Connectivity-Based Skeleton Extraction in Wireless Sensor Networks

Hongbo Jiang, *Member*, *IEEE*, Wenping Liu, Dan Wang, *Member*, *IEEE*, Chen Tian, *Member*, *IEEE*, Xiang Bai, Xue Liu, *Member*, *IEEE*, Ying Wu, *Senior Member*, *IEEE*, and Wenyu Liu, *Member*, *IEEE*

Abstract—Many sensor network applications are tightly coupled with the geometric environment where the sensor nodes are deployed. The topological skeleton extraction for the topology has shown great impact on the performance of such services as location, routing, and path planning in wireless sensor networks. Nonetheless, current studies focus on using skeleton extraction for various applications in wireless sensor networks. How to achieve a better skeleton extraction has not been thoroughly investigated. There are studies on skeleton extraction from the computer vision community; their centralized algorithms for continuous space, however, are not immediately applicable for the discrete and distributed wireless sensor networks. In this paper, we present a novel Connectivity-bAsed Skeleton Extraction (CASE) algorithm to compute skeleton graph that is robust to noise, and accurate in preservation of the original topology. In addition, CASE is distributed as no centralized operation is required, and is scalable as both its time complexity and its message complexity are linearly proportional to the network size. The skeleton graph is extracted by partitioning the boundary of the sensor network to identify the skeleton points, then generating the skeleton arcs, connecting these arcs, and finally refining the coarse skeleton graph. We believe that CASE has broad applications and present a skeleton-assisted segmentation algorithm as an example. Our evaluation shows that CASE is able to extract a well-connected skeleton graph in the presence of significant noise and shape variations, and outperforms the state-of-the-art algorithms.

Index Terms—Sensor networks, algorithm/protocol design, skeleton extraction.

1 Introduction

Sensor networks today are widely used as they are able to capture the phenomena of the physical world that were originally difficult or impossible to obtain by traditional techniques. Existing examples include disaster relief [6], habitat monitoring [23], battlefield surveillance [12], etc. For these applications, however, the sensor networks cannot be deployed well-planned in advance as conventional networks. The geographical locations and deployment methods may vary greatly, and the topology of the sensor networks is affected by such factors as obstacles, deployment randomness, holes, etc. The shape of the sensor network hardly conforms to simple shapes such as a square or a disk; which are mainly used in research studies [13], [21]. As a concrete example, a butterfly-shape sensor network (e.g., Fig. 1a) is representative for airport terminals or train maps shown in Fig. 2.

The performance of the sensor network, e.g., the efficiency of data routing, localization, and path planning, heavily depends on the distribution of the sensors and the overall network topology. Some previous studies have shown that the topological *skeleton* (or so-called *medial axis*) information of the sensor network can greatly improve routing performance [3]. In this paper [3], the routing efficiency is compared with GPSR, another wireless routing scheme where only local neighbor information is available; and great improvement is observed. Follow-ups have used skeleton information for landmark selection to provide location service [18], segmentation [25], and navigation algorithms [5]. We argue that geometric skeleton information can be useful far more than routing, location, segmentation, and path design. For instance, for a mobility-assisted sensor network for field coverage, the skeleton can be used for path planning since the nodes in the skeleton represent a medial axis, and thus, cover most area where all nodes are observing. Thus, to understand the characteristics of the sensor topology and extract some important geometric information are essential to the sensor networks.

The previous works focus on the application of the skeleton information for various purposes in the sensor networks. The performance of the *skeleton extraction*, however, is not the focus of these studies. In this paper, we present a novel distributed Connectivity-bAsed Skeleton Extraction (CASE) algorithm which can better represent the topological shape of the sensor network than previous studies while incurring comparable communication cost.

- *H. Jiang, W. Liu, C. Tian, X. Bai, and W. Liu are with the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, Hubei 430074, P.R. China. E-mail: {hongbojiang2004, openheart611, xiang.bai}@gmail.com, {chentian, liuwy}@hust.edu.cn.*
- *D. Wang is with the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. E-mail: csdwang@comp.polyu.edu.hk.*
- *X. Liu is with the Department of Electrical and Computer Engineering, School of Computer Science, McGill University, Montreal, Quebec, Canada. E-mail: xueliu@cs.mcgill.ca.*
- *Y. Wu is with the EECS Department, Northwestern University, Evanston, IL 60208. E-mail: yingwu@northwestern.edu.*

Manuscript received 4 Mar. 2009; revised 1 June 2009; accepted 9 June 2009; published online 26 June 2009.
Recommended for acceptance by T. Abdelzaher.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2009-03-0097.
Digital Object Identifier no. 10.1109/TPDS.2009.109.

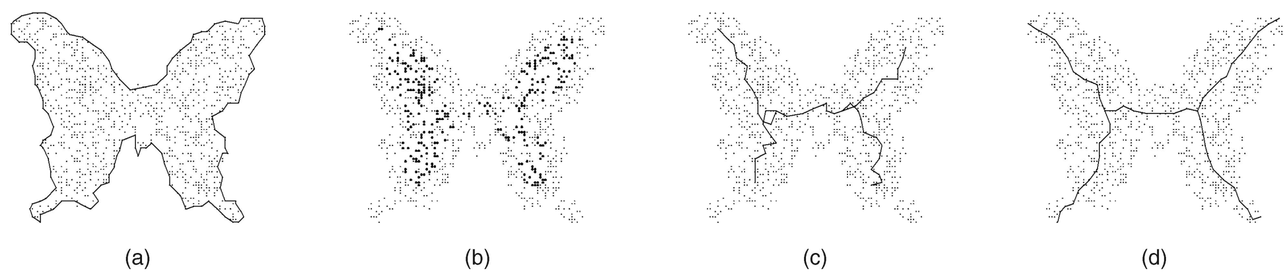


Fig. 1. An example network. (a) Boundary by the approach in [24]. (b) Medial axis points according to the definition in [3], [4]. (c) Medial axis graph computed by MAP in [3], [4]. (d) Skeleton graph computed by CASE.

We illustrate the comparison by a simple example in Fig. 1. We see that Fig. 1c shows the medial axis graph computed by [3], [4] and Fig. 1d is the skeleton graph computed by CASE. Clearly, CASE grasps the geometric information better, and avoids possible loops.

Skeleton or medial axis extraction schemes have been studied in computer vision [8] and graphics [22] communities. While our proposed algorithm carries some of their advantages, there are some intrinsic differences due to the unique characteristics of the wireless sensor networks; for example, the sensor networks are discrete and the skeleton should be computed given only local connectivity information. Thus, we face numerical challenges in designing an efficient skeleton extraction algorithm. *First*, sensor nodes randomly deployed often have no location information or distance information. This makes topology recognition difficult as various techniques widely used in computer vision fields for skeleton extraction that request centralized computation are not suitable for a sensor network. *Second*, in a discrete network, the definition of the distance between skeleton nodes has to use hop-count distance instead of euclidean distance. This will result in noise, e.g., incorrect estimation of boundaries. It has been shown that a little noise or deformation at the boundary often seriously disturbs the topology of the skeleton graph [3]. As a result, many well-known algorithms [8], [1], [14] applied in other fields to extract skeleton cannot be used directly. Effective elimination of the unstable segments in skeleton to keep the genuine geometric features is challenging. *Third*, even after the skeleton nodes are identified, connecting all nodes in a proper way is not as straightforward as that in the continuous case. In many cases, there are not enough skeleton nodes that can be extracted to construct a connected skeleton. In addition, the paths generated to connect skeleton nodes may either not be the shortest paths or introduce a cycle as shown in Fig. 1c.

Toward taming these challenges, in this paper, we propose a novel skeleton extraction approach—CASE, and make the following contributions:

- We present the design and implementation of a practical and distributed skeleton extraction algorithm which requests only connectivity information and preserves the topology well.
- Based on boundary partition techniques, our proposed algorithm is robust against noise, and thus, capable of keeping the genuine geometric features of the sensor network.
- We have proved that CASE is a scalable algorithm as its time and message complexity are linearly proportional to the network size.
- Our work has implications for topology recognition in general. The results suggest that the user should keep track of the importation points such as skeleton joints and corner points in order to better understand the topology.

The remaining part of this paper proceeds as follows: Section 2 presents related work. Section 3 outlines the background of skeleton extraction and the motivation of the work. Section 4 is devoted to the skeleton extraction algorithm. Issues related to overhead and refinement are discussed in Section 5. We evaluate our scheme in Section 6. Finally, Section 7 summarizes the paper and future plans.

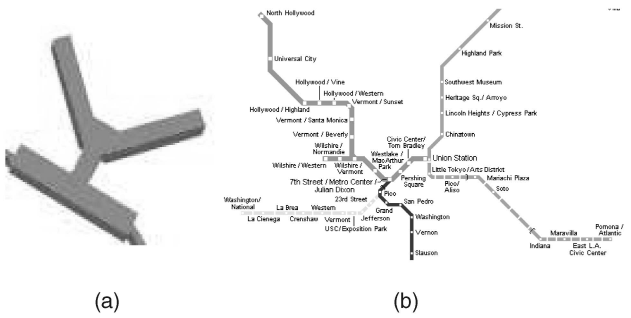


Fig. 2. (a) Chicago airport terminal 2. (b) A part of Los Angeles train map.

2 Related Work

Topology recognition is crucial to many sensor network applications. Boundary detection and skeleton extraction are by far the most widely used methods for topology recognition in sensor networks. These two categories of methods have been explored in recent years. The first category, boundary detection, focuses on detecting nodes on the inner or outer boundaries. Fang et al. [10] proposed a simple algorithm built on the idea that a packet can only get stuck at a node on hold boundaries, while assuming the nodes know their geographical location information. Fekete et al. [11] proposed an algorithm based on the observation that the boundary nodes often have much lower degree than inner nodes. Without the assumption of location geographical information, Ghrist and Muhammad [13] proposed a centralized algorithm to detect holes via homology. Wang et al. [24] developed a practical distributed algorithm also only using connectivity information. Motivated by the observation that holes create irregularities in hop count distances, they identify those nodes where shortest paths of distinct homotopy types terminate and touch each other, trapping the holes between them. Saukh et al. [21] proposed to determine the inner nodes using geometric constructions based on the d-quasi unit disk graph model for radio propagation. Their algorithm works well even if there is no hole in the network.

For the second category, little has been done on skeleton extraction even though it is considered to be an important step for many other sensor applications. Lin and Lee [19] proposed a dynamic medial axis model that represents shapes and changes of shapes in a geometric space. One work that is close to our work is MAP [3], [4]. They considered to define skeleton points using hop count such that nodes with equal distances to two closest boundary nodes can be identified to skeleton points. In addition, connecting these skeleton points provides a construction of the skeleton graph. However, we emphasize the differences from their work. *First*, and most importantly, motivated by the process of Discrete Curve Evolution, we partition the boundary according to the corner points, and thus, the global information, instead of local, significant of the shape is maintained. Our algorithm is more stable under small deformations. *Second*, MAP does not guarantee that the skeleton points it identifies form a connected component, and thus, incurs the problem that the shortest path between two nodes may not represent the shape of the geometric environment. Instead, our algorithm preserves the shape via constructing a connected component using skeleton points, and thus, the path in this component is able to represent the exact shape of the environment. *Third*, we refine the coarse skeleton graph using local flooding on it and achieve better results. *Fourth*, we have proven the skeleton extraction algorithm's time and message complexity. Although we have a higher message overhead than [3], [4], it is noted that our experiments in Section 6 show that the number of average messages needed per node is only three to eight. We believe it is acceptable for most applications. Therefore, this sacrifice for a better skeleton extraction is justifiable.

3 Background and Motivation

Before introducing the proposed approach, we give a description of some definitions which will be used in the rest of this paper. According to Blum's definition of the medial axis [2], the skeleton S of a set P is the locus of the centers of the maximal disks. A maximal disk of P (see Fig. 3a for the disks centered at A-F) is a closed disk contained in P that is interiorly tangent to the boundary ∂P and that is not contained in any other disk in P. Each maximal disk $B(s)$ with a center s must be tangent to the boundary at two (or more) different points which are called *generated points*. By Theorem 8.2 in [7], the skeleton S is a geometric graph, which means that S can be decomposed into a finite number of connected arcs, called *skeleton arcs*, composed of points of degree two, e.g., all the points on AD, ED, BA, DF, and CA in Fig. 3a. These points are called *connection points*. The arcs meet at *skeleton joints* (or *bifurcation points* [14]) that are points of

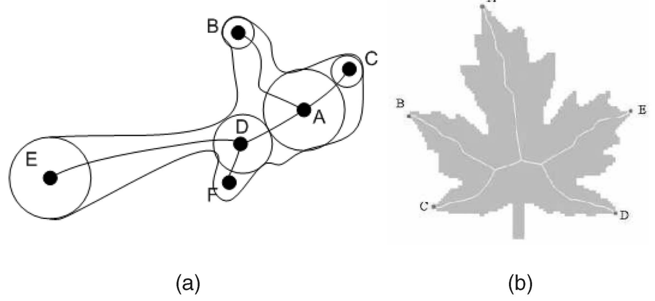


Fig. 3. (a) A skeleton with different types of skeleton points. (b) Boundary partitioned by five corner points on the boundary.

degree three or higher (A, D in Fig. 3a). The skeleton point having, for example, B, C, E, F in Fig. 3a.

In computer vision [14], [1] studies have obtained good skeletons with the aid of the boundary *corner points*. We purposely delay the definition of corner points. Intuitively, it is one kind of critical points and using them as the endpoints of the skeleton allows explicit preservation of the topological information of the shape. As shown in Fig. 3b, the boundary of the maple leaf is partitioned by five corner points A, B, C, D, E; which leads to five boundary segment points having their generated points on these segments. As such, via boundary partition, the topology can be fixed and redundant arcs of skeletons can be disregarded [1]; leading to good skeletons.

Unfortunately, the aforementioned methods are very sensitive to boundary noise and deformation [8], [1]; thus, they cannot be directly applied to the discrete case. In the discrete case, skeleton points are usually referred to the nodes that have equal hops to two closest boundary nodes [4]. Unlike the continuous case, the boundary can have serious noise and deformation as the hop count is only an approximation of euclidean distance [4]. In other words, extracting skeleton in a sensor network is not stable due to the undesirable nodes that have equal distance to two close-by nodes on the boundary [4]. The solutions proposed in [3], [4] remove the unnecessary nodes in the skeleton, see Fig. 1b, via setting a global threshold. This leads to problems when the removed skeleton arcs represent important geometric patterns or deformation exists on the boundary.

Motivated by Discrete Curve Evolution (DCE) [16], which results in a fine-grained boundary partition, our skeleton extraction algorithm is based on associating a boundary partition with skeleton graph generation. As such, the main idea behind our algorithm is to remove all skeleton points whose nearest boundary nodes lie on the same boundary segment. We emphasize the difficulties of the design and implementation on boundary partition using DCE in the wireless sensor networks. These are as follows:

1. we have to extend the definition of corner points and the definition of joint points;
2. we need to deal with the case where multiple joint points emerge;
3. we need to find a path between two joint points to approximate the skeleton arcs, and
4. we need to refine the coarse results based on connectivity information, etc.

We will detail our proposed algorithm in the next section.

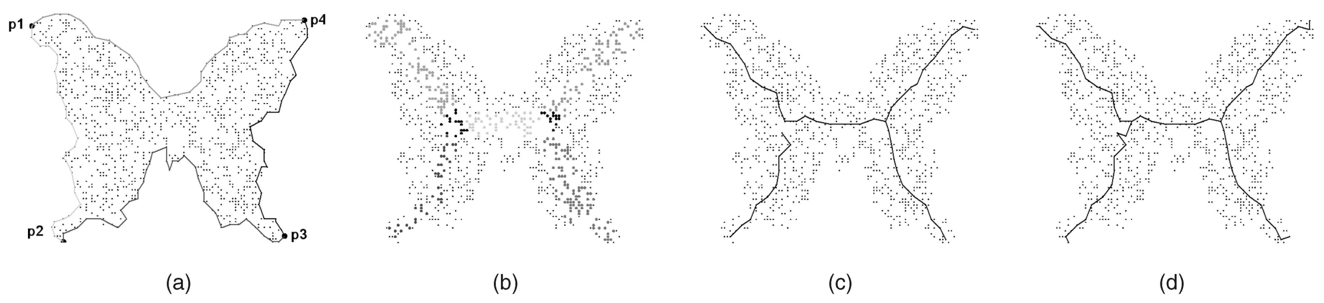


Fig. 4. Skeleton extraction algorithm of an example sensor network with the average degree of 20.9. The final refined skeleton is shown in Fig. 1d. (a) Boundary segments. (b) Skeleton points. (c) Skeleton arcs. (d) Coarse skeleton.

4 Skeleton Extraction Algorithm

As we mentioned in Section 3, to determine whether a node is a skeleton point, the corresponding nearest boundary point should be determined. To that end, the first step is to detect boundary points. While our skeleton extraction algorithm is derived from the boundary, the boundary construction is out of the scope of this work. Numerous recent studies, for instance, [9], [21], [24], have provided boundary detection and recognition algorithms. We bear this in mind and state that many of these algorithms can be used in conjunction with our approach. That is, we assume the boundary points are given as a system input in our algorithms. Overall, the CASE algorithm includes five main building blocks:

1. Find corner points on the boundary such that the whole boundary can be decomposed into a finite number of boundary segments.
2. Identify the skeleton points such that they form a connected component.
3. Generate a set of skeleton arcs by connecting the skeleton points.
4. Generate a coarse skeleton result by connecting skeleton arcs and corner points.
5. Refine the coarse result to obtain the final skeleton.

4.1 Boundary Segments

The first step of CASE is to decompose the boundary ∂P by *corner points* into a finite number of open boundary segments. To that end, we need to identify the corner points on the boundary at first, as mentioned in Section 3.

While the corner point can be defined in terms of the curvature of the local shape in a continuous case, it is not straightforward to use it in discrete networks. We formalize here the discrete curvature as a simplification criterion, which is able to represent geometric shape well. For any boundary point p, we refer to its h-hop neighbors as $N_h(p)$. Let the maximal hop count between two nodes in $N_h(p)$ be denoted by $M_h(p)$. Intuitively, for an inner point or some boundary points, $M_h(p)$ is often $2h$. Thus, we define the discrete curvature by $\rho_p = \max_{h=1,\dots,H} M_h(p)/2h$ (in experiments, we found that $h = 3$ or 4 provides good results in most cases). For a point $p \in \partial P$, if ρ_p is less than a given threshold δ_ρ, say 0.5 in our implementation (later, we will show different thresholds that allow to generate multi-resolution skeleton results), the point p is defined to be a *corner point*. With a set of corner points, marked with thick black circles in Fig. 4a, the boundary ∂P can be decomposed into four open *boundary segments* $p1p2$, $p2p3$, $p3p4$, and $p1p4$, as shown in Fig. 4a.

4.2 Skeleton Points Identification

As mentioned in Section 3, skeleton S can be decomposed into skeleton arcs which are composed of skeleton points. To that end, we turn to identifying the skeleton points each of which is associated with at least two boundary segments.

The definition that the nodes that have the exactly equal distances to at least two boundary segments are skeleton points [4] is not suitable in many applications, especially for a sparse sensor network where few nodes are identified to be skeleton points. We observed that the skeleton points based on this definition will be quite few in sensor networks with a moderate average degree (see Fig. 12a in Section 6). It is noted that, as opposed to the continuous case [1], [8], [14], in a discrete network, there is no guarantee that all skeleton points constitute a connected component. Thus, we refer to *skeleton points* associated with two boundary segments, say C_i and C_j, as those nodes such that: 1) the difference of the absolute distances to two nearest boundary segments is less than a given threshold, denoted by $\delta_p \geq 0$; and 2) the component composed of those nodes, denoted by $S(C_i, C_j)$, is a connected component. It is noted that we do not claim these two boundary segments are neighbors to each other. Instead, two nearest boundary segments to the node are considered; for example, those nodes marked with green color in Fig. 4b (that is, the center part of skeleton points) are associated with two nonneighbor boundary segments $p1p4$ and $p2p3$. Via this definition, parts of skeleton points can be well connected into skeleton arcs which will be described in the next section. Our process to identify skeleton points works as follows: First, every node in the network initiates a flooding to obtain its distances to boundary segments. Those nodes having equal distance to C_i and C_j (that is, $\delta_p = 0$) definitely belong to $S(C_i, C_j)$. Then, any $p \in S$ initiates a flooding to check whether S is a connected component or not. If not, we iteratively increase δ_p value by one to include more nodes into the set S until $S(C_i, C_j)$ is a connected component. It is noted that different points have different thresholds. For instance, in Fig. 4b, the δ_p value of the component composed of blue points is 0, but for other components, this value is 1.

Next, we turn to investigating the complexity of the connected component generation procedure. Let $d(p, C_j) = \min_{q \in C_j} dist(p, q)$ be the distance between a sensor node p

and a boundary segment C_i. We refer to $\hat{d}(C_i, C_j) = \max\{\max_{p \in C_i} d(p, C_j), \max_{q \in C_j} d(C_i, q)\}$ as the maximum distance between two segments. The following theorem shows that the above-mentioned procedure, i.e., resetting δ_p, is limited by a threshold: As a concrete example, in Fig. 4, the maximal value of δ_p is only 1. Besides, in all our test scenarios in Section 6, the maximal values of this parameter are less than 3.

Lemma 1. *For each point $p \in S(C_i, C_j)$, we have $d(p, C_j) \leq \hat{d}(C_i, C_j)$. That is, for any skeleton point, its distance to the segment is less than the maximum distance between two segments.*

Theorem 1. *The times of resetting δ_p are limited by a value of $\hat{d}(C_i, C_j)$.*

Proof. We prove it by contradiction. Assume that when $\delta_p = \hat{d}(C_i, C_j)$, $S(C_i, C_j)$ is still not connected. That is, there exists at least one point p and one of its neighbor $q \notin S(C_i, C_j)$. On the other hand, the absolute value of the difference of the distances $|d(q, C_i) - d(q, c_j)|$ should be less than $\hat{d}(C_i, C_j)$. This is a contradiction with the fact that $q \notin S(C_i, C_j)$ since $\delta_p = \hat{d}(C_i, C_j)$. □

As a result, by resetting δ_p, we obtain the set of skeleton points, denoted by $S(P)$ and shown in Fig. 4b, where the skeleton points associated with different boundary segments are marked with different color thick circles. This result shown in Fig. 4b, obviously, is much better than that using the method in [3], [4], shown in Fig. 1b. Here, the better results are obtained as: 1) the set of our skeleton points is connected; and 2) by identifying skeleton points that are coupled with two boundary segments, CASE is able to remove many undesirable nodes introduced by MAP [3], [4] (see Fig. 10a in Section 6). Unlike the continuous case, we refer to the *joint points* as the nodes that belong to at least two components $S(C_i, C_j)$ and $S(C_i, C_k)$ in a discrete network, that is, $S(C_i, C_j) \cap S(C_i, C_k)$. The joint points are marked with thick black circle in Fig. 4b.

4.3 Skeleton Arcs Generation

Since skeleton S can be decomposed into a set of skeleton arcs, an important step in the construction of the skeleton is to generate skeleton arcs by connecting the skeleton points $S(P)$ in a correct way.

Unlike previous work [4], we do not claim to include all skeleton points into our skeleton arcs. Instead, we construct the skeleton arcs based on the fact that parts of skeleton points are enough to construct a good approximation of the theoretical skeleton in a discrete network.

We select two such nodes x' and y' that have the longest length path in the component corresponding to two segments C_i and C_j, that is, $(x', y') = argmax_{x,y \in S(C_i,C_j)} \hat{dist}(x, y)$. We emphasize that $\hat{dist}(x, y)$ is different from $dist(x, y)$ in Section 4.1 since $\hat{dist}$ represents the length of the shortest path that is limited within the component $S(C_i, C_j)$ (it is easy to show that $\hat{dist}(x, y) \leq dist(x, y)$). We refer to the path (called *diametral path* [8]) from x' to y' as the longest path in $S(C_i, C_j)$. Here, we shed light on the means to calculate the distance $i\hat{dist}(\cdot, \cdot)$. Each node $p \in S(C_i, C_j)$ initiates a flooding with its own ID and a counter to indicate how many hops the message has traveled, as well as each node ID across this path.

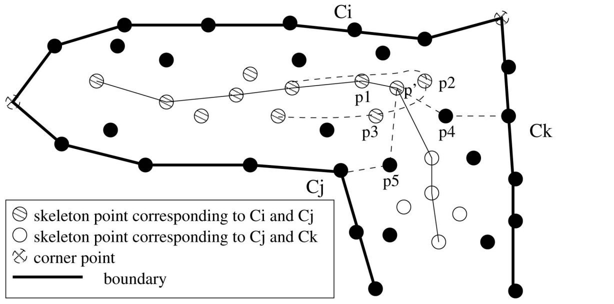

Ci
Ck
Cj
p1
p2
p'
p3
p4
p5
skeleton point corresponding to Ci and Cj
skeleton point corresponding to Cj and Ck
corner point
boundary

Fig. 5. The joint point p' has equal distance to three boundary segments C_i, C_j, and C_k.

This flooding process is limited within the connected component $S(C_i, C_j)$. For p's all neighbors, those nodes having exactly equal distances to C_i and C_j begin with forwarding the message and incrementing the counter. For those nodes whose distances to C_i and C_j are unequal, they then forward the flooding message. By doing so, we allow to calculate the distance between two nodes. Meanwhile, the path between two nodes is recorded and this path is almost traveling through those nodes having equal distances to C_i and C_j. The following theorems show that one endpoint of the longest path should be the joint point if there exist joint points:

Lemma 2. *For the connected component $S(C_i, C_j)$, if there exist joint points, not all joint points are inner points of $S(C_i, C_j)$. That is, there exists joint point $p' \in S(C_i, C_j)$ and p' is on the boundary of $S(C_i, C_j)$.*

Proof. We prove by contradiction. Suppose all joint points are inner points of $S(C_i, C_j)$. According to the definition of the joint point, without loss of generality, assume that the joint points correspond to boundary segments C_i, C_j, and C_k. There exist a joint point p' and at least three boundary points $p1$, $p2$, and $p3$ of $S(C_i, C_j)$. These three boundary points are not joint points and form a polygon such that p' is an inner point of this polygon [21] as shown in Fig. 5.

Note that p' is associated with C_i and C_k and assume the minimal length path from p' to C_k is traveling through $p4$. Since p' is an inner point of the polygon $p1p2p3$, at least one point, say $p2$ in Fig. 5, has the same neighbor $p4$ as well. In this sense, $p2$ has equal distance to C_i and C_k, that is, $p2$ corresponds to C_k. This is a contradiction with the fact that the points $p1$, $p2$, and $p3$ only correspond to C_i and C_j; otherwise, they are joint points as well. Thus, the joint point p' cannot be an inner point. □

Theorem 2. *If there exist joint points in the connected component $S(C_i, C_j)$, at least one endpoint of the longest path should be a joint point.*

Proof. It is noted that when one skeleton point p is associated with boundary segment C_i, there exists one generated point $C_p \in C_i$ which has the minimal distance to p. There exists a joint point p' which is on the boundary of the component $S(C_i, C_j)$. Thus, its generated $C_{p'}$ should be the closest one to the corner (that is, the endpoint of C_i) among all corresponding nodes associated with other skeleton points $\{p | p \in C_i\}$. The longest path from p', accordingly, has the other endpoint p'' whose corresponding node in C_i is the closest one to the endpoint of C_i. Otherwise, this is not a longest path since there exists another node which has a longer path to p' than p''. □

One implication of this theorem is that when the component $S(C_i, C_j)$ contains joint points, we do not have to ask all skeleton points initiating a flooding to find a pair of nodes with longest distance. Instead, only a joint point $p' \in S(C_i, C_j) \cap S(C_i, C_k)$ is required to initiate a flooding to find the other endpoint with longest distance between them. That is, when there exists a joint point in the component, we reduce the traffic cost to a considerable extent. Unlike the continuous case [14], it is noted that the joint points $S(C_i, C_j) \cap S(C_i, C_k)$ may contain many nodes, as shown in Fig. 4b where joint points are marked with thick black circles. In this case, our idea is to select the node closest to the center of $S(C_i, C_j) \cap S(C_i, C_k)$. Hence, the joint point with the largest degree in the subgraph $S(C_i, C_j) \cap S(C_i, C_k)$ is selected to be the endpoint of the skeleton arc.

So far, we have obtained two endpoints of the longest path in the subgraph $S(C_i, C_j)$. One characteristic of this path is that all nodes on the path are skeleton points. We call this path to be *skeleton arc*, denoted by $A(C_i, C_j)$. For every pair of boundary segments C_i and C_j, one skeleton arc can be extracted accordingly, as shown in Fig. 4c. Next, we discuss how to connect all skeleton arcs in a right way such that a coarse skeleton graph is obtained.

4.4 Coarse Skeleton: Connecting Skeleton Arcs and the Corner Points

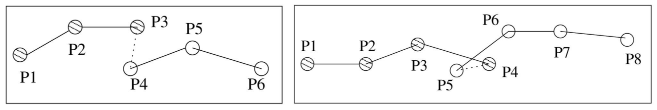
While in a continuous case, the skeleton S can be seamlessly decomposed into a set of skeleton arcs which are intrinsically connected, that is not always the case in a discrete sensor network. Hence, an additional step of CASE is to connect all the skeleton arcs and corner points, in order to generate a *coarse* skeleton, as shown in Fig. 4d.

For two skeleton arcs A_1 and A_2, we refer to $\hat{dist}(A_1, A_2) = \min \hat{dist}_{x \in A_1, y \in A_2}(x, y)$ as the distance of these two arcs. In a discrete network, it is possible that two close skeleton arcs A_1 and A_2 are not neighbor, that is, $\hat{dist}(A_1, A_2) > 1$. It is known that each skeleton corresponds to boundary segments. When these two skeleton arcs correspond to a same boundary segment, say C_i, these two arcs are supposed to be connected. In this case, we find two nearest points x' and y', where $\hat{dist}(x', y') = \hat{dist}(A_1, A_2)$. To find a path from x' to y', the approach to flooding is similar to that in Section 4.3. Again, during flooding, the path is based on the best-effort style to travel those nodes which have similar distance to different boundary segments.

Another step is to connect the corner points and skeleton arcs to extract better skeleton used for topology recognition. To that end, each corner point initiates a flooding to find a path to connect its closest arc's endpoint. By doing so, the corner points become the endpoints of skeleton S, as mentioned in Section 3.

4.5 Coarse Skeleton Refinement

A careful investigation of the above connecting method reveals that this coarse skeleton often cannot be very useful for topology recognition in practice. First, based on the flooding process in Sections 4.3 and Section 4.4, the path from one endpoint to another skeleton arc is not necessarily the shortest path. Fig. 6a shows the case when $p3$ is connected to $p4$ despite the fact that $p3$ is a neighbor of $p5$. Second, unlike the continuous case, the cycle could be obtained when we connect two skeleton arcs, as shown in Fig. 6b. These two problems were also observed in [3], [4].



(a) (b)

Fig. 6. Some situations where refinement is required.

To overcome these disadvantages, we randomly select one corner point to initiate a final flooding along the coarse skeleton graph. We argue that it does not introduce much traffic since the flooding is limited within the coarse skeleton. Each node on the coarse skeleton only forwards the message once. For example, when node $p3$ in Fig. 6a receives the message from $p2$, it forwards it to $p4$ and $p5$. In this case, when $p4$ tries to forward the message to $p5$, node $p5$ will discard the message but only record its parent to be $p3$. As a result, node $p4$ is removed from the skeleton and the new path is $p1p2p3p5p6$. As for the case shown in Fig. 6b, node $p3$ forwards the message to $p5$ and $p4$, but only the first message, say from $p4$, will be sent to $p6$ and $p6$ records its parent to be $p4$ as well. Thus, the new path is $p1p2p3p4p6p7p8$ (it is noted that the new path could be $p1p2p3p4p5p7p8$ as well due to the random fashion of the flooding). Finally, the refined skeleton is obtained after the last refinement, as shown in Fig. 1d.

5 Discussion

5.1 Time and Message Complexity

Time and message complexity are important factors for an efficient skeleton extraction algorithm. Let n be the total number of nodes in the network, and $\sqrt{n}$ represents the number of nodes on the boundary. Here, we assume that all nodes are roughly uniformly distributed over the sensor area. We do not consider some severe cases such as a circle-like network, where the number of nodes on the boundary could be n instead of $\sqrt{n}$ as the skeleton extraction is not useful and meaningful in those scenarios. Actually, in all cases where we conduct experiments, this assumption is not violated. The following theorem shows the scalability of CASE:

Theorem 3. *Both the time and message complexity of CASE are $O(n)$, where n is the network size.*

Proof. Let's see time complexity at first. First, the boundary partition based on DCE has a complexity of $O(\sqrt{n} \log \sqrt{n})$, which has been proved in [16], [1]. Second, to find skeleton points, each boundary node initiates a flooding within local area to enable each node to calculate its distance to three nearest boundary segments. This process usually has a time complexity of $O(1)$ as performed in a distributed fashion. In worst case, however, when the node is far away from boundary segments, say $O(\sqrt{n})$ in

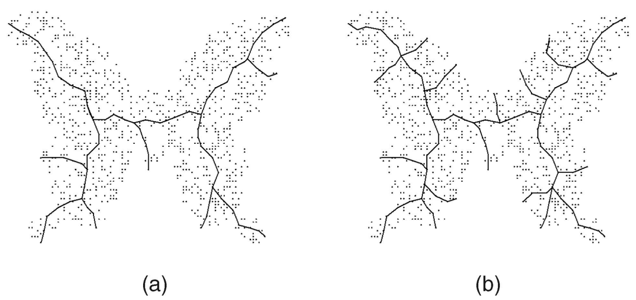


Fig. 7. Multiresolution skeletons when we set (a) $\delta_\rho = 0.75$ and (b) $\delta_\rho = 0.86$. It is noted that we set $\delta_\rho = 0.5$ in Fig. 1d.

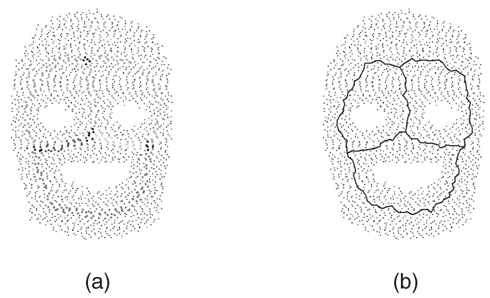


Fig. 8. An example network without the existence of corner points. (a) Skeleton points. (b) Skeleton graph.

the case where all nodes are randomly deployed in a square area, the time complexity becomes $O(\sqrt{n})$. Third, to generate the skeleton arcs, we can traverse the skeleton points and find the path with maximum length. Similar to Bellman-Ford algorithm [17], the time complexity is $O(\sqrt{n} * E)$, where E is the number of edges. Usually, the number of edges is proportional to that of nodes, and thus, the time complexity to generate skeleton arcs is $O(n)$. Fourth, local flooding performed to connect corner points and skeleton arcs has a time complexity of $O(1)$ in a distributed way. Finally, we need to traverse the coarse skeleton for refinement in a linear time $O(\sqrt{n})$. Overall, the time complexity of our algorithm is $O(n)$.

Next, let's see message complexity of CASE, that is, the traffic cost our algorithm incurs. First, to identify corner points, each node on the boundary has to initiate a flooding within local area, limited by a TTL value. Thus, the traffic cost for local area totally is $O(\sqrt{n})$. Second, the flooding by boundary nodes to identify skeleton point incurs $O(n)$ traffic since each node receives at most three messages from its nearest three boundary branches, as mentioned in Section 4.2. Third, to find maximum length path in a connected component $S(P)$ composed of a set of skeleton points, the communication cost is $O(\sqrt{n}\log\sqrt{n})$. Fourth, the message complexity of the local flooding to connect skeleton arcs and corner points is also proportional to the number of skeleton points $O(\sqrt{n})$. Finally, traveling the coarse skeleton will incur $O(\sqrt{n})$ traffic cost. Hence, the message complexity of our proposed algorithm is $O(n)$. □

Theorem 3 implies that while our proposed algorithm always uses flooding scheme to collect necessary information in each step of CASE to extract skeleton, the communication cost will not increase too much as we limit the flooding in a local area. For instance, in the step to refine coarse result, the flooding is only executed within the skeleton, containing few nodes compared with the whole sensor network.

5.2 Multiresolution Skeleton

While acknowledging our skeleton result is based on the threshold δ_ρ for identifying corner points to partition the boundary described in Section 4.1, the corner points naturally provide a good partition of the boundary into segments. The corner points' identification by providing a threshold can be considered to be the process of DCE [1], [16]. Briefly speaking, we assume there exists a subset of sample points that lie on the boundary that constitutes a polygon result approximating the underlying geometric environment. The process of DCE performs by recursively removing polygon points with the smallest shape contribution [16].

After the process of DCE, a subset of points (corner points) are accordingly extracted to provide a good approximation of the shape of a given boundary. This subset can also be viewed as a partitioning of the original boundary polygon into boundary segments. A multiresolution skeleton structure obtained by providing different thresholds is shown in Fig. 7. One desirable characteristic of DCE is the boundary partition stability [1], [16].

Multiresolution representations are desirable, as they provide a flexible tool that fits the user's needs better than single-resolution systems [20]. In sensor's applications, they can also be useful for location, navigation, and path planning (here, we do not detail these applications due to the space limit).

5.3 Special Cases for Boundary Partitioning

While our boundary partitioning is based on the corner point identification, it is possible that, in some cases, no corner point exists. Fig. 8 shows an example network where none of the nodes has a curvature higher than the threshold 0.5. This example network has many corresponding geometric environments in reality. For instance, sensors are deployed in an outdoor area, say university campus, with some buildings inside. It can be seen that it has three holes, and thus, has a total of four boundary segments. Joint points are marked with thick black circles in Fig. 8a. Recall that the basic idea behind our algorithm is that we associate each connected component (shown in Fig. 8a with different colors, composed of skeleton points) with two boundary segments. Hence, skeleton arcs can still be generated followed by coarse and refined skeleton graphs, as shown in Fig. 8b.

5.4 Skeleton-Assisted Segmentation

Segmentation algorithm in sensor networks is aimed at partitioning an irregular sensor field into nicely shaped parts to allow the assumption of a uniform and dense sensor distribution in each part [25]. The procedure of our skeleton extraction algorithm inherently introduces a follow up of shape segmentation algorithm. Recall that each skeleton point corresponds to two boundary segments

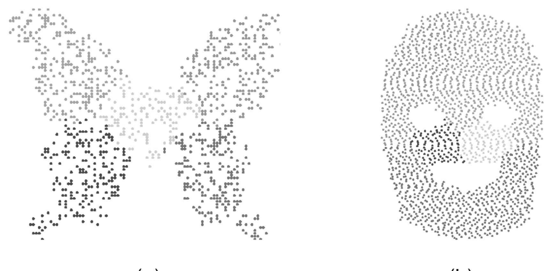


Fig. 9. Segmentation results on the networks in Figs. 1 and 8.

marked with different colors in Fig. 4b. Let each skeleton point initiate a flooding of message which contains the boundary segments' IDs it corresponds to. Each node forwards the message when it receives the first message; otherwise, the message is discarded. As such, each node in the sensor network is aware of the boundary segments it corresponds to. That is, segmentation is obtained, as shown in Fig. 9. This skeleton-assisted segmentation algorithm, as opposed to the previous work [25] which requires classifying the medial axis nodes and merging parts of them into a sink cluster such that each node finally corresponds to a sink cluster, is quite simple and efficient. Note that the joint point marked with thick black circles in Fig. 4 corresponds to at least three boundary segments. In this case, we label every pair of boundary segments a unique ID and then assign the joint points corresponding to the pair of segments with the smallest ID.

6 Simulations

We have implemented the simulator, conducted a series of simulations on various simulated communication graph topologies, and compared with the approach used in [3], [4]. Due to the space limit, we only present some representative results. As mentioned before, the major difference between our algorithm and previous work is that, instead of giving a threshold like [3], [4] to disregard those nodes whose nearest boundary nodes are too close, we remove all unstable nodes whose nearest boundary nodes all lie on the same boundary segment. The effectiveness of the boundary partition will be illustrated in this section.

Fig. 10 shows results on a network with many small deformations on the boundary. This network contains 1,714 sensor nodes and the average degree is 5.68. To identify medial axis nodes using the definition in [3], [4], we set the threshold to be four hops, in order to disregard unstable nodes whose two nearest boundary nodes are too close. Fig. 10a depicts the identified medial axis nodes.

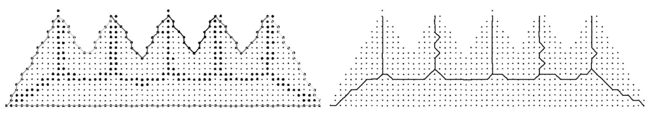


Fig. 11. The skeleton of a sensor network with the serration-like shape. (a) Skeleton points and segments. (b) Skeleton graph.

Obviously, due to deformations on the boundary, many unstable nodes are regarded but have no contribution for skeleton extraction. That is, the deformations on the boundary incur significant noise on medial axis nodes. Besides, the component composed of all medial axis nodes is not connected, and thus, connecting these nodes becomes challenging. Fig. 10b shows the identified skeleton points computed by CASE, where $\delta_\rho = 0.5$. These skeleton points are connected and exhibit the significant pattern of the shape as we identify them via boundary segments (one segment is marked with red line and the other is marked with green line). The skeleton graph is accordingly well extracted as shown in Fig. 10c.

We further study the case where the deformations are large enough that they should be considered as an important part of the shape. We see a serration-like shape in Fig. 11. Recall that, in Section 4.1, to find the corner points, we define the discrete curvature with a system input h. Here, we set h to 3 or 4 in our implementations. That is, when a deformation spans over an area beyond h hops, it is assumably important for the pattern recognition.

Fig. 12 shows the comparison on a small-scale scenario based on JFK Airport Terminal 5 map, where 561 sensor nodes are deployed with a grid model and with small perturbation. The average degree of the communication graph is 6.66. Due to the small scale, the threshold is set to be two hops while using MAP [3], [4]. Fig. 12a illustrates that MAP only identifies a few nodes such that it is hard to connect them. The medial axis graph is shown in Fig. 12b. It is noted that, when a minimum length path is directly used to connect two nodes, the geometric features of the environment are hardly represented (see the middle part of the graph). Besides, at some parts (see the right-bottom part of the graph), no medial axis is extracted at all. However, CASE generates a better skeleton graph in Fig. 12d since we appropriately identify enough skeleton nodes shown in Fig. 12c.

We then conducted simulations on additional network topologies from [18] to further demonstrate CASE's effectiveness. Figs. 13a and 13b show two skeleton graphs using CASE on two networks each of which has more that 2,000 sensor nodes. It is observed that the performance of

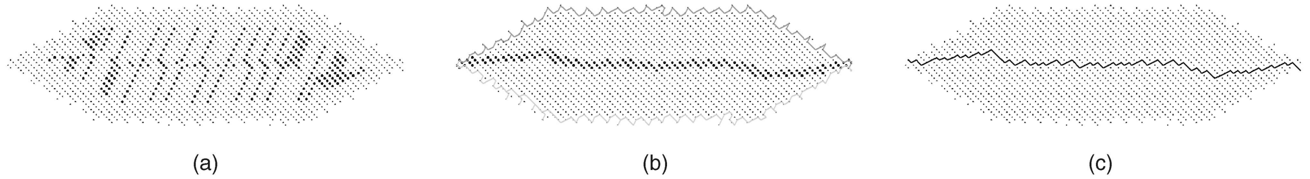


Fig. 10. Comparison on an example network with many small deformations on the boundary. (a) Medial nodes by the method in [4]. (b) Skeleton points and segments by our algorithm. (c) Skeleton graph by our algorithm.

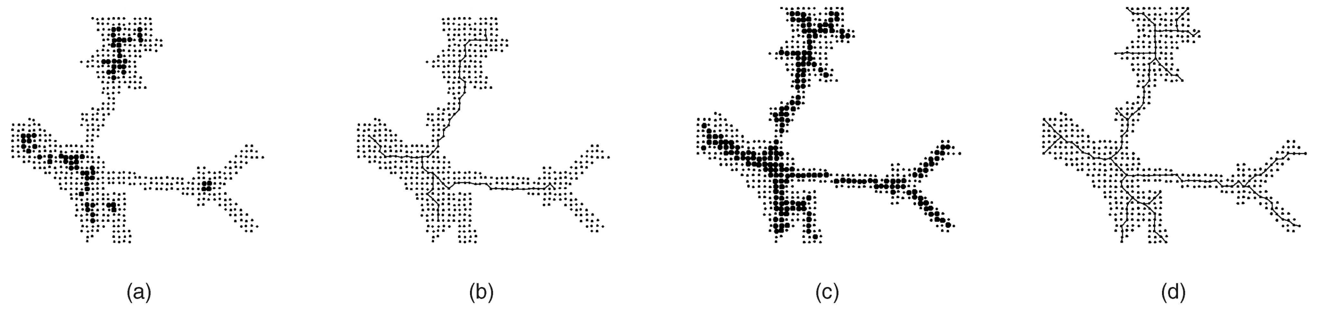


Fig. 12. Comparison on the scenario of JFK airport terminal 5. (a) Medial axis nodes computed by MAP [4]. (b) Medial axis computed by MAP [4]. (c) Skeleton points computed by CASE. (d) Skeleton graph computed by CASE.

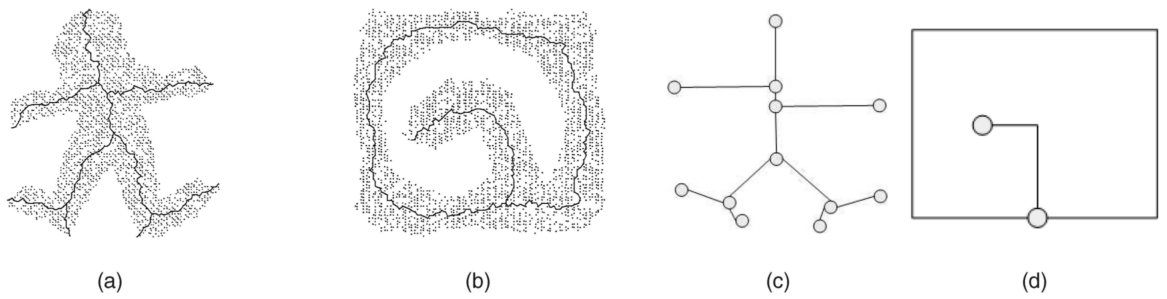


Fig. 13. (a) and (b) Skeletons on additional two networks in [18]. (c) and (d) Simplified skeleton graphs stored in each node.

our algorithm is fairly stable for these two kinds of shapes. Similar to [3], [4], Figs. 13c and 13d depict the simplified skeletons stored at each sensor for applications and the vertex consists of joint points and corner points. Here, we highlight the difference from [3], [4] since besides joint points, we involve the corner points into the vertex in the simplified skeleton stored at each sensor. We observed that, by doing so, the extracted skeleton graph is better than that using MAP [4]. These results imply that both the corner points and joint points are crucial for topology recognition.

We next study the impact of the network density on the skeleton construction. We conducted experiments on butterfly network in Fig. 1. Fig. 14 shows the skeleton when the numbers of nodes are 250 and 500; the average degrees are 5.65 and 12.64. Compared to the skeleton built out of 1,025 nodes in Fig. 1, we can clearly see that higher node density often results in a better result. The reason is that the number of hops in the shortest path in the communication graph may not accurately approximate the distance of two sensor nodes [4]. In addition, when the density of nodes is very low, it is not easy to find a smooth path like Fig. 1 and there are often long branches in the skeleton.

We next consider the performance of CASE when the sensor nodes are nonuniformly distributed. In Fig. 15, a total of 500 nodes are deployed. Half of all nodes are deployed in the bottom-right region of the butterfly-like network, and the other half of nodes are deployed in the rest of the area. Similar to Fig. 14, we found that the skeleton result in low-density network looks different while CASE maintains stable performance on the construction of skeleton.

We also compared the communication cost of the algorithms. Table 1 shows the total communication cost in terms of the total number of transmitted messages for skeleton extraction algorithms. While providing much better skeleton results than MAP in most scenarios, CASE only introduces a little higher communication cost (around two to five times compared with MAP despite the network scale). It is noted that in Butterfly map, the communication cost using CASE is around five times than that using MAP. The reason is that, in this case, the number of boundary nodes is considerable and the average degree is high. Thus, the total traffic incurred by the identification of corner points using CASE is significant compared to others. Besides, noted that in the map of Hexagon, the communication cost using CASE is close to that using MAP as MAP

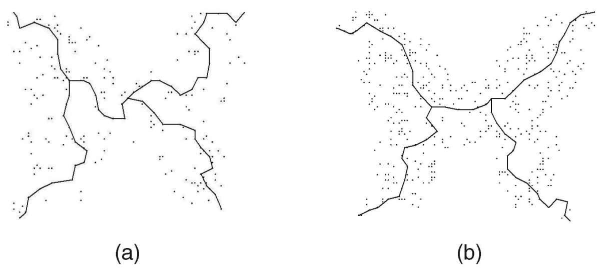


Fig. 14. The skeletons of the butterfly network with different node densities. (a) 250 nodes. (b) 500 nodes.

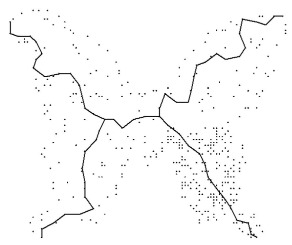


Fig. 15. Skeleton result in the presence of nonuniform distribution of nodes.

TABLE 1
Performance of **CASE** on Sensor Networks in Different Scenarios

	Size		Number of nodes			Total no. of message	
Network	No. of nodes	Avg degree	Boundary	Skeleton	Medial axis [4]	CASE	MAP [4]
Butterfly(see Fig. 1)	1,025	20.9	126	280	271	10,016	2,100
Face(see Fig. 8)	1,967	6.27	346	234	300	6,512	2,964
Hexagon(see Fig. 10)	1,714	5.68	306	138	408	5,009	3,289
Serration(see Fig. 11)	791	7.31	155	115	65	3,408	1,436
JFK(see Fig. 12)	561	6.66	220	191	69	3,931	1,144
Man(see Fig. 13(a))	2,179	5.63	414	477	476	7,252	3,983
Spiral(see Fig. 13(b))	2,812	5.15	885	398	612	10,929	4,856

identifies much more medial axis nodes due to the noise on the boundary. Overall, CASE can also be viewed as a lightweight solution for skeleton extraction.

Finally, we conduct experiments under a different network model: the Quasi-UDG model [4] to the butterfly network. We set $\alpha = 0.8$; that is, a link exists between two nodes if the distance between the two nodes is smaller than 0.2 times of radio range; a link does not exist between two nodes when the distance between two nodes is greater than 1.8 times the radio range, and a link between two nodes exists with a probability of $p = 0.3$ if the distance of the two nodes is between 0.2 and 1.8 times of radio range. The skeleton generated by CASE is shown in Fig. 16. We can see that though CASE has a different result from that in Fig. 1d, the skeleton still captures the network topology well.

7 Conclusion

We have presented CASE: A novel distributed and scalable algorithm for skeleton extraction in sensor networks. It is the first study with the focus on skeleton extraction algorithm in sensor networks. This algorithm requests the connectivity information only and it does not make such assumptions as in advance knowledge of location information and distance information, which might not be realistic in practice. We first identify the skeleton points by partitioning the boundary. These skeleton points are then connected to form the skeleton arcs, each of which corresponds to two boundary segments. We connect all the skeleton arcs and conduct a refinement to achieve the final skeleton. We have demonstrated that CASE provides more accurate skeleton results, and is robust against boundary variations. It outperforms a state-of-the-art algorithm under various configurations. We prove the proposed algorithm is scalable since its traffic complexity is linearly proportional to the network size. The experimental results also demonstrate its scalability as it only causes a little higher traffic than previous algorithm while providing more accurate skeleton results.

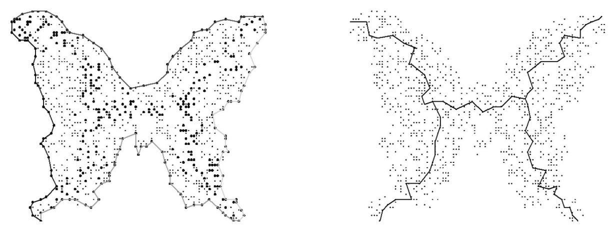

(a) (b)

Fig. 16. CASE on the Quasi-UDG in the butterfly network. (a) Skeleton points and segments. (b) Skeleton graph.

We are interested in several directions in the future. First, we seek more efficient algorithms to reduce the traffic overhead of our extraction technique. In particular, an approximation of skeleton graph could be applicable. Second, we would like to evaluate other boundary detection techniques for our skeleton extraction algorithm. We believe improvement is possible when other approaches are adopted. Third, evaluation of CASE on larger scale networks will be carried out. Fourth, we find that the CASE algorithm does not work well with the network (e.g., a unit disk-like area where the sensor nodes are uniformly distributed) where there could be no corner point that can be identified. Another direction is to further study how to deal with this kind of severe cases.

Acknowledgments

This work was supported in part through the Chinese National 863 project (No. 2007AA01Z223) and the National Natural Science Foundation of China (Nos. 60572063, 60803115, and 60873127). One of the authors (Dan Wang)'s work is supported by HKPU/ICRG G-YG78, A-PB0R, and RGC/GRF PolyU 5305/08E. The authors would like to thank Yue Wang for her help in the implementation of boundary detection [24] as our system inputs, and the anonymous referees for their valuable comments for an earlier version of this work which appeared as [15].

References

[1] X. Bai, L.J. Latecki, and W. Liu, "Skeleton Pruning by Contour Partitioning with Discrete Curve Evolution," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 3, pp. 449-462, Mar. 2007.
[2] H. Blum, "Biological Shape and Visual Science (Part i)," *J. Theoretical Biology,* vol. 38, pp. 205-287, 1973.
[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. 609-616, 2007.
[5] C. Buragohain, D. Agrawal, and S. Suri, "Distributed Navigation Algorithms for Sensor Networks," *Proc. IEEE INFOCOM,* 2006.

[6] E. Cayirci and T. Coplu, “Sendrom: Sensor Networks for Disaster Relief Operations Management,” *Wireless Networks*, vol. 13, no. 3, pp. 409-423, 2007.

[7] H.I. Choi, S.W. Choi, and H.P. Moon, “Mathematcal Theory of Medial Axis Transform,” *Pacific J. Math*, vol. 181, no. 1, pp. 57-88, 1997.

[8] W.-P. Choi, K.-M. Lam, and W.-C. Siu, “Extraction of the Euclidean Skeleton Based on a Connectivity Criterion,” *Pattern Recognition*, vol. 36, pp. 721-729, 2003.

[9] 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.

[10] Q. Fang, J. Gao, and L. Guibas, “Locating and Bypassing Routing Holes in Sensor Networks,” *Proc. Mobile Networks and Applications*, 2006.

[11] S.P. Fekete, A. Kroller, D. Pfisterer, S. Fischer, and C. Buschmann, “Locating and Bypassing Routing Holes in Sensor Networks,” *Proc. Int’l Workshop Algorithmic Aspects of Wireless Sensor Networks*, 2004.

[12] 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.

[13] R. Ghrist and A. Muhammad, “Coverage and Hole-Detection in Sensor Networks via Homology,” *Proc. IEEE Information Processing in Sensor Networks (IPSN)*, 2004.

[14] P. Golland and E. Grimson, “Fixed Topology Skeletons,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 10-17, 2000.

[15] H. Jiang, W. Liu, D. Wang, T. Chen, X. Bai, X. Liu, Y. Wu, and W. Liu, “CASE: Connectivity-Based Skeleton Extraction in Wireless Sensor Networks,” *Proc. IEEE INFOCOM*, pp. 2916-2920, Apr. 2009.

[16] L.J. Latecki and R. Lakamper, “Shape Similarity Measure Based on Correspondence of Visual Parts,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1185-1190, Oct. 2000.

[17] E.L. Lawer, *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, 1976.

[18] S. Lederer, Y. Wang, and J. Gao, “Connectivity-Based Localization of Large Scale Sensor Networks with Complex Shape,” *Proc. IEEE INFOCOM*, 2008.

[19] L. Lin and H. Lee, “A Dynamic Medial Axis Model for Sensor Networks,” *Proc. IEEE Int’l Conf. Embedded and Real-Time Computing Systems and Applications*, pp. 146-156, Aug. 2007.

[20] H. Rom and G. Medioni, “Hierarchical Decomposition and Axial Shape Description,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 973-981, Oct. 1993.

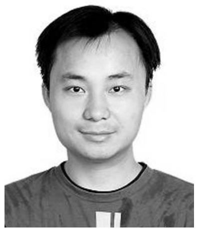
[21] O. Saukh, R. Sauter, M. Gauger, P.J. Marron, and K. Rothernel, “On Boundary Recognition without Location Information in Wireless Sensor Networks,” *Proc. Information Processing in Sensor Networks (IPSN)*, 2008.

[22] S. Schaefer and C. Yuksel, “Example-Based Skeleton Extraction,” *Proc. Eurographics Symp. Geometry Processing*, 2007.

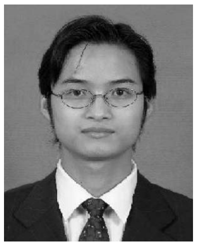
[23] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, “Habitat Monitoring with Sensor Networks,” *Comm. ACM*, vol. 47, no. 6, pp. 34-40, 2004.

[24] Y. Wang, J. Gao, and J.S. Mitchell, “Boundary Recognition in Sensor Networks by Topological Methods,” *Proc. ACM MobiCom*, Sept. 2006.

[25] X. Zhu, R. Sarkar, and J. Gao, “Shape Segmentation and Applications in Sensor Networks,” *Proc. IEEE INFOCOM*, pp. 1838-1846, May 2007.



Hongbo Jiang received the BS and MS degrees from Huazhong University of Science and Technology, China, and the PhD degree from Case Western Reserve University in 2008. He then 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.



Wenping Liu is currently working part-time toward the PhD degree at MC Lab, Department of Electronics and Information Engineering, Huazhong University of Science and Technology, supervised by Wenyu Liu. Meanwhile, he is a faculty member of the Department of Statistics and Applied Mathematics, Hubei University of Economics. His research interests include statistical modeling and wireless sensor networks.

Dan Wang (S’05-M’07) received the BSc degree from Peking University, Beijing, China, in 2000, the MSc degree from Case Western Reserve University, Cleveland, Ohio, in 2004, and the PhD degree from Simon Fraser University, Burnaby, British Columbia, Canada, in 2007; all in computer science. He is currently an assistant professor at the Department of Computing, The Hong Kong Polytechnic University. His research interests include wireless sensor networks, Internet routing, and peer-to-peer networks. He is a member of the IEEE.

Chen Tian received the BS and MS degrees from the Department of Electronics and Information Engineering at the Huazhong University of Science and Technology, China, in 2000 and 2003, respectively. He is working toward the PhD degree in the Department of Electronics and Information Engineering at the Huazhong University of Science and Technology, China. His research interests include distributed networks, wireless networks, and network architecture. He is a member of the IEEE.

Xiang Bai received the BS and MS degrees in electronics and information engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2003 and 2005, respectively. He is currently working toward the PhD degree at HUST and has recently joined the University of California, Los Angeles, as a joint PhD student. From January 2006 to May 2007, he was with the Department of Computer Science and Information, Temple University. His research interests include computer graphics, computer vision, and pattern recognition.

Xue Liu received the BS degree in applied mathematics and the MEng degree in control theory and applications from Tsinghua University and the PhD degree in computer science from the University of Illinois, Urbana-Champaign, in 2006. He is currently an assistant professor in the School of Computer Science, McGill University. He was briefly with the Hewlett-Packard Laboratories and IBM T.J. Watson Research Center. His research interests include real-time and embedded computing, performance and power management of server systems, sensor networks, fault tolerance, and control. He is the author/coauthor of more than 20 refereed publications in leading conferences and journals in these fields. He is a member of the IEEE.

Ying Wu received the BS degree from Huazhong University of Science and Technology, Wuhan, China, the MS degree from Tsinghua University, Beijing, China, and the PhD degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign (UIUC), in 1994, 1997, and 2001, respectively. From 1997 to 2001, he was a research assistant at the Beckman Institute for Advanced Science and Technology at UIUC. During summer 1999 and 2000, he was a research intern with Microsoft Research, Redmond, Washington. In 2001, he joined the Department of Electrical and Computer Engineering at Northwestern University, Evanston, Illinois, as an assistant professor. He is currently an associate professor of electrical engineering and computer science at Northwestern University. His current research interests include computer vision, image and video analysis, pattern recognition, machine learning, multimedia data mining, and human-computer interaction. He serves as an associate editor for the *IEEE Transactions on Image Processing*, the *SPIE Journal of Electronic Imaging*, and the *IAPR Journal of Machine Vision and Applications*. He received the Robert T. Chien Award at UIUC in 2001, and the National Science Foundation CAREER Award in 2003. He is a senior member of the IEEE.

Wenyu Liu received the BS degree in computer science from Tsinghua University, Beijing, China, in 1986, and the diploma and doctoral degrees, both in electronics and information engineering, from Huazhong University of Science and Technology (HUST), Wuhan, China, in 1991 and 2001, respectively. He is now a professor and associate chairman of the Department of Electronics and Information Engineering, HUST. His current research interests include computer graphics, multimedia information processing, and computer vision. He is a member of the IEEE.