applied sciences

MDPI

*Article*

# Three-Dimensional Formation Control for Robot Swarms

**Jonghoek Kim**

Electronic and Electrical Department, Sungkyunkwan University, Suwon 16419, Korea; jonghoek@gmail.com

**Abstract:** This article addresses a distributed 3D algorithm for coordinating a swarm of autonomous robots (ARs) to spatially self-aggregate into an arbitrary shape based on only local interactions. Each AR has local proximity sensors for measuring the relative coordinates of its nearby AR. We assume that only a single AR, called the leader, can localize itself in global coordinate systems, while accessing the arbitrary user-specified 3D shape. We further assume that the leader has a communication ability superior to all other ARs so that the leader can directly send a communication signal to any other AR inside the shape. Our aim is to make the ARs maneuver while maintaining a user-specified 3D formation such that the network connection of all ARs is maintained during the maneuver. Our approach results in a 3D formation shape and does not need the global localization of an AR, except for the leader. To the best of our knowledge, our study is novel in the construction of a 3D formation for covering an arbitrary shape such that network connection is maintained while ARs maneuver based on local communication. We further show that the proposed control yields reliable accuracy against significant AR failures and movement error. Utilizing MATLAB simulations, we demonstrate the outperformance of the proposed formation controls.

**Keywords:** distributed network systems; 3D formation control; mobile sensor network; network connection; user-specified 3D shape



**Citation:** Kim, J. Three-Dimensional Formation Control for Robot Swarms. *Appl. Sci.* **2022**, *12*, 8078. https://doi.org/10.3390/app12168078

Academic Editor: Alessandro Gasparetto

Received: 4 July 2022
Accepted: 11 August 2022
Published: 12 August 2022



## 1. Introduction

In this article, we tackle the problem of organizing autonomous robots (ARs) into arbitrary self-sustaining 3D formations. Keeping formations is crucial for various tasks, especially when the task requires collective action [1]. For instance, ARs may aggregate for coordinated search and rescue, collectively maneuvering large objects, or exploring unknown environments.

Formation controls in 3D environments (e.g., underwater space), where global positioning system (GPS) is not accessible, is not trivial [2–4]. In underwater mobile sensor networks, many difficulties remain to be studied, especially in the communication between robots, owing to unfriendly environments that limit the communication channel [5].

In this paper, we assume that each AR has local proximity sensors for measuring the relative coordinates of its nearby AR. Moreover, it is assumed that only a single AR, called the leader, can access the arbitrary user-specified 3D shape. We further assume that the leader has a communication ability superior to all other ARs so that the leader can directly send a communication signal to any other AR inside the shape. We further assume that only the leader can localize itself in global coordinate systems. For instance, the leader can use the localization approach based on inertial measurement units (IMU) [6], or visual–inertial simultaneous localization and mapping (VI-SLAM) [7,8]. Here, VI-SLAM integrates a high frequency imaging sensor and IMU for the robot's localization.

Our aim is to make all ARs maneuver while forming a user-specified 3D formation such that the network connection of all ARs is maintained during the maneuver. Our approach results in a 3D formation shape, and one only requires global localization of the leader.

To the best of our knowledge, our study is novel in the construction and control of a 3D formation for covering an arbitrary shape such that the network connection is preserved while every AR maneuvers based on local communication. We assume that there exists a

multi-hop communication link between any two ARs initially. This initial connection is crucial since every AR maneuvers based on local communication only, while not depending on global coordinate information.

In the proposed formation controls, we locate ARs one by one, until the network entirely covers the arbitrary user-specified 3D shape. In the proposed 3D formation controls, each AR spreads the network by tracking an edge of the network.

There may be a case in which the networked ARs entirely cover the user-specified 3D shape, but we want to add more ARs into the shape. In this case, we apply distributed rendezvous controls [9], while not moving the leader. Thereafter, the network begins clustering toward the leader. After the network begins clustering, coverage holes are newly formed in the 3D shape. Additional ARs are further located until they cover the holes in the shape entirely.

Once a swarm of ARs are aggregated into a user-specified 3D shape, we can control the ARs so that they maneuver in a desired direction while maintaining the network connection. For the flocking maneuver, all ARs maneuver synchronized to the leader. We assume that the leader has a communication ability superior to all other ARs so that the leader can directly send a communication signal to any other AR inside the shape. Therefore, the velocity command of the leader can be transmitted to all ARs inside the shape in real time. The velocity of each AR is synchronized to that of the leader utilizing the communication command transmitted from the leader.

As far as we know, our manuscript is novel in control of 3D formation for covering an arbitrary shape, such that network connection is preserved while each AR maneuvers based on local communication. We show that the proposed control yields reliable accuracy against significant AR failures and movement error. Utilizing MATLAB simulations, we demonstrate the outperformance of the proposed formation controls.

The remainder of this article is as follows: Section 2 reviews the literature related to this study. Section 3 discusses assumptions and definitions utilized in our paper. Section 4 addresses the 3D formation controls. Section 5 addresses how to cover the sensing holes in the user-specified 3D shape. MATLAB simulations are presented in Section 6. Conclusions are addressed in Section 7.

## 2. Literature Review

There are many papers on mobile sensor systems in 2D environments. Refs. [10,11] considered deploying a swarm of ARs into an unknown 2D environment for achieving complete sensor coverage of the environment. The authors of [10,11] utilized Voronoi tessellations for making all ARs evolve in time for increasing the network coverage based on local communication. Furthermore, the authors of [12] implemented their coverage controls utilizing real ARs.

There are many papers on formation controls in 2D environments. Considering 2D environments with obstacles, Ref. [13] tackled the distributed formation control of multiple robots. Ref. [14] considered the problem of letting multiple nonholonomic robots follow a desired leader–follower formation under omnidirectional vision. The authors of [15] addressed distributed multi-robot formation control in the face of dynamic obstacle interference. In [16], a deep reinforcement learning (DRL)-based method was proposed to guide multiple robots through unknown complex 2D environments, where the centroid of the robot team aims to arrive at the goal while avoiding collisions and maintaining connectivity. Considering mobile ARs in 2D environments, Ref. [17] proposed an online distributed algorithm based on the lattice of configurations, in order to recover the formation when a sensing failure causes the network to lose its rigidity.

Considering 2D environments, Ref. [1] discussed a decentralized algorithm for coordinating multiple ARs to spatially self-aggregate into an arbitrary shape, utilizing local communication only. The authors of [1] achieved multi-robot formation by allowing the robots to disperse inside the given 2D shape. If an AR believes that it is inside the shape, then it behaves like a gas particle with the shape as a closed container. Otherwise, the AR

wanders randomly. However, random walk may lead to disconnected networks, which leads to the loss of an AR. Ref. [18] presented a distributed algorithm that enables multiple robots to form arbitrary shapes and regenerate the shapes when cut.

There are many papers on multi-agent formation control in 3D environments. Considering a fish-inspired robot swarm, Ref. [19] showed that dynamic 3D collective behaviors can be achieved by measuring a robot's neighbors, without any centralized controls. The authors of [20] introduced a distributed control strategy for multiple agents, so that the agents can achieve a desired 3D formation based on local relative position measurements. Ref. [21] proposed a formation control method that enables multiple aerial robots to enclose a 3D target by generating a given geometric formation surrounding the target. The authors of [22] proposed a progressive assignment algorithm and formation control scheme that extends leader–follower formations for enabling multi-robot cooperation with minimal, one-way, local communication among agents. Ref. [23] developed a path planner for 3D robot formations and addressed how to adapt the formation shape for avoiding obstacles. Ref. [24] presented collision-free formation flight and reconfiguration for a team of autonomous helicopters. The control scheme in [24] was based on potential fields with the concept of a virtual leader. However, these papers on 3D formation controls did not consider the construction of a 3D formation for covering an arbitrary shape, as presented in our paper.

To the best of our knowledge, the proposed multi-agent system is novel in the construction and control of a 3D formation for covering an arbitrary shape such that network connection is preserved while every AR maneuvers based on local communication. This article verifies that the proposed control scheme yields reliable accuracy against significant AR failures and movement error. MATLAB simulations are used to demonstrate the outperformance of the proposed formation controls.

## 3. Assumptions and Definitions

### 3.1. Definitions

We address the definitions used in this paper. Let $min(a, b)$ return a smaller value between $a$ and $b$. According to graph theory [25], a graph $G$ is defined as $G = (V(G), E(G))$, in which $V(G)$ represents the vertex set, and $E(G)$ represents the edge set. In a graph, a *path* is an alternative sequence of vertexes and edges. We say that $G$ is *connected* if one can find a connected path between any two vertexes in $G$. Any two end vertexes of an edge in $E(G)$ are *neighbors* to each other.

A *tree* $T$ is a connected graph containing no cycles. A *spanning tree* of $G$ is a tree having all vertexes in $G$.

One vertex of $T$ is selected as the *root*. Let $v$ present one vertex in $T$. In $T$, $p(v)$, the *parent* of $v$, defines the neighbor of $v$ along the path to the root. Furthermore, $c(v)$, *child* of $v$, defines a vertex, satisfying that $v$ is the parent of $c(v)$.

A vertex having no children is called the *leaf*. A *descendant* of $v$ represents a vertex which is either $c(v)$ or is the descendant of $c(v)$ (recursively). An *ancestor* of $v$ represents a vertex which is either $p(v)$ or is the ancestor of $p(v)$ (recursively).

Suppose $N$ ARs are located in a 3D environment. Let $u_i$ represent the $i$-th AR ($i \in \{1, 2, \ldots, N\}$). In the inertial frame, let $\mathbf{u}_i \in \mathcal{R}^3$ define the 3D location of $u_i$ ($i \in \{1, 2, \ldots, N\}$).

Every AR has local communication modules. Among all ARs, only the leader, say, $u_1$, is able to locate itself in global coordinates. In addition, the leader has a communication ability superior to all other ARs so that the leader can directly send a communication signal to any other AR inside the user-specified shape.

Initially, all ARs are positioned such that one can find a connected path between any two ARs in the network. This *initial network connection* is crucial since each AR maneuvers through local communication, while not relying on GPS. Utilizing the proposed formation controls, every AR maneuvers, while guaranteeing that the network connection is maintained.

Let $r_s$ present the maximum range of an AR local sensor. In addition, let $r_c$ present the maximum range of AR local communication. Moreover, $r_{sc}$ is defined as

$$r_{sc} = min(r_s, r_c). \tag{1}$$

The *coverage* of an AR, say $u_i$, represents a sphere with radius $r_{sc}$, centered at $\mathbf{u}_i$. Let *senseSphere* of an AR present the boundary of the AR coverage.

$Q$ points are uniformly formed on the senseSphere of an AR $u_i$. For instance, we address the process of building $Q = 20 \times 20$ points on a senseSphere of $u_i$. This method is used in MATLAB simulations (see Section 6). Centered at the $y$-axis, we rotate a vector $[r_{sc}, 0, 0]$ by an angle in $[\frac{\pi}{10}, \frac{2\pi}{10}, \frac{3\pi}{10}, \ldots, 2\pi]$. The rotation of $\theta$ with respect to the $y$-axis is performed through the rotation matrix

$$\mathbf{R}(\theta) = \begin{pmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{pmatrix}. \tag{2}$$

Thereafter, centered at the $z$-axis, we rotate $\mathbf{R}(\theta)[r_{sc}, 0, 0]^T$ by an angle in $[\pi/10, 2 \times \pi/10, 3 \times \pi/10 \ldots, 2\pi]$. The rotation of $\psi$ with respect to the $z$-axis is performed through the rotation matrix

$$\mathbf{R}(\psi) = \begin{pmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3}$$

Thereafter, we calculate

$$\mathbf{fp}(\psi, \theta) = \mathbf{R}(\psi)\mathbf{R}(\theta)[r_{sc}, 0, 0]^T. \tag{4}$$

By adding $\mathbf{u}_i$ to $\mathbf{fp}(\psi, \theta)$, we obtain the 3D coordinates of a point on a senseSphere of $u_i$. $\psi$ and $\theta$ are selected from $[\pi/10, 2 \times \pi/10, 3 \times \pi/10 \ldots, 2\pi]$, respectively. Therefore, one calculates $Q = 20 \times 20$ points on a senseSphere of $u_i$.

Among $Q$ points of an AR $u_i$, a *shapePt* is a point that is in the user-specified 3D shape. In practice, shapePts can be determined, since $u_1$ can access the 3D shape, and we can localize $u_i$. How to locate $u_i$ in global coordinates is proved in Theorem 2. $u_1$ sends the information of the user-specified shape to $u_i$ so that $u_i$ can find a shapePt in its local frame.

A *frontierPt* $f(u_i)$ of an AR $u_i$ is a shapePt of $u_i$, where every point in $f(u_i)$ is outside the sensing range of every other AR. Note that a frontierPt is in the user-specified 3D shape since it is a shapePt. As $Q \to \infty$, we obtain densely spaced frontierPts on a senseSphere.

Let *frontierInfty* present the set of frontierPts as $Q \to \infty$. This implies that a frontierInfty is a subset of a senseSphere. If every AR has no frontierInfty, then all of the ARs' coverSpheres cover the entire space inside the user-specified 3D shape.

Let $L(u_i, u_j)$ represent a straight line segment connecting two ARs $u_i$ and $u_j$. Recall that $r_{sc}$ was defined in (1). We assume that two ARs $u_i$ and $u_j$ can sense each other, in the case where $L(u_i, u_j)$ is not blocked by obstacles and the length of $L(u_i, u_j)$ is shorter than $r_{sc}$. Moreover, $u_i$ and $u_j$ can communicate with each other, in the case where $L(u_i, u_j)$ is not blocked by obstacles and the length of $L(u_i, u_j)$ is shorter than $r_{sc}$. We say that two ARs $u_i$ and $u_j$ are *neighbors* in the case where $u_i$ and $u_j$ can sense and communicate with each other.

Figure 1 shows the case in which an AR $u_i$ maneuvers toward a frontierPt $f(u_i)$. In Figure 1, a sphere depicts $u_i$. Figure 1 shows that $u_i$ maneuvers along a narrow tunnel. The path of $u_i$ is plotted as yellow lines. Large dots on the path of $u_i$ indicate the ARs, which are positioned along the path of $u_i$. The senseSphere of each AR is plotted as a sphere with dots. On the senseSphere of $u_i$, frontierPts are plotted with dots.

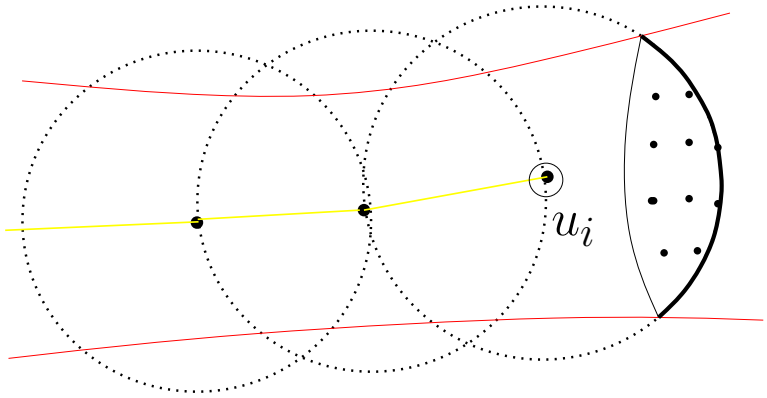


**Figure 1.** The AR $u_i$ maneuvers toward a frontierPt $f(u_i)$. and the path of $u_i$ is plotted as yellow lines. Large dots on the path of $u_i$ plot the ARs, which are positioned along the path of $u_i$. On the senseSphere of $u_i$, frontierPts are plotted with dots.

Let $\mathbf{u}_i(k)$ define $\mathbf{u}_i$ at sample-index $k$. In the inertial frame, the process model of $u_i$ is

$$\mathbf{u}_i(k+1) = \mathbf{u}_i(k) + \mathbf{V}_i(k) \times dt. \tag{5}$$

Here, $\mathbf{V}_i(k)$ defines the velocity command of $u_i$ at sample-index $k$. The simple dynamics in (5) are applicable to many autonomous vehicles. In practice, an AR maneuvers with a limited speed. Therefore, let $S_i$ define the maximum speed of $u_i$, i.e., $\|\mathbf{V}_i(k)\| \leq S_i$ for all $k$.

### *3.2. Assumptions*

Two ARs $u_i$ and $u_j$ are *nearby* to each other, if $\|\mathbf{u}_i - \mathbf{u}_j\| < 2 \times r_{sc}$ is met. $u_i$ is a *closeAR* of $u_j$ if $\|\mathbf{u}_i - \mathbf{u}_j\| < 2 \times r_{sc}$ is met. Every AR $u$ uses the following assumptions:

(A1) An AR $u$ can detect the relative coordinates of any of its closeAR.
(A2) An AR $u$ stores the relative coordinates of a frontierPt in $f(u)$.
(A3) Initially (at sample-index 0), one has a connected path between any two ARs.
(A4) $u_1$ can locate itself in global coordinates, while accessing the user-specified 3D shape. In addition, $u_1$ has a communication ability superior to all other ARs so that the leader can directly send a communication signal to any other AR inside the shape.

This paper considers an AR which can measure the relative coordinates of its closeAR utilizing proximity sensor, thus Assumption (A1) is plausible. For instance, an AR, say, $u$, generates signal pings for measuring a closeAR. Once the pings generated from $u$ are reflected from its closeAR, say, $A$, then $u$ can estimate the elevation angle, azimuth angle, and range to $A$ through 3D multiple signal classification (MUSIC) algorithm [26].

An AR $u$ obtains the relative coordinates of its closeAR through Assumption (A1). Therefore, $u$ is able to localize the senseSphere of its closeAR. Therefore, $u$ is able to calculate the relative coordinates of a frontierPt in $f(u)$. Therefore, Assumption (A2) is plausible.

Every AR maneuvers utilizing local sensing measurements, thus Assumption (A3) is crucial. To the best of our knowledge, other papers [27–31] on distributed multi-AR control applied the initial connection assumption.

## 4. Formation Controls

The considered problem is as follows: *construct a formation with a user-specified shape in a 3D environment utilizing multiple ARs. While ARs maneuver, they preserve the network connection.*

### *4.1. Distributed Generating of a Spanning Tree*

Before tacking the above problem, we address how to form a spanning tree in a distributed fashion. We form a spanning tree $T$ rooted at an AR $u$ by applying a distributed breadth first search (BFS) algorithm in [32]. Ref. [32] presented a distributed algorithm, so that a node in the network can guide a moving object across the network to a designated

goal. Algorithm 2 in [32] can be applied to make a spanning tree $T$ rooted at $u$. The spanning tree $T$ is rooted at $u$, and $T$ has an unique path from an AR to any other AR.

The goal sensor in Algorithm 2 of [32] represents the root, say $u_r$, in this article. Each AR $u$ stores and updates $hops_g(u)$ (the hop distance to the root $u_r$). The root $u_r$ initializes $hops_g(u_r) = 0$. Every other AR initializes $hops_g(u) = \infty$ where $u \neq u_r$.

Initially, $u_r$ broadcasts $hops_g(u_r)$ to all its neighbors. Suppose an AR $u$ receives a hop distance message from its neighbor, say $n$. In this case, $u$ updates its hop distance information as

$$hops_g(u) = min(hops_g(u), hops_g(n) + 1). \tag{6}$$

Under (6), $hops_g(u)$ can be updated to $hops_g(n) + 1$. In this case, the parent of $u$ is updated to $n$. Thereafter, $u$ broadcasts $hops_g(u)$ to its neighbors. Refs. [32,33] proved that the number of message broadcasts of each AR is 1 in this algorithm. This implies that the distributed BFS has the computational complexity $O(1)$.

### *4.2. Distributed Rendezvous Control*

Our aim is to construct a formation with a user-specified shape in a 3D environment utilizing multiple ARs. While ARs maneuver, they must preserve the network connection. This article addresses the formation controls (Algorithm 1) as follows. Before running the formation controls, all ARs rendezvous at the leader in a distributed fashion. Once the rendezvous is achieved, one satisfies that

$$\|\mathbf{u}_j - \mathbf{u}_1\| < \epsilon \tag{7}$$

for all $j \in \{2, \ldots, N\}$.

Here, $\epsilon \ll r_{sc}$ is a small positive constant.

We form a spanning tree $T$ whose root is the leader, by utilizing the distributed BFS in Section 4.1. Based on the formed spanning tree $T$, Algorithm 2 in [9] is applied to make all ARs rendezvous at the leader, while preserving network connection. Algorithm 2 in [9] initiates by letting a leaf AR visit its ancestors in $T$ sequentially, until it meets the leader. At the instant when an AR meets all its descendants, the AR begins visiting its ancestors in $T$ sequentially, until it meets the leader. However, [9] handled 2D environments. Since our article considers ARs in 3D environments, we address distributed rendezvous controls to make an AR visit its ancestors in 3D environments.

Let $path_T(u_i, u_1)$ define the path in $T$, from $u_i$ to the leader $u_1$. Suppose that $path_T(u_i, u_1)$ consists of an AR set $p_1 \to p_2 \to p_3 \ldots \to p_{end} = u_1$. Consider the case where $u_i$ has just met $p_l$ and the next AR to meet is $p_{l+1}$ ($l \in \{1, 2, \ldots, end - 1\}$).

In order to make $u_i$ maneuver toward $p_{l+1}$, $\mathbf{V}_i(k)$ of $u_i$ in (5) is set as

$$\mathbf{V}_i(k) = S_i \frac{\mathbf{p}_{l+1} - \mathbf{u}_i(k)}{\|\mathbf{p}_{l+1} - \mathbf{u}_i(k)\|}. \tag{8}$$

Here, $\mathbf{p}_{l+1} - \mathbf{u}_i(k)$ is available under Assumption (A1). This implies that $u_i$ maneuvers toward $p_{l+1}$ utilizing local sensing measurements.

Under (8), $u_i$ maneuvers toward $p_{l+1}$ with a constant speed $S_i$. In the case where $u_i$ meets $p_{l+1}$, $u_i$ maneuvers toward the next AR $p_{l+2}$. This process repeats until $u_i$ meets $p_{end} = u_1$.

Suppose that all ARs rendezvous at the leader and that the leader cannot detect a frontierPt in the user-specified 3D shape. Under Assumption (A4), the leader is able to localize itself while accessing the user-specified 3D shape. Therefore, the leader maneuvers toward the user-specified 3D shape slowly, until it can detect a frontierPt in the user-specified shape.

While the leader maneuvers slowly, all other ARs keep maneuvering toward the leader based on their local sensing measurements. Since all ARs are already rendezvoused at the

leader, maneuvering toward the leader is feasible based on the local sensor of each AR. If necessary, rendezvous controls can be applied by regenerating a tree $T$ by applying the distributed BFS addressed in this subsection.

### *4.3. Formation Controls for Expanding the Network*

Once all ARs rendezvous at the leader, then every AR is controlled one at a time, such that as more ARs arrive at their assigned positions, an uncovered space in the user-specified 3D shape reduces incrementally. The proposed formation controls (Algorithm 1) work by sequentially locating ARs one by one, until the network fills the user-specified 3D shape.

Algorithm 1 is explained as follows. Once the rendezvous is finished, a tree $T$ is formed in a distributed fashion. For generating $T$, every AR $u_i$ stops maneuvering, while utilizing its proximity sensor for measuring the relative coordinates of its neighbors. The distributed BFS algorithm in Section 4.1 is applied for the distributed generation of $T$.

Initially, the leader $u_1$ switches on its proximity sensor for covering the space close to $u_1$. $u_1$ localizes its global position utilizing Assumption (A4). Thereafter, frontierPts of $u_1$ are built.

Suppose an AR arrives at its assigned position. Thereafter, the AR switches on its proximity sensor for covering its surroundings. For generating a connected network, an AR maneuvers into an uncovered space within $r_{sc}$ distance units. As an AR arrives at its assigned position, the AR switches on its proximity sensor with range $r_{sc}$. See *EnableSensor* function (Algorithm 2).

**Algorithm 1** Formation controls

1: Initially, every AR will rendezvous at the leader under distributed rendezvous control in Section 4.2;
2: $S = [u_1, u_2, \ldots, u_N]$;
3: The leader $u_1$ switches on its proximity sensor with range $r_{sc}$ and builds a frontierPt inside the user-specified 3D shape;
4: $i = N$;
5: **repeat**
6: By applying the distributed BFS in Section 4.1, $u_i$ finds an AR with a frontierPt, which has the smallest hop distance from $u_i$;
7: **if** no frontierPt is detected by running the distributed BFS **then**
8: This algorithm is finished;
9: **end if**
10: Let $u_f$ present the found AR;
11: A frontierPt on the senseSphere of $u_f$ is set as $f_{u_i}$;
12: The AR $u_i$ maneuvers along the path to $u_f$, then tracks the edge from $u_f$ to $f_{u_i}$;
13: **if** The AR $u_i$ arrives at its assigned position $f_{u_i}$ **then**
14: EnableSensor($u_i$,$u_f$);
15: **end if**
16: $i = i - 1$;
17: **until** the index $i$ becomes 1

**Algorithm 2** EnableSensor($u$,$u_f$)

1: $u$ switches on its proximity sensor;
2: $u$ localizes itself by measuring the relative coordinates of $u_f$;
3: $u$ builds a frontierPt inside the user-specified 3D shape;

In Algorithm 1, every AR is controlled one at a time with the following order: $u_N \rightarrow u_{N-1}, \ldots \rightarrow u_2$. The AR $u_N$ maneuvers to arrive at its frontierPt $f_{u_N}$. Thereafter, $u_N$ switches on its proximity sensor for covering its close space. FrontierPts of $u_N$ are built accordingly. See Algorithm 2.

Once $u_N$ arrives at its assigned position, $u_{N-1}$ maneuvers until reaching its frontierPt $f_{u_{N-1}}$. Keep iterating this procedure until all ARs arrive at their assigned positions.

Consider the case in which $u_{i+1}$ just turned on its proximity sensor for covering its close space. Thereafter, $u_i$ finds an AR, say $u_s$, with a frontierPt, which has the smallest hop distance from $u_i$. The distributed BFS in Section 4.1 is applied for this search. Thereafter, the frontierPt on the senseSphere of the found AR is set as $f_{u_i}$.

An AR $u_i$ maneuvers along a path, say $P$, in the spanning tree $T$ until reaching its frontierPt $f_{u_i}$. Considering the definition of the neighbor, the length of each line segment along this path is shorter than $r_{sc}$. In addition, the path is collision-free, since no obstacle hinders the path. In this way, the proposed control considers obstacle avoidance of each AR.

At the instant when $u_i$ arrives at an AR in $P$, $u_i$ maneuvers toward the next AR in $P$, since an AR can sense its neighbor in $P$. (8) addresses how to make $u_i$ maneuver toward an AR in $P$ utilizing local sensing measurements. In order to track along the path $P$, $u_i$ measures an AR in $P$ using its local sensor. At the instant when $u_i$ arrives at an AR, say $A$, in $P$, $u_i$ detects the relative coordinates of the next AR in $P$. See Assumption (A1).

As $u_i$ arrives at $u_s$ (the last AR in $P$), $u_i$ is able to maneuver toward $f_{u_i}$ under Assumption (A2). After $u_i$ arrives at $f_{u_i}$, $u_i$ switches on its proximity sensor to sense its close space. See *EnableSensor* function (Algorithm 2).

Once $u_i$ switches on its proximity sensor, frontierPts inside the senseSphere of $u_i$ disappear. This disappearance is achieved through the local sensing of $u_i$. Consider the case in which a point in $f(m)$ is in the senseSphere of $u_i$. The distance between $m$ and a point in $f(m)$ is shorter than $r_{sc}$. Thus, $m$ is a closeAR of $u_i$. $u_i$ is able to sense the relative coordinates of $m$ under Assumption (A1).

Consider the case where $u_i$ measures the relative coordinates of its closeAR $m$. The vector from $u_i$ to a point in $f(m)$ is the addition of the following two vectors, which are accessible using Assumptions (A1) and (A2):

1. The vector from $m$ to the point in $f(m)$.
2. The vector from $u_i$ to $m$.

At the moment when $u_i$ switches on its proximity sensor, all frontierPts in the senseSphere of $u_i$ disappear. Then, $u_i$ broadcasts the disappearance of the frontierPts to all ARs through multi-hop communication. $u_{i-1}$ then searches for its frontierPt $f_{u_{i-1}}$, and maneuvers toward the found frontierPt. This repeats until $i$ becomes 1 or no frontierPt is detected utilizing the distributed BFS. Once $i$ decreases to 1, then one has no remaining ARs for covering a remaining frontierPt.

There may be a case in which the networked ARs entirely cover the user-specified 3D shape, but we want to add more ARs into the 3D shape. In this case, we apply distributed rendezvous controls in Section 4.2, while not moving the leader. Based on distributed rendezvous controls, the network begins clustering toward the leader. After the network begins clustering, coverage holes are newly formed in the 3D shape. Additional ARs are further located until they cover the holes in the shape entirely. Covering the sensing holes in the 3D shape is further handled in Section 5.

Analysis

We analyze the computational complexity of Algorithm 1. $u_i$ finds an AR with a frontierPt, which has the smallest hop distance from $u_i$. This search can be performed through distributed BFS, whose computational complexity is $O(1)$ [32,33]. Since $i$ decreases from $N$ to 1 in the worst case, the computational complexity of the loop in Algorithm 1 is $O(N)$.

As $Q$ increases to infinity, frontierPts on a senseSphere become the frontierInfty. Theorem 1 proves that if an AR cannot find a frontierInfty, then the entire space inside the user-specified shape is covered by coverSpheres.

**Theorem 1.** *If an AR cannot find a frontierInfty, then the entire space inside the user-specified shape is covered by coverSpheres.*

**Proof.** Under the transposition rule of propositional logic, the following statement is proved: if one has an uncovered space inside the user-specified shape, then an AR is able to find a frontierInfty.

Suppose that one has an uncovered space in the user-specified shape. Let $O$ present this uncovered space. Using the definition of a frontierInfty, at least one AR can find a frontierInfty on the boundary of $O$. Therefore, an AR is able to find this frontierInfty. □

If $Q$ is not so large, then frontierPts may represent a frontierInfty coarsely. Therefore, Theorem 1 may not always hold.

In Algorithm 1, an AR $u_i$ maneuvers along a path, say $P$, until reaching $f_{u_i}$. Theorem 2 implies that $u_i$ applies local communication only for its maneuver. In addition, network connection is assured during the maneuver. Theorem 2 further proves that as $u_i$ arrives at $f_{u_i}$, we are able to locate $u_i$ in global coordinates.

**Theorem 2.** *Consider the case where all ARs maneuver under Algorithm 1. While an AR $u_i$ maneuvers along a path, say $P$, until reaching $f_{u_i}$, the maneuver of $u_i$ does not lead to disconnected networks. While $u_i$ maneuvers along $P$, $u_i$ applies local communication (one hop communication) only. As $u_i$ arrives at $f_{u_i}$, we are able to locate $u_i$ in the global coordinates.*

**Proof.** In Algorithm 1, all ARs rendezvous at the leader before the formation controls are initiated. Therefore, the maneuver of $u_i$ does not lead to disconnected networks.

An AR $u_i$ maneuvers along a path, say $P$, until reaching $f_{u_i}$. Let $\{m_1 \rightarrow m_2 \rightarrow \ldots \rightarrow m_{end}\}$ represent the order of ARs along $P$ until reaching $f_{u_i}$. After reaching $m_j$ ($j \leq end - 1$), $u_i$ maneuvers toward $m_{j+1}$. Furthermore, after reaching $m_{end}$, $u_i$ maneuvers toward $f_{u_i}$.

One proves that while $u_i$ maneuvers along $P$, $u_i$ applies local sensing measurements only. At the instant when $u_i$ encounters $m_j$, $u_i$ derives the relative coordinates of $m_{j+1}$. $m_j$ is a neighbor of $m_{j+1}$. Thus, $u_i$ applies local sensing measurements to maneuver from $m_j$ to $m_{j+1}$. This local measurement is feasible under Assumption (A1).

One proves that when $u_i$ arrives at its associated position $f_{u_i}$, we can locate $u_i$ in global coordinates. We prove by deduction.

In Algorithm 1, $i$ starts from $N$ and decreases to 1 as time elapses. At the initial stage of Algorithm 1, one considers the case where $i$ is $N$. The global coordinate of $u_N$ is derived from the addition of the following two vectors:

1. The global coordinate of $u_1$.
2. The vector from $u_1$ to $u_N$.

The vector from $u_1$ to $u_N$ is available utilizing the proximity sensor of $u_N$. Additionally, the leader $u_1$ is localized in global coordinates.

Next, consider the case where $i \neq N$. For $i \neq N$, the global coordinate of $u_i$ is calculated as the addition of the following two vectors:

1. The global coordinate of $m_{end}$.
2. The vector from $m_{end}$ to $u_i$.

The vector from $u_i$ to $m_{end}$ is available utilizing the proximity sensor of $u_N$.

This theorem is proved. □

## 5. Covering the Sensing Holes Inside the User-Specified Shape

There may be a case where the networked ARs entirely cover the user-specified 3D shape, but we want to add more ARs into the shape. In this case, we apply distributed rendezvous controls in Section 4.2, while not moving the leader. After the network begins clustering under the rendezvous controls in Section 4.2, coverage holes are newly formed in the user-specified 3D shape. Additional ARs are further located until they cover all holes in the user-specified shape entirely. Covering the sensing holes in the 3D shape is handled in this section.

Furthermore, covering the sensing holes is useful when AR failures happen. Suppose that ARs are positioned for covering the user-specified 3D shape through the formation

controls in our paper. As time elapses, some ARs may fail. In addition, an intruder may destroy some ARs, hence coverage holes inside the user-specified shape are formed. To tackle this case, we address formation controls for covering all sensing holes inside the user-specified shape.

Suppose that only $N_h$ ARs are located inside the user-specified shape. Suppose that new $M$ ARs $v_i$ ($i \in \{1, 2, \ldots, M\}$) are located such that at least one AR is connected to the existing tree $T$, which contains $N_h$ ARs.

We use Algorithm 3 for covering all sensing holes inside the user-specified shape. We locate these new $M$ ARs, in order to cover all holes in the shape. Initially, $M$ ARs rendezvous at the leader under the rendezvous algorithms in [9]. Thereafter, a new tree $T$ is initialized by applying the distributed BFS in Section 4.1 to the new $M$ ARs. In Algorithm 3, $T = T_{new} \cup T$ implies that $T_{new}$ is merged with the existing tree $T$ to form a new tree $T$.

**Algorithm 3** Cover the sensing holes

1: Suppose that only $N_h$ ARs are located inside the user-specified shape, while forming a tree $T$;
2: Initially, $M$ ARs rendezvous at the leader;
3: $S = [v_1, v_2, \ldots, v_M]$;
4: New tree $T_{new}$ is initialized by applying the distributed BFS in Section 4.1 to the new $M$ ARs;
5: $T = T \cup T_{new}$;
6: $i = M$;
7: **repeat**
8: By applying the distributed BFS in Section 4.1, $v_i$ finds an AR with a frontierPt, which has the smallest hop distance from $v_i$;
9: Let $u_s$ present the found AR;
10: A frontierPt on the senseSphere of $u_s$ is set as $f_{v_i}$;
11: The AR $v_i$ maneuvers along the path to $u_s$, then tracks the edge from $u_s$ to $f_{v_i}$;
12: **if** $v_i$ arrives at its assigned position $f_{v_i}$ **then**
13: EnableSensor($v_i$,$u_s$);
14: **end if**
15: $i = i - 1$;
16: **until** the index $i$ becomes 0 or no frontierPt is detected under the distributed BFS;

*Analysis*

Suppose that $M$ is sufficiently large to cover all sensing holes in the 3D shape. Theorems 1 and 2 are used to analyze Algorithm 3.

This subsection analyzes the computational complexity of Algorithm 3. $v_i$ finds an AR with a frontierPt, which has the smallest hop distance from $v_i$. This search can be performed utilizing distributed BFS, whose computational complexity is $O(1)$ [34]. Since $i$ decreases from $M$ to 1 in the worst case, the computational complexity of the loop in Algorithm 3 is $O(M)$.

## 6. MATLAB Simulation Results

Table 1 summarized the parameters in our paper.

**Table 1.** The simulation parameters.

| *Parameters* | *Values* |
|---|---|
| $N$ (number of ARs) | 100 |
| $r_{sc}$ (radius of a senseSphere) | 50 (m) |
| $S_i$ (AR's maximum speed) | 5 (m/s) |
| $Q$ (number of rays surrounding an AR) | 400 |
| $dt$ (sampling interval) | 1 (s) |

Initially, the leader $u_1$ is located at the origin. We simulate the case where ARs are randomly deployed inside the restricted space with size $70 \times 70 \times 70$ in meters. Thus, the $i$-th AR ($i \in \{2, \ldots, N\}$) is located randomly as follows:

$$\mathbf{u}_i = [70 \times rand, 70 \times rand, 70 \times rand]^T. \tag{9}$$

Here, *rand* returns a random number in the interval [0,1].

Under the rendezvous controls in Section 4.2, we first make all ARs rendezvous at the leader located at the origin. Thereafter, Algorithm 1 is applied to make the ARs cover the user-specified 3D shape.

In practice, there is process noise in the the process model of $u_i$ (5). Therefore, instead of (5), $u_i$ maneuvers under

$$\mathbf{u}_i(k+1) = \mathbf{u}_i(k) + \mathbf{V}_i(k) \times dt + [N \times randn, N \times randn, N \times randn]^T. \tag{10}$$

Here, $N$ is the strength of the process noise. We use $N = 1$ meter. In (10), *randn* returns a value with zero mean Gaussian noise with standard deviation 1.

### *6.1. MATLAB Simulation of Algorithm 1*

One demonstrates the effectiveness of Algorithm 1 through MATLAB simulations. We consider the case in which the user-specified 3D shape is a sphere with radius 100, centered at the origin. Figure 2 shows the path of each maneuvering AR until the user-specified 3D shape is entirely covered. Here, Algorithm 1 is utilized for formation controls. The path of a maneuvering AR is depicted with yellow line segments. The final position of a maneuvering AR is depicted as a blue diamond. See that blue diamonds are marked at the end points of each line segment.

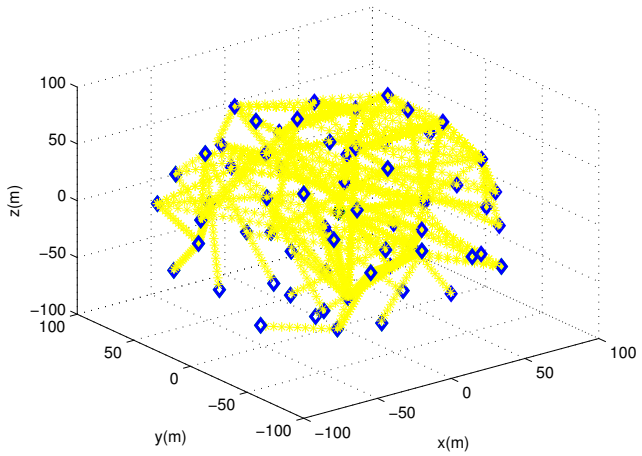


**Figure 2.** The path of each maneuvering AR until the user-specified 3D shape (sphere with radius 100) is entirely covered ($r_{sc} = 50$ m). The path of a maneuvering AR is depicted with yellow line segments. The final position of a maneuvering AR is depicted as a blue diamond. See that blue diamonds are marked at the end points of each line segment.

Figure 3 presents the ARs' final positions under Algorithm 1. Among 100 ARs, 66 ARs maneuver for covering the user-specified 3D shape, while (100-66) ARs are located at the leader. In other words, Algorithm 1 is finished as $i = 100 - 66$ since no frontierPt is detected at this instant. The ARs consume 4115 s until the 3D shape is entirely covered. MATLAB takes 42 s to run the simulation.

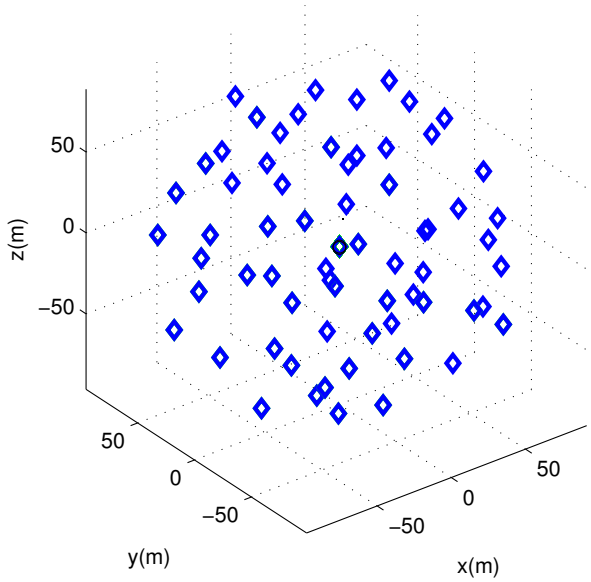


**Figure 3.** The ARs' final positions under our formation controls (Algorithm 1) for covering the user-specified sphere shape ($r_{sc} = 50$ m).

Once a swarm of ARs are aggregated into a user-specified 3D shape, one can control all ARs so that they maneuver in a desired direction while maintaining the network connection. For the flocking maneuver, all ARs maneuver while synchronized to the leader. Under Assumption (A4), the velocity command of the leader can be transmitted to all ARs inside the shape in real time. The velocity of each AR is synchronized to that of the leader, utilizing the communication command transmitted from the leader.

#### 6.1.1. Change the Sensing Range

One next presents the effect of changing $r_{sc}$. The sensing range $r_{sc}$ varies from 50 to 70 m. This implies that the radius of a senseSphere is 70 m.

Under the rendezvous controls in Section 4.2, we first make all ARs rendezvous at the leader located at the origin. Thereafter, Algorithm 1 is applied to make the ARs cover the 3D sphere, while setting $r_{sc}$ as 70 m. Figure 4 shows the path of each maneuvering AR until the 3D sphere is entirely covered. The path of a maneuvering AR is depicted with green line segments. The final position of a maneuvering AR is depicted as a blue diamond. See that blue diamonds are marked at end points of each line segment.

Figure 5 presents the final position of each AR under Algorithm 1. Among 100 ARs, only 34 ARs maneuver for covering the user-specified sphere, while (100-34) ARs do not maneuver at all. Compared to the case in which $r_{sc} = 50$ m, increasing $r_{sc}$ to 70 decreases the number of maneuvering ARs. The ARs consume 2323 s until the user-specified sphere is entirely covered. A total of 17 s are consumed to perform the MATLAB simulation.

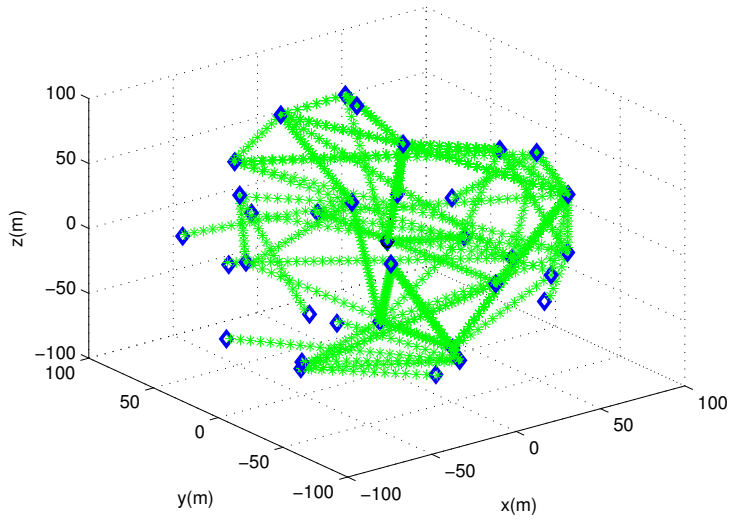


**Figure 4.** The path of each maneuvering AR until the 3D sphere is entirely covered ($r_{sc} = 70$ m). The path of a maneuvering AR is depicted with green line segments. The final position of a maneuvering AR is depicted as a blue diamond. See that blue diamonds are marked at end points of each line segment.

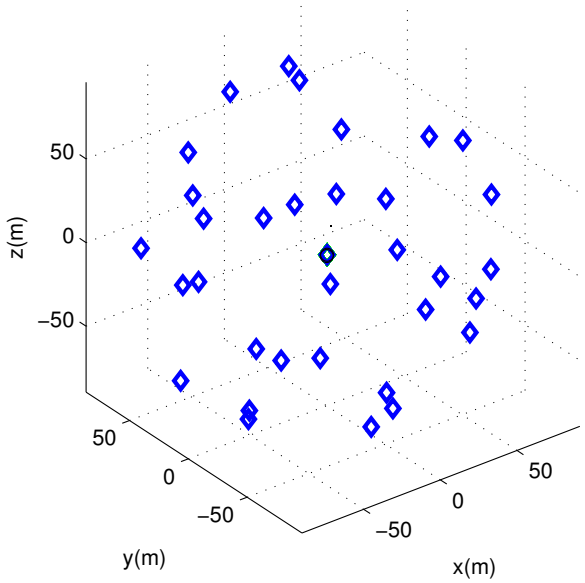


**Figure 5.** The final position of each AR under our formation controls (Algorithm 1) for covering the user-specified sphere shape. We use $r_{sc} = 70$ m.

#### 6.1.2. Change the User-Specified 3D Shape

There may be a case in which the networked ARs entirely cover the user-specified 3D shape, but we want to change the shape to a new shape. We next address how to change the user-specified 3D shape from a sphere to a box shape. The user-specified shape is a box, which has height (z) as 200/3, width (y) as 200/3, and length (x) as 300 in meters.

The sensing range $r_{sc}$ is set as 50 m. Utilizing the rendezvous controls in Section 4.2, we first make all ARs rendezvous at the leader located at the origin. Thereafter, Algorithm 1 is applied to make the ARs cover the user-specified 3D box, while setting $r_{sc}$ as 50 m.

Figure 6 shows the path of each maneuvering AR until the user-specified box is entirely covered. Here, Algorithm 1 is utilized for formation controls. The path of a maneuvering AR is depicted with magenta line segments. The final position of a maneuvering AR is

depicted as a blue diamond. See that blue diamonds are marked at the end points of each line segment.

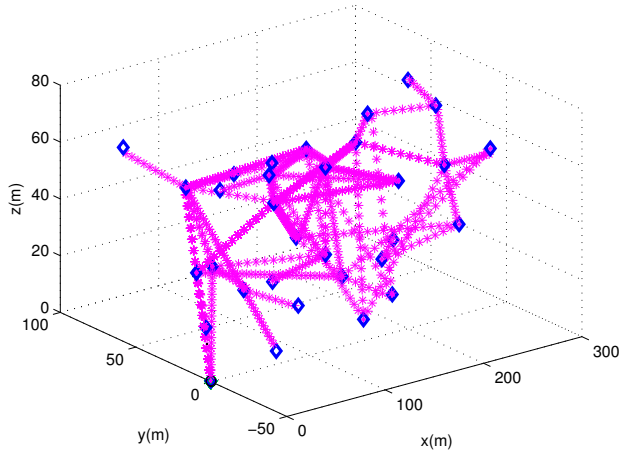


**Figure 6.** The path of each maneuvering AR until the user-specified box is entirely covered ($r_{sc} = 50$ m). The path of a maneuvering AR is depicted with magenta line segments. The final position of a maneuvering AR is depicted as a blue diamond. See that blue diamonds are marked at end points of each line segment.

Figure 7 presents the ARs' final positions under Algorithm 1. Among 100 ARs, 32 ARs maneuver for covering the user-specified box shape, while $(100 - 32)$ ARs are located at the leader. In other words, Algorithm 1 is finished as $i = 100 - 32$, since no frontierPt is detected at this instant. The ARs consume 1689 s until the 3D shape is entirely covered. MATLAB consumes 16 s to perform the simulation.

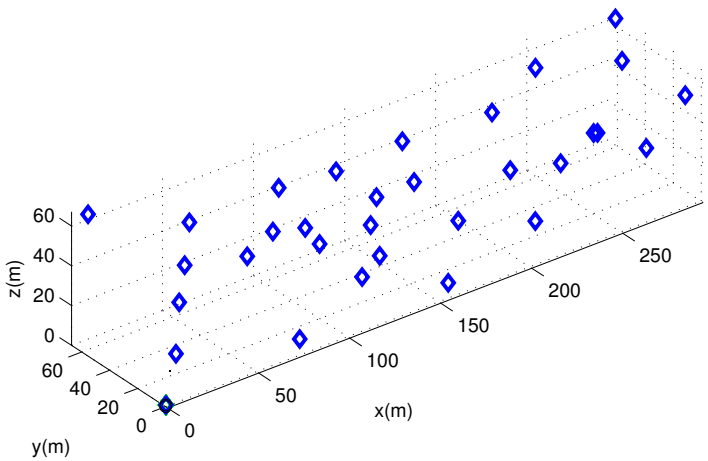


**Figure 7.** The ARs' final locations under our formation controls (Algorithm 1) for covering the user-specified box shape ($r_{sc} = 50$ m).

#### 6.1.3. Forming a Complicated 3D Shape

We next address how to change the user-specified 3D shape from a box shape to a complicated shape (the union of two boxes). The user-specified shape is complicated, which is set as the union of two connected boxes: one box has height (z) as 200/3, width (y) as 200/3, and length (x) as 300 in meters, and another box has height (z) as 200, width (y) as 200/3, and length (x) as 200/3 in meters.

The sensing range $r_{sc}$ is set as 50 m. Figure 8 shows the path of each maneuvering AR until the user-specified complicated shape is entirely covered. Here, Algorithm 1 is

utilized for formation controls. The path of a maneuvering AR is depicted with red line segments. The final position of a maneuvering AR is depicted as a blue diamond. See that blue diamonds are marked at the end points of each line segment.

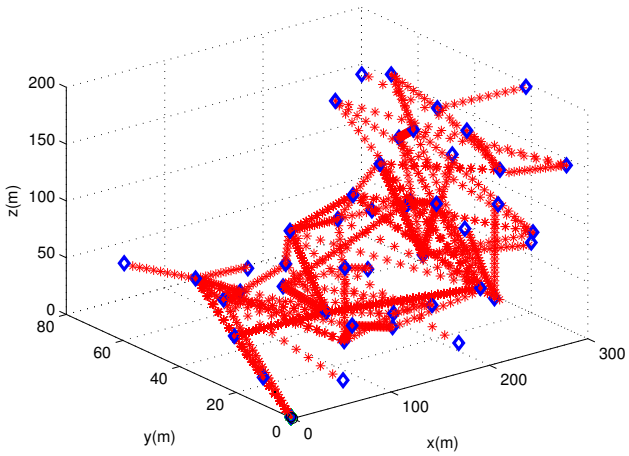


**Figure 8.** The path of each maneuvering AR until the user-specified complicated shape is entirely covered ($r_{sc} = 50$ m). The path of a maneuvering AR is depicted with red line segments. The final position of a maneuvering AR is depicted as a blue diamond. See that blue diamonds are marked at the end points of each line segment.

Figure 9 presents the ARs' final positions under Algorithm 1. Among 100 ARs, 46 ARs maneuver for covering the user-specified complicated shape, while (100-46) ARs are located at the leader. In other words, Algorithm 1 is finished as $i = 100 - 46$, since no frontierPt is detected at this instant. The ARs consume 3116 s until the 3D shape is entirely covered. MATLAB consumes 26 s to perform the simulation.

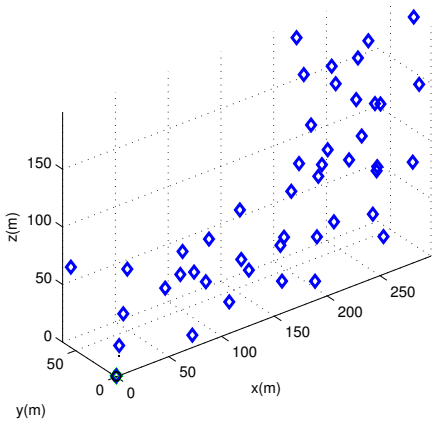


**Figure 9.** The ARs' final locations under our formation controls (Algorithm 1) for covering the user-specified complicated shape ($r_{sc} = 50$ m).

## 7. Conclusions

This article addresses a distributed 3D algorithm for coordinating a swarm of ARs to spatially self-aggregate into an arbitrary shape, utilizing local communication only. We make all ARs form a user-specified 3D formation such that the connection of all ARs is preserved during the maneuver. Our control scheme leads to a 3D formation for covering the given shape. Moreover, our approach only requires the global localization of the leader among all ARs.

To the best of our knowledge, this study is novel in building a 3D formation for covering a given shape such that network connection is maintained while ARs maneuver

based on local communication only. We show that the proposed formation control yields reliable accuracy in handling significant AR failures and movement error. Once a swarm of ARs is aggregated into a user-specified 3D shape, the leader can maneuver synchronized to all other ARs by sharing its velocity command with all other ARs.

Utilizing MATLAB simulations, we demonstrate the outperformance of the proposed formation controls. The proposed formation controls can be used in the case where the user-specified 3D shape varies as time goes on. Whenever the shape varies, we apply distributed rendezvous controls in Section 4.2, while not moving the leader. Once all ARs achieve rendezvous, they are redistributed by applying Algorithm 1 again. In the future, we plan to perform experiments to demonstrate the proposed formation controls using real ARs.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Cheng, J.; Cheng, W.; Nagpal, R. Robust and Self-Repairing Formation Control for Swarms of Mobile Agents. In Proceedings of the 20th National Conference on Artificial Intelligence, AAAI'05, Pittsburgh, PA, USA, 9–13 July 2005; pp. 59–64.
2. Liu, J.; Wang, Z.; Cui, J.H.; Zhou, S.; Yang, B. A Joint Time Synchronization and Localization Design for Mobile Underwater Sensor Networks. *IEEE Trans. Mob. Comput.* **2016**, *15*, 530–543. [CrossRef]
3. Misra, S.; Ojha, T.; Mondal, A. Game-Theoretic Topology Control for Opportunistic Localization in Sparse Underwater Sensor Networks. *IEEE Trans. Mob. Comput.* **2015**, *14*, 990–1003. [CrossRef]
4. Han, G.; Zhang, C.; Shu, L.; Rodrigues, J.J.P.C. Impacts of Deployment Strategies on Localization Performance in Underwater Acoustic Sensor Networks. *IEEE Trans. Ind. Electron.* **2015**, *62*, 1725–1733. [CrossRef]
5. dell'Erba, R. Determination of Spatial Configuration of an Underwater Swarm with Minimum Data. *Int. J. Adv. Robot. Syst.* **2015**, *12*, 97. [CrossRef]
6. Allotta, B.; Costanzi, R.; Fanelli, F.; Monni, N.; Paolucci, L.; Ridolfi, A. Sea currents estimation during AUV navigation using Unscented Kalman Filter. *IFAC-PapersOnLine* **2017**, *50*, 13668–13673. [CrossRef]
7. Melim, A.; West, M. Towards autonomous navigation with the Yellowfin AUV. In Proceedings of the OCEANS'11 MTS/IEEE KONA, Waikoloa, HI, USA, 19–22 September 2011; pp. 1–5.
8. Kelly, J.; Sukhatme, G.S. Visual-inertial simultaneous localization, mapping and sensor-to-sensor self-calibration. In Proceedings of the 2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation-(CIRA), Daejeon, Korea, 15–18 December 2009; pp. 360–368.
9. Kim, J. Distributed Rendezvous of Heterogeneous Robots with Motion-Based Power Level Estimation. *J. Intell. Robot. Syst.* **2020**, *100*, 1417–1427. [CrossRef]
10. Cortés, J.; Martínez, S.; Karatas, T.; Bullo, F. Coverage control for mobile sensing networks. *IEEE Trans. Robot. Autom.* **2004**, *20*, 243–255. [CrossRef]
11. Stergiopoulos, Y.; Kantaros, Y.; Tzes, A. Connectivity-aware coordination of robotic networks for area coverage optimization. In Proceedings of the 2012 IEEE International Conference on Industrial Technology, Athens, Greece, 19–21 March 2012; pp. 31–35.
12. Siligardi, L.; Panerati, J.; Kaufmann, M.; Minelli, M.; Ghedini, C.; Beltrame, G.; Sabattini, L. Robust Area Coverage with Connectivity Maintenance. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 2202–2208.
13. Alonso-Mora, J.; Baker, S.; Rus, D. Multi-robot formation control and object transport in dynamic environments via constrained optimization. *Int. J. Robot. Res.* **2017**, *36*, 1000–1021. [CrossRef]
14. Vidal, R.; Shakernia, O.; Sastry, S. Formation control of nonholonomic mobile robots with omnidirectional visual servoing and motion segmentation. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), Taipei, Taiwan, 14–19 September 2003; Volume 1, pp. 584–589.
15. Hu, J.; Sun, J.; Zou, Z.; Ji, D.; Xiong, Z. Distributed multi-robot formation control under dynamic obstacle interference. In Proceedings of the 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Boston, MA, USA, 6–9 July 2020; pp. 1435–1440.
16. Lin, J.; Yang, X.; Zheng, P.; Cheng, H. End-to-end Decentralized Multi-robot Navigation in Unknown Complex Environments via Deep Reinforcement Learning. In Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China, 4–7 August 2019; pp. 2493–2500.

17. Amani, A.M.; Chen, G.; Jalili, M.; Yu, X.; Stone, L. Distributed Rigidity Recovery in Distance-Based Formations Using Configuration Lattice. *IEEE Trans. Control. Netw. Syst.* **2020**, *7*, 1547–1558. [CrossRef]
18. Mishra, R.S.; Semwal, T.; Nair, S.B. A distributed epigenetic shape formation and regeneration algorithm for a swarm of robots. *Proc. Genet. Evol. Comput. Conf. Companion* **2018**, 1505–1512.
19. Berlinger, F.; Gauci, M.; Nagpal, R. Implicit coordination for 3D underwater collective behaviors in a fish-inspired robot swarm. *Sci. Robot.* **2021**, *6*, eabd8668. [CrossRef] [PubMed]
20. Fathian, K.; Safaoui, S.; Summers, T.H.; Gans, N.R. Robust 3D Distributed Formation Control with Collision Avoidance And Application To Multirotor Aerial Vehicles. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 9209–9215.
21. Aranda, M.; López-Nicolás, G.; Sagüés, C.; Zavlanos, M.M. Three-dimensional multirobot formation control for target enclosing. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 357–362.
22. McCord, C.; Queralta, J.P.; Gia, T.N.; Westerlund, T. Distributed Progressive Formation Control for Multi-Agent Systems: 2D and 3D deployment of UAVs in ROS/Gazebo with RotorS. In Proceedings of the 2019 European Conference on Mobile Robots (ECMR), Prague, Czech Republic, 4–6 September 2019; pp. 1–6.
23. Álvarez, D.; Gómez, J.V.; Garrido, S.; Moreno, L. 3D Robot Formations Path Planning with Fast Marching Square. *J. Intell. Robot. Syst.* **2015**, *80*, 507–523. [CrossRef]
24. Paul, T.; Krogstad, T.R.; Gravdahl, J.T. UAV formation flight using 3D potential field. In Proceedings of the 2008 16th Mediterranean Conference on Control and Automation, Ajaccio, France, 25–27 June 2008; pp. 1240–1245.
25. Douglas, B.W. *Introduction to Graph Theory*, 2nd ed.; Prentice Hall: Forsyth, IL, USA, 2001.
26. Li, Y.C.; Choi, B.; Chong, J.W.; Oh, D. 3D Target Localization of Modified 3D MUSIC for a Triple-Channel K-Band Radar. *Sensors* **2018**, *18*, 1634. [CrossRef] [PubMed]
27. Alonso-Mora, J.; Montijano, E.; Schwager, M.; Rus, D. Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus. In Proceedings of the Robotics and Automation (ICRA), 2016 IEEE International Conference, Stockholm, Sweden, 16–21 May 2016; pp. 5356–5363.
28. Cortés, J.; Martínez, S.; Bullo, F. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Trans. Autom. Control* **2006**, *51*, 1289–1298. [CrossRef]
29. Lin, J.; Morse, A.S.; Anderson, B.D.O. The multi-agent rendezvous problem. In Proceedings of the IEEE International Conference on Decision and Control, Maui, HI, USA, 9–12 December 2003; pp. 1508–1513.
30. Ji, M.; Egerstedt, M. Distributed Coordination Control of Multi-Agent Systems While Preserving Connectedness. *IEEE Trans. Robot.* **2007**, *23*, 693–703. [CrossRef]
31. Park, H.; Hutchinson, S. An efficient algorithm for fault-tolerant rendezvous of multi-robot systems with controllable sensing range. In Proceedings of the Robotics and Automation (ICRA), 2016 IEEE International Conference, Stockholm, Sweden, 16–21 May 2016; pp. 358–365.
32. Li, Q.; De Rosa, M.; Rus, D. Distributed Algorithms for Guiding Navigation across a Sensor Network. In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, MobiCom'03, San Diego, CA, USA, 14–19 September 2003; pp. 313–325.
33. Li, Q.; Aslam, J.; Rus, D. Distributed Energy-conserving Routing Protocols for Sensor Networks. In Proceedings of the IEEE Hawaii International Conference on System Science, Big Island, HI, USA, 6–9 January 2003.
34. Lavalle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.