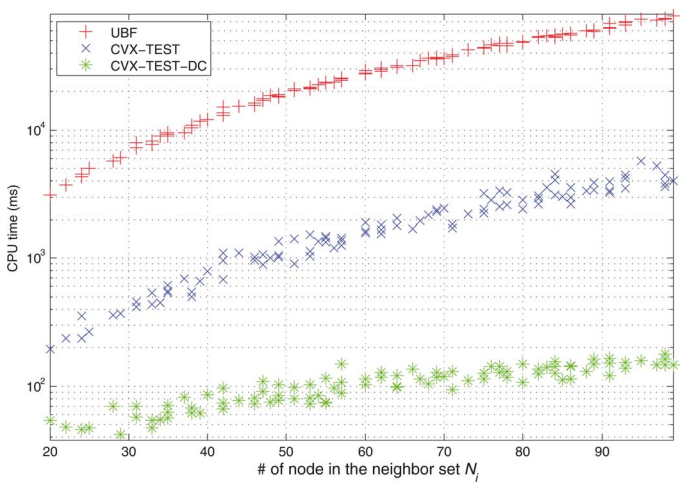


Fig. 8. CPU times for internal nodes.

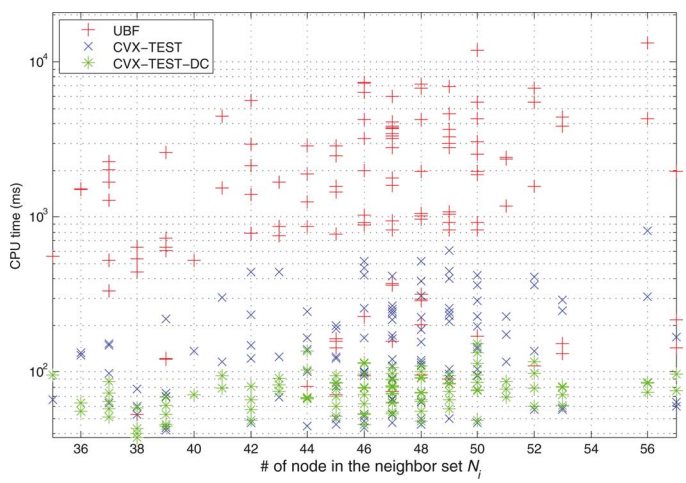


Fig. 9. CPU times for boundary nodes.

edges decreases with the number of execution rounds until no such edges exist, which is shown by Fig. 10(c)–(f).

We also show the impact of π on the convergence time in Fig. 11 for an 800-node WSN boundary. It is clear that the time to convergence increases monotonically in π (22, 3, and 2 s for $\pi = 100$, 10, and 5 ms, respectively), whereas the total number of rounds decreases with π (410, 300, 220 for $\pi = 100$, 10, and 5 ms, respectively). As the number of messages sent is proportional to the number of rounds, the algorithm becomes more energy-efficient but less time-efficient with large values of π. The intuitive explanation is the following. When π gets small, the potential (broadcast) packet collisions become intensive. Consequently, the information each node collects about its neighbors decreases, which in turn requires more rounds to terminate the diffusion process. Therefore, we have a clear tradeoff between the time complexity and the energy efficiency of LBS, and we may tune the value of π to obtain a required (by a certain application) balance between reducing latency and saving energy.

VI. Conclusion

We have investigated the challenging problems of online boundary detection and boundary regularization in 3-D WSNs, and we have proposed LBDP as a concrete solution to these problems. As one component of LBDP, UNFOLD significantly speeds up the boundary detection by performing it in a transformed domain. This makes it perfectly suitable for online boundary detection in 3-D WSNs that are deployed for monitoring time-variant physical events. We have demonstrated the efficiency and efficacy of UNFOLD through both simulations and experiments (using a MICAz-based testbed). Compared to an up-to-date proposal, UNFOLD is, on one hand, more robust

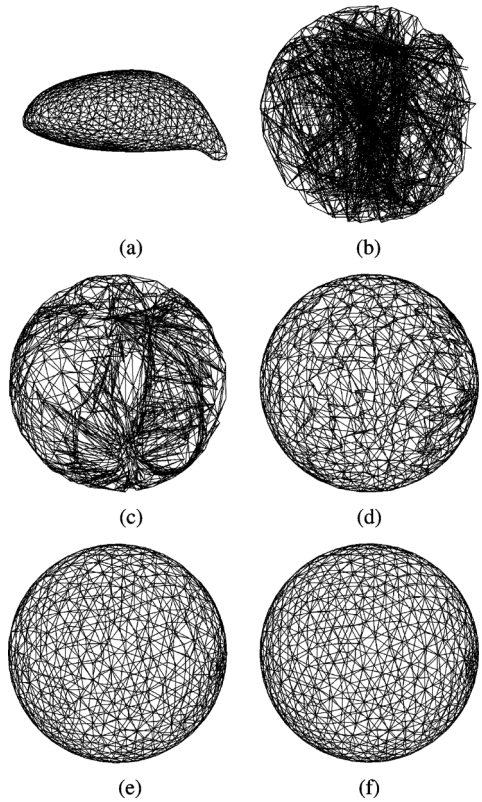


Fig. 10. Illustration of the diffusion process of LBS with about 800 nodes and $\pi = 0.1$ s. (a) Original surface. (b) Gauss map. (c) 10th round. (d) 82nd round. (e) 145th round. (f) 190th round.

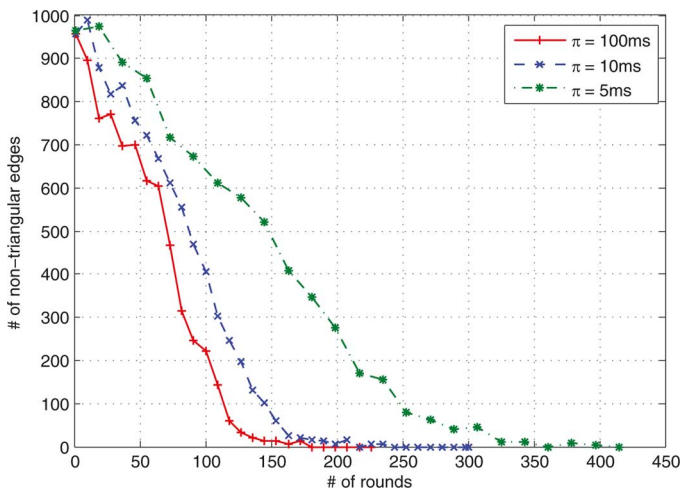


Fig. 11. Convergence of LBS under different values of π.

against the location errors, and on the other hand, UNFOLD is far more efficient in terms of both computation time and energy consumption. As another component of LBDP, LBS serves as a distributed parametrization procedure to regularize an arbitrary boundary surface. The resulting virtual coordinates may enable many WSN functionalities, such as distinguishing internal boundaries from the external one, as well as geometric routing on a boundary.

APPENDIX
PROOFS FOR THE PROPOSITIONS STATED IN THE PAPER

Proof of Proposition 1: Let p_i be on the boundary of $\mathcal{P}$ according to either *Definition 1* or *Definition 2*, and let v be the concerned viewpoint. Without loss of generality, we may expand $\mathcal{N}_i$ to a nontrivial $\mathcal{P}' \subseteq \mathcal{P}$ such that $\mathcal{N}_i \subset \mathcal{P}'$, simply by adding one more point, i.e., $\mathcal{P}' = \mathcal{N}_i \bigcup \{p'\}$. There are two possible cases: p' lies either *on* or *not on* the line segment between v and p_i. As $p' \notin \mathbf{conv}(\mathcal{N}_i)$ (otherwise we would have $p' \in \mathcal{N}_i$, given the convexity of the transmission volume), we can avoid the former case by choosing v very close to the boundary of $\mathbf{conv}(\mathcal{N}_i)$. For the latter case, we simply let $v' = v$. According to the "blowing-up" feature of the transformation $f(\cdot)$ (which will be quantified by *Proposition 4*), there always exists an $f'(\cdot)$ such that $f'(p_i)$ lies on the convex hull of $f'(\mathcal{P}') \bigcup \{v'\}$, as far as there is no point in $\mathcal{P}'$ that lies on the line segment between v' and p_i.

In particular, if p_i is on the external boundary of $\mathcal{P}$, it is "visible" from some viewpoint v'. Therefore, we can directly expand $\mathcal{N}_i$ to $\mathcal{P}$ and be sure that no $p' \in \mathcal{P}$ is on the line segment between v' and p_i. Consequently, there exists an $f'(\cdot)$ such that $f'(p_i)$ lies on the convex hull of $f'(\mathcal{P}) \bigcup \{v'\}$. Conversely, the existence of an $f'(\cdot)$ such that $f'(p_i)$ lies on the convex hull of $f'(\mathcal{P}) \bigcup \{v'\}$ suggests that no $p' \in \mathcal{P}$ is on the line segment between v' and p_i. Since $v' \notin \mathbf{conv}(\mathcal{P})$, p_i has to be on the external boundary of $\mathcal{P}$. Q.E.D.

Proofs of Propositions 2 and 3: It is straightforward to verify that the transformation $f(\cdot)$ preserves surface convexity. In other words, if part of the surface is described by a convex function, this convexity is preserved in the image of $f(\cdot)$. The fact that p_i is on the convex hull of $\mathcal{P}' \subseteq \mathcal{P}$ suggests that the surface through p_i is locally convex. Therefore, simply choosing a viewpoint v that faces that surface and applying a proper $f(\cdot)$ will lead to the claim made in *Proposition 2*.

It is also trivial to show that, if v, p_i, p_i' are collinear, then v, p_i, p_i', $f(p_i)$, $f(p_i')$ are also collinear. Therefore, at most one of p_i and p_i' can be recognized as a boundary node, hence the claim made in *Proposition 3* follows. Q.E.D.

Proofs of Propositions 4 and 5: We only sketch the proof of *Proposition 4* by omitting the tedious trigonometric derivations. To derive the upper bound for d, we consider the worst case where the images of the three points become collinear, as shown in Fig. 12(a), because further increasing d would compromise the convexity of $\breve{p}_i$. Using basic trigonometry, we obtain the relation between d and $(r_i, r_j, r_k, \alpha_1, \alpha_2)$ as

$$d \leq \frac{r_i\left(r_j \sin\alpha_1 + r_k \sin\alpha_2\right) - r_j r_k \sin(\alpha_1+\alpha_2)}{\sqrt{r_j^2 + r_k^2 - 2 r_j r_k \cos(\alpha_1+\alpha_2)}}.$$

Given $(r_j, r_k, \alpha_1, \alpha_2)$, d is increasing in r_i. Therefore, we can also bound d from above by bounding r_i. The key to subsequent derivations is: $\beta_1 + \beta_2 = \pi \Rightarrow \tan\beta_1 + \tan\beta_2 = 0$ in the worst case shown in Fig. 12(a). Using $(R, r_j, r_k, \alpha_1, \alpha_2)$ to represent $(\tan\beta_1, \tan\beta_2)$, we have

$$r_i \leq 2R - \frac{(2R - r_j)(2R - r_k)\sin(\alpha_1+\alpha_2)}{(2R - r_j)\sin\alpha_1 + (2R - r_k)\sin\alpha_2}.$$

The case where $\alpha_1 = \alpha_2$ and $r_j = r_k$ follows trivially.

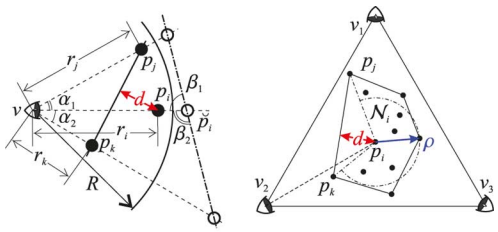


Fig. 12. Illustrations for proofs. (a) DoN in 2-D—the extreme case. (b) p_i is "visible" from at least one of the three vertices of an equilateral triangle enclosing $\mathbf{conv}(\mathcal{N}_i)$.

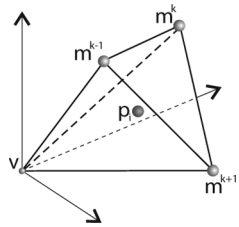


Fig. 13. Illustration for proof of Lemma 2.

According to differential geometry, at a point p of a differentiable surface S, two *principal curvatures* k_1 and k_2 determine the convexity of the surface at p. However, as the *principle directions* associated with these two curvatures are unknown in our case (we do not have the boundary surface), we need to sample three nonidentical sectional planes to determine three sectional curvatures, by which we can uniquely determine the principal curvatures/directions and hence the surface convexity. Therefore, for boundary detection in 3-D WSNs, we choose three nonidentical planes containing v and p_i, with one arbitrarily chosen but the other two perpendicular to each other, and the aforementioned 2-D conditions need to be satisfied in each of these three planes.

The statement that four transformations are sufficient to recognize a boundary point p_i follows from the sufficiency of "seeing" p_i from one of the four viewpoints. Let us still use a 2-D scenario to illustrate the idea in Fig. 12(b). It is straightforward to see that, as far as the line-of-sight from v to p_i passes through the line segment (p_j, p_k), p_i can be detected as a boundary node with a properly chosen transformation. However, for an arbitrary $\mathcal{N}_i$, as the orientation of (p_j, p_k) is unknown, the best choice is to put three viewpoints at the vertices of an equilateral triangle enclosing $\mathbf{conv}(\mathcal{N}_i)$. Consequently, as far as the angle $\angle p_j p_i p_k$ is larger than $2\pi/3$, at least one viewpoint will have its line-of-sight toward p_i passing through (p_j, p_k). It can be shown that $\frac{d}{\rho} < \frac{1}{2}$ is a sufficient condition of $\angle p_j p_i p_k > 2\pi/3$. Extending this idea to 3-D, the viewpoints should be put on the vertices of a regular tetrahedron enclosing $\mathbf{conv}(\mathcal{N}_i)$, which leads to the same requirement on $\frac{d}{\rho}$ claimed in *Proposition 5*. Q.E.D.

Proof of Proposition 6: For the kth outer iteration, let $\vec{n}^k$ and $\vec{n}^k \cdot \vec{p}_i$ denote the plane used for the test, i.e., $\vec{n}^k \cdot \vec{m} = \vec{n}^k \cdot \vec{p}_i$, $\forall\, m \in \mathbb{R}^3$, and let $\breve{\mathcal{N}}_i^k$ be the subset of $f(\mathcal{N}_i)$ that belongs to the convex hull of $\{v, m^1, \ldots, m^k, \breve{p}_i\}$, where m^k refers to the point chosen for the kth iteration. We first show that $\{\breve{\mathcal{N}}_i^k\}$ is a strictly increasing sequence.

Lemma 1: $\breve{\mathcal{N}}_i^k \subset \breve{\mathcal{N}}_i^{k+1}$.

Proof: Given the construction, it is clear that $\breve{\mathcal{N}}_i^k \subseteq \breve{\mathcal{N}}_i^{k+1}$. Assume, by contradiction, that $\breve{\mathcal{N}}_i^k = \breve{\mathcal{N}}_i^{k+1}$. This implies that $m^{k+1} \in \breve{\mathcal{N}}_i^k$, and hence contradicts the fact that $\vec{m}^{k+1}$ is the largest among $f(\mathcal{N}_i)$ in terms of its dot product with $\vec{n}^k$. Q.E.D.

As $f(\mathcal{N}_i)$ is a finite set, Lemma 1 shows that CVX-TEST-DC terminates in finite time regardless of the status of $\breve{p}_i$. If $\breve{p}_i$ is on the convex hull of $f(\mathcal{N}_i) \bigcup \{v\}$, the correctness of the algorithm is obvious. Otherwise, the correctness is confirmed by the following result.

Lemma 2: If neither m^{k-1} nor m^k can be replaced at the kth iteration and the current plane is not supporting, $\breve{p}_i$ is not on the convex hull of $f(\mathcal{N}_i) \bigcup \{v\}$.

Proof: Let us consider the simplex induced by the set $\{v, m^{k-1}, m^k, m^{k+1}\}$ in Fig. 13(b), where m^{k+1} is a candidate to be added for the next iteration. The assumption that m^{k+1} cannot replace either m^{k-1} or m^k implies that a plane determined by $\breve{p}_i$ and any two points in $\{m^{k-1}, m^k, m^{k+1}\}$ separates the third from v. It is straightforward to show that this is possible only if $\breve{p}_i$ is inside the simplex, hence proving $\breve{p}_i$ is not on the convex hull of $f(\mathcal{N}_i) \bigcup \{v\}$. Q.E.D.

It is straightforward to see that CVX-TEST-DC is similar to *quicksort*, as the new point chosen in each iteration acts in a similar way as the *pivot* in quicksort. Moreover, CVX-TEST-DC only needs to "conquer" points in $\mathcal{N}_i \backslash \breve{\mathcal{N}}_i^k$ after dividing. Therefore, we have the following recurrence for the time complexity $T(\eta)$, given totally η points in $f(\mathcal{N}_i)$:

$$T(\eta) = T(c\eta) + \mathcal{O}(\eta), \qquad 0 < c < 1.$$

According to *master theorem* [5], the average-case complexity of CVX-TEST-DC is $\mathcal{O}(\eta \log \eta)$. Q.E.D.

It can be shown that the average-case of complexity of CVX-TEST-DC in 2-D is only $\mathcal{O}(\log \eta)$, demonstrating again the nontrivialness of extending from 2-D to 3-D.

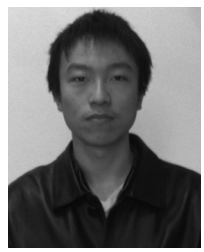
REFERENCES

[1] "TinyOS home page," 2012 [Online]. Available: http://www.tinyos.net/
[2] N. Amenta and M. Bern, "Surface reconstruction by Voronoi filtering," *Discrete Comput. Geom.*, vol. 22, no. 4, pp. 481–504, 1999.
[3] N. Amenta, S. Choi, T. K. Dey, and N. Leekha, "A simple algorithm for homeomorphic surface reconstruction," *Int. J. Comput. Geom. Appl.*, vol. 12, no. 1–2, pp. 125–141, 2002.
[4] M. C. Do, *Differential Geometry of Curves and Surfaces*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1976.
[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2 ed. Cambridge, MA, USA: MIT Press, 2001.
[6] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks," in *Proc. 24th IEEE INFOCOM*, 2005, vol. 2, pp. 902–913.
[7] D. Dong, Y. Liu, and X. Liao, "Fine-grained boundary recognition in wireless ad hoc and sensor networks by topological methods," in *Proc. 10th ACM MobiHoc*, 2009, pp. 135–144.
[8] S. Durocher, D. Kirkpatrick, and L. Narayanan, "On routing with guaranteed delivery in three-dimensional ad hoc wireless networks," in *Proc. 9th ICDCN*, 2008, pp. 546–557.
[9] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 4, pp. 551–559, Jul. 1983.

[10] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing routing holes in sensor networks," in *Proc. 23rd IEEE INFOCOM*, 2004, vol. 4, pp. 2458–2468.
[11] M. Fayed and H. Mouftah, "Localised alpha-shape computations for boundary recognition in sensor networks," *Ad Hoc Netw.*, vol. 7, no. 6, pp. 1259–1269, 2009.
[12] S. Funke, "Topological hole detection in wireless sensor networks and its applications," in *Proc. 3rd ACM DIAL-M-POMC*, 2005, pp. 44–53.
[13] S. Funke and C. Klein, "Hole detection or: "How much geometry hides in connectivity?"," in *Proc. 22nd ACM SCG*, 2006, pp. 377–385.
[14] R. Ghrist and A. Muhammad, "Coverage and hole-detection in sensor networks via homology," in *Proc. 4th ACM/IEEE IPSN*, 2005, p. 34.
[15] D. Goldenberg, P. Bihler, M. Cao, J. Fang, B. Anderson, A. S. Morse, and Y. R. Yang, "Localization in sparse networks using sweeps," in *Proc. 12th ACM MobiCom*, 2006, pp. 110–121.
[16] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S.-T. Yau, "Genus zero surface conformal mapping and its application to brain surface mapping," *IEEE Trans. Med. Imag.*, vol. 23, no. 8, pp. 949–958, Aug. 2004.
[17] K. Han, L. Xiang, J. Luo, and Y. Liu, "Minimum-energy connected coverage in wireless sensor networks with omni-directional and directional features," in *Proc. 13th ACM MobiHoc*, 2012, pp. 85–94.
[18] J. H. Hubbard and B. B. Hubbard, *Vector Calculus, Linear Algebra, and Differential Forms: A Unified Approach*, 3rd ed. Ithaca, NY, USA: Matrix Editions, 2007.
[19] B. Joe, "Three-dimensional triangulations from local transformations," *SIAM J. Sci. Stat. Comput.*, vol. 10, no. 4, pp. 718–741, 1989.
[20] S. Katz, A. Tal, and R. Basri, "Direct visibility of point sets," in *Proc. 34th ACM SIGGRAPH*, 2007, p. 24.
[21] A. Kröller, S. Fekete, D. Pfisterer, and S. Fischer, "Deterministic boundary recognition and topology extraction for large sensor networks," in *Proc. 17th ACM/SIAM SODA*, 2006, pp. 1000–1009.
[22] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *Proc. 1st ACM SenSys*, 2003, pp. 126–137.
[23] F. Li, J. Luo, S. Xin, W. Wang, and Y. He, "LAACAD: Load balancing k-area coverage through autonomous deployment in wireless sensor networks," in *Proc. 32nd IEEE ICDCS*, 2012, pp. 566–575.
[24] 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. 12th ACM MobiHoc*, 2011, pp. 141–152.
[25] J. Luo and Y. He, "GeoQuorum: Load balancing and energy efficient data access in wireless sensor networks," in *Proc. 30th IEEE INFOCOM (Mini-conf.)*, 2011, pp. 616–620.
[26] J. Luo, F. Li, and Y. He, "3DQS: Distributed data access in 3D wireless sensor networks," in *Proc. IEEE ICC*, 2011, pp. 1–5.
[27] J. Luo, H. V. Shukla, and J.-P. Hubaux, "Non-interactive location surveying for sensor networks with mobility-differentiated ToA," in *Proc. 25th IEEE INFOCOM*, 2006, pp. 1241–1252.
[28] D. Niculescu and B. Nath, "Trajectory based forwarding and its applications," in *Proc. 9th ACM MobiCom*, 2003, pp. 260–272.
[29] R. Nowak and U. Mitra, "Boundary estimation in sensor networks," in *Proc. 2nd ACM/IEEE IPSN*, 2003, pp. 80–95.
[30] Y. Shang and W. Ruml, "Improved MDS-based localization," in *Proc. 23rd IEEE INFOCOM*, 2004, vol. 4, pp. 2640–2651.
[31] G. Tan, H. Jiang, S. Zhang, and A.-M. Kermarrec, "Connectivity-based and anchor-free localization in large-scale 2D/3D sensor networks," in *Proc. 11th ACM MobiHoc*, 2010, pp. 191–200.
[32] Y. Wang, J. Gao, and J. Mitchell, "Boundary recognition in sensor networks by topological methods," in *Proc. 12th ACM MobiCom*, 2006, pp. 122–133.
[33] C. Zhang, J. Luo, L. Xiang, F. Li, J. Lin, and Y. He, "Harmonic quorum systems: Data management in 2D/3D wireless sensor networks with holes," in *Proc. 9th IEEE SECON*, 2012, pp. 1–9.
[34] 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.
[35] H. Zhou, S. Xia, M. Jin, and H. Wu, "Localized algorithm for precise boundary detection in 3D wireless networks," in *Proc. 30th IEEE ICDCS*, 2010, pp. 744–753.
[36] X. Zhu, R. Sarkar, J. Gao, and J. Mitchell, "Light-weight contour tracking in wireless sensor networks," in *Proc. 27rd IEEE INFOCOM*, 2008, pp. 1175–1183.

Feng Li received the B.S. degree in computer science from Shandong Normal University, Jinan, China, in 2007, and the M.S. degree in computer science from Shandong University, Jinan, China, in 2010, and is currently pursuing the Ph.D. degree in computer engineering at Nanyang Technological University, Singapore.

His research interests are computational geometry and its applications in wireless sensor networks.



Chi Zhang received the B.S. degree from Zhejiang University, Hangzhou, China, in 2011, and is currently pursuing the Ph.D. degree in computer engineering at Nanyang Technological University, Singapore.

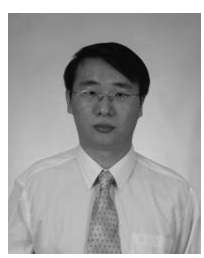
His research interests are embedded systems, social networks, and wireless sensor networks.

Jun Luo (M'09) received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1997 and 2000, respectively, and the Ph.D. degree in computer science from the Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland, in 2006.

From 2006 to 2008, he has worked as a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. In 2008, he joined the faculty of the School of Computer Engineering, Nanyang Technological University, Singapore, where he is currently an Assistant Professor. His research interests include wireless networking, mobile and pervasive computing, distributed systems, multimedia protocols, network modeling and performance analysis, applied operations research, as well as network security.

Shi-Qing Xin received the Ph.D. degree in applied mathematics from Zhejiang University, Hangzhou, China, in 2009.

He had a two-year work experience with Autodesk, Shanghai, China. Since 2009, he has been working with Nanyang Technological University, Singapore, as a Research Fellow. His research interests include computational geometry, computer graphics, and topology.



Ying He (M'08) received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1997 and 2004, respectively, and the Ph.D. degree in computer science from Stony Brook University, Stony Brook, NY, USA, in 2006.

He is currently an Associate Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests are in the broad areas of visual computing, with a focus on the problems that require geometric computation and analysis.