Connectivity-Based Boundary Extraction of Large-Scale 3D Sensor Networks: Algorithm and Applications

Hongbo Jiang, *Member, IEEE*, Shengkai Zhang, Guang Tan, *Member, IEEE*, and Chonggang Wang, *Senior Member, IEEE*

Abstract—Sensor networks are invariably coupled tightly with the geometric environment in which the sensor nodes are deployed. Network boundary is one of the key features that characterize such environments. While significant advances have been made for 2D cases, so far boundary extraction for 3D sensor networks has not been thoroughly studied. We present CABET, a novel Connectivity-Based Boundary Extraction scheme for large-scale 3D sensor networks. To the best of our knowledge, CABET is the first 3D-capable and pure connectivity-based solution for detecting sensor network boundaries. It is fully distributed, and is highly scalable, requiring overall message cost linear with the network size. A highlight of CABET is its *non-uniform critical node sampling*, called r'*-sampling*, that selects *landmarks* to form boundary surfaces with bias toward nodes embodying salient topological features. Simulations show that CABET is able to extract a well-connected boundary in the presence of holes and shape variation, with performance superior to that of some state-of-the-art alternatives. In addition, we show how CABET benefits a range of sensor network applications including 3D skeleton extraction, 3D segmentation, and 3D localization.

Index Terms—Sensor networks, algorithm/protocol design, 3D boundary

1 Introduction

Today sensor networks [16] are used to capture the phenomena of the physical world that were originally difficult or impossible to obtain by traditional techniques. Examples include disaster relief [5], habitat monitoring [20], battlefield surveillance [8], and so on. The geographical environment and deployment method of a sensor network can vary greatly. Boundary, one of the key network features, usually has physical correspondences, such as a building floor plan, holes, and obstacles. Therefore, understanding network boundaries is of great importance in the design of basic networking operations. For example, for location-free geographic routing, knowing the boundary of a network region allows one to find perimeter/boundary nodes [18], which are used to set up a virtual coordination system. In [18] with the knowledge of boundary nodes, briefly speaking, perimeter nodes discover the distances (hop-count) between all perimeter nodes via flooding the network and then use a triangulation algorithm to compute positions from the distance matrix. After that, non-perimeter nodes is able to use the relaxation algorithm to compute their own virtual coordinates. Besides, applications such as 3D sensor network coverage [2] and localization [6] can also benefit from the knowledge of boundaries.

While some methods have been developed for 2D boundary extraction, they cannot be applied to 3D networks because many geometric structures and tools cannot be extended from 2D to 3D. The only work that we are aware of addressing 3D sensor network boundary extraction is [24]. The main idea there is to have each node establish a local coordinate system and then construct a unit ball, whereby a node is able to determine whether it is on a boundary, since a hole/boundary can always contain an empty unit ball. This solution needs to know inter-node physical distance in advance. This requires special hardware support which may not always be available in practice.

Indeed, 3D boundary extraction is a well studied topic in graphics research fields. While bearing a resemblance to the task therein, our problem is more challenging because of the characteristics of a distributed network. *First*, the network is discrete with a fairly random node distribution, implying that the properties of geometric objects in the continuous domain do not hold, or come with unacceptable errors. *Second*, in practice sensor nodes may not have any location information, making geometric information such as distance and direction difficult to collect. Inter-node distance, for example, is often estimated by hop count [19]. *Third*, a distributed sensor network often consists of a set of autonomous sensor nodes which collectively perform tasks without help from any central servers. Then the demand of the distributedness drives the task of boundary extraction more challenging. *Last*, even after the boundary nodes are identified, connecting all nodes into a proper boundary line or surface is nontrivial [23], [24].

- *H. Jiang and S. Zhang are with Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, Hubei 430074, P.R. China. E-mail: hongbojiang2004@gmail.com.*
- *G. Tan is with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, P.R. China.*
- *C. Wang is with InterDigital Communications, PA 19406.*

Manuscript received 14 May 2011; revised 12 Mar. 2013; accepted 24 Mar. 2013; date of publication 2 Apr. 2013; date of current version 21 Feb. 2014. Recommended for acceptance D. Xuan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.97

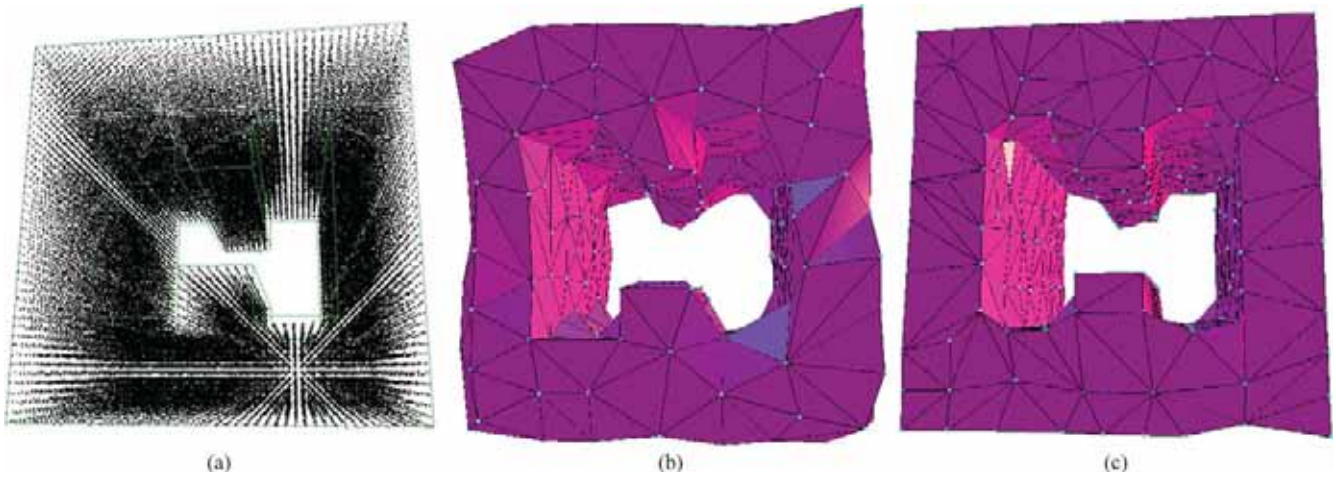


Fig. 1. An example 3D network with 64,240 nodes in an hollow-H shape. (a) The original network. (b) Boundary extraction result by [24]. (c) Boundary extraction result by CABET. (We draw the lines in this figure and in this paper, in order to visualize the boundary, while in practise, each landmark node only has the knowledge of its neighbors in the triangulated graph.)

To tame these challenges, we propose a novel boundary extraction method, named *CABET*, that recovers the network shape well, using connectivity information only. CABET is fully distributed and is scalable, with overall message complexity linear with the network size. The success of CABET has an implication for distributed topology recognition in general: that one should pay special attention to nodes that make distinguished contributions to the network's topological features. We also describe the benefits of having 3D boundary extraction in various 3D sensor network applications, such as skeleton extraction and network segmentation. Besides, it is noted that, like existing algorithms such as [23], [24] based on connectivity information only, CABET relies on the assumption that network nodes are relatively evenly distributed, for example, a perturbed grid or uniform random one, as well as the inter-node distance has a much smaller variance than the global network size. This is required for a reasonable accuracy when the distance is estimated by hop count.

The remainder of this paper proceeds as follows: Section 2 is devoted to the 3D boundary extraction algorithm. We evaluate our scheme in Section 3. Section 4 presents several 3D applications using 3D boundary extraction. Finally, Section 5 summarizes the paper and describes future plans.

2 The Boundary Extraction Algorithm

In this section, we describe *CABET*, a connectivity-based boundary extraction algorithm for 3D sensor networks. The algorithm includes five steps: 1) boundary node identification; 2) critical node identification; 3) landmark selection; 4) coarse boundary extraction; and 5) boundary refinement.

2.1 Boundary Node Identification

In this step, every node p makes a decision as to whether itself is a boundary node, using connectivity information only. As mentioned earlier, previous 2D solutions [7], [23] based on boundary cycle generation cannot be used here. We use the traditional neighborhood size-based method based on the observation that nodes on the boundaries have much smaller neighborhood size than interior nodes. This does not offer a precise result, but here we only want a coarse sampling of the boundaries. This will suffice for our triangulation scheme to be used later.

Definition 1. *The* r-hop neighborhood, $\mathcal{N}_r^{(3)}(p)$, *of a node* $p \in \mathcal{S}$ *is defined as the nodes in* $\mathcal{S}$ *that are at most* r *hops away from* p. *The average* r*-hop neighborhood size of a sensor network is denoted by* $\overline{\mathcal{N}}_r^{(3)}$.

Following an observation in [21], we can see that when a node p is located at the middle of a locally "flat" boundary surface, its neighborhood area will be approximately a half ball.

Definition 2. *The* r-hop criticality *of node* p *is given by* $\mathcal{C}_r(p) = \frac{|\mathcal{N}_r^{(3)}(p)|}{\overline{\mathcal{N}}_r^{(3)}/2} - 1$.

When the boundary surface is locally "flat" enough the criticality of nodes around this area tends to be close to 0. Similarly, the local shape should be convex when the criticality is negative, or concave when positive. For interior nodes, the criticality is typically close to 1. Fig. 2 shows the criticality values of nodes in a 2D network.

Our goal is to identify the set of boundary nodes $\mathcal{B}_\mathcal{S}$. The idea is quite simple: each node $p \in \mathcal{S}$ labels itself as a boundary node when its criticality is less than a given threshold δ_0. The selection of δ_0 should be careful: a small value, say $\delta_0 = 0.1$, may result in omission of nodes exactly on boundaries, called *exact boundary nodes* (see the nodes in the concave area in Fig. 2), whereas a large one may lead to many near-boundary nodes being falsely taken as exact ones. Despite the inaccuracy, we prefer a relatively large δ_0, in order to avoid the more severe consequence of missing exact boundary nodes as they are more representative of the network topology. Meanwhile, to alleviate the disadvantage of a large δ_0, we will differentiate the nodes by defining *critical nodes*. This will be described in the next section.

2.2 Critical Node Identification

In this step, some nodes mark themselves as critical nodes based on local shape recognition. There are three types of such nodes: namely *convex critical nodes*, *concave critical*

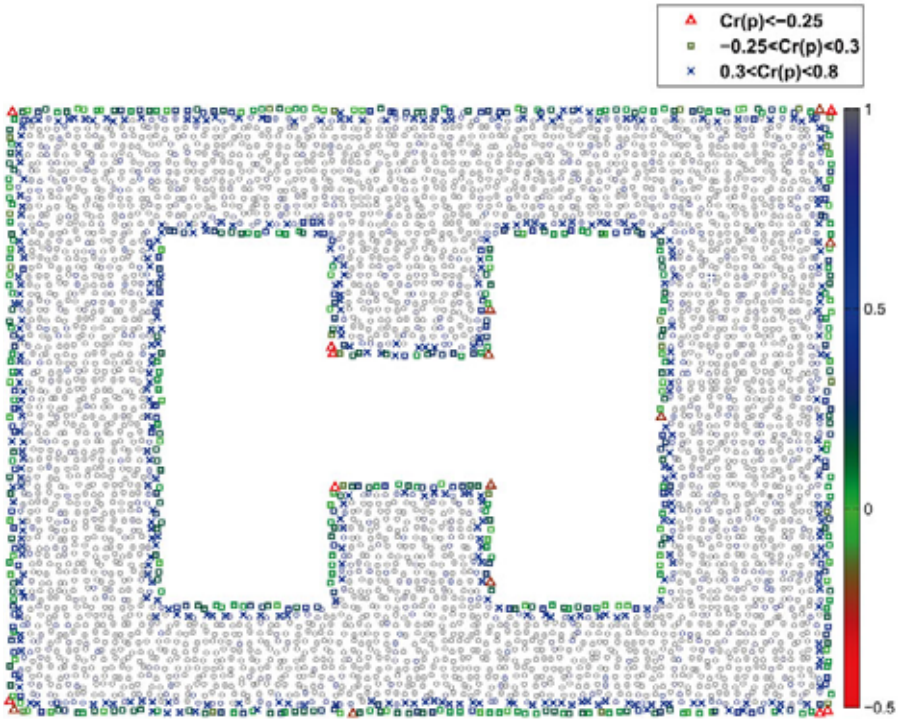


Fig. 2. A 2D network with 4,537 nodes and average degree 8.01. The color map shows the criticality values. Squares represent the potential boundary nodes(the critical value is close to 0); triangles represent the potential convex nodes (the critical value is close to -0.5); crosses involve the potential concave nodes (the critical value is close to 0.5).

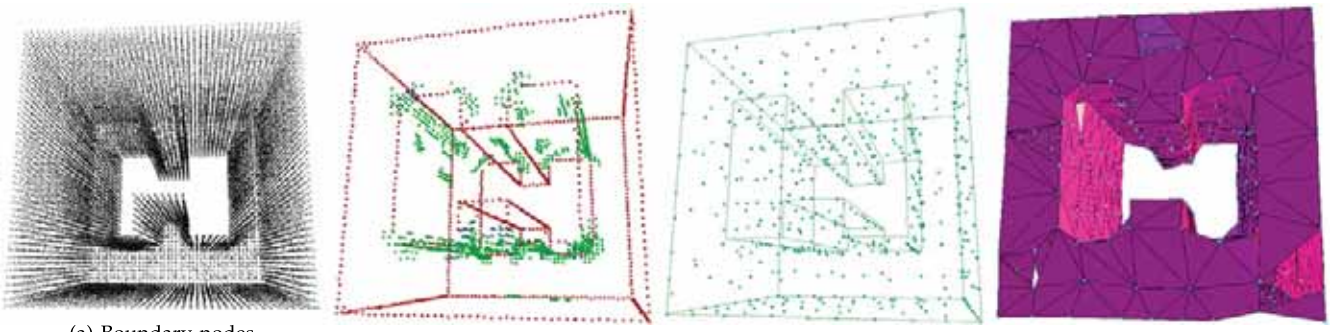
nodes, and *saddle critical nodes*. These nodes are specially important for capturing the "distinct features" of the geometric environment. Intuitively, nodes located at the area where boundary surface is locally "flat" are less important than those at corner areas.

2.2.1 Convex Critical Nodes

The convex critical nodes are easy to determine as their criticality is typically a local minimum. A node p compares its own criticality with that of its r-hop neighbors. If its criticality is less than that of most (δ_0, say in our case 70 percent) neighbors, and is less than a given threshold δ_1, then p labels itself as a convex critical node. It it noted that a convex node is not required to be the minimum among neighbors in terms of the criticality value, that is, $\delta_0 \neq 100\%$. The reason is that in 3D networks, the convex areas could be long and some neighbors of convex nodes are still convex nodes (please see the red points in Fig. 3b). Besides, the boundary extraction is insensitive to this parameter, leading to similar results even it ranging from 60 to 90 percent (while ranging from 90 to 60 percent leads to the false positive to convex critical node identification, in the process of landmark selection, each critical node asynchronously labels itself as a landmark and then notifies its (r/ρ_p)-neighborhood via local flooding, which can alleviate the influence of this false positive).

2.2.2 Concave Critical Nodes

Concave critical nodes cannot be determined by local maximum, since a node with local maximum value may well be an interior node. Since the shortest path could be often bent (not straight) at the concave area [15], [22]. As such, intuitively the shortest paths along a boundary often go through those nodes which contribute for the concave area and can



(a) Boundary nodes (b) Critical nodes (c) Selected landmark nodes (d) Coarse boundary

Fig. 3. Boundary extraction for the Hollow-H-shaped network. In (b), the critical convex/concave/saddle nodes are marked with red/green/blue points. The final result of the boundary extraction is shown in Fig. 1c.

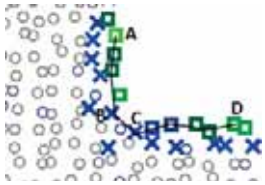


Fig. 4. A close-up view of Fig. 2.

be considered as potential concave nodes. As an example, Fig. 4 shows a shortest path from node A to node D containing the potential concave critical nodes B and C.

The pseudocode of identifying concave critical nodes is shown in Algorithm 1. All boundary nodes are first divided to many neighborhoods. This can be done by a process similar to r-sampling [17]. Each node i asynchronously assigns its neighborhood an ID $\mathcal{N}_i$ by a local flooding. If a node receives such a flooded message from some other node before its own flooding, it cancels its flooding and takes assigns itself the received ID.

Algorithm 1 Concave Critical Nodes Identification()

Require: All boundary nodes $\mathcal{B}_\mathcal{S}$ are divided into many neighborhoods, individually denoted by $\mathcal{N}_i$.

1: **for** each r-hop neighborhood $\mathcal{N}_i$ **do**
2: Find N_0 nodes with the smallest criticality in $\mathcal{N}_i$, denoted by $\mathcal{M}_i$;
3: **for** each $p \in \mathcal{M}_i$ **do**
4: Find a node p' such that $p' = \texttt{argmax}_{q \in \mathcal{M}_i} \hat{\texttt{dist}}(p, q)$;
5: Each node $q \in \texttt{path}(p, p')$ labels itself as a concave critical node iff $\mathcal{C}_r(q) \geq \delta_2$.
6: **end for**
7: **end for**

In each neighborhood, we first find some nodes residing on the boundary (Line 2) which have the smallest criticality values. Let $\texttt{path}(\cdot,\cdot)$ be the shortest path between two nodes that lies entirely on the boundary. We next find the paths that may contain concave critical nodes (Line 4). Here $\hat{\texttt{dist}}(\cdot,\cdot)$ represents the path length in hop count. Finally, the concave nodes typically have a large criticality in the path (Line 5).

2.2.3 Saddle Critical Nodes

In computing geometric theory, a saddle point is a point in the domain of a function that is a stationary point but not a local extreme. A saddle often curves up in one direction, and curves down in a different direction (like a horse saddle or a mountain pass). Similarly, in a discrete 3D network, a saddle point is often located at the area (see blue points in Fig. 3b) which is simultaneously close to a convex area in one direction and close to a concave area in the other. As such, boundary nodes that are close to both convex and concave critical nodes are possibly saddle critical nodes. In addition, a local saddle area can be viewed as the displacement of a flat surface. Therefore the average shortest path length between nodes around the saddle area on the boundary surface should be larger than that at flat areas. We determine the saddle critical nodes using the following heuristic. For each boundary node p close to both convex and concave critical nodes, find the perimeter nodes of its r-hop neighborhood. Let $\mathcal{A}_p$ be the average shortest path length between the perimeter nodes. p is a saddle critical node if and only if $\mathcal{A}_p > (1+\epsilon)\overline{\mathcal{A}}$, where $0 < \epsilon < 1$ is a small number (e.g., 0.2) and $\overline{\mathcal{A}}$ is the average $\mathcal{A}$ of all neighborhoods. Our experiment results show that CABET is not sensitive to this parameter. Often smaller value of epsilon means more saddle nodes could be identified, and vice versa.

2.2.4 Node Importance

Fig. 3b shows an example network where convex/concave/saddle critical nodes are marked with thick red/green/blue circles. As can be seen, these critical nodes well capture the "distinct features" of the geometric environment. To highlight those features we propose a concept called *node importance*, defined as $\rho_p = |\mathcal{C}_r(p)| + 1$. For a saddle critical node p, since it is close to convex/concave critical nodes, we let its importance be the highest among the convex/concave critical nodes close to p. For other boundary nodes, the importance is set to 1.0.

2.3 Landmark Selection

In the third step, a subset of boundary nodes are selected as landmarks, denoted $\mathcal{R}_\mathcal{S}$. The landmarks present an approximation of the network boundary. More importantly, they constitute the 0-*simplices* (vertices) of Δ_Σ. Previously a uniform sampling technique has been proposed, known as *r-sampling* [17]. A disadvantage of this approach is that it ignores the non-uniformity of nodes' ability in representing the network's topological features.

Inspired by the work in graphics community [11] where sampling is biased toward areas exhibiting distinct features, we propose a non-uniform random sampling scheme, called *r′-sampling*.

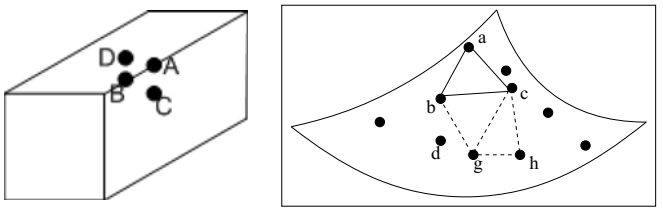
Definition 3. *Given a set of boundary nodes $\mathcal{B}_\mathcal{S}$, an* r'-sampling *is a subset $\mathcal{R}_\mathcal{S} \subset \mathcal{B}_\mathcal{S}$ such that each node $p \in \mathcal{B}_\mathcal{S}$ is at most (r/ρ_p)-hops away from some node $q \in \mathcal{R}_\mathcal{S}$.*

In *r′-sampling*, each critical node p asynchronously labels itself as a landmark and then notifies its (r/ρ_p)-neighborhood via local flooding. If a node receives such a flooded message from some other node before its own flooding, it cancels its action and labels itself as a non-landmark node. Later on, the same procedure is performed by non-critical nodes to select landmark or non-landmark nodes among them. In Fig. 3c, the selected landmarks $\mathcal{R}_\mathcal{S}$ constitute the 0-*simplices* (vertices) of $\triangle_\Sigma$. In the following section, we detail how to obtain 1,2-*simplices* of $\triangle_\Sigma$.

2.4 Getting a Coarse Boundary

In the fourth step, we develop a connectivity scheme from the selected landmarks by boundary triangulation. There are two sub-steps: partial virtual edges extraction and triangle evolution, both carried out in a fully distributed way.

Boundary triangulation is based on critical nodes. Fig. 5a shows a case where the nodes A, B, C, D are landmarks. A possible result of triangulation might include $\triangle ACD$ and $\triangle BCD$, given the random connections among landmarks [9], [24]. Instead of using the



(a) (b)

Fig. 5. Triangular evolution: (a) triangulation selection; (b) evolution steps.

random connections, our algorithm gives priority to critical nodes, producing triangles $\triangle ABC$ and $\triangle ABD$. Fig. 5b shows an example of triangle evolution where the nodes b and c are critical nodes. We first connect b and c to generate a virtual edge. Since finally each edge should be associated with two triangles, from nodes adjacent to both b and c, we find a node a with the largest importance value, and then the triangle $\triangle abc$ is generated (as long as $\texttt{edge}(a,b)$ and $\texttt{edge}(a,c)$ do not lead to an intersection). Similarly, the triangle $\triangle bcg$ is generated so that $\texttt{edge}(b,c)$ is used twice. This process continues until every edge joins exactly two triangles or no such neighbor node can be found. We detail the algorithm in the following.

To efficiently connect landmarks, *Voronoi Cells* were previously applied where a non-landmark node $p \in \mathcal{B}_\mathcal{S}/\mathcal{R}_\mathcal{S}$ is associated with its closest landmark $i \in \mathcal{R}_\mathcal{S}$ [9], [24]. All nodes associated with the same landmark i are called the tile $\mathcal{T}_i$. Accordingly, the adjacency of landmarks is based on the *combinatorial Delaunay graph* (CDG), a dual graph of Voronoi cells, which has a solid mathematical foundation in graph theory.

Definition 4. *Two landmarks $i, j \in \mathcal{R}_\mathcal{S}$ are* adjacent *to each other in a CDG if there exist two adjacent nodes $p_1 \in \mathcal{T}_i$ and $p_2 \in \mathcal{T}_j$.*

Note we do not connect all adjacent landmarks in a CDG as this may result in a non-planar graph due to degenerate cases (for example, where there exists one node having the same distance to at least four landmarks) [9].

When the adjacency of two landmarks i and j is detected, a virtual edge $\texttt{edge}(i,j)$ is established, all nodes on which are labelled *edge nodes*. In addition, to maintain the information of how many triangles each virtual edge is associated with, we add an attribute to the edge, denoted by $\texttt{edge}(i,j).n$. To obtain a planar graph, an important problem is to avoid the intersection of virtual edges.

Definition 5. *A path $\texttt{path}(i,j)$ is said to lead to an* intersection *if and only if there exists a node p' on the path that is adjacent to some other node p not belonging to either $\mathcal{T}_i$ or $\mathcal{T}_j$.*

2.4.1 Distributed Partial Virtual Edges Extraction

(Algorithm 2) Virtual edges (parts of 1-*simplices* of $\triangle_\Sigma$) are first generated which often exhibit distinct features of the boundary. To that end, each critical node (Line 1) distributedly contacts its neighbors (on the CDG) with high node importance (Lines 3-8) where only local information is required. The most time-consuming procedure is to identify whether an intersection occurs where each node on the $\texttt{path}(i,j)$ performs a local operation to tell whether any neighbors do not belong to $\mathcal{T}_i$ and $\mathcal{T}_i$.

Algorithm 2 Distributed Partial Virtual Edges Extraction()

1: **for** each critical (landmark) node i not associated with a virtual edge **do**
2: Find all adjacent landmarks to i, denoted $\mathcal{A}_i$; $\mathcal{A}'_i \leftarrow \emptyset$
3: **for** each $j \in \mathcal{A}_i$ **do**
4: **if** j is not associated with some virtual edge **and** $\texttt{path}(i,j)$ does not lead to an intersection **then**
5: $\mathcal{A}'_i \leftarrow \mathcal{A}'_i + \{j\}$;
6: **end if**
7: **end for**
8: Find the landmark $j' = \texttt{argmax}_{j \in \mathcal{A}'_i} \rho_j$;
9: Connect i and j' by a virtual edge $\texttt{edge}(i,j')$; $\texttt{edge}(i,j').n \leftarrow 0$.
10: Each node $p \in \texttt{path}(i,j')$ labels itself as an edge node.
11: **end for**

2.4.2 Distributed Triangle Evolution

(Algorithm 3) Given a set of virtual edges, the triangulation process is essentially triangle evolution, based on the fact that each virtual edge should be associated with exactly two triangles. In particular, a new triangle should be generated if a virtual edge is associated with only zero or one triangle (Line 1). The step of triangle evolution guarantees no intersection with any identified virtual edges (Lines 6-7), and no edges associated with more than two triangles (Lines 4-5). The landmark with a high importance value is then involved in the triangle evolution (Line 10). All nodes in the paths are taken as edge nodes for intersection check (Line 14).

Fig. 3d shows a result of the coarse boundary result after the fourth step. Some holes in this result are found, which are undesirable. Next we discuss how to refine the coarse boundary so as to avoid holes and establish a closed and well-connected final boundary surface.

2.5 Refining the Coarse Boundary

A careful investigation of the above connecting method reveals that this coarse boundary surface may not be very useful for topology recognition in practice. The reason is that we used a 2D method to address the 3D intersection problem, potentially leading to holes. Fig. 6 shows a close-up view of the Hollow-H topology where A, B, C, D, E are landmarks. The nodes A and D are adjacent in the CDG while the nodes C and E are not. The $\texttt{edge}(A,D)$, shown in thick green circles, leads to an intersection with $\texttt{edge}(B,C)$, shown in thick blue circles. (Recall that intersection detection is limited within the boundary nodes.) As a result, a hole is created by the nodes A, C, D and E.

To correct this error, we first need to discover these holes. Let each landmark node (say the node j) check all virtual edges it is associated with. If it finds two virtual edges $\texttt{edge(i,j)}$ and $\texttt{edge(j,k)}$, each associated with only one triangle, then the two edges should be on the

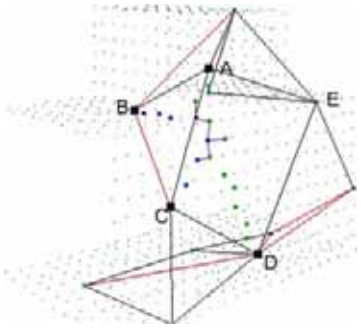


Fig. 6. A close-up view of the network with the Hollow-H topology in Fig. 1. Small black points are boundary nodes and straight lines represent the virtual edges. The green/blue thick points are shortest paths from A to D/from B to C, and thus $\texttt{edge}(A, D)$ leads to an intersection with $\texttt{edge}(B, C)$.

hole. Afterwards, the node k checks all its associated virtual edges to find the next virtual edge on the hole. This process continues until we find an $\texttt{edge}(\texttt{l}, \texttt{i})$ and the cycle $i, j, k, \ldots, l, i$ constitutes a hole. To triangulate the detected hole, the landmark node with the smallest ID on the hole sends a message to the neighboring landmark of its neighbor on the hole (the node i sends a connecting message to k in this case) to generate a new virtual edge and a new triangle without checking intersection. This process continues until the hole is triangulated finally. The refined boundary is shown in Fig. 1c.

Algorithm 3 Distributed Triangles Evolution()

1: **while** A virtual edge $\texttt{edge}(i, j)$ exists such that $\texttt{edge}(i, j).n < 2$ **do**
2: Find all landmarks adjacent to i and j, denoted by $\mathcal{A}_{i,j}$; $\mathcal{A}'_{i,j} \leftarrow \emptyset$;
3: **for** each $k \in \mathcal{A}_{i,j}$ **do**
4: **if** $\texttt{edge}(i, k)$ exists $\wedge$ $\texttt{edge}(i, k).n == 2$ **then continues**;
5: **if** $\texttt{edge}(j, k)$ exists $\wedge$ $\texttt{edge}(j, k).n == 2$ **then continues**;
6: **if** $\texttt{path}(k, i)$ leads to an intersection **then continues**;
7: **if** $\texttt{path}(k, j)$ leads to an intersection **then continues**;
8: $\mathcal{A}'_{i,j} \leftarrow \mathcal{A}'_{i,j} + \{k\}$;
9: **end for**
10: Find the landmark $k' = \texttt{argmax}_{k \in \mathcal{A}'_{i,j}} \rho_k$;
11: Connect i and k'; $\texttt{edge}(i, k').n \leftarrow \texttt{edge}(i, k').n + 1$
12: Connect j and k'; $\texttt{edge}(j, k').n \leftarrow \texttt{edge}(i, k').n + 1$
13: $\texttt{edge}(i, j).n \leftarrow \texttt{edge}(i, j).n + 1$
14: Each node $p \in \texttt{path}(i, k') \cup \texttt{path}(j, k')$ labels itself as an edge node.
15: **end while**

All the virtual edges and triangles obtained based on Algorithm 2, Algorithm 3, and the boundary refinement constitute 1, 2-*simplices* of $\triangle_\Sigma$. We argue that this connectivity scheme provides a good approximation of the 3D boundary surface, in that almost all nodes are located inside this tessellation. Moreover, the triangulation is a locally planarized two-manifold [24]. This is an important consideration for many practical applications because available 2D graphic tools can be applied on 3D surfaces.

2.6 Limitations of CABET

In addition to uniform distribution assumption, CABET presumes the perfect link between nodes. In CABET, each node maintains a neighbor list. To address the problem of packet loss, after a node sends a message by means of unicast or broadcast, it will receive an ACK message from its neighbor; otherwise it will send out the message again. By doing so, CABET can be robust again packet loss.

3 Simulations

We have implemented a simulator and conducted a series of simulations on various communication graphs in comparison with the approach in [24]. As mentioned earlier, a major difference between our method and the one in [24] is that we do not need the knowledge of physical internode distance. Aside from requiring less support from hardware, our method outperforms the previous one by using a non-uniform sampling scheme based on critical node identification.

In the topologies we tested, sensor nodes are deployed with the perturbed grid model where the grid has a width around 4.5. We also consider a uniform random distribution model. The radio range of all three topologies is around 6.8. Both regular and irregular radio models are considered. By default $r = 4$ for the r-sampling.

3.1 Evaluation Results

Figs. 7a, 7b, and 7c show the results of a large-scale scenario based on the Chicago Airport Terminal 2 map. It can be seen that CABET delivers stable performance and successfully recovers the network boundary, resulting in a better approximation of the network boundary than the algorithm in [24]. In particular, for the area C in Fig. 7c, CABET better recovers the distinct features at network boundary compared with the results of [24] at the area B in Fig. 7b. Using the algorithm in [24], the result in Fig. 7b shows an undesirable shape of the network boundary.

We also conducted simulations on an S-shaped topology shown in Figs. 7d, 7e, and 7f. Again, the algorithm in [24] performs worse than CABET. In particular, in the area E in Fig. 7e, a hole on the boundary can be seen, the reason behind which is that the algorithm in [24] merely avoids edge intersection without refinement as we describe in Section 2.5.

Beside visual results, we also compare the accuracies of the two algorithms. Table 1 compares the two algorithms in terms of the number E_1 of *outside nodes*, the nodes outside the boundary surface. Ideally, all nodes should be within or lie on the boundary surfaces determined. When errors occur some nodes will be outside the surfaces. We also use E_2 to represent the average distance from the nodes to the boundary surfaces. For those non-outside nodes, the distance is set to 0. Besides the average distance, we measure the variance of the distance to the boundary surface, denoted by V_1.

Table 1 shows that CABET produces 20-40 percent fewer outside nodes than [24] does, indicating a better extracted boundary. In terms of E_2, CABET incurs about half the error

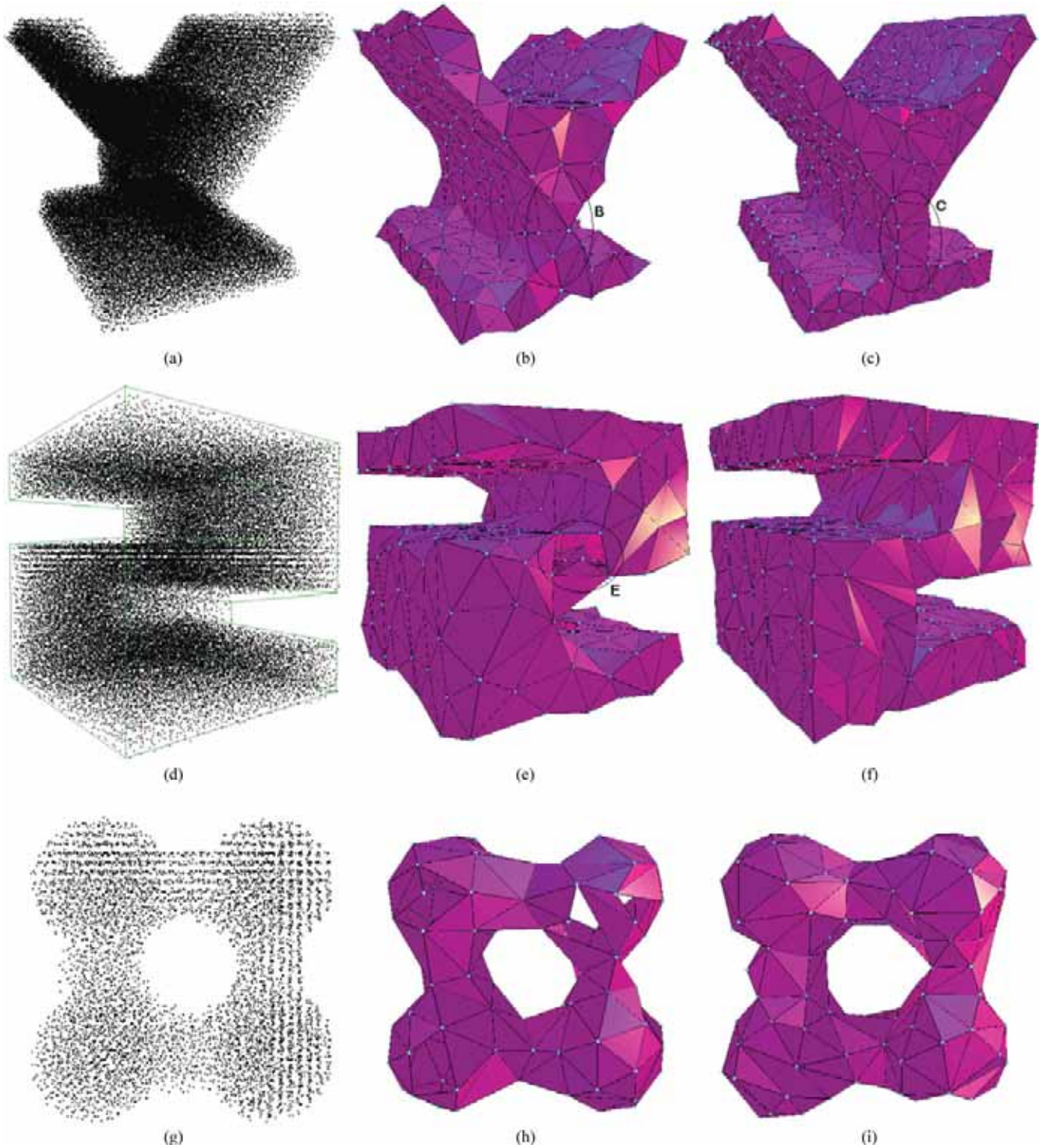


Fig. 7. Boundary extraction results. (a) The original network (Chicago Airport Terminal 2) with 77,251 nodes. (b) Boundary extraction result by [24]. (c) Boundary extraction result by CABET. (d) A network with 41,192 nodes. (e) Boundary extraction result by [24]. (f) Boundary extraction result by CABET. (g) A network with 6,854 nodes. (h) Boundary extraction result by [24]. (i) Boundary extraction result by CABET.

of [24]. In the Hollow-H topology for example, compared with [24] (let $r = 4$, See Row 1, Table 1), CABET generates only half landmarks (let $r = 8$, See Row 3, Table 1) while their errors in both E_1 and E_2 are comparable: the number of outside nodes is around 9,000 and the average distance to boundary is around 0.3. These results again demonstrate CABET's superiority in boundary extraction. One observation is that the error of E_1 is non-trivial: around 10-15 percent nodes are outside the extracted boundary surface. One reason is that often the variation of actual boundary shape is significant. Therefore, when we reconstruct the boundary with triangulation to achieve an approximation, many boundary nodes could be outside. Besides, compared with the radio range of 6.8, the average distance from the nodes to the boundary surfaces, around 0.1-0.3, is trivial. Also, the variance is not significant: around 1.0-2.5. That is, while often 10-15 percent nodes are

TABLE 1
Performance of CABET on Sensor Networks in Different Scenarios

	Size		Boundary $\lvert\mathcal{B}_S\rvert$		Landmarks $\lvert\mathcal{R}_S\rvert$		Error E_1		Error E_2		Variance V_1	
Networks	nodes	Avg. deg.	[24]	CABET	[24]	CABET	[24]	CABET	[24]	CABET	[24]	CABET
Hollow-H ($r = 4$)	64,240	16.8	15,381	16,127	372	575	9,796	7,924	0.31	0.155	1.502	1.013
Hollow-H ($r = 6$)	64,240	16.8	15,381	16,127	172	261	12,319	8,636	0.867	0.239	2.668	1.376
Hollow-H ($r = 8$)	64,240	16.8	15,381	16,127	93	167	16,044	9,581	1.827	0.342	4.357	2.323
Fig. 7, Row 1	77,251	13.6	24,656	27,172	354	557	15,516	10,873	0.652	0.284	1.263	1.080
Fig. 7, Row 2	41,192	15.0	10,376	11,897	299	433	7,130	5,027	0.530	0.277	1.605	1.120
Fig. 7, Row 3	6,854	14.3	2,047	2,135	63	102	1319	884	0.486	0.179	1.509	1.040

outside the extracted boundary surface as we mentioned, most of them are very close to the boundary. This makes the extracted boundary results meaningful.

3.2 Impact of Node Densities

Fig. 8 shows CABET's results on the Hollow-H shape network for a range of node densities. The real radio range is varied with a fixed small step size. In Fig. 8, the lowest average node degree is 6.997, below which some nodes start getting isolated, giving us a disconnected network Fig. 8 also gives the accuracy results in terms of E_1 and E_2. These results show that CABET is quite robust to different node densities. This robustness can be also observed in other networks in Fig. 7.

3.3 Impact of Radio Models

We conduct experiments under a different radio model: Quasi-UBG radio model. With the parameter $0 \leq \alpha < 1$, a link exists between two nodes if their Euclidian distance is less than $(1-\alpha)R$ where R is the radio range; a link exists with probability $0 < p < 1$ between two nodes when the distance is between R and $(1+\alpha)R$; no link exists when the distance is greater than $(1+\alpha)R$. We vary the α value and adjust p so that the average node degree in the tested networks remains nearly the same. Fig. 9 shows the boundary results, along with error results, for varying α values. It can be seen that CABET achieves reasonably good results for even large α values. Although the performance becomes worse for a larger α, the extracted boundary still captures the network boundary topology faithfully.

We next conduct the experiments when the communication radio model is log-normal where the probability that a link exists between nodes i and j is based on the log-normal shadowing radio model [10] given by

$$p(\hat{r}) = \frac{1}{2}\left[1 - erf\left(\alpha \frac{log\ \hat{r})}{\text{ɬ}}\right)\right], \text{ɬ} \triangleq \sigma/\eta \qquad (1)$$

where $\hat{r}$ is the normalized distance between nodes i and j, $\alpha = 10/(\sqrt{2}log10)$ is a constant, σ is the standard deviation, and η is the pathless exponent. Empirically, ɬ may vary between 0 and 2 [10]. Fig. 10 shows the results under log-normal radio model for three different values of ɬ, namely, 0.5, 1.0, 1.5. As we can be seen, with the increase of ɬ, the average node degree increases accordingly, while CABET generates

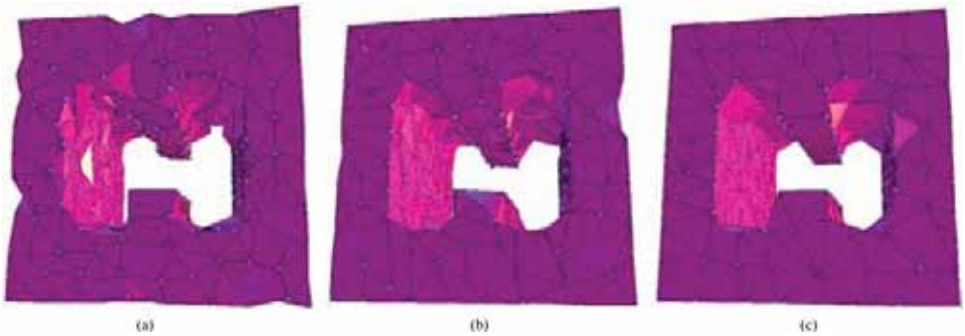


Fig. 8. Effect of node densities on boundary extraction. The average node degree in (a), (b) and (c) is 6.997, 10.286, 18.390, respectively. (a) $E_1 = 12,969$, $E_2 = 0.665$; (b) $E_1 = 9,739$, $E_2 = 0.344$; (c) $E_1 = 6,170$, $E_2 = 0.069$.

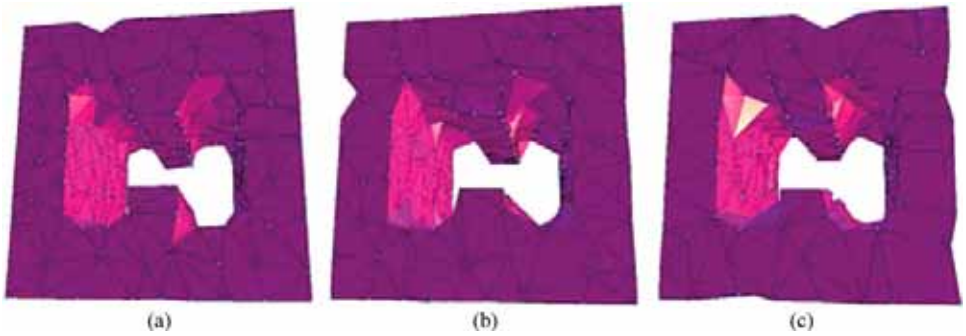


Fig. 9. Performance of CABET under the Quasi-UBG radio model. The average node degree in (a), (b) and (c) is 16.72, 16.62, 16.53, respectively. (a) $\alpha = 0.2$, $E_2 = 0.249$; (b) $\alpha = 0.3$, $E_2 = 0.283$; (c) $\alpha = 0.4$, $E_2 = 0.273$.

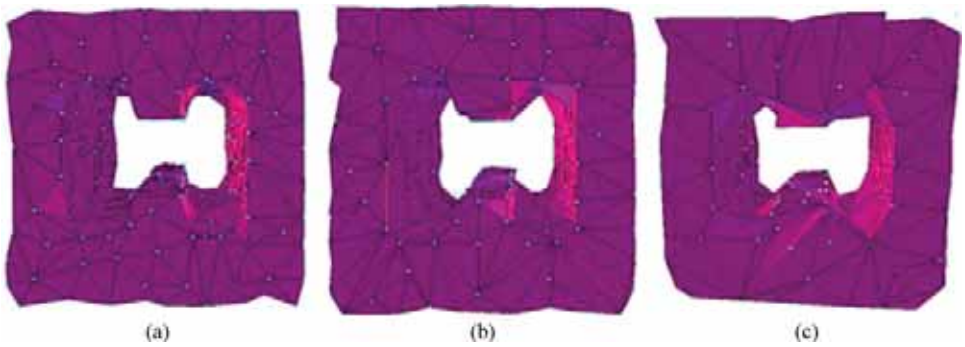


Fig. 10. Performance of CABET under the log-normal communication radio model. The average node degree in (a), (b) and (c) is 13.57, 15.66, 20.77, respectively. (a) $\natural = 0.5$, $E_2 = 0.239$; (b) $\natural = 1.0$, $E_2 = 0.288$; (c) $\natural = 1.5$, $E_2 = 0.298$.

acceptable results still. It is noted that, for larger $\natural$, the side length of triangles is apparently greater than smaller $\natural$. The reason is that for larger $\natural$, the probability of having a link between two faraway nodes is larger than that for smaller $\natural$.

4 Applications

In this section we describe how 3D sensor network applications can benefit from a boundary extraction algorithm. We consider two applications: skeleton extraction and network segmentation.

4.1 3D Skeleton Extraction Algorithm

The knowledge of the skeleton (also called medial axis in [3], [4]) [12], [13] of a sensor network can greatly improve routing performance. Skeleton can be used to achieve load balancing during geographical routing. Roughly speaking, each node, instead of greedy routing, forwards the packet to it neighbor, parallel along with the skeleton of the network in this scheme. By doing so, it can avoid the node overload at the boundary and concave area. The technical details how to achieve load balance during geographical routing can be found in [3], [4]. A node is on the skeleton if it is equidistant from at least two closest boundary nodes. Establishing the skeleton structure can naturally start with our boundary extraction algorithm.

First, every node in the network initiates a flooding to obtain its distances to two closest landmarks on the boundary. If the distances are equal or differ by only one, then the flooding node is marked as a skeleton node. To avoid noise we depress skeleton nodes whose two closest landmarks are neighbors to each other. This way the skeleton extraction will be more robust to irregularity in node distribution and network shape variation. The skeleton node result is shown in Fig. 11a.

In a 3D space, a skeleton is two-manifold [1]. To comply with this notion, we can partition the skeleton nodes. Define *joint nodes* of a skeleton as nodes that are equidistant from at least three closest boundary nodes. The thick black points in Fig. 12a show the joint nodes. Joint node help partition the skeleton into components, shown in different colors in Fig. 12a. For each component, a triangulation process similar to that in Section 2.4 is performed (here the surface may be not closed). The final two-manifold skeleton surface is shown in Fig. 11b.

4.2 3D Segmentation Algorithm

Many sensor network protocols, such as geographic routing and information gathering, implicitly assume a relatively dense and uniform sensor field in a simple geometric region. When applied to an irregular sensor field (e.g., with holes), their performance may degrade significantly. The *segmentation algorithm* in sensor networks is used to partition an irregular sensor field into nicely shaped pieces where protocols relying on a regular field tend to work well [25]. Prior work has considered only 2D networks and cannot be used in a 3D space. Our 3D skeleton extraction algorithm provides a possible way to fulfill this task.

Recall that the skeleton nodes can be partitioned into components by joint nodes. Let each skeleton node flood a message containing the ID of the skeleton node's host

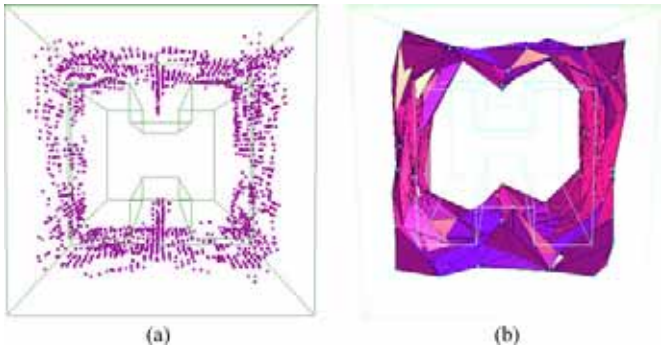


Fig. 11. Skeleton result: (a) skeleton nodes; (b) skeleton surface.

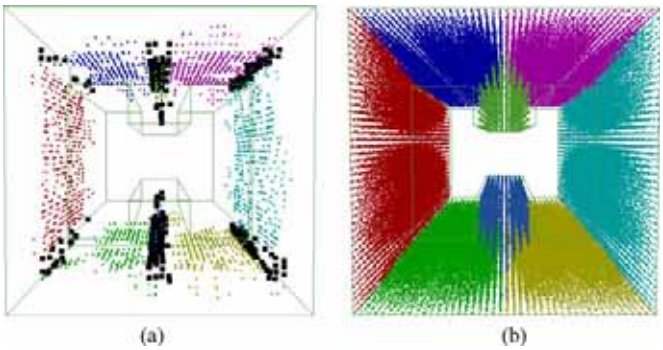


Fig. 12. Segmentation result: (a) skeleton nodes; (b) segmentation results.

component with approximate synchronization. Each node forwards the message when it receives the first message, otherwise the message is discarded. This way every node in the network will be aware of the segment it belongs to; see Fig. 12b. This skeleton-assisted segmentation algorithm is quite simple and efficient for 3D sensor networks. One possible problem is that a joint node marked with a thick black circle in Fig. 12a is associated with at least three boundary landmarks. For these joint nodes, we label a unique ID for every pair of boundary landmarks of a joint node and then assign the joint node corresponding to the pair of landmarks with the smallest ID.

5 Conclusion

We have presented CABET, a Connectivity-based Boundary Extraction scheme for large-scale sensor networks in a 3D space. The algorithm requires connectivity information only, without using location and distance information. By highlighting the critical nodes CABET can well capture the distinct features for recovering the original boundary topology. We have demonstrated CABET's efficacy and robustness against region variations. It outperforms a state-of-the-art algorithm under various configurations. Besides, we describe how 3D sensor network applications such as skeleton extraction, network segmentation, multi-resolution representation, and localization, can benefit from a boundary extraction algorithm.

We are interested in several directions of future work. We next anticipate CABET's improvement to reduce the boundary error (E_1 and E_2). We will seek more efficient algorithms to reduce the traffic overhead of our extraction technique. We also anticipate CABET's applications to more sensor network applications such as 3D coverage and 3D geographic routing. Besides, one limitation of CABET is that it relies on the assumption that network nodes are relatively evenly distributed as mentioned. First, we would like to qualitatively formulate this assumption. Second, we hope to extend this work for more general cases in the future. Finally, we plan to conduct experiments under realistic testbeds.

Acknowledgments

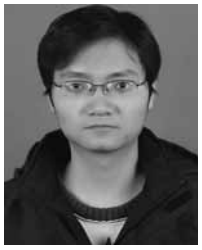
This work was supported in part by the National Natural Science Foundation of China and Microsoft Research Asia under Grant 60933012; by the National Natural Science Foundation of China under Grants 61073147, 61173120, 61103243, 61202460, 61271226, and 61272410; by the Fundamental Research Funds for the Central Universities under Grant 2012QN078; by the CHUTIAN Scholar Project of Hubei Province; by the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry); by the National Natural Science Foundation of Hubei Province under Grant 2011CDB044; by the Fok Ying Tung Education Foundation under Grant 132036; by the Hong Kong Scholars Program under Grant XJ2012019; and by the Program for New Century Excellent Talents in University under Grant NCET-10-408 (State Education Ministry). Dr. Guang Tan's work was supported by the National Natural Science Foundation of China under Grant 61103243, Youth Innovation Promotion Association, the Ministry of Science and Technology 863 Key Project No. 2011AA010500, and Shenzhen Overseas High-level Talents Innovation and Entrepreneurship Funds KQC201109050097A. An earlier version of this work appeared as [14].

References

[1] N. Amenta, M. Bern, and M. Kamvysselis, "A New Voronoi-Based Surface Recontruction Algorithm," *Proc. ACM SIGGRAPH*, 1998.
[2] X. Bai, C. Zhang, D. Xuan, J. Teng, and W. Jia, "Low-Connectivity and Full-Coverage Three Dimensional Wireless Sensor Networks," *Proc. ACM MobiHoc*, 2009.
[3] J. Bruck, J. Gao, and A.A. Jiang, "Map: Medial Axis Based Geometric Routing in Sensor Networks," *Proc. ACM MobiCom*, 2005.
[4] J. Bruck, J. Gao, and A. A. Jiang, "Map: Medial Axis Based Geometric Routing in Sensor Networks," *Wireless Networks*, vol. 13, no. 6, pp. 835-853, 2007.
[5] E. Cayirci and T. Coplu, "SENDROM: Sensor Networks for Disaster Relief Operations Management," *Wireless Networks*, vol. 13, no. 3, pp. 409-423, 2007.
[6] W. Cheng, A.Y. Teymorian, L. Ma, X. Cheng, X. Lu, and Z. Lu, "Underwater Localization in Sparse 3D Acoustic Sensor Networks," *Proc. IEEE INFOCOM*, 2008.
[7] D. Dong, Y. Liu, and X. Liao, "Fine-Grained Boundary Recognition in Wireless Ad Hoc and Sensor Networks by Topological Methods," *Proc. ACM MobiHoc*, 2009.
[8] M. Fennell and R. Wishner, "Battlefield Awareness via Synergistic SAR and MTI Exploitation," *IEEE Aerospace and Electronic Systems Magazine*, vol. 13, no. 2, pp. 39-43, Feb. 1998.
[9] S. Funke and N. Milosavljevic, "Guaranteed-Delivery Geographic Routing under Uncertain Node Locations," *Proc. IEEE INFOCOM*, 2007.
[10] R. Hekmat and P. V. Mieghem, "Connectivity in Wireless Ad-Hoc Networks with a Log-Normal Radio Model," *Mobile Networks and Applications*, vol. 11, no. 3, pp. 351-360, 2006.
[11] H. Hoppe, T. DeRose, and T. Duchampy, "Surface Reconstruction from Unorganized Points," *Proc. ACM SIGGRAPH*, 1992.
[12] H. Jiang, W. Liu, D. Wang, C. Tian, X. Bai, X. Liu, Y. Wu, and W. Liu, "CASE: Connectivity-Based Skeleton Extraction in Wireless Sensor Networks," *Proc. IEEE INFOCOM*, 2009.
[13] H. Jiang, W. Liu, D. Wang, C. Tian, X. Bai, X. Liu, Y. Wu, and W. Liu, "Connectivity-Based Skeleton Extraction in Wireless Sensor Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 21, no. 5, pp. 710-721, May 2010.
[14] H. Jiang, S. Zhang, G. Tan, and C. Wang, "CABET: Connectivity-Based Boundary Extraction of Large-Scale 3D Sensor Networks," *Proc. IEEE INFOCOM*, 2011.
[15] M. Jin, S. Xia, H. Wu, and X. Gu, "Scalable and Fully Distributed Localization with Mere Connectivity," *Proc. IEEE INFOCOM*, 2011.
[16] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, and L. Mo, "Does Wireless Sensor Network Scale? A Measurement Study on GreenOrbs," *Proc. IEEE INFOCOM*, 2011.
[17] A. Nguyen, N. Milosavljevic, Q. Fang, J. Gao, and L.J. Guibas, "Landmark Selection and Greedy Landmark-Descent Routing for Sensor Networks," *Proc. IEEE INFOCOM*, 2007.
[18] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic Routing without Location Information," *Proc. ACM MobiCom*, 2003.
[19] R. Sarkar, X. Yin, J. Gao, F. Luo, and X.D. Gu, "Greedy Routing with Guaranteed Delivery Using Ricci Flows," *Proc. IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN)*, 2009.
[20] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat Monitoring with Sensor Networks," *Comm. ACM*, vol. 47, no. 6, pp. 34-40, 2004.
[21] G. Tan, M. Bertier, and A.-M. Kermarrec, "Convex Partition of Sensor Networks and Its Use in Virtual Coordinate Geographic Routing," *Proc. IEEE INFOCOM*, 2009.
[22] G. Tan, H. Jiang, S. Zhang, and A.-M. Kermarrec, "Connectivity-Based and Anchor-Free Localization in Large-Scale 2D/3D Sensor Networks," *Proc. ACM MobiHoc*, 2010.

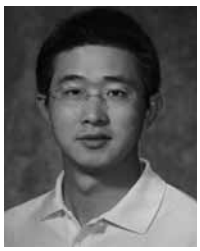
[23] Y. Wang, J. Gao, and J.S. Mitchell, "Boundary Recognition in Sensor Networks by Topological Methods," *Proc. ACM MobiCom*, 2006.
[24] H. Zhou, S. Xia, M. Jin, and H. Wu, "Localized Algorithm for Precise Boundary Detection in 3D Wireless Networks," *Proc. IEEE 30th Int'l Conf. Distributed Computing Systems (ICDCS)*, 2010.
[25] X. Zhu, R. Sarkar, and J. Gao, "Shape Segmentation and Applications in Sensor Networks," *Proc. IEEE INFOCOM*, 2007.

Hongbo Jiang received the BS and MS degrees from Huazhong University of Science and Technology, Wuhan, China, and the PhD degree from Case Western Reserve University, Cleveland, Ohio, in 2008. After that, he joined the faculty of Huazhong University of Science and Technology as an associate professor. His research interests include computer networking, especially algorithms and architectures for high-performance networks and wireless networks. He is a member of the IEEE.



Shengkai Zhang received the BS degree from Huazhong Normal University, China. He is working toward the MS degree at Huazhong University of Science and Technology since 2009. His research interests include wireless networking, especially algorithms and architectures for sensor networks.

Guang Tan received the BS degree from the Chongqing University of Posts and Telecommunications, China, the MS degree from the Huazhong University of Science and Technology, China, and the PhD degree in computer science from the University of Warwick, Coventry, United Kingdom, in 1999, 2002, and 2007, respectively. He is currently an associate researcher at Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China, where he works in the area of distributed systems and networks. From 2007 to 2010, he was a postdoctoral researcher at INRIA-Rennes, France. He is a member of the IEEE.



Chonggang Wang received the PhD degree in computer science from Beijing University of Posts and Telecommunications, China. He has conducted research with NEC Laboratories America, AT&T Labs Research, and the University of Arkansas and Hong Kong University of Science and Technology. His research interests include future Internet, machine-to-machine (M2M) communications, and cognitive and wireless networks. He has published more than 80 journal/conference articles and book chapters. He is a senior member of the IEEE.