DisLoc: A Convex Partitioning Based Approach for Distributed 3-D Localization in Wireless Sensor Networks

DisLoc: a convex partitioning based approach for Distributed 3D Localization in Wireless Sensor Networks

Jin Fan*, Yidan Hu†,Tom H. Luan‡, Mianxiong Dong§
*Hangzhou DIANZI University,China
†University of Delaware,USA
‡Deakin University, Australia
§Muroran Institute of Technology, Japan
fanjin@hdu.edu.cn,yidanhu@udel.edu, tom.luan@deakin.edu.au,mx.dong@csse.muroran-it.ac.jp

***Abstract*—Accurate localization in wireless sensor networks (WSNs) is fundamental to many applications, such as geographic routing and position-aware data processing. This, however, is challenging in large scale 3D WSNs due to the irregular topology, such as holes in the path, of the network. The irregular topology may cause overestimated Euclidean distance between nodes as the communication path is bent, and accordingly introduces severe errors in 3D WSN localization. As an effort towards the issue, this work develops a distributed algorithm to achieve accurate 3D WSN localization. Our proposal is composed of two steps, segmentation and joint-localization. In specific, the entire network is first divided into several subnetworks by applying the approximate convex partitioning. A spatial convex node recognition mechanism is developed to assist the network segmentation which relies on the connectivity information only. After that, each subnetwork is accurately localized by using the multi-dimensional scaling (MDS) based algorithm. The proposed localization algorithm also applies a new 3D coordinate transformation algorithm, which helps reduce the errors introduced by coordinate integration between subnetworks and improve the localization accuracy. Using extensive simulations, we show that our proposal can effectively segment a complex 3D sensor network and significantly improve the localization rate in comparison with existing solutions.**

I. Introduction

The 3D WSNs, where nodes are deployed in a 3D topology, have been witnessed in many Internet-of-Things applications, such as safety monitoring of buildings, indoor fire detections. The real-world topology of a 3D WSN, however, is typically complicated and rarely in a simple or regular shape such as a square or a disk. The complexity of the network topology accordingly brings significant challenges to existing localization algorithms. For instance, the centralized localization schemes [1] could fail when the network topology of a large scale WSN is irregular. The irregular network shapes also challenge the accuracy of localization scheme. For example, a number of localization algorithms in literature assume the straight-line shortest paths between nodes. The assumption is not valid when the network topology is in a complex 3D shape. This is because the shortest path between nodes may be bent, which would result in significant errors in path distance estimation and accordingly inaccurate localization [2].

Considering the challenges brought by the irregularity of network topology, a number of methods have been developed in literature[3][4][5] etc.. Tan et al. in[6] propose a convex partitioning protocol to tackle the 2D/3D topology complexity issue. However, these existing methods have been challenged in many ways as the fundamental differences between the 2D and 3D scenarios. For example, in 3D WSNs, the network boundary condition becomes more complicated and the connectivity of the network can be more diverse compared to the 2D scenario. There are a few centralized localization schemes, such as three dimensional Distance Vector-Hop algorithm(3D-DV-HOP)[7] and three dimensional multidimensional scaling algorithm (3D-MDS)[8], which are capable of estimating node location in a 3D scenario. However, these schemes either suffer from the high computation complexity, or heavily depend on strict assumptions of network configurations, such as dense deployment of anchor nodes, the availability of depth information, a relatively small network size [9] and *etc*. To summarize, the existing schemes can hardly work effectively and efficiently in a distributed manner with connectivity information only (or with a constant number of anchors) in a large-scale 3D WSN of irregular topology.

This paper addresses the above challenges by developing a distributed 3D localization approach in a large-scale 3D WSN. Notably, as a 3D sensor network grows larger, it becomes more intricate in topology on account of its close relations with the surrounding deployment context. A distributed localization algorithm is therefore highly desirable, which is the key to achieve network localization at a low computational cost. In this paper, a new localization protocol for 3D wireless sensor networks based on network segmentations is proposed. By accurately identifying the concave nodes in the topology, our proposal first decomposes a 3D WSN into a number of approximate convex subnetworks. In each subnetwork, the improved MDS algorithm is adopted to realize the relative map. After that, a new three-dimensional coordinate transformation algorithm inspired by Camera Calibration Principle of computer vision is developed to integrate subnetworks for absolute locations. The proposed spatial concave node

recognition algorithm is not exclusive, which can be applied in different network connectivities.

The contribution of this paper can be summarized as follows:

- Algorithm Design: we proposed a fully distributed localization algorithm, named DisLoc, for large-scale 3D WSNs with irregular network topology.
- Sub-optimal network partition: by accurately identifying the concave nodes, DisLoc contains an efficient network partition scheme, which decomposes a 3D WSN into a number of approximate convex subnetworks. The number of subnetworks is also optimized by proposed scheme.
- Real-world model Validation: by using extensive simulations, we show that DisLoc can achieve accurate localizations at low cost in different real-world 3D WSN topologies.

The rest of this paper is organized as follows. Section II describes the related work in localization and the background of convex decomposition scheme. Section III elaborates the proposed approximate convex partition based localization algorithm. We assess the performance of proposed algorithm using simulations in Section IV, and the paper is concluded in Section V.

II. Related works

A. Distributed Vs Centralized Localization Schemes

A dominant approach of 3D localization is by employing the centralized algorithm to compute the actual coordinate of each unknown node. The centroid-based approach [10] is one of the earliest works of connectivity-based localization. In [10], a node estimates its location simply by calculating its distance to the centroid of anchor nodes which are in the communication range; and based on that the appropriate subarea will be selected as the node location. The localization of this scheme can be determined with high accuracy if the anchors are uniformly distributed with high density all over the network. However, it cannot be applied in large scale ad hoc WSNs for accurate localization as the high density deployment of anchor nodes is not affordable for most of large scale WSN applications in reality. Other centralized algorithms, such as 3D-DV-HOP [7] and 3D-MDS-MAP [8] are modified from 2D-plan scenarios. In a large scale wireless sensor network, with the limited power and computation capability of each sensor node, the centralized algorithms can be extremely expensive at cost of computational and time complexity, and therefore is not practical.

There is another attention in localization which uses the connectivity information only. These schemes aim to produce a relative coordinate system for a network without reliance on extra hardware supplement [2][11]. Liu *et al*. [12] define concavity to recognize concave nodes and decompose plane wireless sensor networks. The experimental results of [3] demonstrate that the error of the localization algorithm can be reduced by 60-90% after the processing. CATL [6] is recently proposed for 3D WSN localization. In CATL, notch nodes where the hop-count of the shortest path between the nodes deviates from the true Euclidean distance are firstly identified. Then an iterative notch avoiding multilateration scheme is adopted to realize the localization. The accuracy of CATL is closely tied up with the proper deployment of anchor nodes. As a result of the iterative operations, CATL has the drawback of error propagation. Zhang *et al*. [4] use the concave node definition to recognize concave nodes and perform convex partitioning on the sensor network. It is demonstrated in [4] that the convex partitioning approach could considerably enhance the performance of virtual-coordinate-based geographic routing algorithms and connectivity-based localization routing algorithms. It is also suggested that the concave nodes can be seen as a typical supporting structure for general geometry-related applications in wireless sensor networks.

B. Network Partitioning

Network segmentation has been intensively studied in literature [13], [14]. It is usually regarded as an optimization problem that divides the network while minimizing or maximizing some given criteria or property in the computational geometry. Most of these problems are, however, known to be NP-hard [15].

Convex decomposition, with or without Steiner point, has been researched for many years in computer graphics community [16], [17], [18]. Approximate convex decomposition (ACD) aims at minimizing concavity along with obtaining balanced partitions with perceivable components [19]. Wan [20] extends ACD to incorporate both concavity and curvature and prevents over-segmentation by avoiding cuts inside pockets. However, these algorithms often work in a centralized fashion and take the full coordinate information for granted, which is not applicable for large-scale sensor networks. Three-dimensional models are partitioned by projecting the model into 2D plane [21], [22]. The projections are segmented and mapped back on the original object. Nevertheless, the segmentation boundaries become restricted with respect to the choice of the projection planes and their orientation. The application of this method into a 3D WSN scenario could be infeasible due to the complex projection which involves the node locations.

The approximate convex partition based localization in 3D wireless sensor networks face several challenges. First of all, there is a lack of understanding of concavity of a 3D scenario. Secondly, recent convex partition schemes from the literature are designed for continuous shapes in a centralized manner. While for 3D distributed localization, the number of approximate convex parts has to be minimized as the merge of various coordination could bring non-negligible errors. Third, due to the nature of the random deployment of 3D sensor networks, it is impractical to manually identify convex/concave regions during deployment or just extract a graph of the network directly. Furthermore, in real-world application, the sensor network is discrete and the nodes can only obtain local information by message exchange. A low communication complexity scheme is therefore necessarily required consider-

ing the limited resource at each sensor node. Lastly, it is not straightforward to reconstruct a global map with the relative coordinates after the network segmentation. The number of partitions also has to be optimized for the localization accuracy and algorithm complexity. Our work is then motivated to address the above challenges in one framework.

III. Algorithm Design

This section develops the proposed algorithm for large-scale 3D WSNs based on network segmentations. By accurately identifying the concave nodes, the proposed scheme decomposes a 3D WSN into a number of approximate convex subnetworks, and we adopted the state-of-art MDS algorithm to achieve relative localization in each subnetwork. Finally, the proposed scheme DisLoc unifies location of all the subnetworks.

A. Overview

Since a real-world application of wireless sensor networks is discrete and has subsequent boundary noise, it is impossible to segment the network into strictly convex sections. As in [8], the MDS-based localization can tolerate localization error well due to its over-determined nature. Therefore, the discrete 3D network can be approximately divided into several convex sections. It is assumed that the boundaries of the network can be identified[1]. The proposed algorithm consists of the following three steps:

1) *3D network segmentation*. It decomposes a 3D wireless sensor network into several approximate convex pieces. It involves spatial concave node recognition, segmentation and partition recognition. The number of partitions optimization is discussed in this part as well.
2) *Distributed map establishment*. An advanced MDS technique is adopted here to realize relative localization for each approximate convex section.
3) *Global map establishment*. At last, we will merge all partitions into a global map and identify the absolute coordinates for each node. We develop a method which is inspired by camera calibration principle of computer vision. This coordinate merging method can reduce the errors of coordinate integration between subnetworks and improve the localization precision.

B. Network Segmentation

Network segmentation is designed to decompose a wireless sensor network into several approximate convex pieces. It proceeds three steps including concave node recognition, segmentation and partition recognition as follows.

1) Concave nodes Recognition: A concave node is a node where the inward angle (the angle spanning across the sensing area) is greater than π [6] as shown in Fig. 1 . Concave nodes presented in the network can seriously disrupt the straight-line course between nodes and result in the inaccuracy of localization algorithms. As a special kind of nodes, concave nodes also have following two features.

Feature I. Compared with the ordinary boundary nodes (including convex nodes), the spatial coverage of a concave node with k hops contains the largest number of nodes, as shown in Fig. 1(a).

In a 3D WSN where nodes are evenly deployed, a boundary node which has a greater coverage indicates that this node have more neighbors within *k* steps (a step here indicates a hop between two nodes). However, in general, sensor nodes are not uniformly distributed in a network. As such, we define relative coverage rate φ_i^k , which can be represented as (1). The relative coverage rate φ_i^k is proposed to capture the effect of network density.

$$\varphi_i^k = \frac{\sum_{j=1}^{k} x_i^j}{B_i} \qquad (1)$$

$$B_i = \begin{cases} \frac{1}{n}\sum_{t=1}^{n} x_t^1 & n \geq 1 \\ 1 & n = 0 \end{cases}$$

where B_i is the background distribution density of node i. x_i^j denotes the number of neighboring node for node i in j steps , and n denotes the boundary neighbor number of node i in one step.

Feature II. The maximum arc length of a spatial concave node is longer than those of other boundary nodes, and this maximum arc can be approximately described as the route from the left *k*th neighbor to the right *k*th neighbor, as shown in Fig. 2(b).

For a single node, as shown in Fig. 2(a), there is a pair of neighbors A and B having the largest central angle than other pairs such as a and b. Besides, this pair nodes are also boundary nodes. We define the concept of path concavity Ψ and calculate the path concavity for each candidate we selected after rough selection.

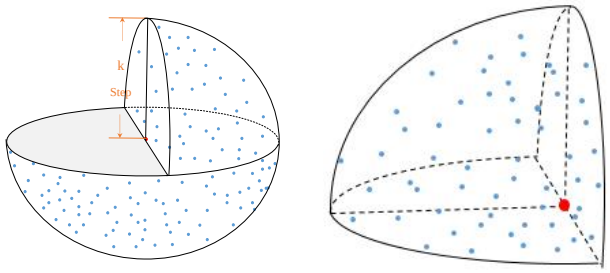
$$\Psi_i = \frac{Arc_i}{\pi \times k \times d_{hop}} \qquad (2)$$

where k represents a radius of k steps, d_{hop} is the average length of one step, and Arc_i is the longest arc length that combines node neighbors in k steps. When $\psi > 1 + \zeta$ (i.e., the central angle is larger than π), set node i as a concave node; otherwise, eliminate node i.

That is to say, we can identify concave nodes by calculating the distance of all such pairs of boundary nodes and estimating whether the longest one is longer than a threshold or not. If the answer is yes, the current node is a concave node. The shortest path between two neighbor nodes of a concave node is used to simulate the corresponding arc length of a potential concave node.

Using the shortest path of a pair of boundary nodes to simulate the longest arc length, the following cases must be considered when the network has a low connectivity (i.e., a node has a fewer neighbors in k steps):

[1]boundary identification is out of the scope of this work. We pointer interested readers to [23], [24], [25] for related schemes. In this paper, we adopt the scheme in [25] in our simulation.

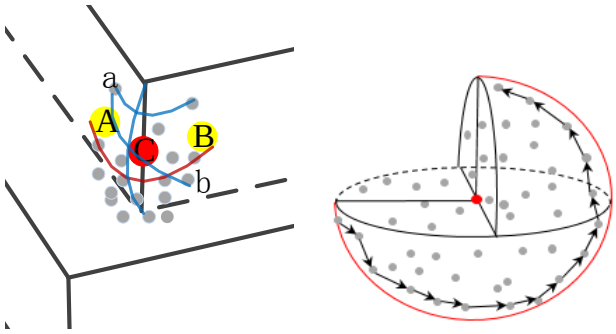


(a) The coverage of a concave node with k steps.

(b) The coverage of a convex node with k steps.

(c) The coverage of an ordinary node with k steps.

Fig. 1. Three types of boundary node coverage. The red dots denote the currently selected boundary node, and the blue dots denote the boundary nodes neighbor nodes in k steps



(a) A maximum route of concave node.

(b) Possible distance between neighbor pairs.

Fig. 2. The maximum route of concave nodes.

1) There is no shortest path between two nodes when the node has low connectivity as shown in Figure 3. This could lead to the situation where two boundary nodes do not have a connecting path in a network. To solve this problem, we introduce $k+l$-step neighbors as pseudo k-step nodes to help determine the shortest path, as shown in Figure 4(a). We do not take the $k-l$ neighbor nodes as the assistant nodes because they may bring significant reduction the length of the shortest path.
2) Spatial curve folding. Spatial curve folding is a unique problem in 3D space, as shown in Figure 4(a). Although there is a shortest path between two neighbor nodes in k steps, due to the node connectivity and irregular distribution, shortest path wrapping emerges in a segment of the path, which increases the length of the shortest path. This problem can be solved by adding assistant neighbor nodes. If the shortest path is shortened after $k+l$th step nodes are introduced for the nominated nodes, then the path folding occurs when only using the k-step neighbor

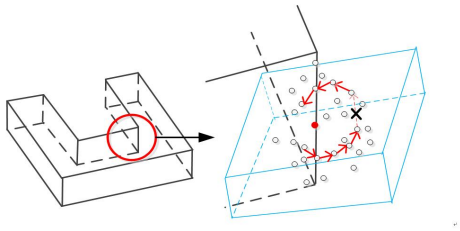
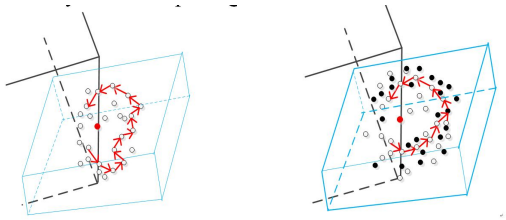


Fig. 3. The neighbor nodes of the nominated node cannot find the shortest path in k steps.



(a) Spatial curve folding

(b) Solution: Introduce nodes in $k+1$ steps

Fig. 4. Spatial curve folding problem and its solution.

nodes. Hence, the algorithm specifies the rule

$$dis(p_1, p_2) = min(f(A_k^i), f(A_{k,k+1}^i)) \quad (3)$$

where $dis(p_1, p_2)$ represents the distance between nodes (p_1, p_2), $f(A_k^i)$represents the shortest path between nodes (p_1, p_2) in the network topology, which is formed by the k-step neighbor nodes of node i.

3) Fake concave node: fake concave nodes are those nominated nodes that are a network boundary but that are not close to the boundary line. Such nodes may have a central angle slightly larger than π , as shown in Figure 5. To avoid the incorrect recognition of concave nodes, we adjust the threshold in Equation 2, i.e., $\psi_i > 1 + \zeta + \xi$, where $\xi > 0$ is an empirical number. The ξ is designed to rule out the fake concave node.

Based on above analysis, we propose Algorithm 1 to identify concave nodes based on the features presented. Algorithm 1 mainly consists of two phases: rough identifying and refined recognition. The rough identifying phase, the algorithm selects

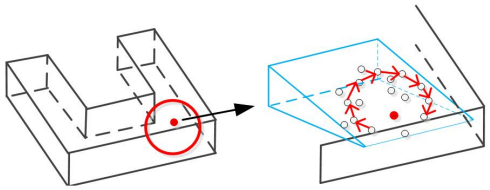


Fig. 5. Fake concave node

the nodes which have more neighboring nodes in k step coverage and identify them as potential concave nodes. The refined recognition is derived from the second feature.

Algorithm 1: Concave Nodes recognition

Require:
All boundary nodes.
The number of boundary nodes *N*.
A neighbor matrix *A* describes the connectivity relationship between all nodes.
Threshold δ and ξ.

Ensure:
Concave nodes set *CN*.

for each boundary nodes i **do**
  DO
  Calculating the number of neighbors n whose distances is less than k steps from a certain node i.
  Calculating background B whose ;
  if $n/b < \delta$ **then**
    candidate set CNc=CNc $\bigcup$ i;
  end if
end for
for each nodes j in CNc **do**
  calculating the neighbor set with k steps, A_k^j based on A;
  max=0;
  for each two nodes (p_1, p_2) in A_k^j **do**
    if $A(p_1, p_2) = 1$ **then**
      $dis(p_1, p_2)$;
      if $dis(p_1, p_2) > max$ **then**
        $max = dis(p_1, p_2)$;
      end if
    end if
  end for
  if $max > \psi$ **then**
    CN=CN $\bigcup$ j;
  end if
end for

In Algorithm 1, max is a temp flag which is used to find the longest distance. dis refers to the hop distance from p_1 to p_2. Threshold δ is used for rough selection of concave node recognition. δ is set to 3% for our following simulations, which means the top 3% boundary nodes with highest relative coverage rate is selected for the refinement. This empirical threshold is identified through simulations. It is verified that the nominated node set CN contains nearly all of the concave nodes when using this threshold. ψ is adopted to identify whether a candidate node is a concave node or not.

2) Convex Segmentation: Convex segmentation aims to decompose a WSN W including concave nodes into several convex pieces D, which is specified in Definition 1.

Definition 1. In a decomposition D of a wireless sensor network, W is defined as a set of components $\{W_i\}$, such that the union of the W_i is W, and every pair of M_i is interior disjoint:

$$\sum W_i = W$$
$$\forall_{i \neq j} M_i \bigcap M_j = \emptyset \tag{4}$$

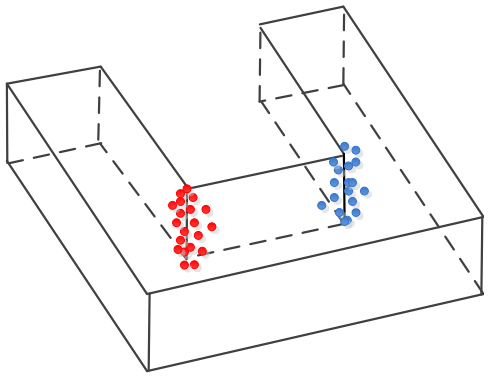


Fig. 6. The principle of segmentation.

To achieve this goal, we need to obtain all concave subsets from the original concave set. A concave subset contains all concave nodes in one concave area. As shown in Fig. 6, the original concave set can be divided into two concave subsets which are denoted by red and blue nodes, respectively.

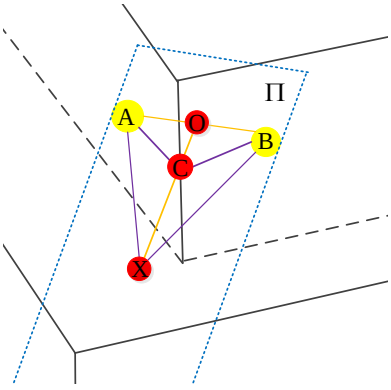
Intuitively, the hop distance between red concave nodes is less than the step length from a red concave node to a blue one. It means that the inter-class distance is different from the intra-class distance. In this way, we only need to set an appropriate distance threshold to classify all concave nodes.

Theorem 1. Converse of the mid-perpendicular plane theorem: If a point is equidistant from the endpoints of the segment, it is on the mid-perpendicular plane of a segment.

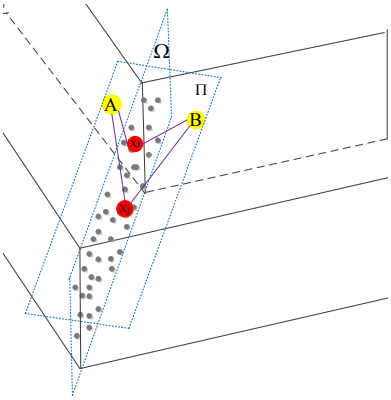
After the process of concave nodes recognition using Feature 2, each concave node has a pair of nodes distributed on the left and right, respectively. As shown in Fig. 7(a), a red concave node C represents a concave region and its two endpoints of the arc are denoted as yellow nodes A and B. Because point A and B are the k steps neighbors of the node C, then, we can believe that the distance AC is equal to the distance BC. We also denote the middle point of line AB as point O. Then, there is no doubt that $AO = BO$. According to theorem 1, we can deduce that point O and C are on the mid-perpendicular plane of line AB. In the same way, we also can find other arbitrary points X are all on the mid-perpendicular plane. Line AB and OC can determine a plan π. With the help of plane π, we can derive Feature 3.

Feature 3. For all nodes $X = x_1, x_2, \cdots, x_n$ in the segmental plane, the distance from x_i to the left boundary node is equal to the distance between x_i and the right boundary node.

Proof: Firstly, we assume that there is a plane Ω which is perpendicular to plane Π and point O is on the intersecting line of two surface. Certainly, line AB is perpendicular to plane Ω at point O. Then we choose an arbitrary point X in plane Ω. Because point O and X are both in plane Ω, then



(a) Characteristic of segmental nodes.



(b) Segmental lines and its plane

Fig. 7. The principle of segmentation.

we can deduce that line XO is perpendicular to line AB and $\angle AOX = \angle BOX$. At here, in $\triangle AOX$ and $\triangle BOX$, $OA = OB$,$OX = OX$ and $\angle AOX = \angle BOX$, then $\triangle AOX \cong \triangle BOX$. So, $XA = XB$. It demonstrates that, for any nodes X in the segmentation plane Ω, distance between the node A and X is equal to the distance from B to X.

In this way, to define a segmental plane Ω, we only need to find all the nodes which satisfies Feature 3 and denote them as segmental nodes. The plane Ω is also a bisector of the inward angle at each concave area.

3) Optimization the number of convex pieces: In general, the bisector process typically has an acceptable segmental result and all partitions generated by this convex segmentation are convex polyhedrons, as it is proved in [26]. However, more convex pieces indicates that more distributed coordinates need to be merged at the final stage of localization. This may bring significant errors for coordinate merging, and therefore decrease the accuracy of localization. To address this issue, we investigate in schemes to minimize the number of convex pieces during network partition.

Using the segmentation method in Section III-B2, we can get a segmental plane which bisect a concave area and divide the area into several parts. The segmental planes can be classified by two cases. One is that a plane will extend through a concave region until it reaches the boundary area. The other one is that a segmental plane will cross with other segmental planes before it extends to the boundary area. As shown in Fig. 8, plane ω belongs to the first category; and plane α interacts plane β at line μ, which belongs to the second circumstances. For the first case, there is no doubt that the segmentation process can terminate and two convex subnetworks can be obtained because of plane ω. However, for the second case, the number of convex subnetworks will be four. In a limited area, two intersected segmental planes indicate that the concave regions are very close to each other and the direction of each plane should not be parallel. It indicates that these area may be overly segmented. For example, as shown in Fig. 8, plane α and β belong to the second circumstance. Two concave areas are not far from each other and the difference between angle ϑ and ε is very obvious. Then we try to merge the original segmentation by searching for a line connecting the representative concave nodes C and F.

Although connecting two concave nodes directly will decrease the degree of concave certainly, it is necessary to check whether the generated partitions are all convex polyhedrons or not. If there is at least one partition still concave, the merged result should be discarded and the original result is reused. Otherwise, an optimized result is obtained and used for the following process, distributed localization. For checking, we use inward angle to ensure the result of approximate convex decomposition. In detail, as shown in Fig. 9(a), we will find a node x in the line CF firstly. The node x, A and B are all the kth neighbors of a representative concave node C. Then we will calculate the approximated arc length $\widetilde{Ax}$ and $\widetilde{Bx}$, respectively, according to (5).

$$l = \theta \times r \tag{5}$$

where θ is an angle and l is the arc length of this angle. r represents a radius. Here, taking node C as an example, θ is equal to angle $\angle ACx$ and $\angle BCx$. r is the length of hop distance.

Therefore, if $\widetilde{Ax}$ or $\widetilde{Bx}$ is larger than $\pi \times k$, this merge result should be discarded. This is because that the calculating value indicates that at lest one generated polyhedron is still concave. Similarly, for the concave area F, we do the same evaluated process to determine that whether the optimized result is effective.

In all, the process of optimized convex segmentation can be described as algorithm 2 and the pseudo optimal results are shown in Fig. 9(b).

4) Partition Recognition: The partition recognition phase aims to recognize the partition edges of two neighboring subnetworks and notify the nodes of the partition they belong to. The flooding mechanism is adopted here. The key challenge is how to confine the flooding scheme within certain partition without penetrating to other parts of the network. To achieve this goal, we proceed three operations, isolated nodes classification, flooding and extending.

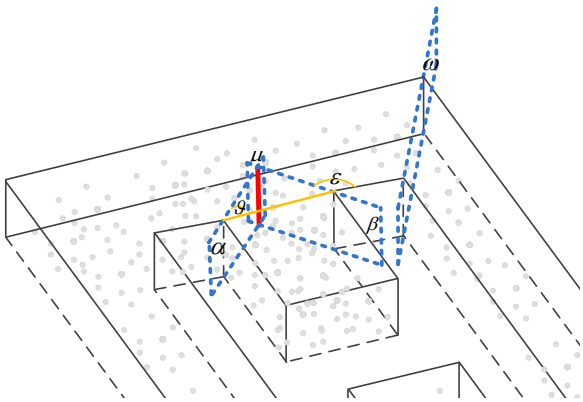
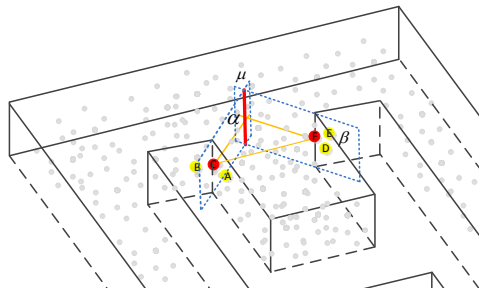
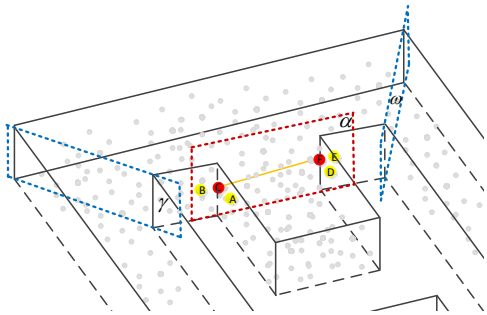


Fig. 8. Merge circumstance.



(a) Merge principle



(b) Optimized segmental result

Fig. 9. Optimized process and result.

Isolated nodes classification The result of segmentation phase can provide a natural border plane between two partitions. However, it is only a rough boundary and can not indicate which partition these nodes actually belong to. We construct a thick volume area to prevent the flooding scheme penetrating from different polyhedron. The isolated nodes are used to mark the termination of flooding process. We assume that there exist a coordinator in the network which is in charge of node classification of the network. In our implementation, all the k-hop neighbors of the segmental nodes (nodes in the segmental plane), including the segmental nodes themselves, constitute a thick volume area. All those k-hop neighbors are called as volume nodes. Then, no matter segmental nodes or volume nodes, their neighbor relationship is seen as temporary invalid from the neighbor matrix. These nodes then be recognized as isolated nodes. Then we successfully classified sensor

Algorithm 2: Optimized Convex Segmentation

Require:
Two concave area nodes whose segmental plane is crossing.

Ensure:
Optimized segmentation for concave nodes.

1: Find a shortest route Sr between two representative concave nodes;
2: Find the $k-hop$ neighbor X;
3: Calculating the corresponding arc length l for each segmental inward angle.
4: **if** $l > \pi \times k$ **then**
5: Readopt bisector segmental plane;
6: **else**
7: Sr and it first neighbors will be denoted as segmental plane.
8: **end if**

nodes into four categories: boundary nodes, isolated volume nodes, isolated segmental nodes and general nodes.

Flooding. Choosing a certain number of general nodes which belong to different polyhedron and keeping flooding for each piece. Recall that the number of concave area is known which has been determined in the very first phase of convex segmentation and the optimized phase.

It is assumed that each node will have a flag to represent its attribute after the isolated node identification. First of all, we randomly select one general node and denote it an unique clustering number. The node selected then flood the cluster number to its one-hop neighbors. These neighbors classify themselves as the same clustering number and keep flooding to their neighbors again. This flooding process of a single original node is terminated until all connected neighbors(which has to be general nodes) are denoted as the same number. Another original node is then selected in the remaining general nodes and keeps flooding as the same process. In this end, all general nodes are classified as general nodes which know their clustering number.

Extending. Segmental volume nodes are still isolated nodes and they need to be classified to different parts as well. For this nodes, we make those classified general nodes extending their boundaries. For a certain polyhedron P , if a general node, p, is close to P's boundary , it recovers its relationship with volume nodes and extends one-hop to make those one-hop neighbors as the same cluster with p. After that, we move to another polyhedron to do the same extension. The process of extension terminates until all volume nodes are classified into different polyhedrons.

In detail, algorithm of recognition can be described as algorithm 3.

C. Distributed Map Establishment

After network segmentation phase, the whole 3D network is divided into several subnetworks. Each subnetwork is regarded as a cluster. We can obtain the relative coordinates of nodes in

Algorithm 3: Partition Recognition

Require:
All nodes and their attributes.
A neighbor matrix *A* describes the connectivity relationship between all nodes.
The optimized number N_o of polyhedrons.

Ensure:
All nodes with a specific cluster number.

for each isolated segmental nodes i **do**
  DO
  Search all neighbors n whose distances is less than *k* steps from a certain node *i*.
  IBN $\leftarrow$ IBN $\bigcup$ n;
  A(n,:)=0,A(:,n)=0;
end for
for each optimized polyhedron j **do**
  randomly choose a node n_j and denote the cluster number j;
  create a new queue Q;
  Q.push(the node n_j);
  while !Q.empty **do**
    n_{j1}=Q.pop;
    Q.push(all first neighbors of the node n_{j1});
    denote the nodes n_{j1} as cluster j;
  end while
end for

each cluster using the MDS-MAP algorithm, with the details described as below.

We assume in each subnetwork, there exists a localization coordinator(could be arbitrary node), and the coordinator is in charge of the process of local relative map establishment. ni is denoted as the number of sensor nodes in each subnetwork.

In each subnetwork, the shortest hop count between two arbitrary adjacent nodes in the whole network can be obtained by the Floyd algorithm, and all of these hop count distances is saved in a distance matrix named by $neibMatrix$. Based on the $neibMatrix$ and MDS-MAP algorithm, the relative coordinates of each cluster can be calculated. The MDS-MAP algorithm deployed consists of 4 steps:

1) Step 1 The shortest hop counts between all pairs of nodes in the whole network can be computed by Floyd algorithm, and it can construct the $n \times n$ distance matrix(n nodes in the network) $neibMatrix$. Here, the time complexity of this algorithm is $O(n_i^3)$, where n_i is the number of nodes for a subnetwork.
2) Step 2 Compute the square of all elements in the matrix $neibMatrix$ and obtain the square matrix $neibMatrix^2$. Then compute the inner product matrix B of $neibMatrix^2$, $B = -(1/2)EneibMatrix^2E$, here E represents nth identity matrix.
3) Step 3 MDS applies SVD(singular value decomposition) on Matrix B to extract all eigenvalues and their corresponding eigenvectors of matrix B.
4) Step 4 Take the first three largest eigenvalues to construct the relative coordinates of this cluster and the relative map of this cluster is established. Note that in MDS-based localization scheme, only the first three largest eigenvalues are used for 3D coordinates. Thus we can use the power method of a matrix only 3 times to obtain these three eigenvalues and eigenvectors, instead of all eigenvalues and eigenvectors. The power method (also known as the power iteration) of a matrix B is designed for extracting the dominant eigenvalue (i.e., the first eigenvalue with the largest magnitude) and the corresponding eigenvector.

For each subnetwork, it will execute MDS-MAP algorithm to localize those nodes. The procedure of local map establishment is fully distributed for each subnetwork.

D. Global Map Establishment

After obtaining the relative map of each cluster, we need to convert the distributed map into a global map. The global map is established from merging these different clusters. The merging method is to find two adjacent clusters which have the most common nodes and then merge them into one cluster (This is discussed in our previous work [9]). When all clusters are merged into one cluster , the merging process stops and the global map establishment is completed. At the end, all the absolute coordinates of the unknown nodes have been estimated. In our algorithm, we choose 3D Coordinate Transformation Algorithm as the practice of merging two adjacent clusters. The merging process is shown as follow.

Suppose two adjacent clusters are cluster M and cluster N, and node p is a common node of cluster M and cluster N. The 3D relative coordinates of P are (x_M, y_M, z_M) in cluster M, while in cluster B is (x_N, y_N, z_N). We merge the two coordinates based on the following formula.

$$\begin{bmatrix} x_M \\ y_M \\ z_M \\ 1 \end{bmatrix} = M_{NM} \begin{bmatrix} x_N \\ y_N \\ z_N \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ O & 1 \end{bmatrix} \begin{bmatrix} x_N \\ y_N \\ z_N \\ 1 \end{bmatrix} \quad (6)$$

M_{NM} is the coordinate transformation matrix from cluster N to cluster M. there are 12 variables, which expressed the rotation matrix R and translational matrix $T = [T_x, T_y, T_z]^T$, the rotation matrix $R = \begin{bmatrix} R_1 & R_2 & R_3 \\ R_4 & R_5 & R_6 \\ R_7 & R_8 & R_9 \end{bmatrix}$ should meet the orthogonal condition:

$$\begin{cases} R_1^2 + R_4^2 + R_7^2 = 1 \\ R_2^2 + R_5^2 + R_8^2 = 1 \\ R_3^2 + R_6^2 + R_9^2 = 1 \\ R_1R_2 + R_4R_5 + R_7R_8 = 0 \\ R_1R_3 + R_4R_6 + R_7R_9 = 0 \\ R_2R_3 + R_5R_6 + R_8R_9 = 0 \end{cases}$$

Then the translation formula can be described more specifically,

$$\begin{bmatrix} x_M \\ y_M \\ z_M \\ 1 \end{bmatrix} = \begin{bmatrix} R_1 & R_2 & R_3 & T_x \\ R_4 & R_5 & R_6 & T_y \\ R_7 & R_8 & R_9 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_N \\ y_N \\ z_N \\ 1 \end{bmatrix} \tag{7}$$

Then put n pairs of common nodes coordinates of cluster M and cluster N (x_{Mi}, y_{Mi}, z_{Mi}), (x_{Ni}, y_{Ni}, z_{Ni}) (i = 1, 2, ..., n) into (3), we can get (4) as follows:

$$\begin{aligned} &\begin{bmatrix} x_{M1}, y_{M1}, z_{M1}, x_{M2}, y_{M2}, z_{M2}, \ldots, x_{Mn}, y_{Mn}, z_{Mn} \end{bmatrix}^T \\ &= \\ &\begin{bmatrix} x_{N1} & y_{N1} & z_{N1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{N1} & y_{N1} & z_{N1} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{N1} & y_{N1} & z_{N1} & 1 \\ x_{N2} & y_{N2} & z_{N2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{N2} & y_{N2} & z_{N2} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{N2} & y_{N2} & z_{N2} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{Nn} & y_{Nn} & z_{Nn} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{Nn} & y_{Nn} & z_{Nn} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{Nn} & y_{Nn} & z_{Nn} & 1 \end{bmatrix} \\ &\times \begin{bmatrix} R_1, R_2, R_3, T_x, R_4, R_5, R_6, T_y, R_7, R_8, R_9, T_z \end{bmatrix}^T \end{aligned} \tag{8}$$

Having the equation solved, we can get 12 parameters of transformation matrix. We can then figure out the merging coordinates of cluster M and cluster N based on the parameters derived. Repeat the merging step until there is only one cluster which contains all the relative nodes coordinates of all clusters. In these steps, the time complexity is $O(n_i^3)$, where n_i is the number of nodes in a subnetwork. Based on the anchor nodes we deployed, we can calculate the absolute coordinates of all nodes based on the above 3D Coordinate Transformation Algorithm, and the global map can be established.

E. Time complexity

The time consuming of the algorithms is relative to the computational complexity. The whole framework consist three steps: 3D segmentation, Distributed map establishment and Global map establishment. Given n sensor nodes in a 3D WSN and the network has been partitioned into k subnetworks.

For 3D segmentation, it mainly includes four parts: concave node recognition, isolating segmental nodesoptimizing polyhedron and partition recognition. For concave node recognition, we only deal with the boundary nodes which could be seen as a relative small portion of the whole network. In segmental node isolation part, we only focus on all segmental nodes and find neighbors within limited steps. Thus, the time complexity is related to the number of segmental nodes and boundary nodes which are very small when compared with the number of all nodes in WSN. For optimizing polyhedron and partition recognition, the time complexity is related to the number of nodes belong to one cluster. Generally, all nodes in WSN will be visited and get its cluster number only once. Therefore, the time complexity in algorithm 3 is pretty small.

Distributed Map Establishment phase definitely is the most time-consuming part of the proposed DisLoc algorithm. The time complexity of identifying the coordinates in different partition is $O(kn_i^3)$, where k represents the number of partitions and n_i represents the number of sensor node in each partition.

The merging phase has the time complexity of $O(n)$. Therefore, the time complexity of DisLoc is $O(kn_i^3)$, which decreases the time complexity significantly compared with existing techniques such as 3D-MDS and 3D-DVHOP.

IV. Simulation

A. Simulation Setup

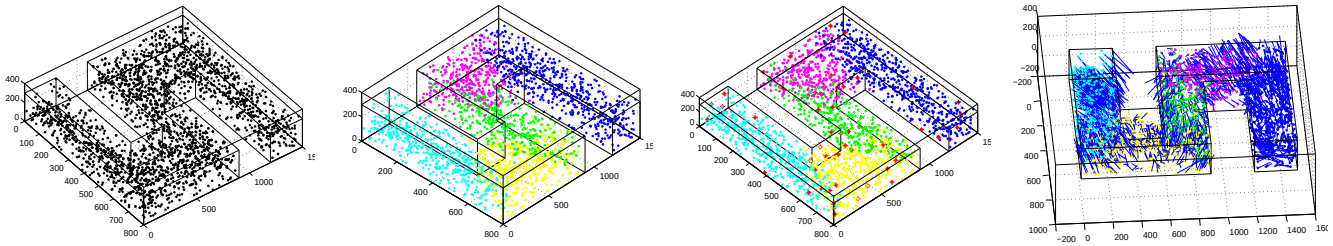
We evaluate proposed DisLoc algorithm on four representative topologies of real-world applications: the 5-shape topology as a 5-shaped coal mine tunnel, the Fei-shape topology as Hangzhou East Express railway station, the H-shape topology as an H-Shape terminal building, and the C-shape topology as an ordinary building entrance. The simulated topologies are shown in Fig.10 to Fig.13, respectively. For simplicity, we use type5, typeF, typeH, typeC to refer to the four different network topologies in the rest of the paper.

For all these topologies above, the sensor nodes are uniformly distributed in networks. All the sensor nodes have the same communication range by default, denoted by R. Each pair of nodes are connected if the Euclidean distance between them is no greater than R. The Logarithmic Attenuation Model is selected as the communication model to simulate the real-time communications among sensors.

The performance of proposed algorithm is assessed by the average localization error which is defined as follows. Let d denote the distance between two neighboring nodes computed based on the established coordinates, and let $\hat{d}$ denote the ground-truth distance between the two neighboring nodes. The average location error is defined as $\left|d - \hat{d}\right|/R$.

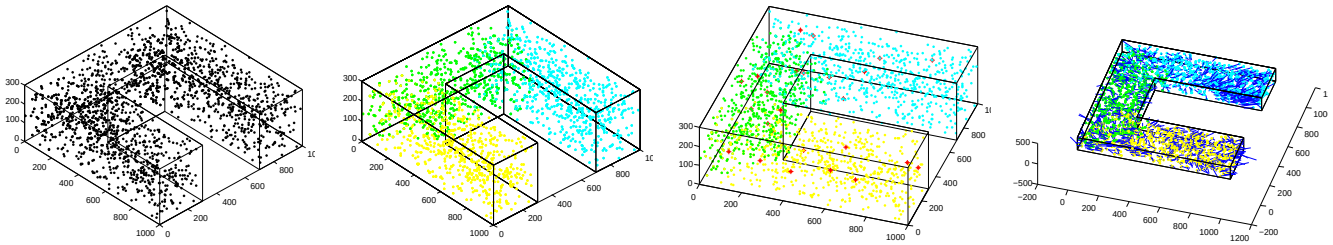
Fig. 10 to Fig. 13 show the localization results under different topologies, respectively, including the network segmentation results, anchor nodes deployment and average localization error. There are 2249 sensor nodes randomly deployed in 5-shape topology region shown in Fig. 10(a). 1799 sensor nodes are deployed in typeC topology shown in Fig. 11(a). 3198 sensor nodes are deployed in Fei-sharp topology region shown in Fig. 12(a), and 3994 sensor nodes are deployed in H-sharp topology region as shown in Fig. 13(a).

As displayed in Fig. 10 to Fig. 13, the first column chart shows the network topology deployment. The second colunm shows the results of network segmentation. The third column shows the deployment of anchor nodes colored by red deployment, and the last column shows the localization error. To make the localization results easier to read, both the true location and estimated location of a node are plotted in Fig. 10(d), Fig. 11(d), Fig. 12(d) and Fig. 13(d), respectively. A light-blue line which connects these two locations, represents the sheer localization error. The longer line apparently represents the larger error.



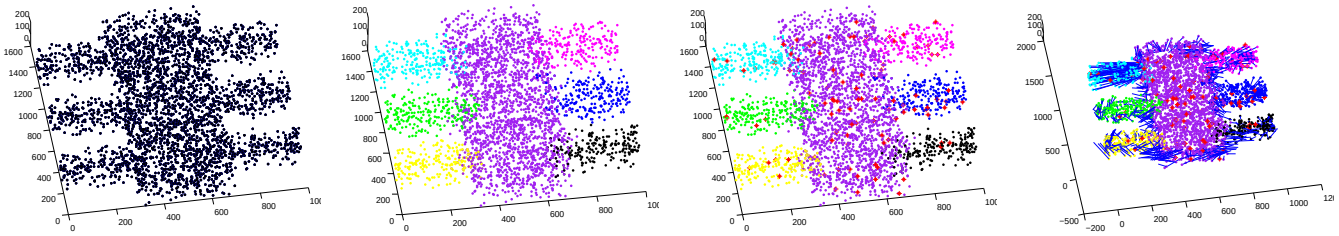
(a) Network topology (b) Segmentation results (c) partition recognition and anchor nodes deployment (d) Localization results

Fig. 10. Simulation Results for 5-shape



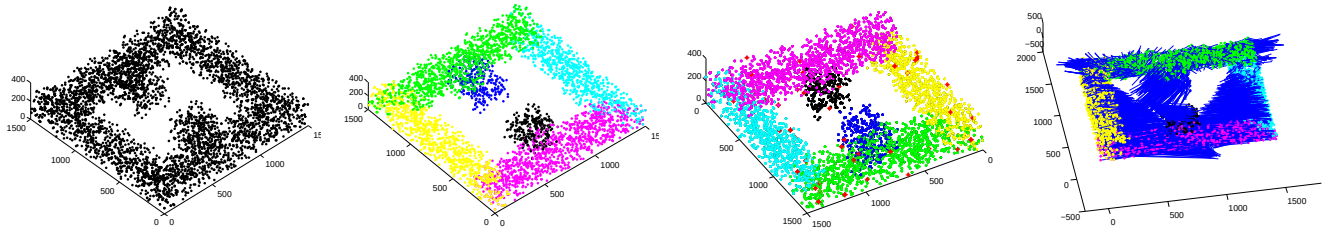
(a) Network topology (b) Segmentation results (c) Partition recognition and anchor nodes deployment (d) Localization results

Fig. 11. Simulation Results for C-shape



(a) Network topology (b) Segmentation results (c) Partition recognition and anchor nodes deployment (d) Localization results

Fig. 12. Simulation Results for Fei-shape



(a) Network topology (b) Segmentation results (c) Partition recognition and anchor nodes deployment (d) Localization results

Fig. 13. Simulation Results for H-shape

As shown in Fig. 10(b), Fig. 11(b), Fig. 12(b) and Fig. 13(b), respectively, the proposed algorithm could successfully decompose 3D networks and achieve an acceptable number of partitions. Given a reasonable number of anchor nodes deployed, the respective localization error is displayed in Fig. 10(d), Fig. 11(d), Fig. 12(d) and Fig. 13(d), respectively. It is apparent that DisLoc performs better in terms of average localization error when the topology is relative simple (the number of partitions is relatively small). It can be explained as that the error could be accumulated during the process of merging subnetworks, especially when the network has to be segmented into a good many ones.

We then compare the algorithm performance with previous work D3D-MDS in terms of accuracy. We choose C shape topology as the experimental environment. For comparison, localization error is observed under different network connectivity. The comparison result is shown in the following Fig. 14. As can be obviously seen from the simulation results of Fig. 14, the localization error decreases when the network connectivity increases. It can be seen that the proposed algorithm can achieve better performance than the D3D-MDS algorithm. For the same network connectivity, the localization error is smaller when the proposed algorithm is applied in the same WSNs environment. For example, when the network connectivity is 14, the localization error of the proposed algorithm is about 59%, while the localization error of D3D-MDS is about 99.65%.

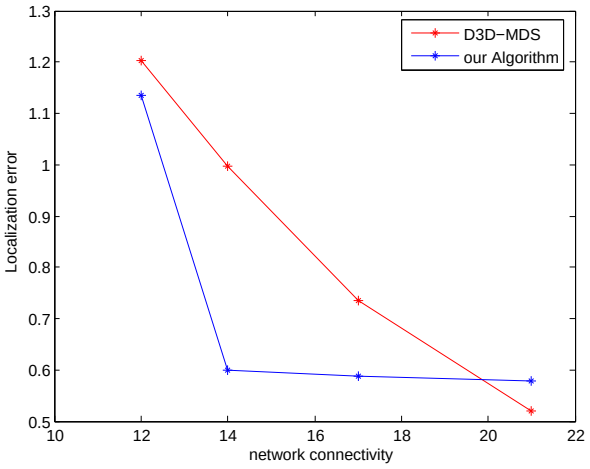


Fig. 14. Comparison result with D3D algorithm

B. performance evaluation

Through intensive experiments, we realize that the number of common nodes in III-D and anchor nodes deployments both have great influence on the localization results.

From the previous discussion of network segmentation, it is known that common nodes are extremely crucial to the network segmentation and the following merge. It relates to the volume area mentioned in III-B4. In this section, the number of common nodes is tuned in Fig. 15 with respective algorithm accuracy. Anchor nodes deployment will significantly affect the localization result as well. Fig. 15 shows the performance variance with different anchor node deployments.

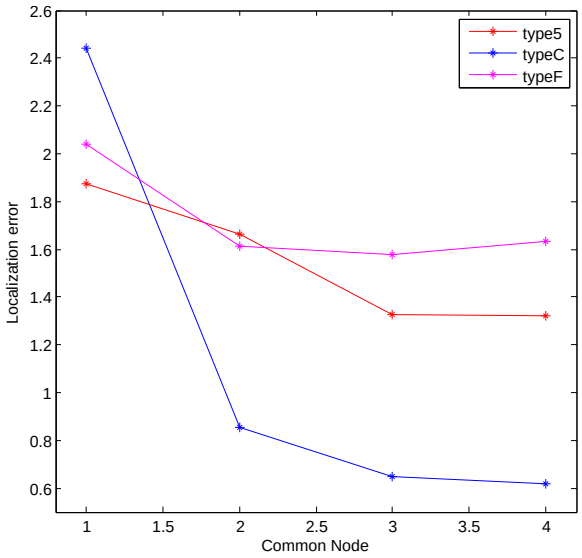


Fig. 15. The impact of common nodes

1) Impact of common nodes: In Fig. 15, the number of common nodes is based on the common nodes set c, which includes the initiative set and its neighbors. $c = 1$, represents that the common node set is the initiative set. When common node value c is set to 2, 3 or 4, it indicates that the common node set including the initiative set and its one, two or three hop neighbors, respectively. It can be seen from Fig 15 that the localization error is relatively low when the common node set is the initiative set and its two hop neighbors. This conclusion can help merge two adjacent subsection.

2) Impact of anchor nodes: In general, the localization error decreases as the number of anchor nodes increases within the limits. To observe the impact of anchor nodes, we change the number of anchor nodes to investigate its impact on algorithm performance in terms of localization accuracy. It can be seen from Fig. 16 that the variance of localization error is quite consistent when the number of anchor nodes is more than 20. It shows that the localization can be realized only with a very small proportion of anchor nodes. For example, when the number of anchor nodes is 20, this number implies that limited anchor node ratio was adopted for presented four simulation scenarios.(i.e. for 5-sharp, the ratio of anchor nodes is 0.89%, 0.63% in Fei-sharp, 0.5% in H-sharp and 1.1% in C-sharp). It can be explained by MDS algorithm analysisthat theoretically, only four anchor nodes are required for a centralized localization algorithm . Due to the complexity brought by the 3D sensor network, we achieve reasonable results by deploying a small portion of anchor nodes.

V. Conclusion

This paper has developed DisLoc, a novel distributed localization algorithm, for a 3D WSN when the network is in a irregular shape. DisLoc has addressed a series of challenges

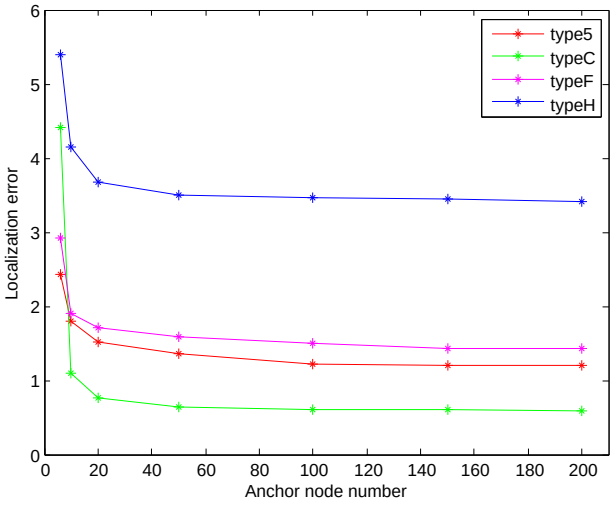


Fig. 16. The impact of anchor nodes

using fully distributed algorithms including spatial convex node identification, network segmentation, local localization and global map reconstruction. DisLoc works in a fashion that is distributed, low message and computation overhead. Using extensive simulations, it has been shown that DisLoc can achieve high efficiency and accuracy as compared with existing proposals.

In future, we plan to set up a real-world 3d sensor network to assess the performance of DisLoc, and apply DisLoc to the advanced service applications in sensor networks.

ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China under Grant Nos. 61401135.

REFERENCES

[1] S. Lederer, Y. Wang, and J. Gao, "Connectivity-based localization of large-scale sensor networks with complex shape," *ACM Transactions on Sensor Networks*, vol. 5, no. 4, p. 31, 2009.

[2] M. Li and Y. Liu, "Rendered path: range-free localization in anisotropic sensor networks with holes," *Networking IEEE/ACM Transactions on*, vol. 18, no. 1, pp. 51–62, 2010.

[3] W. Liu, D. Wang, H. Jiang, W. Liu, and C. Wang, "Approximate convex decomposition based localization in wireless sensor networks," in *IEEE Infocom*. IEEE, 2012, pp. 1853–1861.

[4] S. Zhang, G. Tan, H. Jiang, B. Li, and C. Wang, "On the utility of concave nodes in geometric processing of large-scale sensor networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 1, pp. 132–143, 2014.

[5] M. Jin, S. Xia, H. Wu, and X. Gu, "Scalable and fully distributed localization with mere connectivity," in *IEEE Infocom*. IEEE, 2011, pp. 3164–3172.

[6] G. Tan, H. Jiang, S. Zhang, Z. Yin, and A.-M. Kermarrec, "Connectivity-based and anchor-free localization in large-scale 2d/3d sensor networks," *ACM Transactions on Sensor Networks*, vol. 10, no. 1, p. 6, 2013.

[7] D. Niculescu and B. Nath, "Ad hoc positioning system (aps)," in *IEEE Globecom*, vol. 5. IEEE, 2001, pp. 2926–2931.

[8] Y. Shang and W. Ruml, "Improved mds-based localization," in *IEEE Infocom*, vol. 4. IEEE, 2004, pp. 2640–2651.

[9] J. Fan, B. Zhang, and G. Dai, "D3d-mds: a distributed 3d localization scheme for an irregular wireless sensor network using multidimensional scaling," *International Journal of Distributed Sensor Networks*, 2015.

[10] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, 2000.

[11] W. Xi, Y. He, Y. Liu, J. Zhao, L. Mo, Z. Yang, J. Wang, and X. Li, "Locating sensors in the wild: pursuit of ranging quality," in *International Conference on Embedded Networked Sensor Systems, SENSYS 2010, Zurich, Switzerland, November*, 2010, pp. 295–308.

[12] W. Liu, D. Wang, H. Jiang, W. Liu, and C. Wang, "An approximate convex decomposition protocol for wireless sensor network localization in arbitrary-shaped fields," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3264–3274, 2015.

[13] J.-M. Lien and N. M. Amato, "Approximate convex decomposition of polyhedra," in *Proceedings of the 2007 ACM symposium on Solid and physical modeling*. ACM, 2007, pp. 121–131.

[14] X. Zhu, R. Sarkar, and J. Gao, "Shape segmentation and applications in sensor networks," in *IEEE Infocom*. IEEE, 2007, pp. 1838–1846.

[15] M. Ghosh, N. M. Amato, Y. Lu, and J.-M. Lien, "Fast approximate convex decomposition using relative concavity," *Computer-Aided Design*, vol. 45, no. 2, pp. 494–504, 2013.

[16] P. K. Agarwal, E. Flato, and D. Halperin, "Polygon decomposition for efficient construction of minkowski sums," in *European Symposium on Algorithms*. Springer, 2000, pp. 20–31.

[17] B. Chazelle and D. P. Dobkin, "Optimal convex decompositions," *Computational Geometry*, vol. 4, no. 5, pp. 63–133, 1985.

[18] J.-M. Lien and N. M. Amato, "Approximate convex decomposition of polygons," *Computational Geometry*, vol. 35, no. 1-2, pp. 100–123, 2006.

[19] M. Ghosh, N. M. Amato, Y. Lu, and J.-M. Lien, "Fast approximate convex decomposition using relative concavity," *Computer-Aided Design*, vol. 45, no. 2, pp. 494–504, 2013.

[20] L. Wan, "Parts-based 2d shape decomposition by convex hull," in *Shape Modeling and Applications, 2009. SMI 2009. IEEE International Conference on*. IEEE, 2009, pp. 89–95.

[21] H. Liu, W. Liu, and L. J. Latecki, "Convex shape decomposition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 97–104.

[22] Z. Ren, J. Yuan, C. Li, and W. Liu, "Minimum near-convex decomposition for robust shape representation," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 303–310.

[23] D. Dong, Y. Liu, and X. Liao, "Fine-grained boundary recognition in wireless ad hoc and sensor networks by topological methods," in *ACM MobiHoc*. ACM, 2009, pp. 135–144.

[24] O. Saukh, R. Sauter, M. Gauger, and P. J. Marrón, "On boundary recognition without location information in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 6, no. 3, p. 20, 2010.

[25] Y. Wang, J. Gao, and J. S. Mitchell, "Boundary recognition in sensor networks by topological methods," in *ACM MobiCom*. ACM, 2006, pp. 122–133.

[26] G. Tan, H. Jiang, J. Liu, and A.-M. Kermarrec, "Convex partitioning of large-scale sensor networks in complex fields: Algorithms and applications," *ACM Transactions on Sensor Networks*, vol. 10, no. 3, p. 41, 2014.