Connectivity-Based and Anchor-Free Localization in Large-Scale 2D/3D Sensor Networks

GUANG TAN, SIAT, Chinese Academy of Sciences
HONGBO JIANG, SHENGKAI ZHANG, and ZHIMENG YIN, Huazhong University of Science and Technology
ANNE-MARIE KERMARREC, INRIA/IRISA

A connectivity-based and anchor-free three-dimensional localization (CATL) scheme is presented for large-scale sensor networks with concave regions. It distinguishes itself from previous work with a combination of three features: (1) it works for networks in both 2D and 3D spaces, possibly containing holes or concave regions; (2) it is anchor-free and uses only connectivity information to faithfully recover the original network topology, up to scaling and rotation; (3) it does not depend on the knowledge of network boundaries, which suits it well to situations where boundaries are difficult to identify. The key idea of CATL is to discover the *notch nodes*, where shortest paths bend and hop-count-based distance starts to significantly deviate from the true Euclidean distance. An iterative protocol is developed that uses a *notch-avoiding* multilateration mechanism to localize the network. Simulations show that CATL achieves accurate localization results with a moderate per-node message cost.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Sensor networks, algorithm/protocol design, 3D localization

ACM Reference Format:
Tan, G., Jiang, H., Zhang, S., Yin, Z., and Kermarrec, A.-M. 2013. Connectivity-based and anchor-free localization in large-scale 2D/3D sensor networks. ACM Trans. Sensor Netw. 10, 1, Article 6 (November 2013), 21 pages.
DOI: http://dx.doi.org/10.1145/2529976

6

G. Tan was supported by the NSFC under Grant 61103243, the Youth Innovation Promotion Association, the Chinese Academy of Sciences, the Ministry of Science and Technology 863 Key Project No. 2011AA010500, and Shenzhen Overseas High-level Talents Innovation and Entrepreneurship Funds KQC201109050097A. H. Jiang was supported in part by the National Natural Science Foundation of China under Grant 61073147, Grant 61173120, Grant 61103243, Grant 61202460, Grant 61271226, and Grant 61272410; by the National Natural Science Foundation of China and Microsoft Research Asia under Grant 60933012; by the Fundamental Research Funds for the Central Universities under Grant 2012QN078; by the CHUTIAN Scholar Project of Hubei Province; by the Youth Chenguang Project of Wuhan City under Grant 201050231080; 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).
Authors' addresses: G. Tan, SIAT, Chinese Academy of Sciences, China; email: guang.tan@siat.ac.cn; H. Jiang (corresponding author), S. Zhang, and Z. Yin, Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China 430074; email: {hongbojiang2004, shengkai.zhang, yinzhimeng}@gmail.com; A. Kermarrec, INRIA/IRISA, Rennes, France; email: annemarie.kermarrec@irisa.fr.

DOI: http://dx.doi.org/10.1145/2529976

1. INTRODUCTION

In many wireless sensor networks applications, it is essential for the sensor nodes to automatically discover their locations. Assuming a certain number of anchor nodes whose locations are known in advance, nodes can localize themselves through measuring physical distance to those anchors and between themselves using special hardware. Recently, there has been growing interest in localization schemes that use only connectivity information [Niculescu and Nath 2003; Li and Liu 2010; Lederer et al. 2008; Wang et al. 2009], without reliance on any ranging techniques. To further reduce cost, a small number (ideally a constant number) of anchors are often assumed.

An implicit assumption underlying the connectivity-based and constant-anchor (CBCA) approach is that in practice, sensor nodes are often placed with a certain degree of uniformity (e.g., in a grid-guided distribution [Li and Liu 2010; Bruck et al. 2005; Lederer et al. 2008; Wang et al. 2009]), so that the Euclidean distance between two nodes can be approximated by their minimum hop count (multiplied by average physical hop distance). While this is often the case in regularly shaped fields, this assumption may be seriously violated in *anisotropic* networks with concave regions, where a shortest path may significantly bend, resulting in the hop-count-based estimate widely deviating from the true Euclidean distance. This leads to large localization errors, as reported in previous studies (e.g., [Li and Liu 2010]). Efforts have been made to address this problem, and several algorithms [Shang and Ruml 2004; Lederer et al. 2008; Wang et al. 2009] now exist that localize topologically complex networks with good accuracy.

Despite the success on 2D cases, so far there has been no CBCA design for localizing 3D anisotropic sensor networks. Previous solutions mostly assume, either explicitly or implicitly, a 2D environment, and exploit the network's 2D geometric properties. Such properties often take a very different form or are more difficult to utilize in 3D spaces. For example, merging partially localized network pieces to get a global network layout is the basic idea of some schemes [Shang and Ruml 2004; Lederer et al. 2008; Wang et al. 2009]. It is generally more difficult to merge polyhedra nearby than to merge polygons on the plane, especially when *flip ambiguities* [Lederer et al. 2008; Wang et al. 2009] need to be dealt with and when those components do not fit together due to errors.

This article presents a localization scheme that works for 3D networks with concave regions, using connectivity information only. The scheme, named *CATL*, assumes that sensors are placed in a 2D/3D space with relatively even node distribution (e.g., a perturbed grid or uniform random one), possibly with holes or concave regions. Nodes nearby can directly communicate with each other according to certain radio patterns, but have no means of measuring their physical distances—the only information available to a node is its immediate neighbor list. Under such assumptions, CATL is able to produce a *relative* coordinate system that accurately recovers the original network topology, up to scaling and rotation. The relative coordinate system can be translated to an *absolute* system provided the physical locations of a few anchor nodes.

The key idea of CATL is to identify the *notch nodes* (or *notches* for short), where shortest paths bend and deviate from their straight-line course (see Figure 1). Usually located near concave network corners, such nodes are critical for internode distance measurement, yet their detection is by no means trivial. Here, it may be tempting to consider nodes with an unusual neighborhood size, since notch nodes tend to have a smaller neighborhood than interior nodes, and a larger one than average boundary nodes. Such an approach, however, suffers from the difficulty in selecting a proper size gap for distinguishing nodes: it either generates a lot of false positives due to noise or fails to capture notches at narrow concave corners (see node a in Figure 1), whose neighborhood size is very close to that of average interior nodes. Ignoring these notches

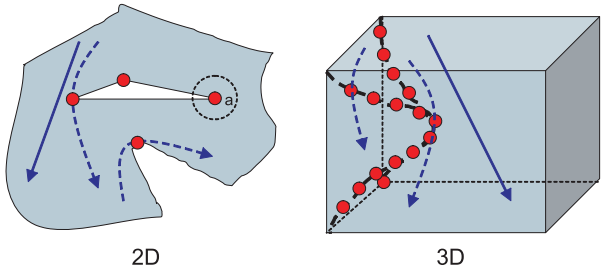


Fig. 1. Notch nodes (red circles) and shortest paths. Paths bending around those notches (dashed lines) are not used for estimating Euclidean distance, while notch-free paths (solid lines) are straight so the Euclidean distance between their endpoints can be approximated with minimum hop count.

can lead to great errors in path length estimate, as the shortest path bend around them most seriously. An additional problem is that, in order to differentiate notches from ordinary boundary nodes, this method needs to be aware of network boundaries in advance, whose determination is nontrivial by itself, especially in a 3D environment [Zhou et al. 2010, 2012; Li et al. 2011].

Our method of detecting notch nodes is based on a new observation: that given a global shortest path (SP) tree, notches often have a much fatter subtree than that of other nodes. By detecting the abnormality in a node's SP subtree size, we are then able to identify notches. Looking beyond local neighborhoods, this method is more robust to local nonuniformity of node distribution than the neighborhood-size-based approach. On the other hand, it is sensitive to concavity in network shape: the sharper the network corners, the more pronounced the fat subtree effect, thus allowing a more reliable way to find notches. With the knowledge of notches, we can avoid using paths passing through them, so two nodes' Euclidean distance can be safely estimated with their minimum hop count. On top of this, we use a simple multilateration technique to localize the nodes.

An important feature of CATL is that it does not rely on the knowledge of network boundaries. With its functional independence and localization capability, CATL readily lends itself to a number of other fundamental tasks in 2D/3D networks, including boundary detection, routing, network segmentation, etc.

The next section introduces related work; an overview of CATL is given in Section 3; Sections 4 through 6 describe the CATL scheme in detail; Section 7 analyzes the message cost of CATL; Section 8 presents simulation results, Section 9 discusses the limitations and potential applications of CATL, and Section 10 concludes.

2. RELATED WORK

Localization has been an extensively studied topic in sensor networks research. A major current trend on this subject is to reduce network configuration cost to a minimum (e.g., no ranging, no anchors, no boundary information), while retaining maximum applicability (e.g., anisotropic networks) and scalability (e.g., thousands of nodes). In the following, we briefly review the work in the background of connectivity-based localization.

The centroid-based approach by Bulusu et al. [2000] is among the earliest work of connectivity-based localization. A node estimates its location simply by calculating the centroid of audible anchors' locations. In DV-hop by Niculescu and Nath [2003], a node measures its hop distances to at least three anchors and then obtains its location by triangulation.

A well-known approach to connectivity-based and anchor-free localization is based on Multidimensional Scaling (MDS) techniques [Shang et al. 2003]. Taking an internode

Table I. Comparison of Connectivity-Based Localization Schemes

Scheme	2D/3D	hole adaptive ?	boundary info. ?	#anchors	message cost
DV-hop [Niculescu and Nath 2003]	2D, 3D	No	No	$O(1)$	$O(n^2)$
MDS [Shang et al. 2003]	2D, 3D	No	No	$O(1)$	$O(n^3)$
PDM [Lim and Hou 2005]	2D, 3D	Yes	No	$O(n)$	$O(n^2)$
REP [Li and Liu 2010]	2D	Yes	Yes	$O(1)$	$O(n^2)$
LWG08 [Lederer et al. 2008]	2D	Yes	Yes	$O(1)$	$O(n)$
WLG09 [Wang et al. 2009]	2D	Yes	No	$O(1)$	N/A
CATL	2D, 3D	Yes	No	$O(1)$	$O(n \cdot n_e)$

Note: n is the number of nodes. n_e refers to the number of edge nodes detected by CATL. "N/A" indicates that the literature has not addressed the relevant issue.

hop distance matrix as input, this method generates a set of relative coordinates for each nodes in a centralized way. Like DV-hop, it performs poorly in the presence of holes [Lederer et al. 2008]. In their subsequent work, Shang and Ruml [2004], address the hole problem by creating a local map at each node and then merging them to form a global map. Their design only considers 2D cases, and the networks examined are relatively small with simple layouts.

Li and Liu [2010] propose a method that utilizes the geometric characteristics of hole corners to rectify the distorted length estimate of a path passing by a hole. The idea is to create a virtual hole around a hole corner which induces a new shortest path. The difference between the original and new shortest paths then contains geometric information for inferring the real Euclidean distance between two nodes. This method depends on a full boundary detection algorithm and is limited to 2D networks. Lederer et al. [2008] develop a novel localization method with an emphasis on graph rigidity and flip ambiguities avoidance. Their design depends on the knowledge of network boundaries as well. Wang et al. [2009] further propose an incremental Delaunay refinement method that gets rid of the boundary assumption. The new design is shown to be more robust to different network environments. Jin et al. [2011] propose an interesting solution for connectivity-based localization using a Ricci flow method. This solution is shown to be highly adaptive to various factors, such as irregularities in network layout, in node distribution, and in radio model. Unfortunately, all these approaches are limited to 2D cases, due to their inherent reliance on the 2D properties of network graphs.

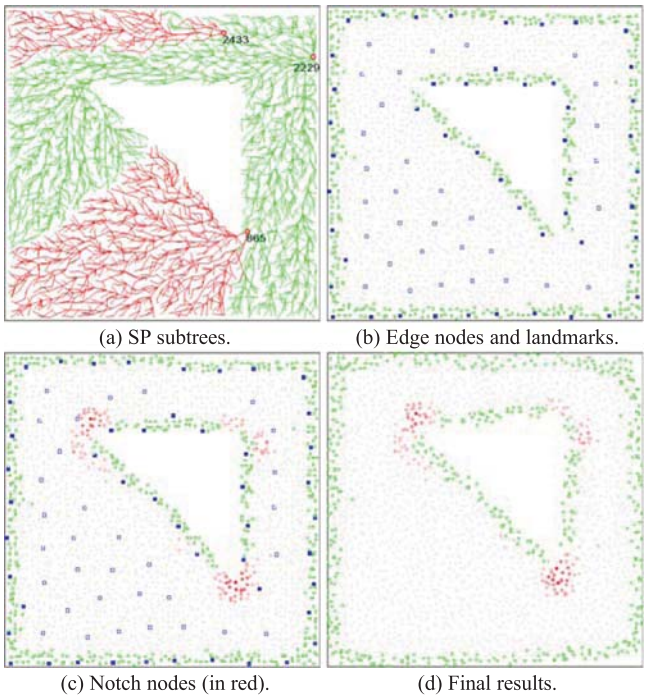
3D localization has been investigated in the area of underwater sensor networks [Cheng et al. 2008; Zhou et al. 2007]. The previous schemes depend on various assumptions about the network environments, such as dense anchor deployment, the availability of depth information, a relatively small network size, etc. None of these schemes can work with connectivity information alone and with a constant number of anchors.

The notion of notch nodes has been also considered [Tan et al. 2009], where the notch nodes (or concave nodes) are used for convex partition of the network. In Tan et al. [2009], the detection of notch nodes assumes the knowledge of node positions and of network boundaries. Our CATL does not rely on these assumptions.

In short, previous solutions either do not work for networks with holes, work for only 2D cases, or rely on network boundary information. CATL is not subject to these restrictions, thus applies to a wider range of scenarios. Table I compares CATL with state-of-the-art connectivity-based localization schemes in several important aspects.

3. AN OVERVIEW OF CATL

In this section, we give an overview of CATL. For ease of visualization, we often use 2D examples for illustration purposes, although the design applies to 3D just as well.



(a) SP subtrees. (b) Edge nodes and landmarks.

(c) Notch nodes (in red). (d) Final results.

Fig. 2. CATL run on a network of 2,645 nodes. (a) A global SP tree rooted at node 2229 and two SP subtrees at nodes 865 and 2433. (b) Edge nodes (green squares) and edge landmarks (blue solid squares). (c) Notch nodes. Darker color means higher notch degree. (d) Localized network.

At the heart of CATL is a method to identify the notch nodes. Suppose a node p floods a message, creating a global *shortest path (SP) tree* $\mathcal{T}_p$. This tree also induces for every other node q a *shortest path (SP) subtree* (or subtree for short). Suppose q is at the lth level of $\mathcal{T}_p$. If we limit q's subtree to a depth of at most h hops, then such a subtree is called q's h-hop subtree, with its size denoted $S(l, h)$.

If the network does not have any boundaries or contain any holes, then we would expect that every node at the same level l of $\mathcal{T}_p$ has approximately the same $S(l, h)$, for any $h > 0$, due to the equal chance of the message propagating in all directions. This uniformity, however, will be broken in a real network because of the disruption from holes or boundaries. In particular, nodes near the concave corners tend to have an unusually fat subtree up to a certain depth. For example, Figure 2(a) shows a global SP tree rooted at the node 2229, and two subtrees rooted at the nodes 865 and 2433. One can see that the node 865 has a much fatter subtree than that of node 2433. By standing in the way of the flooded message to many nodes behind the triangular hole, node 865 has the chance to see more shortest-path flows passing by than it can see in a non-hole environment. In other words, many data flows that would otherwise have followed a nearly straight-line way are now diverted and are forced to go via node 865, contributing to a big subtree under it.

This insight forms the basis for our notch detection method. When a message is flooded, we let each node calculate its SP subtree sizes for a range of depths by collecting reports from descendants. A node then compares these sizes with some standard. If a node discovers a subtree that is larger than the standard by a certain threshold, then its *notch degree* is incremented. We select a small subset of the nodes (see solid squares in Figure 2(b)) to flood messages, each resulting a number of nodes incrementing

Table II. Notation and Definitions

Notation	Definition
n	number of nodes in the network
n_e	number of edge nodes in the network
$\overline{d}$	average interior degree of the network
$\mathcal{T}_p$	a global shortest path (SP) tree rooted at node p
N_{elms}	number of edge landmarks in the network
$Q(\mathcal{P})$	quality of path $\mathcal{P}$
$C(p)$	confidence of a multilaterated result p
`LandmarkSpacing`	number of hops between two landmarks
`NotchThreshold`	a fraction of N_{elms}, used for notch determination

their notch degrees. After this process, a set of nodes may find themselves assigned a positive notch degree. Nodes whose notch degrees are higher than a threshold (for avoiding noise) are taken as the final notch nodes; see the red nodes in Figure 2(c).

The localization of nodes is done with an iterative notch-avoiding multilateraion protocol. Multilateraion computes coordinates for a node using distance measurements to at least four (or three in 2D cases) reference points. The protocol is bootstrapped by four (or three in 2D cases) nodes, termed the *seed* nodes, that are identified and assigned coordinates in a preparatory stage of the protocol. The protocol runs in rounds; in each round a number of location-unknown nodes are assigned coordinates by means of hop distance measurements to already localized nodes. When measuring distances, shortest paths containing notch nodes will be ignored or used with a low priority. Figure 2(d) shows the localized result of the network in Figure 2(a).

In the following, we describe the three mains steps of CATL.

(1) Establishing auxiliary network structures.
(2) Detecting notch nodes.
(3) Notch-avoiding multilaterion-based localization.

We assume that there is a *localization coordinator* (or simply *coordinator*) that controls the localization process by flooding a command at the beginning of each step. The coordinator can be an arbitrary node in the network or a base station. It decides the beginning of a step by checking network status or using a (conservative) estimate of the time needed to complete a step and is not involved in localization otherwise. The description of the CATL algorithm will also use a number of notations and symbols, some of which are listed in Table II for ease of reference.

Before presenting the details, we show several examples of CATL to help with our intuition. The networks to be localized are taken from Wang et al. [2009], where they propose a connectivity-based scheme, hereinafter called *WLG09*. WLG09 is shown to perform very well for complex networks, though its heavy reliance on several 2D geometric structures restricts it to 2D networks. Figure 3 shows the results of CATL and WLG09 for four networks with complex shapes. It is helpful to see the notch nodes (in red) detected by CATL and how those nodes coincide with concave corners.

4. AUXILIARY NETWORK STRUCTURES

Before localization begins, two auxiliary structures, *network edges* and *landmark network*, are established. The purpose of these structures is to select a set of flooding nodes that are to play a central role in detecting notches and localization. We want these flooding nodes to help discover all notches and help all other nodes calculate correct locations, while keeping their total number to a minimum for lowest message cost.

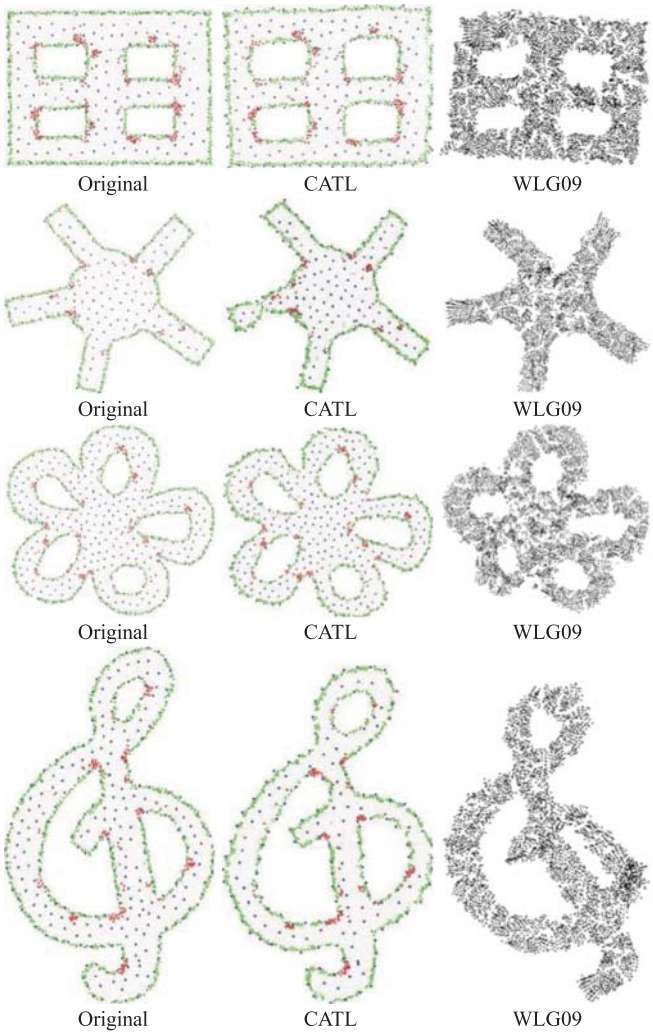


Fig. 3. Results of CATL and WLG09 [Wang et al. 2009] for 2D networks. Columns (from left to right): original networks, CATL's results, WLG09's results. Top row: window shape, 6,495 nodes, average node degree 12.41; second row: sun shape, 5,217 nodes, average node degree 12.37; third row: flower shape, 8,350 nodes, average node degree 9.15; and fourth row: music shape, 6,176 nodes, average node degree 9.16.

The intuition behind these structures is that nodes near the network periphery are more representative of the network layout than interior nodes. Imagine that we need to get a full picture of a sensor field—or more precisely, the corners of such a field—by taking photos at chosen points in the field. Ignoring focus issues, we would be able to get a good approximation if we do so only at the field's periphery and at a certain interval. Inspired by this, we select a set of evenly spaced nodes near the network periphery to do flooding; from there the nodes are able to "see" all the major corners present in the network.

Network Edges. The network edges represent a set of nodes on or near the periphery of the network. Through local communications, every node obtains its two-hop *degree* (one hop is often too susceptible to noise), defined as the number of neighbors within

two-hop distance. This information is exchanged between nodes in a gossip manner so that every node is able to collect a roughly uniform random sample of nodes' degrees in the network. From these samples, a node can calculate the average two-hop *interior degree*, corresponding to the degree of nodes found in the interior of the network. To avoid bias toward boundary areas, the calculation is based on a fraction δ (e.g., 1/3) of top degree samples.

Having obtained the average interior degree $\overline{d}$, a node p compares its own two-hop degree d_p with this average value. If $d_p < (1-\epsilon) \times \overline{d}$, where $0 < \epsilon < 1$ is a parameter (in our setting 0.2 for 2D and 0.25 for 3D), then p marks itself as an *edge node*. Figure 2(b) shows the edge nodes (green squares) found using this method.

The edge nodes should not be equated with boundary nodes, as defined in previous works (e.g., [Wang et al. 2006]), though they share the common property of being close to the network periphery. In particular, the edge nodes may not cover all network boundaries, on the other hand, due to the very rough criterion, the edge nodes often contain nodes that are only close to, but not exactly on, the boundaries. As such, detecting edge nodes is a much simpler task than detecting a full and tight boundary. In fact, edge nodes merely help us narrow down the flooding sources to a small set and do not need to be full or tight in any sense—we need them only for efficiency considerations.

Landmark Network. The landmark network consists of a set of nodes uniformly sampled from the original network with density controlled by a small system parameter `LandmarkSpacing` (e.g., 3). Initially all the nodes are non-landmarks. Every node asynchronously floods (at a random point in time during some time window) a message in its `LandmarkSpacing`-hop neighborhood. If there is no already established landmark in such a neighborhood, it marks itself as one and then notifies the neighborhood. If a node hears such a flood before its own action, then it cancels its flooding. A sufficiently long time window is given to this asynchronous process so the sampling is roughly uniform (strict uniformity is not required here).

The landmark sampling process is run twice: first on edge nodes only, determining what we call *edge landmarks*, and then on other nodes. A landmark creates a virtual link, corresponding to a shortest path, to another landmark when their distance is no greater than $2\times$`LandmarkSpacing` hops. Such links connect the landmarks into a *landmark network*.

5. DETECTING NOTCH NODES

In this step, every edge landmark floods the network to detect notch nodes. The coordinator observes the floodings and records the total number of edge landmarks, N_{elms}.

5.1. Detecting Large SP Subtrees

Suppose some edge landmark p is flooding a message. The message carries a hop counter such that every node learns its level number in the tree. If a node for the first time receives a message from a certain flood, it takes the message transmitter as its parent and replies with a confirmation. A node q that does not receive any confirmation after a certain period of time of its forwarding message regards itself as a leaf node and begins to send a report back up the tree. The report contains an array of q's subtree sizes, $S_q(l, 1), S_q(l, 2), \ldots$, and is initially cleared at the leaf node. A node waits to collect such reports from all of its children and calculates its own SP subtree size array.

After a non-root node q obtains its subtree size array, it checks where there exists a tree depth $h \geq 3$ such that

$$S_q(l, h) > k \times h, \tag{1}$$

where k is a parameter. $h \geq 3$ is for avoiding the effect of local error. The reason for such a function and selection of k will be discussed shortly in Section 5.2.

If the node q discovers that Equation (1) is true, it increments its notch degree by one. Afterwards, q behaves as if it were a leaf node when it sends its subtree size array to its parent, that is, q's subtree size array is cleared. This is to prevent some of its ancestors from incorrectly taking themselves as new notch nodes because of a big shared subtree.

Apparently, a node's notch degree is no greater than the number of flooding sources, that is, the number of edge landmarks N_{elms}. To avoid noise, we set an initial threshold $\texttt{NotchThreshold} = \epsilon N_{\text{elms}}$, where $0 < \epsilon < 1$ is a small number (e.g., 0.1), for the nodes. When the notch detection process finishes, all the nodes with notch degree higher than `NotchThreshold` mark themselves as notch nodes.

5.2. Determining Criteria for Notch Nodes

While the large subtree observation captures a distinguishing feature of notch nodes, it seems unclear how to find a criterion for precisely determining whether a node's subtree is indeed "large" in various network topologies. Furthermore, the impact of such a criterion on the final localization quality is quite indirect. This leads us to decide not to pursue an ideal criterion (if it ever exists); instead, we use a very simple linear function, as in Equation (1), while letting the notch threshold change adaptively in response to the localization progress. This makes the number of initially generated notches not critical, which in turn makes the influence of k in Equation (1) insignificant.[1]

Recall that we set an initial notch degree $\texttt{NotchThreshold} = \epsilon N_{\text{elms}}$ for the nodes, where $\epsilon = 0.1$ in our setting. This threshold can actually be made even lower. It allows nodes to easily consider themselves as notches, which, of course, may generate more notches than there actually are. The consequence is that many path measurements that should have been useful are now considered invalid because the paths contain falsely identified notch nodes. This may further cause the localization process to end prematurely, when no new nodes can obtain valid coordinates. We handle this situation by having the coordinator monitor the localization progress (details to be given in Section 6.2.3), and raise `NotchThreshold` gradually. The increased `NotchThreshold` can lead to fewer nodes to mark themselves as notches, thus bringing opportunities for the localization to continue.

The adaptive mechanism saves us the trouble of searching for optimal notch detecting parameters for all possible scenarios case by case. It also guarantees that every node will be able to localize itself eventually, since in the extreme case where $\texttt{NotchThreshold} = N_{\text{elms}}$, no node will remain as notch node, so every unlocalized node can obtain distance measurement to already localized nodes, thus getting the chance to localize itself. (In practice it rarely needs to get to that extreme point.)

6. LOCALIZATION

The localization of nodes runs in two main substeps. In the first substep, four (or three in 2D) seed nodes are selected and assigned relative coordinates according to their hop distances. In the second substep, an iterative protocol is used to localize nodes. During the iterations, only edge landmarks flood their locations once localized.

6.1. Selecting Seed Nodes

The seed nodes should be assigned coordinates that agree well with their true distances (up to scaling) so that the whole localization process starts with a good reference. This

[1]In our experiments, $k = 15$ turns out to work well. We have also tried alternative criteria without noticing significant differences in the final result.

requires that their shortest paths avoid notches as much as possible. Another desirable condition is that they are not too close to each other so that the distance measurement is less affected by local errors.

To select seed nodes the coordinator floods a message demanding that every edge landmark perform a scoped flood in $5\times$`LandmarkSpacing` hops. Every edge landmark then determines its *notch-free distance*, the number of hops to a nearest notch node minus one. The neighborhood in such a distance is called the edge landmark's *notch-free neighborhood*. The coordinator collects the notch-free distances and selects the edge landmark l_{freest} that has the largest notch-free distance and passes a token to it, which then takes the responsibility of selecting seeds.

l_{freest} first finds a farthest node in its notch-free neighborhood by local flooding and collecting responses. The farthest node is taken as the first seed s_0. Afterwards, s_0 finds a farthest node to itself in the same notch-free neighborhood which is taken as s_1. The third seed s_2 is again determined by s_0 by finding the farthest node in the notch-free neighborhood in terms of accumulative distance to both s_0 and s_1. The fourth seed node s_3 can be found in a similar way by considering its accumulative distance to the first three seeds.

With the four seeds determined, the seeds use their real locations to localize the whole network. In this relative coordinate system, a unit distance corresponds to a hop in the network. After that, the four seed nodes start announcing their locations by global network flooding, and the iterative localization process begins.

6.2. Notch-Avoiding Multilateration

The localization runs in rounds. In each round, some newly localized nodes (edge landmarks) flood their locations, and other nodes calculate/update their locations passively using multilateration.

6.2.1. Path Quality and Location Confidence. Multilateration uses estimated distances to multiple reference nodes. Since some of these estimates may be poorer in quality than others due to the presence of notches in their corresponding paths, the location obtained is assigned a confidence value to guide future improvement. To define the localization confidence, we first need to define the quality of a path $\mathcal{P}$.

When a landmark floods its location, the forwarded packet carries the notch degrees of all nodes on the path. A receiver records this path in the form of a notch degree vector in its local memory. Based on the `NotchThreshold` value, a node extracts the following information: (1) the total hop count of the path; (2) the index of the first notch I_{first} (nodes numbered starting from 0) on the path; (3) the index of the last notch I_{last} on the path. Define the *bent length* as

$$L_{\text{bend}} = \max\left(I_{\text{last}} - I_{\text{first}}, \min(I_{\text{first}}, k - I_{\text{last}})\right).$$

The quality of $\mathcal{P}$ is thus defined as

$$Q(\mathcal{P}) = \begin{cases} 1, & \text{if } k < 2 \text{ or } L_{\text{bend}} = 0; \\ \frac{1}{L_{\text{bend}}}, & \text{if } k \geq 2 \text{ and } L_{\text{bend}} \leq 2; \\ 0, & \text{if } k \geq 2 \text{ and } L_{\text{bend}} > 2. \end{cases} \tag{2}$$

Roughly speaking, the more seriously a path bends, the lower the quality. Figure 4 illustrates how path quality is calculated, and Figure 5 shows several more examples.

The confidence of a multilaterated result p is defined as the average quality of the m shortest paths from which p is calculated, that is,

$$C(p) = \begin{cases} 0, & \text{if } m < 4 \text{ (or } < 3 \text{ in 2D cases)}; \\ \frac{1}{m}\sum_{i=1}^{m} Q(\mathcal{P}_i), & \text{otherwise.} \end{cases} \tag{3}$$

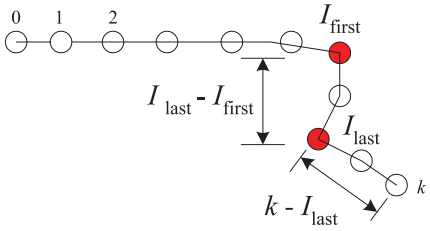


Fig. 4. Path quality calculation. Red circles represent notch nodes.

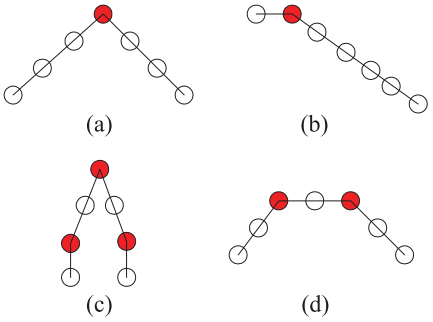


Fig. 5. Quality of the path $\mathcal{P}$. (a) $L_{\text{bend}} = 3$ and $Q(\mathcal{P}) = 0$; (b) $L_{\text{bend}} = 1$ and $Q(\mathcal{P}) = 1$; (c) $L_{\text{bend}} = 4$ and $Q(\mathcal{P}) = 0$; (d) $L_{\text{bend}} = 2$ and $Q(\mathcal{P}) = 0.5$.

6.2.2. Iterative Localization. Initially, the seed nodes set their confidence to 1. During the localization, every unlocalized non-seed node listens to the flooding of edge landmarks and adds the shortest path of positive qualities to its *landmark table*. The shortest paths are selected randomly in the landmark table during the iterative localization process. In each round, edge landmarks that have just obtained their locations in last round flood their locations, and other nodes update their locations silently.

When calculating coordinates with multilateration, a node randomly selects a subset of K_m landmarks from its landmark table as references. In our design, $K_m = \min(K, 10)$ is found to provide enough reference diversity, hence good results, while higher values bring no further improvement due to increasing self-interference between references. To exclude outliers due to badly positioned (e.g., collinear or clustered) references, each node runs the random selection and multilateraion multiple times and tests the distances between these localized results.

As time goes on, a node's landmark table may grow in size (up to K entries) and thus get the chance to improve its localization confidence. We let a node update its coordinates only when its confidence can be improved by a margin Δ_{conf} (0.3 in our setting). If that node happens to be an edge landmark, then it needs to re-flood its new coordinates in the next round. It is easy to see that any landmark floods at most $1/\Delta_{\text{conf}}$ times during the localization.

6.2.3. Notch Threshold Adjustment. The coordinator maintains the number N_f of floods it has ever seen from distinct edge landmarks. If it does not see a location advertising flood in a round, it knows that no new nodes can localize themselves in that round. In this case, the coordinator checks whether $N_f < N_{\text{elms}}$, meaning that at least one edge landmark has not yet localized itself, so the localization process is still incomplete.

Upon detecting premature stopping, the coordinator floods a message containing an increased `NotchThreshold`, with an increase step equal to half the initial notch threshold. This will lead to fewer nodes to be regarded as notch nodes. Nodes receiving this

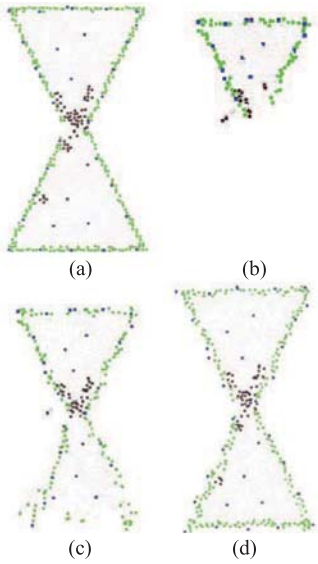


Fig. 6. The effect of notch threshold adjustment. (a) original network; (b) partially localized network when `NotchThreshold` = 3.0; (c) fully localized network for `NotchThreshold` = 10.2; (d) smoothed result.

message and whose confidence is below $1 - \Delta_{\text{conf}}$ then update their landmark tables and try to calculate a valid or new location. Edge landmarks that can newly localize themselves or re-localize will flood their new locations, enabling the localization process to continue.

Figure 6 gives an example of notch threshold adjustment. The hourglass shaped network has 1,737 sensors with average node degree of 11.58. Originally `NotchThreshold` = 3, and three seeds are selected by the program in the upper half of the hourglass. The seeds lead to the upper half being successfully recovered (see Figure 6(b)). However, since the waist part of the hourglass is filled with notch nodes, the localization process cannot make it through the waist. This causes `NotchThreshold` to be raised, which in turn decreases the number of notches near the neck (not shown in the figure). When `NotchThreshold` is increased to 10.2, fewer nodes remain as notch nodes, so that the lower half network gets the opportunity to be localized. Figure 6(c) shows the localization result.

When $N_f = N_{\text{elms}}$, the coordinator announces the completion of the protocol. Note that the protocol is guaranteed to stop because when `NotchThreshold` approaches N_{elms} few nodes will remain as notches so every node will be able to localized itself.

6.2.4. Location Smoothing. In a final step, we use a simple spring-mass algorithm [Leong et al. 2007] to smooth nodes' coordinates. The algorithm is first run on the landmark network, after which all landmarks are fixed, and then on the node level. The landmark-level smoothing is less susceptible to local minima and so helps to restore the network topology at a coarse grain, and the node-level smoothing helps to refine nodes' positions with respect to their immediate neighbors.

7. ANALYSIS OF CATL'S MESSAGE COST

The dominant factor of CATL's message cost is its global network flooding, mainly used in the notch detection and localization processes. The former takes exactly $N_{\text{elms}} \in O(n_e)$ rounds of flooding, where n_e is the number of edge nodes in the network; the latter involves only edge landmarks, each performing at most a constant number of floods, so

the number of floodings by the localization process is also $O(n_e)$. In total, the per-node message cost of CATL is $O(n_e)$. n_e is comparable with the number of boundary nodes of a network and is generally sublinear with the network size. This message complexity is lower than that of some other 3D capable schemes, including DV-hop [Niculescu and Nath 2003], MDS [Shang et al. 2003], and PDM [Lim and Hou 2005].

8. SIMULATIONS

We have developed a simulator[2] to study CATL's performance under various settings. For all settings, network parameters are chosen such that the network is connected. Node density is represented by average node degree. Two radio models are considered: the Unit Ball Graph (UBG) (default setting) and Quasi-UBG models, which are the 3D equivalents to the Unit Disk Graph (UDG) and Quasi-UDG models in 2D, respectively [Flury and Wattenhofer 2008]. Nodes are by default deployed in a perturbed grid pattern [Li and Liu 2010; Lederer et al. 2008]: a node is placed at a grid point with a perturbation governed by a Normal distribution. We also consider a uniform random sensor placement scheme [Li and Liu 2010].

We compare CATL against both 3D and 2D schemes from previous work. Among the few 3D capable schemes, we choose a classic one, DV-hop [Niculescu and Nath 2003], for comparison. We are aware of only two other 3D capable and connectivity-based schemes: PDM [Lim and Hou 2005] and MDS [Shang et al. 2003]. PDM needs $O(n)$ anchor nodes to work to its advantage, and for a small number (e.g., <10) of anchors, it is reported to perform no better than DV-hop [Li and Liu 2010]. MDS requires the global network connectivity information which is unavailable in a distributed setting. Moreover its $O(n^3)$ message and computation cost renders itself impractical for large-scale (e.g., $n \geq 12,000$ as in our cases) networks. For these reasons we believe that DV-hop suffices to show what one can achieve for 3D networks with known techniques under the desired constraints. For 2D comparison, we pick Li and Liu [2010], as it appears to be most successful among several state-of-the-art 2D solutions.

CATL's localization accuracy is studied quantitatively using the metric *location error*, defined as the ratio of the Euclidean distance between the obtained and true locations to the radio range. To make the error calculation meaningful, we have to let CATL generate absolute coordinates. This is done by feeding CATL the physical coordinates of the four seed nodes it determines and the average physical hop distance (the latter can actually be inferred from the seed nodes' coordinates). The resulting network thus has the same scale and orientation as the original one. The DV-hop algorithm also uses these four seed nodes when it localizes the network.

To provide some idea as to how much distortion CATL causes the global network layout, the network *diameter*—the maximum number of hops between all pairs of nodes—is sometimes provided along with location error.

8.1. CATL for Various Network Topologies

Figure 7 shows the localization results of CATL for several 3D networks. CATL takes the connectivity graphs of the left-column networks as input and produces networks shown in the right column. The location errors are provided in Table III. We can see that CATL successfully recovers the network layouts with small errors. For example, in the ring-shaped network, the average location error is only 0.92, in contrast with the network diameter 37, suggesting a very small distortion in network layout. In contrast, DV-hop performs rather poorly, with an average error of 4.45. The large performance difference confirms the importance of concavity-adaptivity in network localization, and that CATL performs very well in this respect.

[2]The source code of the simulator is available for download at `http://code.google.com/p/catl-code/`.

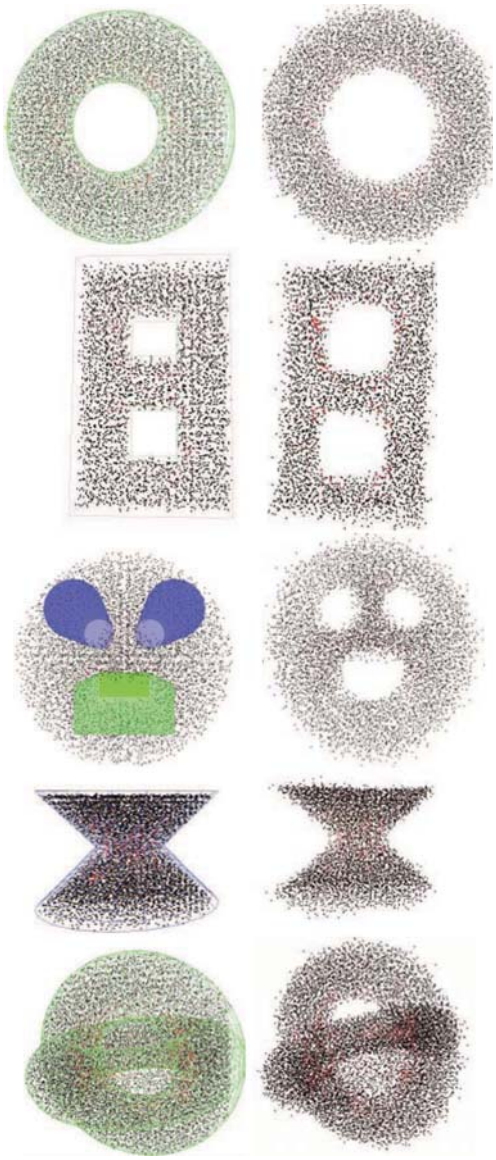


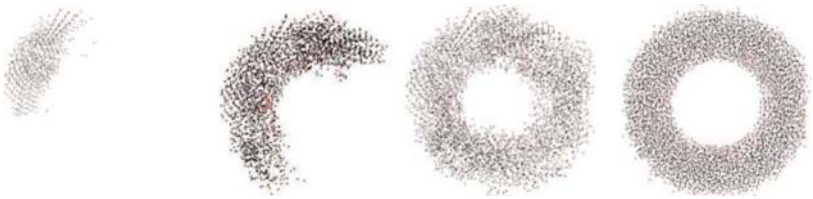
Fig. 7. CATL for 3D networks. The black and red points represent the ordinary and (initial) notch nodes, respectively. Left/right column: original/localized networks. Row 1: (1) Ring shape, 7,171 nodes, average node degree 11.47, localized in 21 rounds; (2) 8 shape, 4,410 nodes, average node degree 10.16, localized in 19 rounds. Row 2: (1) Face shape: a ball with two cylinder holes (eyes) and one rectangular cuboid hole (mouth), 6,908 nodes, average node degree 11.23, localized in 14 rounds. (2) Hourglass-3D, 7,446 nodes, with the average node degree 14.45. localized in 21 rounds. Row 3: (1) Cross-ring: two orthogonal rings, 12,914 nodes, average node degree 13.14, localized in 28 rounds.

8.2. CATL at Different Stages: A Close Look

To provide a close-up look at CATL's behavior during the localization process, Figure 8 presents some intermediate results of CATL running on the ring topology at different stages (rounds). The figure only shows localized nodes with positive confidence. CATL first picks seed nodes, which are all located at a convex corner of the ring (so that

Table III. Location Errors of CATL for 3D Topologies Compared with DV-Hop

Topology	Scheme	Avg Err	5-prtl Err	95-prtl Err	Diameter
Ring	CATL	0.92	0.34	1.74	37
	DV-Hop	4.45	1.16	9.67	37
8	CATL	1.78	0.49	4.16	36
	Dv-Hop	2.83	0.33	10.75	36
Face	CATL	1.08	0.42	1.84	26
	DV-Hop	2.38	1.76	3.83	26
Hourglass-3D	CATL	1.14	0.34	2.11	29
	Dv-Hop	2.12	0.41	9.63	29
Cross-rings	CATL	1.91	0.53	3.24	33
	Dv-Hop	3.12	1.80	2.74	33



(a) After one round. (b) After two rounds. (c) After four rounds. (d) After ten rounds.

Fig. 8. CATL localization results after one, two, four, and ten rounds. The final network is smoothed. The network has 7,111 nodes with an average degree of 11.47.

(a) (b) (c)

Fig. 9. Effect of different node densities on localization. The average node degree in (a), (b), and (c) is 8.56, 9.44, 15.25, respectively.

they are "visible" to each other). These seed nodes will produce localization chances for some nodes that are able to "see" all of them, while other nodes cannot determine their locations with positive confidence because their shortest paths significantly bend around the inner wall of the ring. These localized nodes obviously account for only part of the network, as shown in Figure 8(a). However, some of the newly localized nodes act as new references in the next round and so provide chances for other nodes to get localized. This leads to more nodes to localize themselves after the second round (see Figure 8(b)). This iterative process goes on until the end of the 10th round, when the process finishes, yielding the final shape in Figure 8(d).

8.3. Effect of Node Density

Node density is adjusted by changing the real radio range with a fixed small step size. The lowest average node degree is 8.56, below which the network starts to get disconnected. Figure 9 shows the localized results for the ring-shaped network in Figure 7 for a range of node densities. Table IV gives the accuracy results in terms of average, 5% percentile, and 95% percentile location errors. From the results, we can

Table IV. Location Errors of CATL for Various Node Densities (Average Node Degrees)

Avg deg	Avg Err	5-prtl Err	95-prtl Err	Diameter
8.56	1.50	0.47	2.97	41
9.32	1.73	0.67	3.35	39
11.30	0.92	0.34	1.74	37
14.94	1.65	0.43	3.71	32
17.41	2.99	0.58	5.50	30

(a) (b) (c)

Fig. 10. Performance of CATL under the Quasi-UBG radio model. The original network has 6,811 nodes. (a) $\alpha = 0.05$, average node degree 11.70; (b) $\alpha = 0.10$, average node degree 12.35; (c) $\alpha = 0.15$, average node degree 12.85.

(a) (b)

Fig. 11. CATL localization results for uniformly random node distribution: (a) localized network; (b) localized and smoothed network. There are 8,215 nodes with average node degree 14.32 and network diameter 37.

see that CATL produces good results and is quite robust to varying network densities. The robustness is also observed in other network topologies in Figure 7.

8.4. Effect of Radio Model

In the Quasi-UBG radio model with parameter $0 \leq \alpha < 1$, a link exists between two nodes u and v with probability 1 when $|uv| \leq 1-\alpha$, and with probability $0 < p < 1$ when $1-\alpha < |uv| \leq 1+\alpha$; the link does not exist when $|uv| > 1+\alpha$. We vary α and adjust p so that the average node degree in the tested networks remains nearly the same. When α increases, the link relationship between two nearby nodes becomes more and more uncertain, mimicking the irregular radio patterns found in reality. Figure 10 shows the localized networks, along with error results, for varying α values. It can be seen that CATL achieves reasonably good performance for a range of α values. Although the performance gets worse for a larger α, the localized networks still capture the global network shape faithfully.

8.5. Effect of Node Distribution

We examine CATL's performance under an alternative sensor placement scheme: uniform random distribution. Without using a reference structure (i.e., grid) for node placement, the network connectivity may exhibit more randomness than the perturbed grid approach. For example, a larger variance in node degrees or larger voids may be found in the network. Figure 11 shows CATL's results on the ring topology. (Other topologies have similar performance trends.)

As a result of increased randomness, the localized network layout shows more uneven node distribution and rougher edges (see Figure 11(a)), compared with its counterpart in Figure 7. This is to be expected. However, the localized network still captures the basic shape of the original one. Moreover, the smoothing process successfully reduces the variability of internode distance.

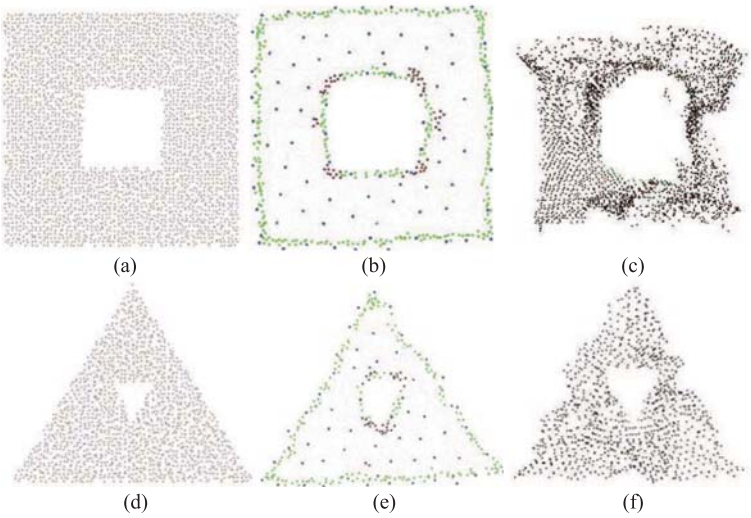


Fig. 12. Comparison between CATL and REP on 2D topologies. (a) The SQUARE topology, with 3094 nodes and average degree 9.62; (b) CATL's result for SQUARE; (c) REP's result for SQUARE; (d) the TRIANGLE topology, with 1456 nodes and average degree 10.98; (e) CATL's result for TRIANGLE; (f) REP's result for TRIANGLE.

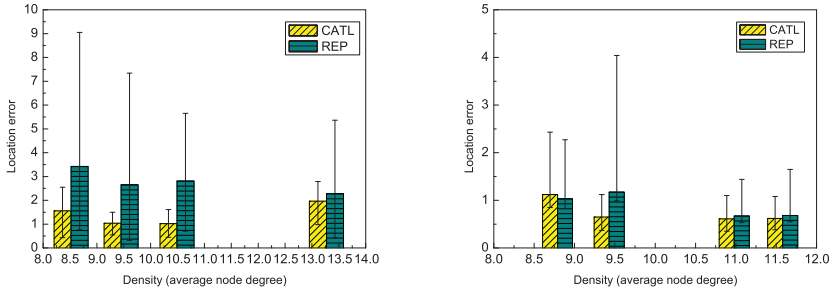
8.6. Comparison with a 2D-Only Scheme

We compare CATL with REP [Li and Liu 2010], a recent design that shows excellent results in dealing with the localization of topologically complex networks. REP uses a different method to estimate the distance between two nodes whose shortest path bends around a hole. For such two nodes, a virtual hole around a hole corner is created, which produces a new shortest path. The geometrical information of the virtual hole and the two shortest paths can then be used to resolve for the true Euclidean distance between the two nodes.

Figure 12 shows two simple network topologies, labeled SQUARE and TRIANGLE, respectively, as well as the localized topologies of both CATL and REP. It can be seen that CATL better recovers the network shape than REP does. Figure 13 compares CATL and REP under different node densities. In 7 out of 8 cases, CATL outperforms REP, with an error reduction over 50% in some cases. Figure 14 presents CATL and REP's location errors under different radio models. Again, in most cases, CATL is superior to REP. The advantage of CATL is most significant in Figure 15, where different node distributions are examined. In 7 out of 8 cases, we can see a dramatically smaller error in CATL's results. The smaller location error of CATL in most cases suggests that it is more accurate than REP in estimating internode Euclidean distance. CATL's spring mass smoothing algorithm further contributes to the smaller error variance, as indicated by the percentile bars.

8.7. Message Cost

The dominant factor of CATL's message cost is from the flooding operations, which directly translate to per-node message cost. Table V presents the number of flooding operations performed by CATL. Generally, CATL needs a few hundred flooding operations to localize the networks. Since the flooded messages are very short and message retransmission due to losses can be reduced by allowing sufficient time for CATL to complete, we believe that this cost is acceptable for common sensor devices, which are often expected to work over a relatively long lifetime (e.g., weeks).



(a) SQUARE. (b) TRIANGLE.

Fig. 13. Average location error (with 5th and 95th percentiles) for different node densities.

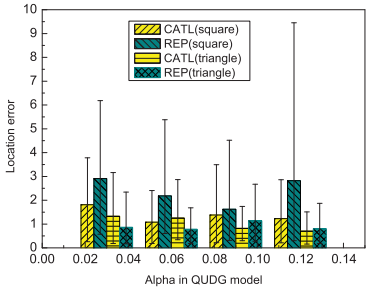


Fig. 14. Average location error (with 5th and 95th percentiles) for different α values in the quasi-UDG radio model.

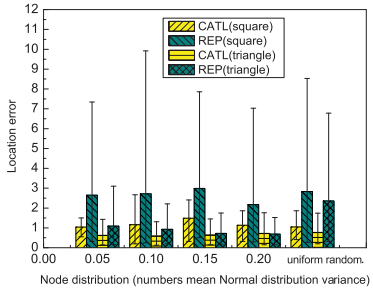


Fig. 15. Average location error (with 5th and 95th percentiles) for different node distributions. The numbers in the X axis mean the variance of the Normal distribution used in the perturbed-grid node placement.

Besides the absolute cost, it is meaningful to look at the *relative message cost*, the ratio of the number of floods to the network size n. This can reflect how heavily per-node message cost depends on the network size, and what is the trend of the cost when the network grows. Table V indicates that the relative cost is all below 0.1. As the network grows in size, we expect that the relative cost tends to zero as the number of floods is sublinear with n.

9. DISCUSSION

In this section we discuss CATL's limitations and potential applications.

9.1. Limitations of CATL

The principle of multilateration and the notch detection heuristic hold precisely in a continuous plane, which can be thought of as a network of perfectly even and infinitely dense node distribution. When the real network has a relatively even and dense node distribution, CATL performs satisfactorily as demonstrated. When this is not the case, however, CATL may not function properly. Some networks, for instance, have a nonuniform node distribution due to heterogeneous node configuration or the nature of sensing tasks. This can seriously undermine the hop-count-based length estimate, which in turn causes large errors in CATL's results. Clearly, this problem is shared by all pure connectivity-based localization schemes.

When a network contain sparse areas where some nodes may never find straight paths to enough references, CATL will also perform poorly. This can be seen from

Table V. Number of Flooding Operations by CATL

Topology	Notch detection	Localization	Total (relative cost)
Ring	100	111	211 (0.029)
8	113	138	251 (0.061)
Face	158	190	348 (0.049)
Hourglass-3D	217	275	492 (0.066)
Cross-ring	218	252	472 (0.036)

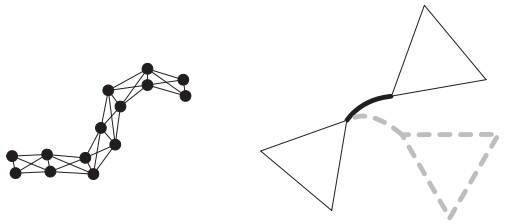


Fig. 16. Network shapes that CATL cannot accurately recover.

Figure 16(a), in which network nodes are placed along a narrow band with very restrictive connections. CATL will fail to recover the original network shape, because multilateration can only be done with highly distorted path measurement, and more fundamentally, because the network itself cannot be deterministically localized. In other words there are infinitely many realizations that satisfy the connectivity constraints. Figure 16(b) shows a more complex example in which CATL may produce a layout (dashed line) that deviates from the ground truth.

Both limitations are inherent for pure connectivity-based localization schemes. That means some networks can by no means embedded into a 2D/3D space in a unique way without additional information about node's positions.

9.2. Potential Applications of CATL

Where CATL's assumptions hold or its limitations do not have a critical impact, we anticipate CATL's applications to several other sensor network tasks.

Connectivity-based 3D boundary detection. By collecting nearby nodes's coordinates and forming a local map, a node can easily recognize whether it is on a network boundary, and if it is, how it is connected to neighbors to form a surface (in 3D) or a boundary line (in 2D). This provides a possible solution to the problem of connectivity-based 3D boundary detection which has remained open so far.

3D geographic routing using virtual coordinates. 3D geographic routing is known to be a challenging problem [Flury and Wattenhofer 2008], and research in this line mostly assumes the availability of node locations. CATL extends those designs to environments where no GPS devices are available. There have been a few efforts [Rao et al. 2003; Fonseca et al. 2005] to assign virtual coordinates to general (2D or 3D) sensor networks for routing. Being blind to the network's geometric features, these methods potentially suffer sever performance degradation in irregularly shaped networks [Tan et al. 2009]. By faithfully revealing the network's salient geometric features, CATL provides those algorithms with useful information to fine-tune their behavior so as to better adapt to complex network environments.

Connectivity-based 3D network segmentation. Network segmentation aims to divide the network into pieces that have nice shapes [Zhu et al. 2008] or satisfy certain

properties [Tan et al. 2009]. This benefits many applications, such as landmark-based routing [Fonseca et al. 2005; Tan et al. 2009], distributed indexing, random sampling, data aggregation, etc. [Zhu et al. 2008]. Despite a few efforts for 2D networks, there has been no previous work for 3D cases. CATL again provides support for this application in 3D.

10. CONCLUSION

CATL is a connectivity-based and anchor-free localization scheme that produces accurate relative coordinates for both 3D and 2D large-scale and anisotropic sensor networks. It makes few assumptions about the network environments; its 3D localization capability also distinguishes itself from several recent sophisticated schemes that are shown to work successfully in (only) 2D networks. We have demonstrated its efficacy with simulations. In the future we will investigate CATL's applications to other sensor network applications, as mentioned in Section 9.2.

ACKNOWLEDGMENTS

We thank Yue Wang for providing us with the data in Figure 3.

REFERENCES

Bruck, J., Gao, J., and Jiang, A. 2005. Map: Medial axis based geometric routing in sensor networks. In *Proceedings of the MobiCom*.

Bulusu, N., Heidemann, J., and Estrin, D. 2000. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Commun. Mag.*

Cheng, W., Teymorian, A., Ma, L., Cheng, X., Lu, X., and Lu, Z. 2008. Underwater localization in sparse 3D acoustic sensor networks. In *Proceedings of INFOCOM*.

Flury, R. and Wattenhofer, R. 2008. Randomized 3D geographic routing. In *Proceedings of INFOCOM*.

Fonseca, R., Ratnasamy, S., Zhao, J., Ee, C. T., Culler, D., Shenker, S., and Stoica, I. 2005. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Proceedings of NSDI*.

Jin, M., Xia, S., Wu, H., and Gu, D. 2011. Scalable and fully distributed localization with mere connectivity. In *Proceedings of INFOCOM*.

Lederer, S., Wang, Y., and Gao, J. 2008. Connectivity-based localization of large scale sensor networks with complex shape. In *Proceedings of INFOCOM*.

Leong, B., Liskov, B., and Morris, R. 2007. Greedy virtual coordinates for geographic routing. In *Proceedings of ICNP*.

Li, F., Luo, J., Zhang, C., Xin, S., and He, Y. 2011. Unfold: Uniform fast on-line boundary detection for dynamic 3D wireless sensor networks. In *Proceedings of Mobihoc*.

Li, M. and Liu, Y. 2010. Rendered path: Range-free localization in anisotropic sensor networks with holes. *IEEE/ACM Trans. Netw.*

Lim, H. and Hou, J. C. 2005. Localization for anisotropic sensor networks. In *Proceedings of INFOCOM*.

Niculescu, D. and Nath, B. 2003. DV based positioning in ad hoc networks. *J. Telecom. Sys.*

Rao, A., Ratnasamy, S., Papadimitriou, C., Shenker, S., and Stoica, I. 2003. Geographic routing without location information. In *Proceedings of MobiCom*.

Shang, Y. and Ruml, W. 2004. Improved MDS-based localization. In *Proceedings of INFOCOM*.

Shang, Y., Ruml, W., Zhang, Y., and Fromherz, M. P. J. 2003. Localization from mere connectivity. In *Proceedings of MobiHoc*.

Tan, G., Bertier, M., and Kermarrec, A.-M. 2009. Convex partition of sensor networks and its use in virtual coordinate geographic routing. In *Proceedings of INFOCOM*.

Wang, Y., Gao, J., and Mitchell, J. S. B. 2006. Boundary recognition in sensor networks by topological methods. In *Proceedings of MobiCom*.

Wang, Y., Lederer, S., and Gao, J. 2009. Connectivity-based sensor network localization with incremental delaunay refinement method. In *Proceedings of INFOCOM*.

Zhou, H., Wu, H., and Jin, M. 2012. A robust boundary detection algorithm based on connectivity only for 3D wireless sensor networks. In *Proceedings of INFOCOM*.

ZHOU, H., XIA, S., JIN, M., AND WU, H. 2010. Localized algorithm for precise boundary detection in 3D wireless networks. In *Proceedings of ICDCS*.

ZHOU, Z., CUI, J.-H., AND ZHOU, S. 2007. Localization for large-scale underwater sensor networks. UCONN CSE Tech. rep. UbiNet-TR07-01, University of Connecticut.

ZHU, X., SARKAR, R., AND GAO, J. 2008. Shape segmentation and applications in sensor networks. In *Proceedings of INFOCOM*.

Received October 2012; revised February 2013; accepted February 2013