A Unified Framework for Constrained Linearization of Sensor Networks with Arbitrary Shapes

YUFU JIA¹, WENPING LIU¹, (Senior Member, IEEE), GUOYIN JIANG², (Member, IEEE), HONGBO JIANG³, (Senior Member, IEEE), YAMIN LI⁴, (Member, IEEE), YANG YANG⁴, (Member, IEEE), JING XING¹, AND ZHICHENG LUI⁵

¹School of Information Management and Statistics, Hubei University of Economics, Wuhan, Hubei 430205, P.R. China.
²School of Public Administration, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, P.R. China.
³College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, P. R. China.
⁴School of Computer Science and Information Engineering, Hubei University, Wuhan, Hubei 430077, P.R. China.
⁵School of Information Engineering, Hubei University of Economics, Wuhan, Hubei 430205, P.R. China.

Corresponding author: Wenping Liu and Hongbo Jiang (e-mail: wenpingliu2009@gmail.com, hongbojiang2004@gmail.com).

An earlier version appeared in the proceedings of ACM TURC 2019 [9]. This work was supported by the National Natural Science Foundation of China under Grants 61672213, 61732017, and 71671060; by the Excellent Young and Middle-aged Scientific and Technological Innovation Team for Universities in Hubei Province under Grant T201714; by the Science and Technology Plan Projects of Wuhan City under Grant 2017050304010325. Part of this work is done when Dr. Yufu Jia visited Prof. Hongbo Jiang.

ABSTRACT
In such application scenarios of sensor networks as motion planning of a mobile robot for data collection or battery recharging, the robot is often required to start from a pre-required location (referred to as Entrance) and quit at another pre-required location (referred to as Exit) before visiting all sensors. Existing solutions can only compute a traversal path (i.e., a space-filling curve, SFC) linking all sensors for either 2D sensor networks or 3D surface/volume networks, where the Entrance and the Exit cannot be specified beforehand. As such, in this paper, we study the constrained linearization problem of sensor networks, i.e., computing a path (referred to as constrained space-filling curve, CSFC) traversing all sensors, starting from the Entrance and ending at the Exit. Motivated by the generation of SFCs in Fractal Geometry, we propose a unified framework, which is novel, simple yet efficient, for the constrained linearization of 2D sensor networks or 3D surface/volume sensor networks merely using connectivity information in an iterative fashion. Specifically, we first compute a shortest path between the Entrance and the Exit via in-network flooding to initialize the CSFC; then, during each round, we simultaneously deform the edges (i.e., replacing each edge with a zig-zag pattern) on the CSFC until no edges can be deformed, and a coarse CSFC, possibly missing some sensors (e.g., due to network sparsity or irregularity), is thus derived. We finally propose to connect the unvisited sensors into the coarse CSFC, and the network is linearized such that all nodes are orderly traversed by the CSFC from the Entrance to the Exit. Extensive simulations show that: 1) our algorithm can efficiently compute the CSFC for 2D sensor networks and 3D surface/volume sensor networks in terms of once-visited node number and time/message complexity, etc., 2) our algorithm is robust to many factors such as network shape, density, scale and communication radio model, etc., and 3) our approach outperforms the state-of-the-art SURF [19], a Space filling cURve computing algorithm for high genus 3D surface sensor networks, where the constraint on the Entrance and the Exit is not considered.

INDEX TERMS Sensor networks, space-filling curve, constrained linearization.
and the Koch curve in Figure 1 (c), etc. These curves are often constructed in an iterative fashion via continuously replacing each segment with a zig-zag (i.e., scaled and rotated) pattern, and the more recursion involved, the denser the curve will be; if the recursion tends to be infinite, each point in the region will be on the curve (i.e., the curve can be arbitrary close to each point).

The linearization of sensor networks computes a single path, the discrete counterpart of the SFC in a continuous domain, “covering” the whole network (i.e., traversing all sensors). Thanks to the capability of visiting various regions and the locality property (e.g., two close points on a 2D plane are also nearby in the SFC), the network linearization has great potential of benefiting many applications [2], such as motion planning of mobile agents for data collection (also termed as data mules) [11], [18], [24] or battery recharging [13], [22], serial/linear data fusion [14], [15], [17], spatial indexing [1], [16], just to name a few.

**Related work.** Recently, a flourish of research efforts have been made for SFC computation and the applications from wireless community. Below we briefly introduce some representative work.

1) **SFC computation.** Ban *et al.* [2] propose to compute an aperiodic dense curve of 2D sensor networks with holes by first mapping all holes but one to “slits”, and then following the line bouncing back and forth between the inner boundary of the only hole and the outer boundary. Yan *et al.* [24] propose to linearize clustered 2D sensor networks and design path planning strategies for robotic data collection, and the simulations on a square area show that it outperforms the serial depth-first traversal scheme and the tree-based aggregation scheme in terms of message cost and detection accuracy. Goswami *et al.* [5] exploit the Hodge decomposition theorem and holomorphic differentials on Riemann surfaces, and propose to conformally map high-genus 3D surface networks to a union of flat tori and then generate a dense SFC on this union. Wang *et al.* [19], [20] propose to construct a Reeb graph and decompose the network into multiple regions, and then design a serial traversal scheme across the regions to generate an SFC of the 3D surface network. BLOW-UP [21] is an SFC computation algorithm designed for 3D volumetric sensor networks based on the unit tetrahedron cell (UTC) structures. It decomposes the network into closed layers such that the sensors can be traversed from the innermost layer to the outmost layer.

2) **SFC application.** Ban *et al.* [2] exploit the computed aperiodic dense curve (i.e., the curve traverses each sensor node at most a constant number of times) of 2D sensor networks for multiple data mules coordination and double ruling based in-network data storage and retrieval. Xie *et al.* [22] show that the shortest Hamiltonian cycle (i.e., a closed Hamiltonian path visiting each sensor) is the optimal traveling path of the wireless charging vehicle, and thus propose a novel energy renewal scheme for sensor networks. In [15], a space-filling curve-based routing mechanism for serial fusion in 2D sensor networks is developed to compute a routing path traversing all sensors. Kamat *et al.* [10] investigate the modification of the Hilbert curve for the coverage problem of Ellipsoidal 2D sensor networks.

**Our contributions.** From the above-mentioned applications of SFC in sensor networks, we find that practically a mobile robot is often required to start and return at the specified location(s) [7], [12] (e.g., the base station [23] of a data mule or the service station [22] of a mobile wireless charging vehicle (WCV)). For instance, in energy-renewal sensor networks, a mobile WCV with sufficient energy will start from a service station, and after charging every sensor with wireless power transfer technology, it returns to the service station for replacing or recharging battery [22]. This leads to a new, and different, problem of SFC compute-
a mobile robot may spend as long as 4 hours to visit all of these nodes \cite{23}.

The Koch curve generation as illustrated in Figure 1 (c): it is constructed iteratively by dividing a line segment into three equal-length segments, generating an equilateral triangle pointing outward by using the middle segment, and replacing the middle segment with two other segments (a zigzag pattern) of the equilateral triangle.

In this paper we make a slight modification since in sensor networks only two nodes within the communication range form an edge. Specifically, we first construct a path (e.g., the shortest path $SP(S, E)$) between pairwise Entrance $S$ and Exit $E$. This path initializes the constrained space-filling curve, denoted by $CSFC(S, E)$, as shown in Figure 1 (d) (f). Then, we deform each edge, say $\overline{p_1p_2}$, on $CSFC(S, E)$ by finding an intermediate node (i.e., a common neighbor of $p_1$ and $p_2$, say $p_{12}$), and replacing $\overline{p_1p_2}$ with a zig-zag pattern, which we call as the constrained linearization problem (CLP), where the SFC is subject to the constraint on a pre-required start location (referred to as Entrance) and a pre-required end location (referred to as Exit).

Occasionally, multiple pairs of Entrance-Exit and/or multiple mobile robots might be introduced to improve the performance (e.g., save the travel time of a robot) of the SFC-based application protocol. For example, when the energy of the WCV is insufficient, there might be multiple service stations (and thus multiple pairs of Entrance-Exit) deployed in the network; for a large-scale network, by allowing multiple robots to travel simultaneously from and end at different service locations we can significantly reduce the whole travel time.\footnote{For a network with only 200 nodes randomly deployed in a sensing field, a mobile robot may spend as long as 4 hours to visit all of these nodes \cite{23}.}

We call the SFC computation in this situation as the constrained linearization problem with the constraint on multiple pairs of Entrances and Exits, denoted by CLP$(k)$ where $k$ represents the number of Entrance-Exit pairs.

Previous solutions on SFC computation usually incur high time complexity (and thus has poor scalability), are designed either for 2D or 3D surface networks, and cannot be readily used for linearizing a sensor network with a pair of pre-required Entrance and Exit. For example, both \cite{2} and \cite{25} find a cut graph of a surface network and then map the underlying network into a 2D plane controlled by a convergence threshold, say $\delta$, where the cut graph serves as the boundary of the plane. However, there exists a tradeoff between the $\delta$, which consequently affects the mapping result, and the computational time $t$; to derive an acceptable result, the computational time will be as long as more than 10 hours. Please see Figure 2 as an example. In addition, in \cite{2}, only the starting position can be predefined and the destination cannot be controlled beforehand.

As such, in this paper we are interested in the constrained linearization problem for more general scenarios where sensors can be deployed in 2D terrains or 3D (surface) spaces, and propose a unified framework for the constrained linearization of 2D, 3D surface or 3D volume sensor networks with arbitrary shapes. Our work is motivated by the Koch curve generation as illustrated in Figure 1 (c): it is constructed iteratively by dividing a line segment into

\begin{figure}[h]
    \centering
    (a) \hspace{2cm} (b) \hspace{2cm} (c) \hspace{2cm} (d)
    \caption{The effect of the threshold value $\delta$ for convergence on the computational time $t$ of mapping the double-torus shaped 3D surface network in Figure 1 to a rectangle. (a) the cut-graph (indicated by the red line); (b) $\delta = 0.01$, $t = 1.5$ minutes; (b) $\delta = 0.001$, $t = 12.3$ minutes; (c) $\delta = 0.0001$, $t = 12$ hours.}
\end{figure}
without considering the constraint on the Entrance and Exit, showing that our algorithm outperforms SURF in terms of time complexity, the length of SFC, etc.

The rest of the paper is organized as follows. We present the problem formulation in Section II, and Section III is devoted to the detailed design of our algorithm. We conduct some discussions in Section IV, and evaluate the performance in Section V. Finally, Section VI concludes the paper.

II. PROBLEM FORMULATION, SOLUTIONS AND CHALLENGES

Given a network $V$ with $n$ sensors and a pair of Entrance-Exit (S, E), our goal is to find a path traversing all sensor nodes (or nodes for short thereafter) with the constraint of the start node (i.e., a node nearest to the start location) being S and end node (i.e., a node nearest to the destination) being E. In addition, to save the cost of a robot along the path, the path should be as short (in terms of Euclidean distance from the Entrance to the Exit) as possible. We formulate this problem as the constrained linearization problem (CLP):

$$\min \text{Length}(Path(v_0, v_1, \cdots , v_n))$$

$$s.t. \begin{cases} \text{card}\{V \cap \{v_0, v_1, \cdots , v_n\}\} = \text{card}\{V\} = n, \\
v_0 = S, v_n = E. \end{cases}$$

where card(·) denotes the cardinality of a set, and $Path(v_0, v_1, \cdots , v_n)$ is the path with the ordered node sequence of $\{v_0, v_1, \cdots , v_n\}$. Note that the constraint $\text{card}\{V \cap \{v_0, v_1, \cdots , v_n\}\} = n$ indicates that all nodes are on the path $Path(v_0, v_1, \cdots , v_n)$ (e.g., the SFC), which differs from the NP-complete traveling salesman problem (TSP) asking for the shortest tour from an end node to another. However, SFC has the potential for approximating the shortest tour in TSP, with shorter computational time yet the longer tour [4].

Note that when there are $k(>1)$ pairs of Entrances and Exits, say $(S^1, E^1), (S^2, E^2), \cdots , (S^k, E^k)$, for multiple robots or destinations of a robot, the constrained linearization problem for computing the SFC with the constraint on these pairs of Entrances and Exits, denoted by CLP(k), can be similarly formulated as the following:

$$\min \text{Length}(Path(v^1_0, v^1_1, \cdots , v^1_{n_1}, \cdots , v^k_0, \cdots , v^k_{n_k}))$$

$$s.t. \begin{cases} \text{card}\{V \cap \{v^1_0, v^1_1, \cdots , v^1_{n_1}, \cdots , v^k_0, \cdots , v^k_{n_k}\}\} = n, \\
v^1_0 = S^1, v^1_{n_1} = E^1, \cdots , v^k_0 = S^k, v^k_{n_k} = E^k. \end{cases}$$

Clearly the CLP is also NP-complete, and CLP(k) can be solved separately. As such, in what follows, of our interest is to solve the CLP, and we propose a heuristic approach to compute the SFC (the so-called constrained SFC, CSFC) of sensor networks with the constraint on the pre-required Entrance-Exit pair. Note that previous study on SFC computation in wireless networks cannot handle the CLP since, they do not consider the constraint of Entrance-Exit and the length of the SFC will be very large if the SFC is forced to start at $S$ and end at $E$. In addition, they often require complex operations, such as mapping [2] or Reeb graph construction [19], and thus incur high time complexity.

Our work is inspired from the generation of Koch curve, where during each round of iteration, each segment is divided into three segments of equal length, and then the middle segment is replaced with two other edges of the equilateral triangle generated by the middle segment with the direction pointing outwards, as shown in Figure 1 (c). We call such an operation as edge/segment deformation or edge/segment expansion. The two end points of the Koch curve, corresponding to a pair of Entrance and Exit in our constrained linearization problem of sensor networks, remain unchanged through all the rounds of iterations.

To design an algorithm of the constrained SFC computation tailored for sensor networks, we propose to make a modification on the expansion-based generation process of Koch curve such that it can be easily extended to discrete sensor networks with the round error of between-sensor distance (i.e., the distance between sensors is measured in hops, instead of other metrics such as Euclidean distance, etc.). Specifically, we first construct a shortest path between $S$ and $E$, and then replace each edge on the path with a zig-zag pattern (namely, a curved edge) via selecting an intermediate node. We call this operation as edge deformation or edge expansion. The deformation/expansion operation continues until no edges can be deformed, say, no intermediate nodes can be found. Due to the network sparsity (and thus the presence of small holes), the deformation process may miss some nodes such that they are not on the CSFC, and we thus propose to connect these isolated/unvisited nodes into the CSFC to form the final CSFC. Desirably, our approach yields a CSFC with the majority of nodes being visited once and thus the CSFC has a small length in terms of the hop count distance. On the other hand, due to the locality property of SFC (i.e., two nearby points are also close along the SFC), the derived CSFC is also short in terms of the Euclidean distance.

Let us take Figure 3 as an example. To visualize the expanding process, we limit our discussion on a 2D continuous domain. Initially, given a pair of Entrance and Exit (e.g., $S_0$ and $S_{10}$), we first compute a path between them at Stage 0. Note that this path is not necessary the shortest one. We then cut this path into $k (=10)$ segments of equal length with nine cut points, i.e., from $S_1$ to $S_9$ marked by the solid red circles. During each stage/iteration, for each segment (or edge) $S_iS_{i+1}(0 \leq i \leq k-1)$, we select an intermediate point $S_{i,i+1}$ such that $\text{Length}(S_iS_{i,i+1}S_{i+1}) = \text{Length}(S_{i+1}S_{i,i+1}) = \text{Length}(S_iS_{i+1})$. Clearly, for each edge, there are two candidate intermediate points on the opposite sides of the edge; we randomly select one of them, and derive two new edges, i.e., $S_iS_{i,i+1}$ and $S_{i+1}S_{i,i+1}$.

2For $k = 1$, we simply denote the constrained linearization problem as CLP.

3This is because many nodes will be visited more than once and thus the SFC has many overlapped segments.
At the same time, the edge $S_iS_{i+1}$ is no longer valid. For example, the intermediates at the first stage are marked by the solid black circles, the dashed black edges represent the newly generated edges after stage 1, and the connection between $S_i$ and $S_{i+1}$ is dropped out.

We repeat the above expansion process on the newly generated path, and the expansion operation terminates after eight stages when no more intermediates can be found. The edges without being shared by two triangles form an SFC. Obviously, if the edge length becomes smaller, the derived SFC covers more the underlying space; if the edge length tends to be infinite small, the SFC will fully cover the space. It is also noticed that during path expansion, we do not allow the edge to deform/expand toward the same direction as the Koch curve does, which otherwise can not yield a path covering the whole space. Another difference is that the edge length is constant for each iteration, as opposed to the decreasing edge length during the generation of the Koch curve. This design is well-tailored for the SFC computation in sensor networks where the hop-count distance between neighbors is always 1.

Specifically, to generate a CSFC of 2D or 3D surface/volume sensor networks with constraint on the pre-required Entrance and Exit, we first construct a (shortest) path between the Entrance and the Exit, and then simultaneously deform each edge on this path in an iterative way. That is, for each edge with two end points, say $p_1$ and $p_2$, we select the intermediate node $p_3$, a common neighbor of $p_1$ and $p_2$ with the smallest Node ID among the common neighbors, and replace the edge $p_1p_2$ with the edges $p_1p_3$ and $p_3p_2$. This expansion operation will derive a new path where each edge on the shortest path is replaced by two new edges via a series of intermediate nodes. By keeping deforming (or, expanding) edges on the newly generated path until there are no intermediate nodes, we are able to obtain a path (i.e., a constrained space-filling curve, CSFC) to linearize the whole network, while the end points of the CSFC are the required Entrance and Exit.

Even though the idea seems to be straightforward, there still exist many challenges for computing a CSFC traversing the whole sensor networks to guide the movement of a mobile robot. For example, due to the discrete nature of sensor networks and the irregularity of network topology (e.g., there might be many small holes), there is no guarantee that the expanded path will cover the whole network. For instance, assume $p_1,p_2,p_3,p_4$ form a small hole (e.g., every pair of sensors are neighboring except two pairs, i.e., $p_2,p_4$ and $p_1,p_3$), and the edge $p_3p_4$ is on the path to be expanded. For this case, both $p_3$ and $p_4$ will not be selected as intermediate nodes. As a result, the common neighbors of $p_3$ and $p_4$ may not be intermediate nodes as well. That is, the computed CSFC contains a majority of the nodes, but there may be some isolated nodes without being selected as intermediate nodes, which form some islands, and thus a refinement is needed to add these isolated nodes to the path. It is a nontrivial task with the consideration of minimizing the covered times of nodes 4, which is a standard goal for SFC computation in discrete sensor networks [2], [19]. In addition, owing to the presence of islands, it is challenging to assign the moving direction for the mobile robot along the CSFC with constraint on the Entrance and Exit.

III. ALGORITHM DESIGN

A. AN OVERVIEW

In this paper our aim is to linearize a 2D/3D sensor network with the constraint on a pair of Entrance and Exit, i.e., compute a constrained SFC (CSFC) traversing the whole network. Without loss of generality, we assume that the Entrance and Exit are not the same; for the motion planning problem of a mobile robot starting from and returning to the same location, e.g., corresponding to the service station in energy-renewal sensor networks, we can randomly select one node nearest to the service station as the Entrance, and one of its neighbors as the Exit.

Our approach includes four steps: CSFC initialization, coarse CSFC establishment, CSFC refinement and direction assignment of CSFC. The pseudocode can be found on Algorithm 1.

1) CSFC initialization. The Entrance $S$ issues an in-network flooding operation, and a tree rooted on $S$ will be created accordingly. As a result, a shortest path $SP(S,E)$ from $S$ to the Exit $E$ can be derived.

2) Coarse CSFC establishment. With the shortest path $SP(S,E)$ between $S$ and $E$, we simultaneously deform the edges on $SP(S,E)$ into “curved” edges via finding candidate intermediate nodes until no nodes join in the path; this process will generate a CSFC where not all nodes are on it (e.g., due to network sparsity or irregularity), and we thus refer to it as coarse CSFC.

3) CSFC refinement. The path expansion process may lead to some isolated nodes which form islands, due to

4That is, a sensor may not be visited once when the robot moving along the SFC to traverse the whole network.
the discrete nature of sensor networks and also there might be some small holes. As such, we propose to identify the isolated nodes and connect them into the coarse CSFC. Eventually, a CSFC traversing all nodes is generated while satisfying the constraint on the Entrance and Exit.

4) Direction assignment of CSFC. To guide the movement of a mobile robot, we assign the movement directions along the CSFC. This step is necessary as the final CSFC may contain some loops after the refining stage where the isolated sensors are connected into the coarse CSFC.

B. CSFC INITIALIZATION

Assume that a robot starts from an Entrance, say Node $S$, and after visiting all nodes, it finishes its trip at the Exit, say Node $E$. To construct a CSFC for the motion planning of the robot with the constraint on the Entrance-Exit pair, we let Node $S$ initiate a flooding within the network, and each node broadcasts this message if it has never received this message before, and discards it otherwise. This way a shortest path $SP(S, E)$ from $S$ to $E$ is generated, as show in Fig 1 (g), and each node $p$ on the shortest path keeps a record of its parent node; Node $S$ marks itself as the parent. Further, Node $E$ sends a message along the reverse direction until reaching Node $S$, and each node $p$ on $SP(S, E)$ marks itself by setting a boolean variable $bFlag$, which is false by default, to be true, and also each node keeps a record of the neighboring nodes on the path $SP(S, E)$ which initializes the CSFC. In addition, we assume that each node maintains a list of NodeID of neighbors via a local flooding.

C. COARSE CSFC ESTABLISHMENT

After CSFC initialization, we conduct the edge deformation (or expansion) operation for the CSFC in an iterative fashion. The key is to select an intermediate node, which is defined below, for each edge on the newly derived CSFC during each round such that a zig-zag pattern can be found.

Definition 1. A node is an intermediate node for edge expansion if its boolean variable $bFlag = false$ and it is a common neighbor of the end nodes of an edge on the CSFC.

To this end, we let the nodes on $SP(S, E)$ (i.e., with $bFlag = true$) simultaneously broadcast a message including the NodeID of itself and the neighbors with $bFlag = true$. When a node $p_3$ receives two such flooded messages, say from $p_1$ and $p_2$ such that $p_1$ is the parent of $p_2$ (i.e., $p_2.parent = p_1$), it conducts the following rules:

- If $bFlag(p_3) = true$, $p_3$ discards the message;
- else if $p_1$ and $p_2$ are not neighbors, $p_3$ holds the message until receiving a message from one neighbor of $p_1$ or $p_2$;
- else, $p_3$ is the common neighbor of $p_1$ and $p_2$, denoted by $CN(p_1, p_2)$, and it notifies $p_1$ and $p_2$ that it is a candidate for joining in the path $SP(S, E)$ to deform the edge $p_1p_2$.

Upon receiving a message from $p_3$, $p_1$ will wait a while for collecting more candidates and select the candidate with the smallest NodeID, say $p_{12}$, as its child. Meanwhile, it notifies $p_2$ to change its parent as $p_{12}$, and notifies $p_{12}$ to change its parent as $p_1$ and its child as $p_2$. This also shows that the edge expansion/deformation will start, which is controlled by the boolean variable $bPathExpanded$. Please see the pseudo code from Line 2 to Line 19 in Algorithm 1. As a result, the edge $p_1p_2$ is replaced with a zig-zag edge $p_{12}p_{12}p_{12}$, and accordingly a new CSFC is generated for further deformation. This process repeats iteratively until no candidate node is found, and a coarse CSFC is generated. Note that crossed edges may occur, e.g., the two edges in the shaded area of Figure 1 (d) (e). This is inevitable due to the discrete nature of sensor networks and the fact that our approach is based on mere connectivity information. However, it does not matter in our problem as we are only interested in finding a curve covering the whole network, i.e., traversing all sensors.

Since for each edge on the path, we only select one common neighbor (e.g., $p_{12}$), as the intermediate node between an edge (e.g., $p_{12}p_{2}$) on CSFC, we thus have the following propositions:

Proposition 1. The above-described deformation/expansion process will generate a CSFC without a loop.

Proposition 2. If the network is regularly shaped, the generated CSFC traverses all sensors.

However, due to the network sparsity or irregularity (e.g., with holes), some nodes may not be included in the CSFC and we thus call the derived CSFC as a coarse CSFC. To derive a CSFC covering the whole network, we need a refinement procedure to link these isolated nodes into the coarse CSFC, which will be detailed in next subsection.

D. CSFC REFINEMENT

During the edge deformation/expansion process, some nodes might be excluded on the coarse CSFC since, only one common neighbor of an edge will be selected as an intermediate for edge deformation. For example, in Figure 1 (e) (f), Node $p_1$ and $p_2$ have two common neighbors, namely, $p_3$ and $p_{12}$, and only $p_{12}$ is included in the expanded path for next round of expansion. In addition, the presence of small holes and network irregularity may also contribute some isolated nodes. These nodes form islands and thus a refinement process is needed for deriving the CSFC to traverse all nodes. Please
Algorithm 1: SFC Computation of 3D surface sensor networks

Input: A 3D surface sensor network \( G = (V, E) \) where \( V \) denotes the set of sensors and \( E \) denotes the edge set; an Entrance \( S \) and an Exit \( E \) of the CSFC

Output: The constructed CSFC with constraint on the Entrance \( S \) and the Exit \( E \)

1. Initialization: Each node \( p \) maintains a list of branch nodes, named \( BL(p) \) which is initialized to be empty; a shortest path \( SP(S, E) \) between \( S \) and \( E \) is built via in-network flooding and serves as the initialized \( CSFC(S, E) \) for expanding

2. bLoopFlag ← true

3. while (bLoopFlag) do
   4.      bLoopFlag ← false
   5.      bPathExpanded ← true;
   6.      while (bPathExpanded) do
      7.          bPathExpanded ← false
      8.      for all pairs of sensors \( p, q \in CSFC(S, E) \) such that \( q \text{.parent} = p \) do
            9.              for all sensors \( v \in N(p) \) do
                10.                 if \( v \in N(q) \) then
                    11.                     \( CN(p, q) = v \cup CN(p, q) \)
                12.                 end if
            13.             end for
            14.             Select Node \( s \in CN(p, q) \) with the smallest \( \text{NodeID} \)
            15.             \( s \text{.child} ← q; s \text{.parent} ← p; p \text{.child} ← s; q \text{.parent} ← s \)
            16.             CSFC \( (S, E) = s \cup CSFC(S, E) \)
            17.             bPathExpanded ← true
      18.      end for
   19.  end while
   20.  for all sensors \( s \notin CSFC(S, E) \) do
      21.      if \( N(s) \subseteq CSFC(S, E) \) then
            22.              Select randomly a node \( p \in N(s) \)
            23.              \( s \text{.parent} ← p; s \text{.child} ← q \)
            24.              \( BL(p) = s \cup BL(q) \)
            25.              CSFC \( (S, E) = s \cup CSFC(S, E) \)
      26.      end if
   27.  end for
   28.  while \( \lvert N(s) \cap \{ V \setminus CSFC(S, E) \} \rvert = 1 \) then
      29.      Assume \( p \in N(s), p \notin CSFC(S, E) \)
      30.      Select Node \( q \in CSFC(S, E) \) such that \( q \in N(s) \cap N(p) \)
      31.      \( s \text{.parent} ← q; s \text{.child} ← p \)
      32.      \( p \text{.parent} ← s; p \text{.child} ← q \)
      33.      \( BL(q) = s \cup BL(q) \)
      34.      CSFC \( (S, E) = \{ s \cup p \} \cup CSFC(S, E) \)
   35.  end while
   36.  if there exist two nodes, namely Node \( p \) and Node \( q \), such that \( p \in N(s), p \in CSFC(S, E), q \notin CSFC(S, E), q \in N(s) \cap N(p) \) then
      37.      \( s \text{.parent} ← p; s \text{.child} ← q \)
      38.      \( q \text{.parent} ← s; q \text{.child} ← p \)
      39.      \( BL(p) = s \cup BL(p) \)
      40.      CSFC \( (S, E) = \{ p \cup q \} \cup CSFC(S, E) \)
      41.      bLoopFlag ← true
   42.  end if
43.  end while
44. return CSFC \( (S, E) \)

see Figure 4 for some intuition. Generally speaking, there are four possible cases of isolated nodes and the corresponding solutions are given as follows:

Case 1: The neighbors of an isolated node are all on the coarse CSFC. In Figure 5 (a) where \( N_1 \) is such an isolated node, we only need to connect \( N_1 \) with one of its neighbors, say Node \( P \), on the coarse CSFC, as indicated by Figure 5 (b). As a result, Node \( P \) regards \( N_1 \) as its parent and \( p \) joins the branch list of \( N_1 \), namely \( BL(N_1) \). Of course, this will incur a loop such that Node \( P \) will be visited twice, and we refer such a node as a branch node. Please see the pseudo code from Line 21 to Line 26 in Algorithm 1.

Case 2: An isolated node has only one neighboring isolated node. Please see Figure 5 (c) where \( N_1 \) and \( N_2 \) are isolated by the coarse CSFC indicated by the solid curves with arrows, and they have a common neighbor, e.g., Node \( p \), on the coarse CSFC. For this case, we can simply connect \( N_1, N_2 \) and \( p \), respectively, such that \( N_1 \) is the parent of \( N_2 \), the child of \( p \), and \( N_2 \) is the parent of \( p \), the child of \( N_1 \). At the same time, Node \( p \) joins the branch list of \( N_1 \), i.e., \( BL(N_1) \). Please see the pseudo code from Line 27 to Line 33 in Algorithm 1.

Case 3: An isolated node has more than two neighboring isolated nodes (e.g., the island on the upper right part of Figure 4 (b)). For such an isolated node \( s \), we first determine whether it has two neighboring nodes \( p \) and \( q \), such that \( p \) is on the coarse CSFC, \( q \) is an isolated node and also a common neighbor of \( s \) and \( p \). If yes, \( p \) is the parent of \( s \), the child of \( q \), and \( q \) is the parent of \( p \), the child of \( s \). In addition, Node \( s \) joins the branch list of Node \( p \), and \( p \) and \( q \) are both included in the coarse CSFC. Afterwards, a new iteration will start, which is controlled by the boolean variable \( b \text{.LoopFlag} \), until we derive the final CSFC covering the whole network, i.e., all sensors are on the CSFC. Please see the pseudo code from Line 34 to Line 42 in Algorithm 1.

E. PATH DIRECTION ASSIGNMENT OF CSFC

With the refined CSFC, we now assign the direction for each node such that under the guide of the CSFC, a mobile robot can move from the Entrance to the Exit while visiting all nodes. For a node with an empty branch list, a robot simply moves to its child. However, there might exist a branch node \( s \) with a non-empty branch list \( BL(s) \). If the branch list \( BL(s) \) has only one node \( q \), the robot first moves from \( s \) to the node \( q \), then to the child of \( q \), and so on, until it moves back to \( s \), and we refer to such a path as the branch path generated by \( q \), denoted by \( BP(q) \). If \( BL(s) = \{ q_1, q_2, \ldots, q_k \} \) where \( k \geq 2 \), then the robot first move along the generated path \( BP(q_1) \), then \( BP(q_2) \), and finally \( BP(q_k) \). After that, the robot moves to the child of \( s \) for traversing other unvisited sensors. This way, the robot can visit all nodes along the CSFC.

IV. DISCUSSIONS
A. COMPLEXITY ANALYSIS

Theorem 3. Our approach has a linear complexity in terms of both the message cost and time cost.

Proof. Our approach for CSFC computation has three major blocks, namely CSFC initialization, coarse CSFC establishment and refinement. During CSFC initialization, a pair of Entrance and Exit issue a flooding to build a shortest path where each node will only forward one package to the neighbors and other late coming messages are immediately discarded. As such, the message and time cost are both $O(N)$ where $N$ denotes the network size. For coarse CSFC establishment, the candidate nodes having neighbors on the CSFC sequentially join in the SFC until no candidate is found. This step clearly only involves a linear complexity of message and time. For the final step of CSFC refinement, as only a very small number of nodes are not in the CSFC and form the island(s), the refinement is only conducted within a small subset of sensors and thus the message and time cost is roughly constant and $O(N)$ at most. In summary, our approach only incurs a linear complexity of time and message.

B. MULTIPLE PAIRS OF ENTRANCE-EXIT

For a large-scale network, as mentioned in Section I, it might take a long time for a mobile robot to traverse all sensor nodes. To greatly save the travel time for data collection or battery recharging, we can allow multiple robots [13] to simultaneously traverse the whole network along different CSFCs with different pairs of Entrance-Exit. To this end, we need to compute a CSFC for each robot such that two robots will not visit the same node.

Interestingly, the computation of multiple SFCs with different Entrance-Exit pairs can be similarly done in the above-described approach, though primarily designed for one pair of Entrance-Exit. Specifically, suppose that there are $k$ ($\geq 2$) pairs of Entrance-Exit $(S_1, E_1), (S_2, E_2), \cdots, (S_k, E_k)$. For each Entrance-Exit pair, we first compute a shortest path $SP(S_i, E_i)$ between them via a flooding from the Entrance $S_i$, and accordingly derive $k$ CSFCs with different pairs of Entrance-Exit, each of which is named, say by the NodeID of the Entrance $S_i$. During the coarse CSFC establishment of each CSFC, for each edge $p_1p_2$ on the CSFC, if Node $p_3$ is a common neighbor of $p_1$ and $p_2$, and the boolean flag $bFlag$ of $p_3$ is false, then $p_3$ will be a candidate node for deforming edge $p_1p_2$. Afterwards, $p_1$ selects the candidate node $p_{12}$ with the smallest NodeID as the intermediate node, and replaces the edge $p_1p_2$ with the curved edge formed by edge $p_1p_{12}$ and edge $p_{12}p_2$. This expansion process continues until no candidate node can be found. Eventually, $k$ CSFCs will be generated in such an iterative fashion.

C. TRAVEL BUDGET VS. CSFC COVERAGE

For a large-scale network, the SFC may be rather long if it covers the whole network. Practically, there might be budget limit for such applications of sensor network as data fusion and battery-charging such that not all sensors will be visited and there exists a trade-off between travel budget and CSFC coverage. To adapt our approach for this case, we can simply change the length of deformed edge during path expansion according to the travel budget. Specifically, if the travel budget only allows $1$ out of $k$ sensors being associated with the CSFC, then for the shortest path $SP(S, E)$ generated during CSFC initialization with the constraint on the given Entrance $S$ and Exit $E$, we find a node $p_1$ on the path with a separation of $k$ hops away from Entrance $S$ and draw a virtual edge between the Entrance $S$ and $p_1$. Afterwards, we find an intermediate node $q_1$ such that $q_1$ is $k$ hops away from $S$ and $p_1$, and deform the virtual edge $Sp_1$ with the curved edge $Sq_1p_1$. Similarly, we deform the virtual edge between $p_1$ and the node $k$ hops away from $p_1$, and so on. Eventually, around $1$ out of $k$ sensors will be visited along the computed CSFC with a limited travel budget governed by the parameter $k$. Obviously, the larger $k$ is, the less the computed CSFC...
V. PERFORMANCE EVALUATION

To our knowledge, there is no previous study on SFC computation with the constraint on the specific pairwise Entrance and Exit, and previous work on SFC computation cannot be readily adapted for this task. To evaluate the performance of our approach for CSFC computation in sensor networks, we develop a program to implement our approach on Windows 7 operating system with Microsoft visual studio 2010. The source code with some example data can be found at [8].

We first conduct extensive simulations on four representative 2D and 3D scenarios with complex shapes, respectively. To further show that our algorithm is robust to network scale and node density, we randomly select some nodes from the original network to derive a new network with less nodes and smaller node density (in terms of average node degree). Please see Figure 6 and Figure 7, respectively.

Next we compare our algorithm with a recent work on SFC computation in sensor networks named SURF [19] for the following reasons: 1) SURF is the state-of-the-art for SFC computation; 2) it is designed for high-genus 3D surface sensor networks with complex shapes, and 3) there is no time-consuming mapping involved in SURF. As the constraint of Entrance-Exit is not considered by SURF, we simply compute the SFC without imposing the constraint on SURF. We conduct the comparison studies on four 3D surface sensor networks with various node densities and genus, namely, the Corridor-shaped network of genus 1 in Figure 8 (a), the Bowknot-shaped network of genus 2 in Figure 8 (b), the Smile-shaped network of genus 3 in Figure 8 (c), and the Window-shaped network of genus 4 in Figure 8 (d). The network size ranges from 710 to 5366, while the average node degrees are all below 11. The metrics used for comparison include the distribution of node visited times, the length of the SFC (in terms of hop-count distance and Euclidean distance) and the computation efficiency.

To study that our approach is immune from the Entrance and the Exit, we randomly select 5000 pairs of nodes serving as the Entrance and Exit, respectively, and accordingly compute 5000 CSFCs for the investigated 3D surface sensor networks. We then examine the effects of hop-count distance between the Entrance and the Exit (or, the locations of the
Entrance and Exit) and the number of Entrance-Exit pairs. Finally, to examine the robustness of our approach to communication radio models, we conduct the simulations on the double-torus shaped 3D surface network in Figure 1 using two other models, i.e., quasi-unit disk graph (QUDG) and Log-normal shadowing model.

A. THE PERFORMANCE UNDER 2D/3D SENSOR NETWORKS WITH COMPLEX SHAPES

Figure 6 depicts the constructed CSFC (marked by the red lines) of four typical 2D sensor networks with complex shapes (i.e., some holes) where the end nodes of the blue lines (i.e., the initialized CSFC) are the Entrance and Exit, respectively. To show the robustness to the network scale and node density, we randomly select some nodes in each original network in Figure 6 (top) to derive a new network with less nodes and low node density, as shown in Figure 6 (bottom). Table 1 depicts the distribution of the node visited times for the investigated 2D sensor networks. It can be shown that our approach yields stable CSFCs, where most of the nodes are only visited once. Also we notice that when the network becomes sparser (e.g., the average node degree is around 6), more nodes are visited more than once. This is because more edges might find no intermediate for deformation and thus more nodes are unvisited by the coarse CSFC. As a result, the nodes neighboring to these unvisited nodes might be visited twice or more. However, the ratio of the number of once-visited nodes to the whole network size (simply referred to as Ratio thereafter) for each network is above 94%. Further, assume that on a CSFC, the i-th node will be visited \(k_i\) times, then for a network with \(n\) nodes, the length of the SFC (referred to as hop-count length) in terms of hop-count can be computed as \(L_H(SFC) = \sum_{i=1}^{n} i \times k_i\). Clearly, the more nodes with large visited times, the longer the path will be, and vice versa. Figure 6 implies that our approach yields the CSFC with nearly the shortest length in terms of hop count distance between nodes, and thus provides a good solution for the CLP given in Section II.

Table 1: The distribution of visited times for 2D sensor networks with complex shapes in Figure 6

<table>
<thead>
<tr>
<th>Topology</th>
<th>Size(avg. deg.)</th>
<th>Visited Times</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Figure 6 (a) (top)</td>
<td>6169</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Figure 6 (b) (top)</td>
<td>6659</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Figure 6 (c) (top)</td>
<td>2922</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Figure 6 (d) (bottom)</td>
<td>5179</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Figure 6 (d) (bottom)</td>
<td>3363</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 7 describes the constructed CSFC (marked by the red lines) of four typical 3D sensor networks, and again for each 3D network we randomly select some nodes to form a smaller scaled network with lower node density. Table 2 presents the distribution of the node visited times for the investigated 3D sensor networks. Similar with the 2D cases, most of the Ratios are over 96%, except for the network in Figure 6 (d) (bottom) which has a very low node density with the average node degree is only 6.6. Yet the Ratio is still over 93%. From these results, we can conclude that that by increasing the communication radio range (and thus the average degree of nodes), the number of nodes being visited once will become larger. This implies that the sparsity (and thus the interior hole) of sensor networks is the root cause of unvisited nodes during edge deformation as described in Section III-C.

B. THE COMPARISON STUDY WITH SURF ON 3D HIGH-GENUS SURFACE SENSOR NETWORKS

Figure 8 depicts the comparison results by our approach and SURF on four investigated scenarios with the genus ranging from 1 to 4. Clearly, our approach and SURF derives an SFC traversing the whole networks. However, compared with SURF, the CSFC by our approach has more nodes covered only once (i.e., the visited times is one), and few nodes are visited more than twice. For example, for the Corridor-shaped network with 887 nodes, 883 nodes are visited once along the CSFC by our approach and 4 nodes are visited twice, while for the SFC by SURF, the number of nodes being visited once, twice, three and four or more times are 695, 109, 24 and 8, respectively. We observe the similar trend for other three scenarios with higher genus. The reason is that in SURF, the network are divided into many sub-regions based on the feature function constructed via in-network flooding. As a result, the contour of the Reeb graph might be incomplete (i.e., loopless), resulting some nodes within the regions of incomplete contour being visited more than once when the SFC traverses across regions. In addition, some leaf nodes (i.e., without any child node) are regarded as dead-ends by SURF, and the parent of a dead end will also be visited more than once. On the contrary, our approach will only visit branch nodes more than once, and the number of branch nodes is generally very small. As a result, our approach outperforms SURF in terms of node visited times.

To comprehensively and quantitatively show that the advantage of our approach over SURF for solving the CLP, we further calculate the SFC length measured by hop count distance. As the CSFC by our approach have more nodes which are visited once (shown in Table 3), the SFC length by SURF is around 1.25 times longer than our approach based.
on hop count distance between nodes.

In practice, we may also concern about the displacement of a mobile robot along the computed SFC, and an important goal is to minimize the SFC length in terms of the Euclidean distance between pairwise neighboring nodes on the SFC. Given an SFC with a node sequence \( p_1, p_2, \ldots, p_n \), we define the SFC length (referred to as Euclidean length) in the Euclidean coordinate system as \( L_E(SFC) = \sum_{i=1}^{n} [(x_{i} - x_{i+1})^2 + (y_{i} - y_{i+1})^2 + (z_{i} - z_{i+1})^2] \) where \( (x_i, y_i, z_i) \) is the 3D coordinate of \( p_i \). Note that our approach does not require the location information of sensors, and here we only use the coordinates for computing the Euclidean length for the seek of the comparison study. Table 3 shows that for the investigated scenarios, the computed SFCs by our approach are all much shorter than SURF. We believe that this is due to the fact that our approach is based on path expansion in an iterative fashion to ensure the locality preservation, while SURF works on the contour sets of the Reeb graph such that the SFC by SURF will have a long zig-zag pattern. As a result, the locality property might be violated. Table 4 presents the computational time of the SFC by our approach and SURF. Desirably, the computational time by our approach is linear to the network size. Compared with SURF, our approach can yield an SFC in much shorter time, while the computational time of SURF is at least five times longer than that by our approach. This shows that our approach can quickly generate an SFC such that when there is delay budget for the mobile robot, our approach would be more promising, letting alone that our approach allows the robot to start at the given Entrance and quit at the given Exit.

### C. THE ROBUSTNESS TO COMMUNICATION RADIO MODEL

Previously we have shown that our algorithm achieves promising results for 2D sensor networks and 3D surface/volume sensor networks with complex shapes, where...
the network communication radio model is assumed to be unit-disk graph (UDG) model, i.e., for any two nodes, if their separation $d$ is less than the communication radio range $R$, there exists a link between them. In this section, to show the robustness to communication radio model, we examine the behavior of our approach using quasi-unit disk graph (QUDG), and log-normal shadowing model, respectively.

Under QUDG model [3], there exists a link between two nodes with probability 1 if their separation $d$ is less than $\alpha \times R$ where $\alpha \in (0, 1)$, with probability $p(\in (0, 1))$ if $\alpha \times R \leq d \leq R$, and no link exists otherwise. Under log-normal shadowing radio model, a link between pairwise nodes exists with probability $p$ given by the following equation [6]:

$$p(\hat{d}) = \frac{1}{2}[1 - \text{erf}(\frac{\log \hat{d}}{\sqrt{2} \log 10})], \xi = \frac{\sigma}{\eta}$$ (1)

where $\text{erf}(\cdot)$ is the error function, $\hat{d}$ is the normalized distance, $\sigma$ is the standard deviation of the shadowing, $\eta$ is the pathloss exponent, and empirically $\xi$ ranges from 0 to 6. Note that under log-normal model, for two nodes with $\hat{d} > 1$, there might have a link, while there might not exist a link between two nodes with $d \leq 1$.

Figure 9 shows the CSFC of the double-torus network.

### Table 5: The distribution of visited times for double-torus network in Figure 1 using different communication models

<table>
<thead>
<tr>
<th>Model</th>
<th>Topology</th>
<th>avg. deg.</th>
<th>Visited Times</th>
<th>Visited Times</th>
<th>Visited Times</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 2 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UDG</td>
<td>Figure 1</td>
<td>15.6</td>
<td>1924</td>
<td>0 0</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>QUDG</td>
<td>Figure 9 (a)</td>
<td>7.24</td>
<td>1621</td>
<td>262 41</td>
<td>84.3%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Figure 9 (b)</td>
<td>9.1</td>
<td>1915</td>
<td>9 0</td>
<td>99.5%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Figure 9 (c)</td>
<td>11.08</td>
<td>1920</td>
<td>4 0</td>
<td>99.8%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Figure 9 (d)</td>
<td>11.85</td>
<td>1924</td>
<td>0 0</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Log-normal</td>
<td>Figure 10 (a)</td>
<td>7.24</td>
<td>1883</td>
<td>40 1</td>
<td>97.9%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Figure 10 (b)</td>
<td>10.37</td>
<td>1871</td>
<td>52 1</td>
<td>97.3%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Figure 10 (c)</td>
<td>19.88</td>
<td>1893</td>
<td>2 2</td>
<td>98.5%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Figure 10 (d)</td>
<td>33.70</td>
<td>1903</td>
<td>21 0</td>
<td>98.9%</td>
<td></td>
</tr>
</tbody>
</table>
in Figure 1 under QUDG with different parameters, and Figure 10 presents the results under Log-normal model with different values of parameter $\xi$. Table 5 summarizes the distribution of node visited times under different communication radio models, including UDG, QUDG, and Log-normal, with the same Entrance and Exit. Under QUDG model, when $\alpha$ is smaller, two nodes with separation small than $R$ are more unlikely to be connected by an edge, and thus the average degree of the network is smaller. As a result, more nodes will be visited twice or more. With the increasing of $\alpha$ (and/or $p$), the average degree increases and thus more nodes will be visited once, which is confirmed by our experiments as shown in Figure 9 and Table 5. We believe that the low Ratio of 84.3% for $\alpha = 0.5$ is mainly due to the network sparsity where the average degree is only 7.24; when the density is comparable (e.g., the average degree is more than 9) with the network under UDG model, the Ratio approaches 100%.

Under Log-normal model, there exist long links between nodes with the normalized distance larger than 1, while two close nodes might not be connected by an edge. As such, the network will have a more complex topology with many locally sparse sub-networks where the nodes are likely to be visited more than once. From Figure 10 and Table 5, we find that with the increasing of $\xi$, the network becomes denser, and more nodes will be visited once, but overall the Ratio remains stable, and approaches the Ratio of UDG model.

All in all, based on the performance evaluation under QUDG and Log-normal models, we argue that our algorithm is robust to communication radio model.

**D. THE ROBUSTNESS TO THE LOCATIONS OF PAIRWISE ENTRANCE-EXIT**

To show the robustness of our approach, we randomly select 5000 pairs of Entrance-Exit and derive 5000 CSFCs accordingly. In particular, here we are more interested in the number (or, the ratio) of nodes which are traversed for one time on the CSFC, as we argue that the CSFC computation for a robot should try to avoid repeatedly visit the sensor nodes. Hence, we present the distribution of Ratio, i.e., the ratio of the number of nodes covered once to the network size, for the 5000 derived CSFCs, as shown in Figure 12. We observe that our approach can yield a very stable result in term of the Ratio metric; the Ratios of the 5000 CSFCs for the Window-shaped network are all between 94% and 95%, while the Ratios for other networks are all between 98.5% and 99.7%. This implies that the locations of the Entrance and the Exit has very little effect on the performance of our algorithm. It also shows that our approach consistently outperforms than SURF with the Ratio all below 85% for the four investigated scenarios as reported in [19] and confirmed by our simulations described in the above comparison study.

**E. THE PERFORMANCE FOR MULTIPLE PAIRS OF ENTRANCE-EXIT**

Figure 11 shows the computed CSFCs for 2-5 pairs of Entrance-Exit on the Bowknot-shaped network in Figure 8 (b), where each pair of sensors are randomly selected to serve as the Entrance and the Exit, respectively. We first obtain two pairs of Entrance-Exit and build a shortest path (i.e., an initialized CSFC) between them (see Figure 11 (a)); then with these two pairs of Entrance-Exit, we additionally select a third pair of Entrance-Exit and accordingly generate three shortest paths (i.e., three initialized CSFCs as shown in Figure 11 (b)), and so on. Here visually we observe that the shortest paths between different pairwise Entrance-Exit intersect with each other. However, this has no effect on the CSFC computation since each node is at most affiliated with one CSFC, and the simulation results show that our approach can compute a CSFC successfully covering the whole network, with constraint on different number of Entrance-Exit pairs. Further, to quantitatively show that our approach is robust to the number of Entrance-Exit pairs, we compute the number of nodes visited once on the CSFC, as shown in Figure 13. It turns out that due to the distributed and iterative nature of the proposed approach, the number of sensors visited once does not significantly change for different pairs of Entrance-Exit.

**VI. CONCLUSIONS**

In this paper we present a unified framework for space-filling curve computation in 2D/3D sensor networks with the constraint on the Entrance and Exit. To our knowledge, this is the first study on the constrained linearization problem for 2D/3D sensor networks to yield a constrained SFC (CSFC). Our approach first initializes the CSFC by constructing a...
shortest path between the Entrance and the Exit, and then iteratively deforms the edges on the CSFC to derive a coarse CSFC, followed by the refinement process to generate the final CSFC. The proposed approach is distributed, of low complexity, easy to implement, and works for 2D sensor networks and 3D surface/volume sensor networks with complex shapes. We conduct extensive simulations to show that our approach is efficient (in terms of node visited times), and robust to many factors such as network scale, density, shape, communication radio range, etc., and the comparison study on 3D high-genus surface sensor networks shows that our approach outperforms the state-of-the-art SURF.

REFERENCES