Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6639574, 23 pages
https://doi.org/10.1155/2021/6639574

Research Article

UWB Base Station Cluster Localization for Unmanned Ground Vehicle Guidance

Yuzhan Wu,[1] Susheng Ding,[2] Yuanhao Ding,[2] and Meng Li [3]

[1]*Beihang University, Beijing, China*
[2]*School of Information Science and Technology, Zhejiang Sci-Tech University (ZSTU), Hangzhou, China*
[3]*College of Big Data and Internet, Shenzhen Technology University, Shenzhen, China*

Correspondence should be addressed to Meng Li; limeng2@sztu.edu.cn

Received 30 December 2020; Revised 7 April 2021; Accepted 10 April 2021; Published 23 April 2021

Academic Editor: Ding Wang

In this paper, we seek to provide unmanned ground vehicles with positioning service using ultrawideband (UWB) technology, a high-accuracy positioning approach. UWB is chosen for two distinct reasons. First, it does not rely on global navigation satellite systems like GPS, making it able to be applied indoors or in an environment where GPS signal is unstable. Second, it is immune to interference from other signals and accurate enough to guide unmanned ground vehicles moving precisely in a complex environment within a narrow road. In this paper, three UWB base stations are aggregated as a group in a 2D space for localization. A large number of tests are performed with a UWB base station cluster in order to validate its positioning performance. Based on the experiment results, we further develop a dynamic particle swarm optimization-based algorithm and a genetic algorithm to deploy multiple clusters of UWB base stations to cover an area of interest. The performance of the proposed algorithms has been tested through a series of simulations. Finally, experiments using unmanned ground vehicles are carried out to validate the localization performance. The results confirm that the robots can follow complex paths accurately with the proposed UWB-based positioning system.

1. Introduction

Accurate positioning is essential for robot control while challenging to achieve, especially in an unstructured environment. Many applications, such as multirobot formation control and robot path planning, strongly rely on positioning accuracy either indoors or outdoors. Depending on the environment, positioning technologies can be roughly classified into outdoor localization, such as GPS, and indoor positionings, among which we can find Bluetooth, WiFi, Zigbee, and UWB technologies. As UWB-based localization can achieve higher accuracy positioning compared with other techniques, it is chosen in this paper to guide unmanned ground vehicles (UGVs).

UWB has been a particularly attractive localization technology, especially for indoor positioning. Normally, UWB base stations, also known as beacons or anchor nodes, are scattered in the area of interest. Then, a UWB tag or agent needs to communicate with several base stations to obtain its position [1, 2]. One advantage of this scheme is an area of interest can be covered with fewer base stations, reducing the cost. Its drawback is that it is challenging to achieve time synchronization among anchor nodes. Thus, complex communication protocols need to be designed to eliminate clock offset among anchors, and multiple communication packages are required between tags and anchors for distance measurements or for localization. As a result, this strategy works well for fewer tags but is unable to provide real-time positioning for a tag swarm. Besides, it is also hard to obtain accurate positions of anchors during deployment or installation when it has a long distance between two anchors, which can introduce positioning errors for the entire system.

In response to these problems, we apply UWB beacon clusters for localization. To be specific, a group of base stations can be regarded as an independent positioning

device, which is able to provide highly accurate position information for tags within its coverage range. This is very useful for swarm robotic control with the leader-follower structure [3–5] because one cluster can be equipped on the leader robot, providing highly accurate positioning information for its followers. Since the beacons in one cluster are near each other, the installation accuracy can be guaranteed. Furthermore, it is feasible to share the same clock source among beacons in the same group, achieving clock synchronization. In this way, it is promising to realize accurate positioning with less communication, so that more tags can be supported simultaneously.

In order to demonstrate our idea, a relatively simple situation is first considered in this paper, where three base stations are combined as a group to provide 2D petitioning for tags. Plenty of experiments are carried out to test the performance of the UWB cluster to determine the proper distances among anchor nodes. Three anchors are installed as equilateral triangles with different side lengths. In order to guarantee the system performance and improve efficiency, we develop a test platform that can adjust the side length continuously and an app used to record the positions of tags automatically. Based on the experiment results, an optimal distance between a pair of base stations and the range a beacon group can cover are obtained. Secondly, two strategies are explored to deploy UWB clusters to cover an area of interest. The main challenges for these strategies are the irregularity of UWB cluster coverage, the randomness of the shape of the area of interest, and uncertainty in the number of UWB clusters. Thus, we develop two algorithms to determine the best solution to deploy beacon clusters, namely, a dynamic particle swarm optimization- (DPSO-) based method and a genetic algorithm- (GA-) based method. Finally, the experiments of UGVs are performed based on UWB positioning to validate the performance of the localization system. The results show that UGVs can follow multiple complex paths. The maximum error is within 30 cm, and the average error is around 6.7 cm. Furthermore, one beacon group can simultaneously support multiple UGVs.

The main contributions of this paper are as follows:

(i) An idea is proposed to combine several UWB base stations as one positioning cluster, which can potentially support more tags simultaneously, laying the ground for future application in swarm robotics

(ii) Experiments are carried out to test the positioning performance of a UWB group, concluding the tendency of positioning accuracy with the coverage range and the change of distances between base stations, which can be used as reference guidelines for the future application of UWB clusters

(iii) Two algorithms are proposed to deploy UWB clusters to cover an area of interest, addressing the issues of the irregularity of UWB cluster coverage, the randomness of the shape of the area of interest, and uncertainty in the number of UWB clusters

The remaining of the paper is organized as follows. In Section 2, current works related to positioning for robots are presented. In Section 3, the developed test platform is introduced and the experiment results are provided. In Section 4, two proposed algorithms for deploying a set of UWB groups into a random area of interest are explained. In Section 5, we introduce the design of a UGV prototype and experiment results of UGVs following different trajectories under the positioning of a UWB cluster. Finally, in Section 6, concluding remarks are provided.

2. Related Work

Positioning plays an essential role in robot control. Various positioning methods have been proposed so far, with different features and suitable applications. The most widely used positioning technique for robots in an outdoor environment is GPS [6, 7]. It is low in cost and applicable to various seniors. Two main drawbacks constrain its further applications, i.e., its dependency on satellites and its low accuracy, with positioning error reaching meters. These two drawbacks encourage the study and application of other positioning techniques on robots.

Positioning techniques that do not rely on satellites include inertial positioning [8–10], point cloud-based positioning [11–13], and a large family of positioning techniques generally called indoor positioning techniques [14–16].

Inertial positioning has a long history, and lots of great achievements have been made so far in this field. This approach estimates positions according to acceleration and is widely used in many applications such as underwater positioning since it does not rely on any external information. However, the integral of acceleration introduces cumulative errors and finally affects the estimation accuracy. So far, various hybrid positioning methods have been proposed by combining inertial positioning with other methods, such as GPS [17, 18], acoustic ranging [19], and Doppler speedometer [20].

The point cloud-based positioning includes a family of methods using kinds of sensors to detect environments around and represent sensing data as a point cloud. From the point cloud, the environment around a robot can be reconstructed. By comparing the environment with a digital map, the robot can recognize its position. Point clouds can be generated using various sensors suitable for different environments. Laser radar [11] can reach an accuracy of millimeters but is expensive. Although millimeter-wave radar costs less, it correspondingly provides few data points in space [12]. An infrared (IR) array can be equipped around a robot to provide a set of range information [21]. The provided data points are so rare that they can only be used for positioning in a structured environment. Ultrasonic sensors are used in a similar way as an IR array. Due to the low cost, they are equipped with various robots, for example, the AmigoBot [22]. From the perspective that positioning is achieved via comparing a reconstructed digital environment with a preinput map, vision-based localization can also be regarded as a special point cloud-based positioning, treating pixels as data points in the cloud. It is a rapidly developed field and various schemes have been proposed, such as positioning by recognizing landmarks [23–25] or detecting

motion by comparing successive images [26]. Novel algorithms are mainly based on deep learning, enabling robots to construct a 3D map with only one camera [13]. However, vision-based positioning heavily relies on a preconstructed map, making it hard to fit for dynamic environments. Furthermore, it is prone to be deceived by similar scenes.

Indoor positioning includes a family of positioning technologies suitable for GPS-denied areas, which include WiFi, Bluetooth, FRID, magnetic field, and UWB. Usually, these methods rely on beacons or base stations. For example, the WiFi-based positioning requires to predeploy a set of APs in the environment [27], and the magnetic field-based positioning method requires to preembed magnetic nails on the floor [28, 29]. At present, WiFi-based and Bluetooth-based methods mainly use the attenuation of signals to estimate distance, probably resulting in high positioning errors reaching 10 meters. The positioning accuracy may be enough for guiding customers in malls, but not enough for robot guidance. FRID-based positioning is realized by recognizing a tag when the tag is near to a reader. However, the distance between them needs to be close enough, constraining its application. Magnetic field-based positioning is precise enough to guide robots, making it suitable for automatic parking and automatic storage [30, 31]. However, the deployment of magnetic nails needs to be completed during the construction of the building. UWB-based positioning is achieved by measuring the distances between a base station and a tag using the time of flight, two-way ranging, or time of arrival, etc. The centimeter-level accuracy can be achieved by using UWB technology [32–34].

In this paper, we seek for a positioning method suitable for robots moving in a complex area. This requires high positioning accuracy and working with fewer constraints. These requirements leave us limited choices. GPS, WiFi, Bluetooth, and RFID are not suitable due to large positioning errors. Furthermore, GPS is prone to fail indoors. The point cloud-based localization schemes such as IR, laser, and ultrasonic either are expensive or require pregiven maps for localization. Image-based positioning methods suffer from the same problem. As a result, UWB positioning is chosen as the solution considering the balance among performance, environment adaptation, and cost. Besides high positioning accuracy, UWB technology also has the advantages of high transmission rate, low power consumption, and strong anti-interference, making it suitable for guiding robots. In this paper, the positioning is realized based on ranging between base stations and a tag using two-way ranging. For more details about current research of UWB-based positioning, refer to [33] and other related literature studies.

3. Performance of UWB Station Cluster

As mentioned in Section 1, three UWB base stations are combined as one group in 2D space, providing positioning information for tags within the group's coverage region, i.e., the overlap coverage region of the three base stations. To avoid bias caused by asymmetry, three base stations are initially placed as an equilateral triangle. Besides, we try to make side lengths of a triangle small so that the UWB cluster can be made as one small device. Therefore, the effect of side lengths on the localization accuracy needs to be determined by checking the positioning accuracy of tags in different locations.

3.1. Test Settings. The test platform is shown in Figure 1, which consists of two tracks with an angle of 60°. The sliders can move continuously along guide rails. TT DC motor is installed on each slider, which is reserved for automatic testing in the future. One UWB beacon is installed at the intersection of the extension of two rails, which is defined as the origin (0, 0) in the Cartesian coordinate system of the UWB cluster. The other two UWB beacons are mounted on top of the two sliders.

In order to facilitate the testing process including data recording and data analysis, a Windows app is developed as shown in Figure 2. The app can record UWB tag distances and positions, show tag trajectories, and control the movement of a UGV. It can also be used to update positions of UWB base stations.

3.2. Test Results. In this part, UWB-based ranging and positioning tests are carried out to determine the proper side edges of one UWB cluster. Due to the limitations of the physical environments, the maximum distance between a UWB tag and a base station is around 8 m in a room and around 38 m in a lobby. In order to achieve the best performance, the tests are conducted under line-of-sight propagation conditions. The absolute positions of sampling points are depicted in Figures 3 and 4 , respectively. For each point, more than 50 samples are collected.

The mean distance errors between a UWB tag and base stations are shown in Figures 5 and 6 , respectively. These experiments are carried out in a room and in a lobby in order to determine the ranging performance of UWB. In a room, the mean distance errors are less than 20 centimeters in most cases. Within 40 meters, the maximum mean distance error is less than 40 centimeters. The distance errors grow with the increase of the distance between a station and a tag, indicating by Figures 5 and 6. This implies that the positioning error will be large when a tag is far from the three base stations.

Meanwhile, mean positioning errors are measured in the same environments. The corresponding results are depicted in Figures 7 and 8 , respectively. At present, two side lengths, i.e., 50 cm and 60 cm, are verified. From Figure 7, it can be seen that the performance with two side lengths is similar. Then, more experiments are conducted in a lobby with long distance, whose results are shown in Figure 8. The mean position error increases approximately in proportion to the distance from the origin in a long-term perspective. The mean position errors are no more than 60 centimeters in most cases within a 10-meter distance. Therefore, 50 cm side length is chosen for a UWB cluster and a 10-meter distance is determined as the coverage radius of a UWB base station.

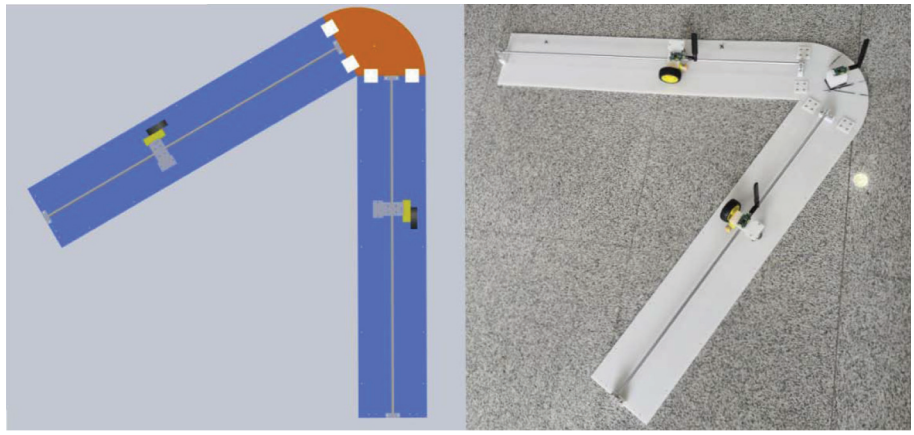

Figure 1: The test platform of a UWB cluster.

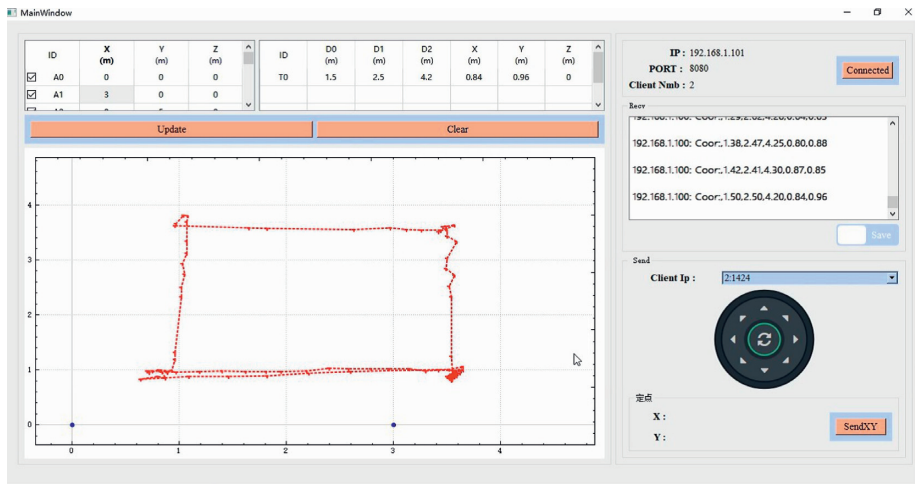

Figure 2: A Windows app developed mainly to show tag traces and to control the movement of a UGV.

4. Region Coverage with Multiple UWB Clusters

Based on the experiment results in Section 3, both the proper side lengths of the triangle composed of three UWB base stations and the range covered by each UWB cluster can be determined.

In general, for a complex area of interest, one group of UWB base stations is unable to cover the whole area. Then, we need to find the best solutions to deploy base station groups. There are three main challenges:

(1) The region covered by all the three base stations are the overlap area of three discs, which is irregular, making it hard to determine if a subarea is covered by the group

(2) The area of interest is of random shape with inaccessible region

(3) Because of (2), the number of UWB groups necessary to cover the area of interest cannot be determined in advance

To solve this problem, we first approximate the overlap region of the three base stations as a disc, as explained in Section 4.1. Then in Sections 4.2 and 4.3, two algorithms are proposed to deploy UWB base station groups to cover a randomly generated area. Finally, in Section 4.4, the performance of the two algorithms is compared.

4.1. Approximate the Covered Region as a Circle. In this paper, UWB positioning is achieved through ranging between base stations and tags. In 2D space, if one robot wants to determine its position, one option is to obtain distance measurements to at least three base stations that are not in a line. In our case, three base stations are located near to each other, forming a group. To determine positions with the group, a robot needs to be within the coverage range of each of them, i.e., it should be within the overlap region of the coverage range of all the three base stations, as shown in Figure 9. We call this overlap region S and obviously it is irregular. To simplify the problem, S needs to be approximated into a regular shape, and there are two choices. We can either use an inscribed triangle S_t or a circumcircle S_c to approximate S, and the one whose area is closest to that of S is the solution. Thus, it is required to calculate the area of S, S_t, and S_c. To simplify the expression, in the following, S, S_t, and S_c are used to represent the area of the corresponding region.

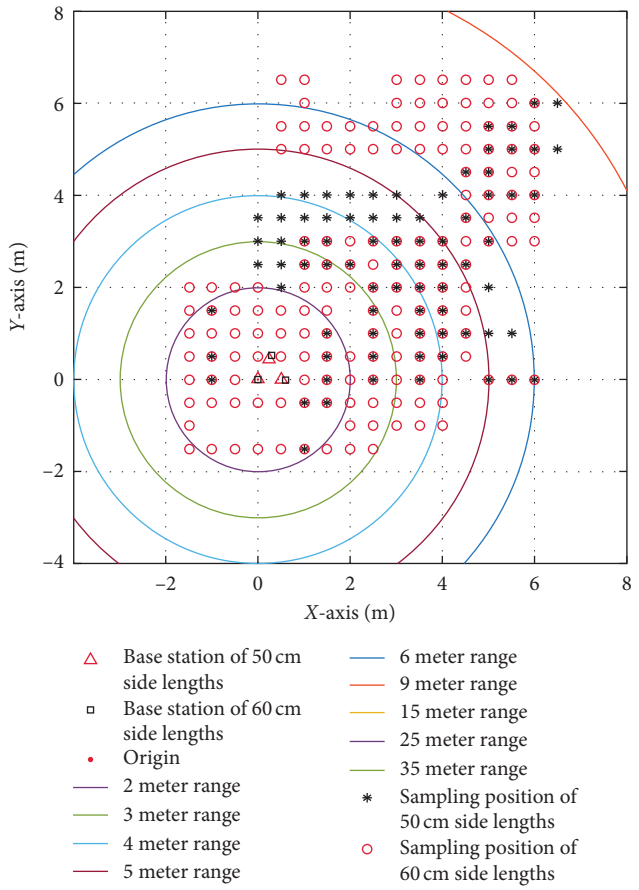

Figure 3: Sampling point distribution in a room with the side length of UWB stations being 50 cm and 60 cm, respectively.

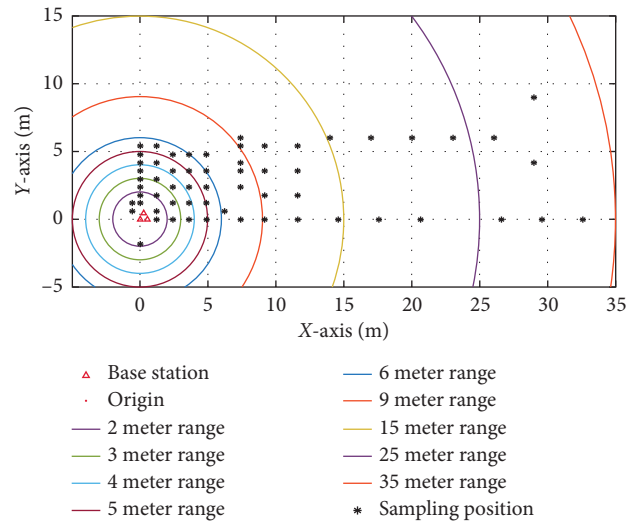

Figure 4: Sampling point distribution in a lobby with the side length of UWB stations being 50 cm.

In our case, three base stations are located at $(x_i, y_i), i = 1, 2, 3$, composing an equilateral triangle. As all the three base stations are identical in hardware configuration, the coverage range is supposed to be the same, which is a disc with a radius of R. With the information, S can be determined. The three vertices of S are also the vertices of S_t, which can be obtained with the following procedure.

Define the vertices as $x_k^{i,j}$, where $i, j, k \in \{1, 2, 3\}$ and i, j, k are different, representing the vertex who is a cross point of disks centered by (x_i, y_i) and (x_j, y_j), and this cross point is within the range of the third disk centered by (x_k, y_k). Then, we have

$$\begin{aligned} \left(x_k^{i,j} - x_i\right)^2 + \left(y_k^{i,j} - y_i\right)^2 &= R^2, \\ \left(x_k^{i,j} - x_j\right)^2 + \left(y_k^{i,j} - y_j\right)^2 &= R^2, \\ \left(x_k^{i,j} - x_k\right)^2 + \left(y_k^{i,j} - y_k\right)^2 &< R^2. \end{aligned} \tag{1}$$

With the expressions above, we can determine the three vertices. Furthermore, we can obtain any edge length of S_t by computing the distance between the corresponding two vertices. For example,

$$l_t^{1,2} = \sqrt{\left(x_3^{1,2} - x_2^{1,3}\right)^2 + \left(\left(y_3^{1,2} - y_2^{1,3}\right)^2\right.}. \tag{2}$$

As the center of discs is placed as an equilateral triangle, thus the three vertices are equivalent, making S_t also an equivalent triangle, whose edges have

$$l_t := l_t^{1,2} = l_t^{1,3} = l_t^{2,3}. \tag{3}$$

Thus,

$$S_t = \frac{\sqrt{3}}{4} l_t. \tag{4}$$

For S_c, its center can be obtained with

$$(x_c, y_c) = \left(\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3}\right). \tag{5}$$

The radius of S_c can be measured as the distance between (x_c, y_c) and a random vertex $(x_k^{i,j}, y_k^{i,j})$, as

$$l_c = \sqrt{\left(x_c - x_k^{i,j}\right)^2 + \left(y_c - y_k^{i,j}\right)^2}. \tag{6}$$

So the area of S_c can be obtained as follows:

$$S_c = \pi (l_c)^2. \tag{7}$$

The overlap region S is a bit tricky, as shown in Figure 10. It consists of the inscribed triangle S_t and three arch form areas, represented as S_{arch}^k, where k indicates the center of the circle corresponding to the arch and $k = 1, 2, 3$. Then,

$$S = S_t + \sum_{k=1}^{3} S_{\text{arch}}^k. \tag{8}$$

The area of S_{arch}^k can be obtained with

$$S_{\text{arch}}^k = S_{\text{fan}}^k - S_{\text{triangle}}^k, \tag{9}$$

where S_{fan}^k is the corresponding fan centered by (x_k, y_k) and S_{triangle} is the triangle with three vertices of (x_k, y_k), $(x_i^{k,j}, y_i^{k,j})$, and $(x_j^{k,i}, y_j^{k,i})$, as shown in Figure 10.

To calculate the area of S_{fan}, it requires its radius R and its span angle θ. R is obtained by experiments, and θ is lacking.

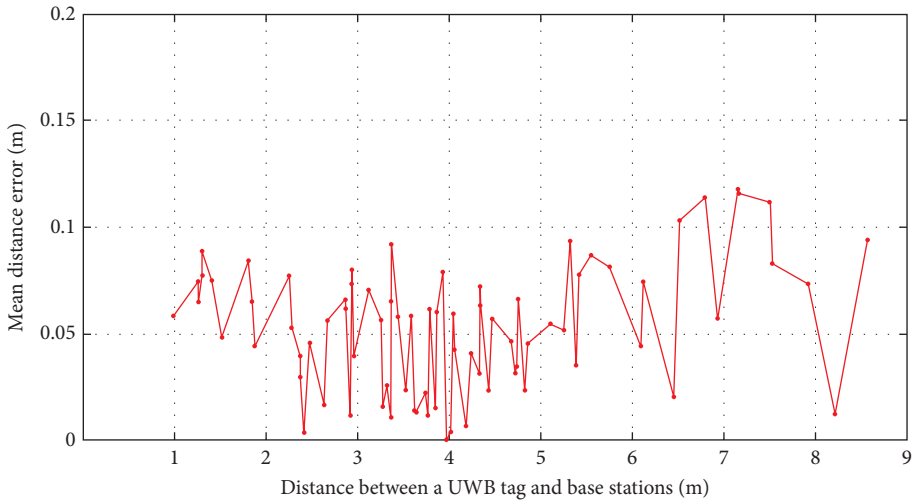

Figure 5: Mean distance errors between a UWB tag and base stations in a room.

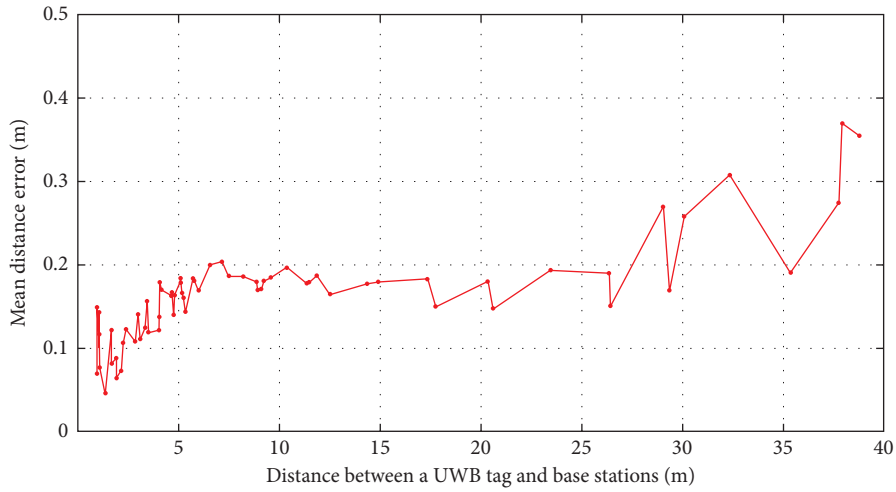

Figure 6: Mean distance errors between a UWB tag and base stations within 40 meters in a lobby.

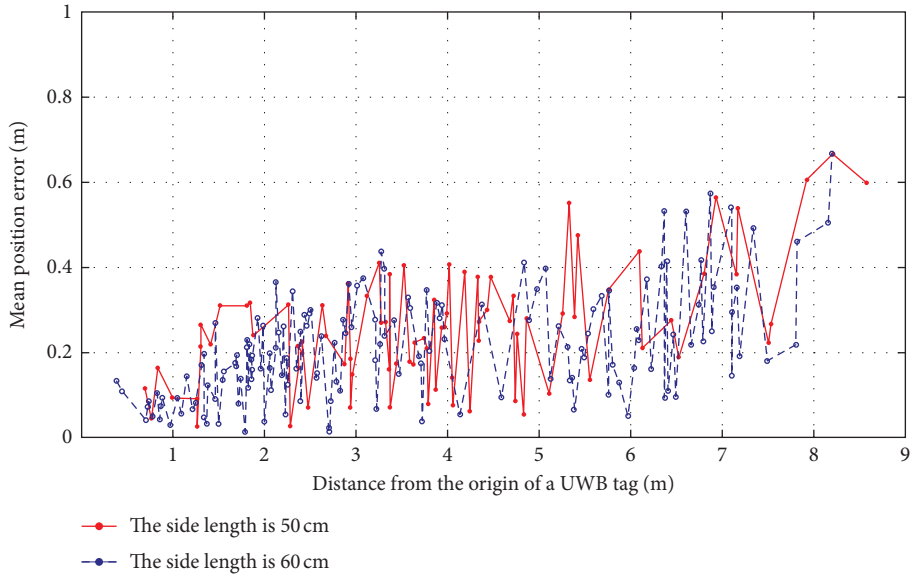

Figure 7: Mean position errors of a UWB tag in a room with different side lengths.

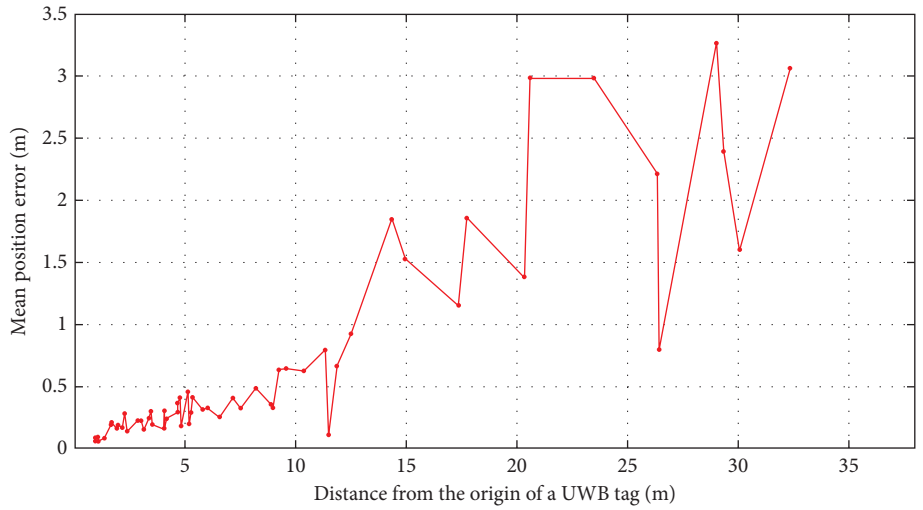

FIGURE 8: Mean position errors of a UWB tag with 50 cm side length in a lobby.

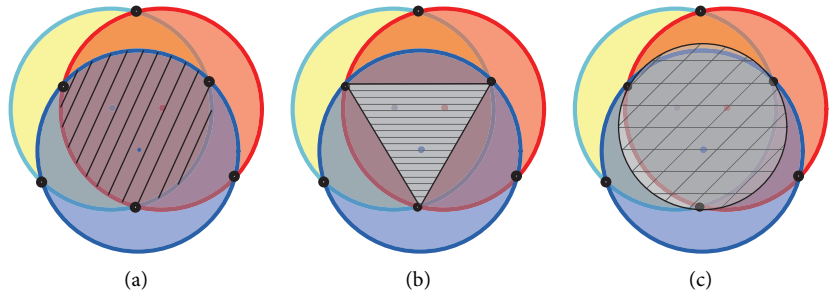

FIGURE 9: The overlap region and its approximation. Shadow in (a) indicates the overlap region of the coverage range of three UWB base stations, i.e., S. Shadow in (b) approximates S with its inscribed triangle S_t, and shadow in (c) approximates S with its circumcircle S_c.

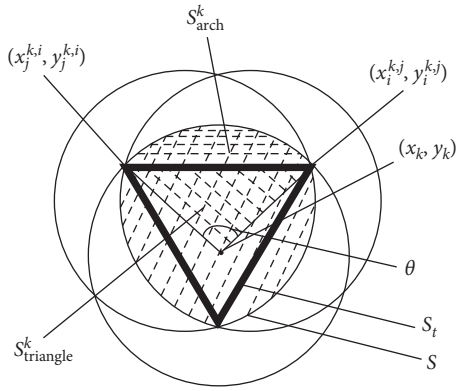

FIGURE 10: An auxiliary figure for calculating S.

To calculate θ, we first get two vectors starting from the center of the disk (x_k, y_k) and pointing at the other two points as

$$\begin{aligned}\vec{v}_{k,i} &= \left(x_i^{k,j} - x_k, y_i^{k,j} - y_k\right),\\ \vec{v}_{k,j} &= \left(x_j^{k,i} - x_k, y_j^{k,i} - y_k\right).\end{aligned} \tag{10}$$

Since

$$\vec{v}_{k,i} \cdot \vec{v}_{k,j} = \left|\vec{v}_{k,i}\right|\left|\vec{v}_{k,j}\right|\cos\theta, \tag{11}$$

then

$$\theta = a\cos\frac{\left(x_i^{k,j} - x_k\right)\left(x_j^{k,i} - x_k\right) + \left(y_i^{k,j} - y_k\right)\left(y_j^{k,i} - y_k\right)}{\sqrt{\left(x_i^{k,j} - x_k\right)^2 + \left(y_i^{k,j} - y_k\right)^2}\sqrt{\left(x_j^{k,i} - x_k\right)^2 + \left(y_j^{k,i} - y_k\right)^2}}. \tag{12}$$

The area of the fan can be obtained as

$$S^k_{\text{fan}} = \pi R^2 \cdot \frac{\theta}{2\pi} = \frac{\theta}{2}R^2, \tag{13}$$

where S_{triangle} is an isosceles triangle with two edges being R and the base line is the chord of the arch, whose length is

$$l_{\text{chord}} = \sqrt{\left(x_i^{k,j} - x_j^{k,i}\right)^2 + \left(y_i^{k,j} - y_j^{k,i}\right)^2}. \tag{14}$$

Then the area of the triangle can be calculated with Heron's formula. Half of the perimeter is given by

$$l_{\text{hp}} = \frac{1}{2}\left(2R + l_{\text{chord}}\right). \tag{15}$$

Then the area is

$$S^k_{\text{triangle}} = \sqrt{l_{\text{hp}}\left(l_{\text{hp}} - R\right)^2\left(l_{\text{hp}} - l_{\text{chord}}\right)}. \quad (16)$$

With S^k_{fan} and S^k_{triangle},

$$S^k_{\text{arch}} = S^k_{\text{fan}} - S^k_{\text{triangle}}. \quad (17)$$

As the three arch form areas are equivalent, the area of S then is

$$S = S_t + \sum_{k=1}^{3} S^k_{\text{arch}} = S_t + 3S^k_{\text{arch}}, \quad (18)$$

where k could be 1, 2, or 3.

Compared with S, S_t is a bit smaller while S_c is a bit larger. Define the rate of approximate error as

$$r_t = \frac{\left|S - S_{\text{approximate}}\right|}{S}, \quad S_{\text{approximate}} \in S_t, S_c. \quad (19)$$

The change of approximation error is shown in Figure 11. If the distances among base stations are small, approximate the overlap region as a disc is a better choice. In our case, according to Section 3, three base stations are placed as an equilateral triangle, whose side length is 50 cm. The error rate is 1.9023%. This means the disk can approximate the actual overlap region with a minor error. Then methods to place multiple such disks to cover an area of a random shape can be considered in the following.

4.2. Deployment with Dynamic Particle Swarm Optimization. We intend to place multiple groups of UWB base stations to provide positioning services for robots. In Section 4.1, it is shown that the irregular overlap region of the three base stations can be approximated as a disk reasonably. Then the problem becomes where to place these disks and how many such disks are needed. Therefore, the deployment scheme is a trade-off between coverage rate and cost.

Inspired by [35], a PSO-based algorithm is proposed in this paper that can dynamically adjust the number of UWB clusters. The algorithm is called dynamic PSO, or short as DPSO.

Generally, DPSO consists of two loops. The outer loop adjusts the number of station groups, while the inner loop adjusts the positions of station groups following the idea of PSO. Assume that in the i_{out} iteration of the outer loop, N^{out}_i station groups are used. Then, for its inner loops, the particles are encoded as the following sequence:

$$P = \left[x_1, x_2, \ldots, x_{N^{\text{out}}_i}, y_1, y_2, \ldots, y_{N^{\text{out}}_i}\right], \quad (20)$$

where $(x_j, y_j) \in S$ and (x_j, y_j), $j = 1, 2, \ldots, N^{\text{out}}_i$, indicates the position of the j^{th} station group.

The fitness function is set to maximize the coverage rate

$$\arg\max f(P) = \frac{S_P}{S}, \quad (21)$$

where S_P is the valid area covered by the station groups and S is the area of the valid area in the map. "Valid" refers to only the region accessible for the robots counts. To measure S_P and S in an irregular area, we mesh the map into a set of grids and estimate areas by counting the number of grids. A swarm of particles is generated randomly in the init process. Then the particles search in the solution space guided by some historical best solutions randomly, as introduced in [36]. Besides, the particle P should not move out of the valid area S, so that a filter is designed. According to the idea above, the pseudocode of DPSO is presented in Algorithm 1. Key functions of Algorithm 1 are detailed in Algorithms 2 and 3, respectively. A general flowchart of Algorithm 1 is shown in Figure 12.

Thus, Algorithm 1 has a complexity of $O((N_{\max} - N_{\min}) \times (N_p + N_{\text{in}} \times N_{\text{swarm}} \times N_g))$. In Algorithm 1, the inputs are the valid area S and the radius R of the approximate coverage disk. The output is a set S_{best}, consisting of best particles with different numbers of base station clusters. The parameters for the algorithm are set in lines 1-2. In line 4, the range of numbers of clusters to be explored is calculated. The floor is the number of groups needed to stuff the area with disks, which is $S/(\pi R^2)$, and the ceil is the number of groups needed to stuff the area with the inscribed square of the disk, which is $S/(\sqrt{2}\,R)^2$. The loop in lines 5–20 is the outer loop, which increases the number of groups by one each time, expending the length of the particle. In each iteration, the particle swarm P, the velocity set V, the historical best solution for each particle p_{best}, and the best solution for all particles g_{best} are reset. Then, within the inner loop in lines 10–18, the velocity and position of each particle are updated randomly, affected by g_{best} and p_{best}. Then g_{best} and p_{best} are updated.

In line 6, we use function GeneratePos to obtain a random particle, as in Algorithm 2. The challenge here is that S is of random shape, and there is a high chance that the particle obtained with a general random number generation algorithm will go out of range. To solve this problem, in GeneratePos, we first scatter S into a set of grids and then stack the grids up into a sequence. The grids in the sequence are then labeled successively. We generate i^{out} random numbers within the range of $[1, N_s]$, where N_s is the number of the grids. These random numbers are the indexes of grids chosen as elements in the particle. The complexity of Algorithm 2 is $O(N_p)$.

In line 7, the function GenerateVel() initializes the velocity of a particle with a zero sequence having the same size of the particle.

In line 14, we use function Constrain to ensure particles will not move out of S, as in Algorithm 3. According to the encoding rule, the particle represents the coordinates of the station groups. If a station group moves out of S, we just ignore the update of its position and corresponding velocity this time. Meanwhile, the updates of velocities and positions for other station groups are not affected. Algorithm 3 has a complexity of $O(N_g)$.

In lines 9, 15, and 17, the fitness function $f(\cdot)$ needs to be calculated to choose the best particle. As S is irregular, a meshing scheme is applied to estimate the coverage rate:

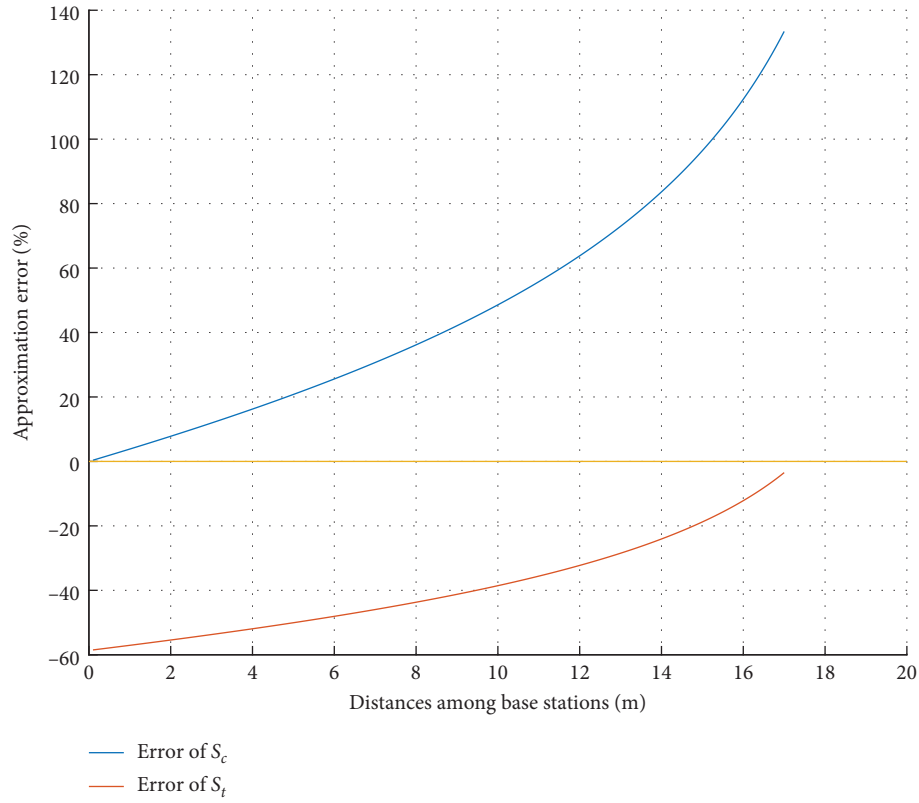

FIGURE 11: Approximation errors of S_c and S_t with the increase of distances among base stations. When the distances among stations are less than 1 m, the approximation error of S_c is less than 5%, while that of S_t reaches 60%.

$$f(p) = \frac{S_P}{S} = \frac{N_{\text{covered}}}{N_S}, \tag{22}$$

where N_{covered} is the number of grids covered by at least one of the station groups and N_S is the number of grids in S. For each grid, we calculate the distance from its center to all the center of station groups as $\text{dis} = \{d_i | i = 1, \ldots, i^{\text{out}}\}$. If $\min(\text{dis}) < R$, we treat the grid being covered. In this way, N_{covered} can be obtained.

Running this algorithm, we notice that there is a chance that the best particle with more station groups performs worse than that with fewer station groups, as shown in Figure 13. This is because in line 6 of Algorithm 1, each time the number of station groups is changed, the particle swarm is regenerated randomly. The red dots in Figure 13(a) indicate that, with 48 stations, the coverage rate is higher than that of 49 and 50 station groups. Clearly, the solution in Figure 13(b) performs better, with fewer station groups. There is a chance that the init swarm is rather bad, and we do not iterate enough times in the inner loop to obtain a good solution. It is not easy to check and decide when to increase the iteration number for the program automatically. With DPSO, the tendency is that more station groups will increase coverage rate, but not stable.

To solve this problem without increasing the iteration number, the DPSO algorithm is improved into the Elite DPSO algorithm, which ensures that the more UWB clusters are used, the better solution can be obtained. Elite DPSO is almost the same as DPSO, but line 6 in Algorithm 1 is changed as

$$\begin{aligned} P &\coloneqq P_{\text{random}} \cup P_{\text{elite}}, \\ P_{\text{random}} &\coloneqq \{p_k | p_k \leftarrow \text{GeneratePos}, \text{k} = 2, \ldots, N_{\text{swarm}}\}, \\ P_{\text{elite}} &\coloneqq \{p_{\text{elite}} \leftarrow \text{GenerateElite}(g_{\text{best}})\}. \end{aligned} \tag{23}$$

This means that each time the group number is increased, $N_{\text{swarm}} - 1$ particles are randomly regenerated, as is done in DPSO. A special particle called elite particle is also generated with function $\text{GenerateElite}(g_{\text{best}})$ based on the best particle g_{best} obtained in the last iteration of the outer loop. It should perform better than g_{best}. Two kinds of $\text{GenerateElite}(g_{\text{best}})$ are explored.

One way to generate elite particle is to add a cluster with a random position in S to g_{best}. In the worst case, the newly added cluster overlaps with another cluster in g_{best}, resulting in $f(p_{\text{elite}}) = f(g_{\text{best}})$. Otherwise, p_{elite} covers more region, ensuring that $f(p_{\text{elite}}) \geq f(g_{\text{best}})$. This scheme is named as Random Elite DPSO.

Another way to generate elite particle is to add a cluster far from other clusters in g_{best}, encouraging station groups to move to a relatively empty region. Assume that the newly added cluster is located at (x, y) and other clusters in g_{best} are located at $(x_i, y_i), i = 1, \ldots, i^{\text{out}}$. Then, the optimal position can be obtained with

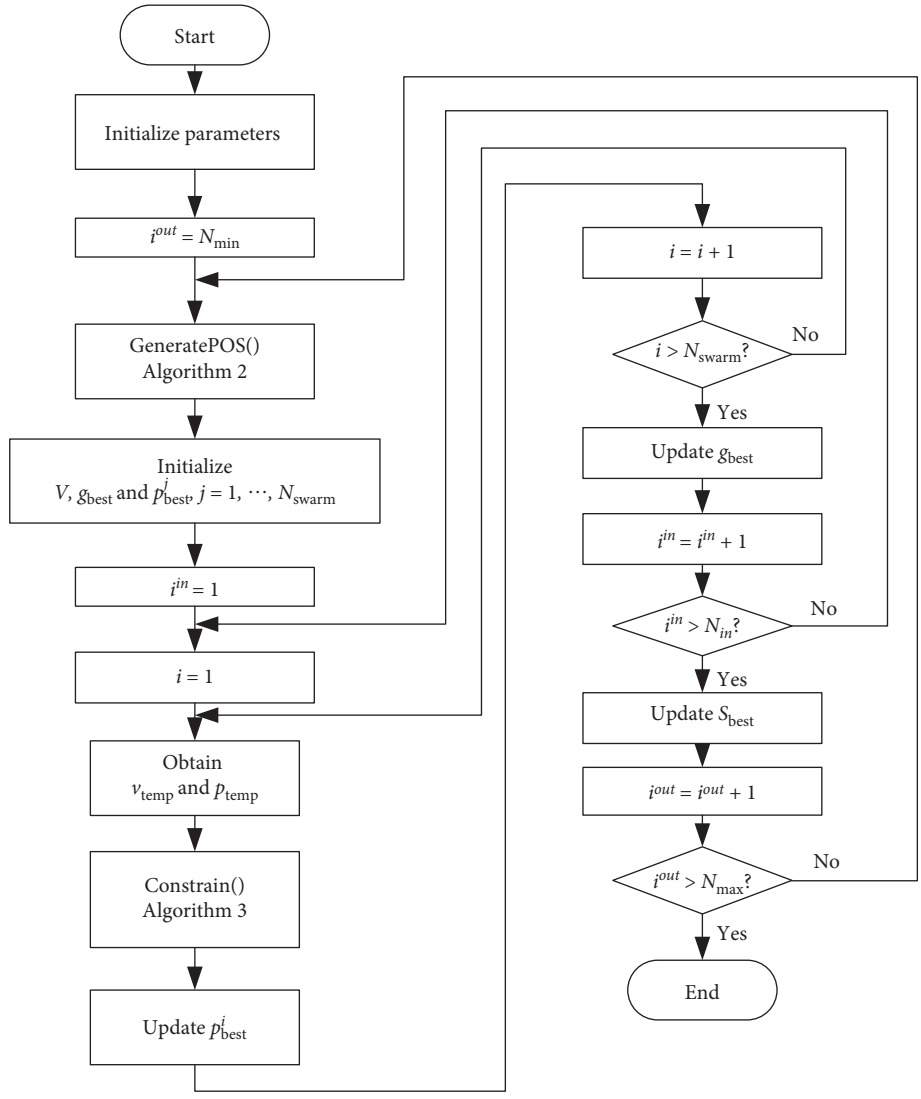

Figure 12: The flowchart of Algorithm 1.

$$\arg\max\{d(x,y)\} = \frac{\sum_{i=1}^{i^{\text{out}}} (x_i - x)^2 + (y_i - y)^2}{i^{\text{out}}}. \quad (24)$$

Apparently, $d(x, y)$ is a convex function, so its maximum value is obtained with (x, y) on the border of the region. All grids at the border of S are exhausted to obtain (x, y). This method is named Sparse Elite DPSO because it intends to encourage the locations of station groups to be sparse.

The performance of both schemes is compared, showing in Figure 14. From the figures, the Sparse Elite DPSO method performs slightly better than the Random Elite DPSO method. However, considering the computation cost on average, the Random Elite DPSO is a more reasonable choice.

4.3. Deployment with Genetic Algorithm. The DPSO methods proposed in Section 4.2 explore a set of group numbers and provide the best solutions correspondingly. However, they are unable to determine the best group number automatically. A GA-based algorithm is then developed, which can provide the best number of clusters needed to cover a region and the deployment scheme, indicating positions of clusters.

We scatter the area of interest with random shape into N_S grids and use the center of the grids to represent them. Thus, these grids are labeled as $g_1, g_2, \ldots, g_{N_S}$. Then in GA, the chromosome is encoded as an element vector with the length of N_S. Each element is either 0 or 1. If the i^{th} element is 1, then a cluster is deployed at the center of grid g_i. Otherwise, a cluster will not be deployed. This implies that

(i) A cluster can only be located at the center of a grid, unable to move continuously

(ii) A chromosome always represents a feasible solution because the grid sequence only contains accessible areas

The fitness function is given by

Input: valid area S, radius of approximate disk R
Output: set S_{best} consists of best particles with different lengths
(1) Set parameters: swarm size N_{swarm}, iteration number N_{in}
(2) Set parameters for PSO: ω, c_1, c_2
(3) $S_{\text{best}} \leftarrow \phi$
(4) Group number range: $N_{\min} = (S/\pi R^2)$, $N_{\max} = (S/(\sqrt{2}R)^2)$
(5) **for** $i^{\text{out}} = N_{\min}, N_{\min}+1, \ldots, N_{\max}$ **do**
(6) $P \coloneqq \{p_k | p_k \leftarrow \text{GeneratePos}(S, i^{\text{out}}), k = 1, \ldots, N_{\text{swarm}}\}$
(7) $V \coloneqq \{v_k | v_k \leftarrow \text{GenerateVel}, k = 1, \ldots, N_{\text{swarm}}\}$
(8) $p^j_{\text{best}} \leftarrow p_j, j = 1, \ldots, N_{\text{swarm}}$
(9) $g_{\text{best}} \leftarrow \arg\max f(p_j), j = 1, \ldots, N_{\text{swarm}}$
(10) **for** $i^{\text{in}} = 1, 2, \ldots, N_{\text{in}}$ **do**
(11) **for** $i = 1, 2, \ldots, N_{\text{swarm}}$ **do**
(12) $v_{\text{temp}} \leftarrow \omega v_i + c_1 r(\cdot)(p_{\text{best}} - p_i) + c_2 r(\cdot)(g_{\text{best}} - p_i)$
(13) $p_{\text{temp}} \leftarrow p_i + v_{\text{temp}}$
(14) $\{p_i, v_i\} \leftarrow \text{Constrain}(S, v_{\text{temp}}, p_{\text{temp}}, v_i, p_i)$
(15) $p^i_{\text{best}} \leftarrow \arg\max f(p), p \in \{p_i, p^i_{\text{best}}\}$
(16) **end for**
(17) $g_{\text{best}} \leftarrow \arg\max f(p), p \in \{p^1_{\text{best}}, \ldots, p^{N_{\text{swarm}}}_{\text{best}}\}$
(18) **end for**
(19) $S_{\text{best}} \leftarrow S_{\text{best}} \cup \{g_{\text{best}}\}$
(20) **end for**

Algorithm 1: Dynamic PSO.

Input: the area of interest S, particle length N_p
Output: random particle p
(1) **function** GENERATEPOS(S, i^{out})
(2) Scatter S into N_S grids, stacking as a sequence S_{sequence}: $S_{\text{sequence}} = [s_1, \ldots, s_{N_S}]$
(3) $p_x \leftarrow, p_y \leftarrow$
(4) **for** $i = 1, 2, \ldots, N_p$ **do**
(5) Index$\leftarrow$ceil$(\text{rand} \times (N_S - 1))$
(6) $p_x \leftarrow [p_x, s_{\text{Index}}(x)], s_{\text{Index}} \in S_{\text{sequence}}$
(7) $p_y \leftarrow [p_y, s_{\text{Index}}(y)], s_{\text{Index}} \in S_{\text{sequence}}$
(8) **end for**
(9) $p \leftarrow [p_x, p_y]$
(10) end function

Algorithm 2: function GeneratePos().

$$f(g) = \frac{S_{g1}}{S} = \frac{N_{\text{covered1}}}{N_S}, \tag{25}$$

where S_{g1} indicates the sum of the area only covered by one cluster, N_{covered1} is the number of grids whose center is covered only by one cluster, and N_S stands for the total number of grids. The ratio of N_{covered1} and N_S is used to estimate the fitness function value. Defining the fitness function in this way encourages clusters to cover more grids, which satisfies the target that the whole area of interest to be covered. Meanwhile, this definition can prevent the algorithm from adding too many clusters to achieve a high coverage rate because the increase of overlap rate results in a decrease of fitness value. In this way, the algorithm can adjust the number of clusters automatically.

Then a swarm of chromosomes is generated. The fitness function is used to evaluate and choose the chromosomes. A cross operator is used to create a new chromosome, and a mutation operator is applied to add randomness to the chromosomes, as the idea introduced in [37]. The algorithm is shown in Algorithm 4. Details of functions in Algorithm 4 are further introduced in Algorithms 5–9 . A general flowchart of Algorithm 4 is shown in Figure 15.

In Algorithm 4, necessary parameters are set in lines 1–3. Then the area S is scattered into grids, helping to calculate the fitness function value and enable us to decode the best chromosome. In line 5, the swarm is generated using a function GenerateChromosome(), and the chromosome with the biggest fitness value is chosen as the current best chromosome, which is presented in detail in Algorithm 5. Then the loop enters lines 7–14. In each iteration, the fitness values of chromosomes in the swarm are first calculated. Then go through selection operation as introduced in Algorithm 6, cross operation as introduced in Algorithm 7, and

Input: the area S, original particle, and velocity p, v and the raw value based on them $p_{\text{temp}}, v_{\text{temp}}$
Output: updated particle and velocity p, v

```
(1) function CONSTRAIN(S, p, v, p_temp, v_temp)
(2)   Number of station groups N_g = length(p)/2
(3)   for i = 1, 2, ..., N_g do
(4)     (p_x^o, p_y^o) ← (p(i), p(i + N_g))
(5)     (v_x^o, v_y^o) ← (v(i), v(i + N_g))
(6)     (p_x^r, p_y^r) ← (p_temp(i), p_temp(i + N_g))
(7)     (v_x^r, v_y^r) ← (v_temp(i), v_temp(i + N_g))
(8)     if (p_x^r, p_y^r) ∉ S then
(9)       p(i) ← p_x^o
(10)      p(i + N_g) ← p_y^o
(11)      v(i) ← v_x^o
(12)      v(i + N_g) ← v_y^o
(13)    else
(14)      p(i) ← p_x^r
(15)      p(i + N_g) ← p_y^r
(16)      v(i) ← v_x^r
(17)      v(i + N_g) ← v_y^r
(18)    end if
(19)  end for
(20) end function
```

Algorithm 3: Function Constrain().

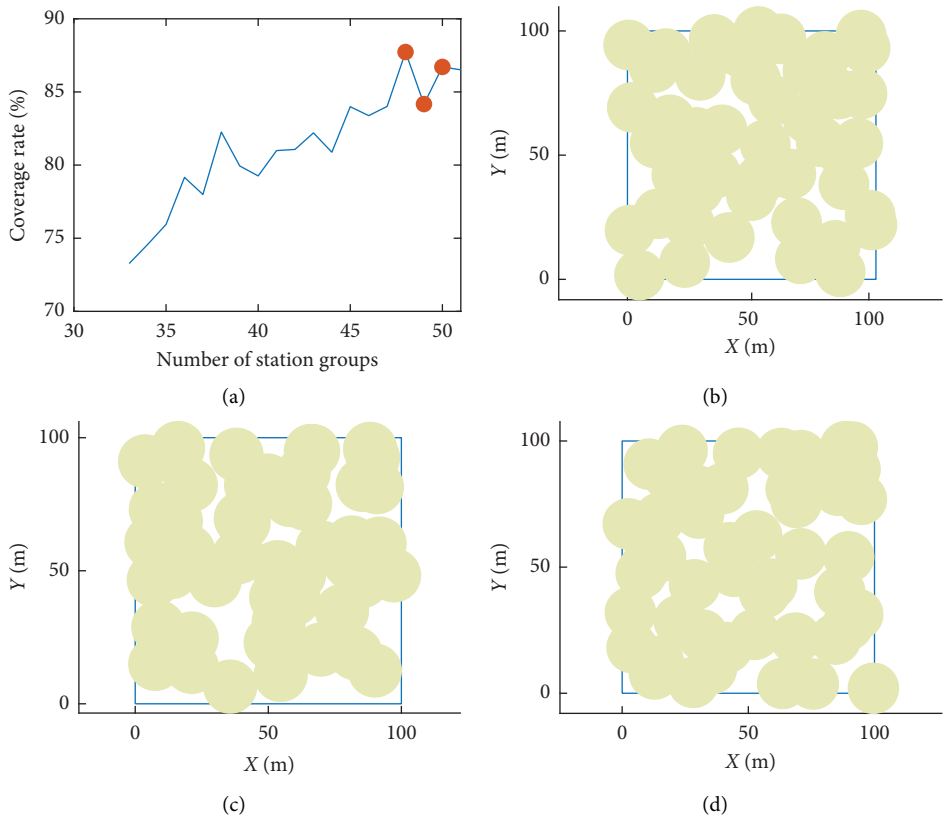

Figure 13: Performance of DPSO. (a) The coverage rate changes with the increase of station groups. (b–d) The corresponding solutions provided by DPSO, with 48, 49, and 50 station groups, respectively. The yellow regions are covered by UWB clusters. (a) Performance of DPSO. (b) Group number = 48. (c) Group number = 49. (d) Group number = 50.

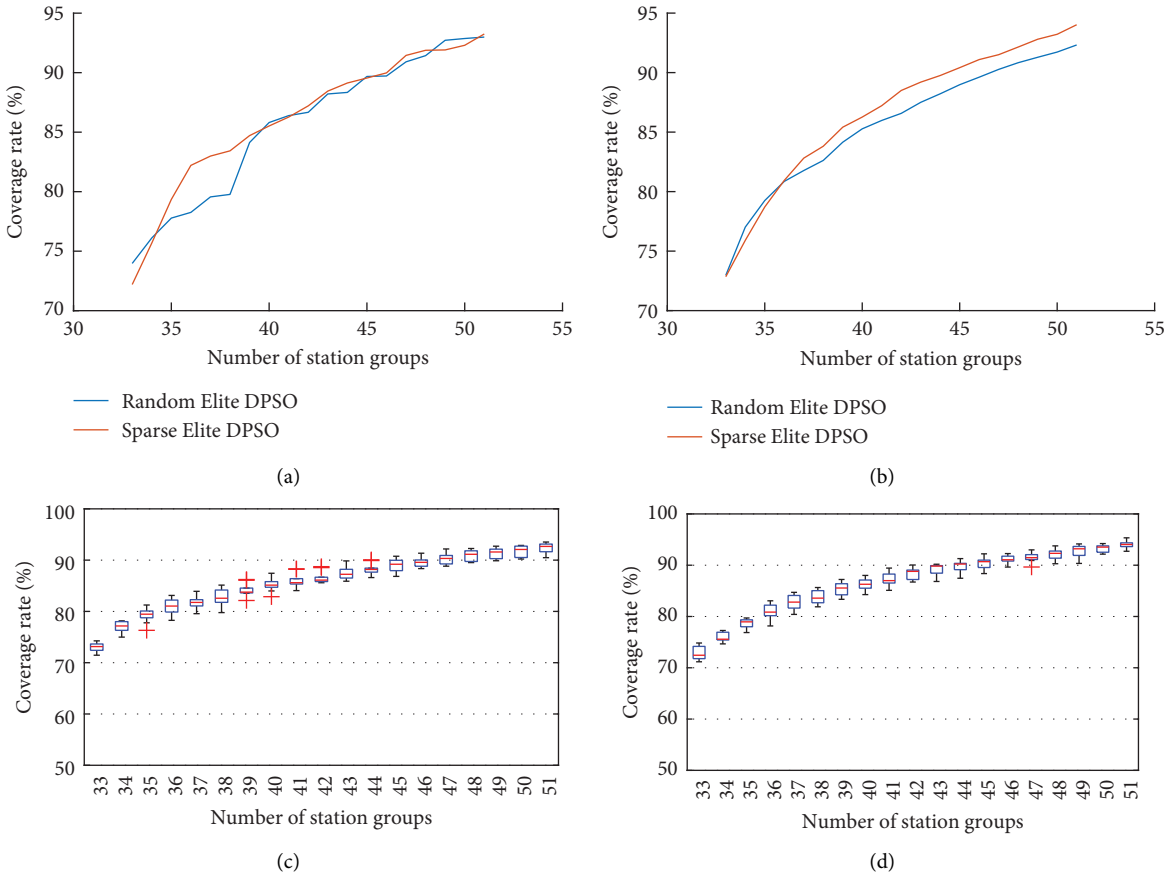

Figure 14: Performance of the Elite DPSOs. (a) Typical performance of the Random Elite DPSO and the Sparse Elite DPSO. With the increase of station groups, the coverage rate always increases. (b) Comparison of the average coverage of the Random Elite DPSO and the Sparse Elite DPSO in terms of computation cost. Boxplots in (c) and (d) show the statics results of the two Elite DPSOs. The data are obtained by repeating experiments 10 times. (a) Typical performance. (b) Average performance. (c) Random Elite DPSO. (d) Sparse Elite DPSO.

Input: valid area S, radius of approximate disk R
Output: positions for station groups: $P = \{(x_i, y_i): i = 1, \ldots, N_{\text{best}}\}$
(1) Set parameters: swarm size N_{swarm}, iteration number N_{iter}
(2) Set parameter: mutation rate μ
(3) Set parameter: swarm creation parameter p_{init}
(4) Scatter S into grids, labeled as $S = [g_1, g_2, \ldots, g_{N_S}]$
(5) Create swarm: $C \coloneqq \{c_k | c_k \leftarrow \text{GENERATECHROMOSOMEN}_S, p_{\text{init}}, k = 1, \ldots, N_{\text{swarm}}\}$
(6) Initialize the best chromosome: $c_{\text{best}} \leftarrow \arg\max f(c), c \in C$
(7) **for** $i = 1, \ldots, N_{\text{iter}}$ **do**
(8) Calculate fitness value: $F \leftarrow \{f(c_i): c_i \in C\}$
(9) Selection operation: $C_{\text{new}} \leftarrow \text{SELECTIONC}, F$
(10) Cross operation: $C_{\text{new}} \leftarrow \text{CROSSC}, C_{\text{new}}$
(11) Mutation operation: $C_{\text{new}} \leftarrow \text{MUTATIONC}_{\text{new}}, \mu$
(12) Update the swarm: $C \leftarrow C_{\text{new}}$
(13) Update the best chromosome: $c_{\text{best}} \leftarrow \arg\max f(c), c \in c_{\text{best}} \cup C$
(14) **end for**
(15) Decode: $P = \text{DECODES}, c_{\text{best}}$

Algorithm 4: Generate deployment scheme with GA.

mutation operation as introduced in Algorithm 8. Finally, update the best chromosome c_{best} by comparing if a better solution is obtained in this iteration. Function Decode() is used to decode the chromosome into cluster positions, which is presented in detail in Algorithm 9. Algorithm 4 has a complexity of $O(N_{\text{swarm}} \times N_s \times N_{\text{iter}})$.

Input: N_S, p_{init}
Output: random chromosome c
(1) **function** GENERATECHROMOSOME (N_S, p_{init})
(2) Generate N_S random numbers obey normal distribution: $c = [c_1, \ldots, c_{N_S}]$
(3) **for** $i = 1, 2, \ldots, N_S$ **do**
(4) **if** $c_i > p_{\text{init}}$ **then**
(5) $c_i \leftarrow 1$
(6) **else**
(7) $c_i \leftarrow 0$
(8) **end if**
(9) **end for**
(10) end function

Algorithm 5: Function GenerateChromosome().

Input: C, F
Output: new swarm C_{new}
(1) **function** SELECTION (C, F)
(2) **for** $i = 1, \ldots, N_{\text{swarm}}$ **do**
(3) $r_i \leftarrow (f_i / \sum_{j=1}^{N_{\text{swarm}}} f_j), f_i, f_j \in F$
(4) **end for**
(5) $R_{\text{floor}} = [0], R_{\text{ceil}} = [r_1]$
(6) **for** $i = 1, \ldots, N_{\text{swarm}} - 1$ **do**
(7) $R_{\text{floor}} \leftarrow [R_{\text{floor}}, \sum_{j=1}^{i} r_j]$
(8) $R_{\text{ceil}} \leftarrow [R_{\text{ceil}}, \sum_{j=1}^{i+1} r_j]$
(9) **end for**
(10) $C_{\text{new}} \leftarrow \phi$
(11) **for** $i = 1, \ldots, N_{\text{selection}}$ **do**
(12) $C_{\text{new}} \leftarrow C_{\text{new}} \cup \{c_k\}$, where $c_k \in C$
With $r_k^{\text{floor}} < \text{rand} < r_k^{\text{ceil}}$
Where $r_k^{\text{floor}} \in R_{\text{floor}}, r_k^{\text{ceil}} \in R_{\text{ceil}}$
(13) $C \leftarrow C \setminus \{c_k\}$
(14) **end for**
(15) end function

Algorithm 6: Function Selection().

Input: C, C_{new}
Output: new swarm C_{new}
(1) **function** CROSS (C, C_{new})
(2) **for** $i = 1, \ldots, |C| - |C_{\text{new}}|$ **do**
(3) $p_1 \leftarrow \text{Rand}(C)$
(4) $C \leftarrow C \setminus p_1$
(5) $p_2 \leftarrow \text{Rand}(C)$
(6) Exchange elements in p_1, p_2 after random point, obtain s_1, s_2
(7) $s \leftarrow \text{Rand}(s_1, s_2)$
(8) $C_{\text{new}} \leftarrow C_{\text{new}} \cup s$
(9) **end for**
(10) end function

Algorithm 7: Function Cross().

Function GenerateChromosome() randomly generates a sequence whose length is the same as the number of grids in S, with each element corresponding to a grid. The sequence only consists of 0 and 1, with 1 indicating a cluster to be deployed at the center of a grid. To generate such a sequence, i.e., a chromosome for the GA, we first

Input: C_{new}, μ
Output: new swarm C_{new}
(1) **function** MUTATION (C_{new}, μ)
(2) **for** $c \in C_{\text{new}}$ **do**
(3) **for** $i = 1, 2, \ldots, N_S$ **do**
(4) **if** rand $< \mu$ **then**
(5) $c_i \leftarrow / c_i, c_i \in c$
(6) **end if**
(7) **end for**
(8) **end for**
(9) end function

Algorithm 8: Function Mutation().

Input: S, c_{best}
Output: positions for station groups P
(1) **function** DECODE (S, c_{best})
(2) $P = \phi$
(3) **for** $i = 1, \ldots, N_S$ **do**
(4) **if** $c_i = 1, c_i \in c_{\text{best}}$ **then**
(5) $P \leftarrow P \cup (x_i, y_i)$
Where (x_i, y_i) is the center of $g_i \in S$
(6) **end if**
(7) **end for**
(8) end function

Algorithm 9: Function Decode().

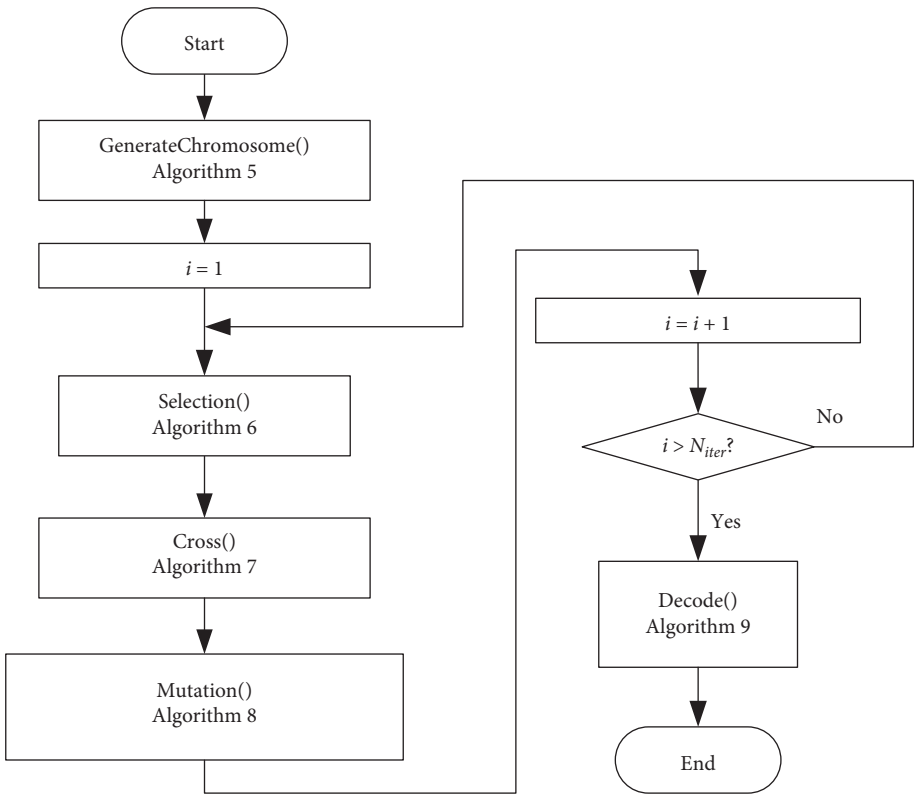

Figure 15: The flowchart of Algorithm 4.

generate N_s random values obeying normal distribution, as line 2 in Algorithm 5. If the value is larger than parameter p_{init}, it is set as 1, and 0 otherwise, as lines 4–8. Algorithm 5 has a complexity of $O(N_s)$.

It is obvious that the choice of p_{init} affects the number of clusters deployed initially and affects the optimization process indirectly. Thus, setting a proper p_{init} is important. Our suggestion of setting p_{init} is to follow

$$N_S P(x > p_{\text{init}}) = \frac{S}{S_c}, \tag{26}$$

where P is a probability distribution and S_c is the area covered by one cluster. In this paper, the normal distribution is chosen for P. Then,

$$P(x \le p_{\text{init}}) = \frac{1}{N_S}\left(N_S - \frac{S}{S_c}\right). \tag{27}$$

A proper p_{init} can be obtained by solving this equation.

Function Selection() presented in Algorithm 6 takes current swarm C and the fitness values of these chromosomes F as inputs. With a roulette scheme, half of the chromosomes are selected into the next generation, which means $N_{\text{selection}} = (1/2)N_{\text{swarm}}$. The roulette scheme is achieved by creating a floor sequence R_{floor} and ceiling sequence R_{ceil} based on the ratio of a chromosome's fitness value to the sum of the fitness values, shown in lines 2–9. R_{floor} and R_{ceil} form a sequence of slots. A random number is generated and falls into a slot. The chromosome corresponding to the slot is selected, as in lines 12-13. Algorithm 6 has a complexity of $O(N_{\text{swarm}})$.

The selection operation presented in Algorithm 6 is a general operation in GA and it works well in various scenarios. However, it performs poorly in our case, as shown in Figure 16. The reason is that the fitness values of a swarm are similar, resulting in a roulette scheme unable to select proper chromosomes for the next generation effectively.

This problem is solved by adopting an elite chromosome scheme. The idea is to choose some best chromosomes and put them directly into the next generation. Then, lines 11–14 in Algorithm 6 are replaced with the following:

(i) Sort elements in C as descending order of elements in F

(ii) $C_{\text{new}} \leftarrow c_1, c_2, \ldots, c_{N_{\text{selection}}} \in C$

$N_{\text{selection}}$ is the number of chromosomes directly passed to the next generation. To add more randomness, we set $N_{\text{selection}} = (1/4)N_{\text{swarm}}$. From Figure 16, it can be seen that this elite chromosome scheme can ensure that the fitness value never decreases. Besides, the algorithm can automatically adjust the number of clusters. Comparing with the roulette scheme, the elite chromosome scheme can achieve a higher fitness value and a coverage rate with fewer clusters. This point is clearly shown in Figure 17, where we exhibit the deployment of clusters according to solutions obtained with GA using two different selection operators. In both cases, 35 clusters are deployed. The solution with the elite scheme resulting in clusters spreading more evenly and covering more area than that of the roulette scheme.

Function Cross() in Algorithm 7 generates the rest chromosomes. It randomly picks two chromosomes as parents, exchanges their elements after a random point, and generates two children (lines 2–6). Then, in lines 7–8, a random child is chosen and added to the swarm. Algorithm 7 has a complexity of $O(N_{\text{swarm}})$.

Function Mutation() in Algorithm 8 randomly generates N_S numbers. If the number is larger than the mutation parameter μ, the corresponding element in the chromosome is inverse from 0 to 1 or from 1 to 0. Attention should be paid that μ should be set as a relatively small number, as a big μ introduces 1s rapidly during iterations as shown in Figure 18, overwhelming the algorithm's ability to adjust cluster numbers. Algorithm 8 has a complexity of $O(N_s \times N_{\text{swarm}})$.

Function Decode() in Algorithm 9 decodes the best chromosome into a set of coordinates for cluster deployment. Algorithm 9 has a complexity of $O(N_s)$.

4.4. Comparison of the Two Schemes. Experiments have been carried out to test the algorithms provided in Sections 4.2 and 4.3. The results in a rectangle region and a randomly generated irregular region are presented here, as shown in Figure 19. For the PSO, the swarm size is set as 20 and the iteration number is 50. The station group number increased from the number of clusters necessary to just fulfill the area ignoring the shape, which is 21. The solution of GA contains 29 clusters. To make it comparable, in DPSO, we also set the maximum number of clusters as 29. It turns out that the coverage can reach 87.98%. For the Elite GA, the swarm size is set as 50 and iteration number 200. Parameter μ is set as 0.00001 and p_{init} is automatically set with the method introduced in Section 4.3. The solution provided by the Elite GA contains 29 clusters, with a coverage rate reaching 89.87%, slightly better than that of the Elite DPSO. Taking the randomness of both algorithms into consideration, we cannot claim the Elite GA is better than the Elite DPSO. In fact, the algorithms have different features, and some discussion is provided in the following.

DPSO can provide a set of particles corresponding to a different number of groups. Thus, users can choose the best one according to the cost and the coverage rate. GA automatically decides how many clusters are needed, and users can directly use the scheme generated. However, users are unable to assign the number of clusters.

In DPSO, base station groups move continually in the area of interest, while, in GA, base station groups jump among grids. This means DPSO has more flexibility than GA. If we are seeking for a more precise solution with GA, the area of interest has to be meshed into more grids, increasing the length of the chromosome and resulting in more computation. For example, if a region of $100 \times 100\,\text{m}^2$ is scattered into $1 \times 1\,\text{m}^2$ grids, the length of the chromosome is $(100 \times 100/1 \times 1) = 10^4$. In case the region is rather complex, we have to divide it into grids of $0.1 \times 0.1\,\text{m}^2$. Then, the length of the chromosome becomes $(100 \times 100/0.1 \times 0.1) = 10^6$, increasing the computation dramatically.

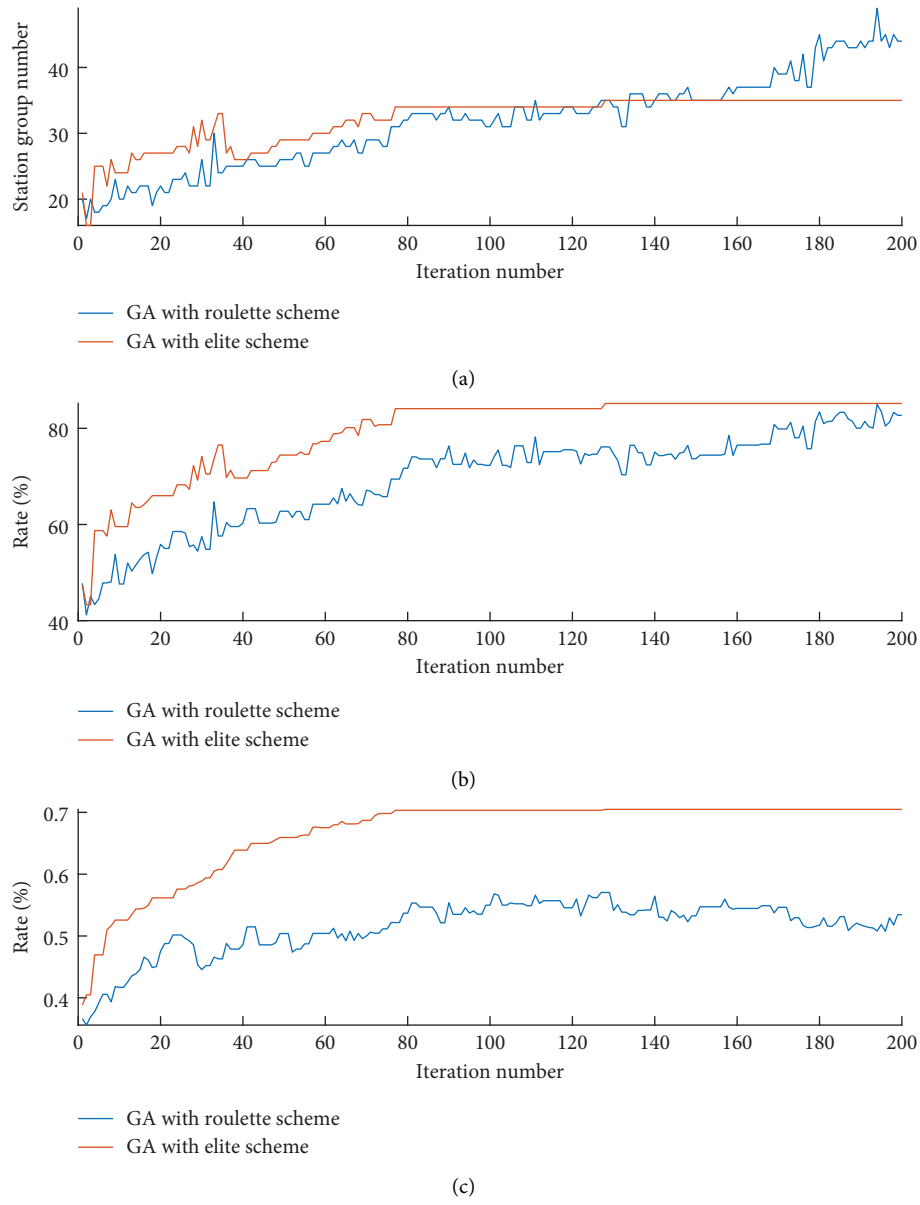

Figure 16: Performance of GA with a roulette scheme and an elite scheme during iterations. (a) The change of cluster number of the best chromosome for each iteration. (b) The coverage rate of the best chromosome. (c) The fitness value, i.e., the rate of the area only covered by one cluster.

In conclusion, both methods can provide a scheme to deploy base station clusters, and the choice depends on the users' preference.

5. Experiment Validation of UGV Navigation Based on UWB Cluster Positioning

In Section 3, the performance of a UWB cluster is tested and validated, and then in Section 4, the deployment methods of UWB clusters are proposed and validated by simulations. In this section, the outcomes of the previous sections are applied to UGVs. Based on the experiments in Section 3, 50 cm side length is chosen for a UWB cluster and 10-meter distance is determined as the coverage radius of a UWB base station. UGVs are then set out to follow different paths, showing that under the positioning of UWB groups, errors of UGV trajectories are constrained around 25 cm. With UWB clusters, UGVs can move smoothly to their destinations.

5.1. Design of UGVs. In order to validate the navigation performance based on the UWB cluster positioning system, UGV prototypes are fabricated. The control kernel hardware is composed of an MCU, a UWB tag, an IMU, a WiFi module, and actuators. The structure is shown in Figure 20. The MCU acts as the "brain" of a UGV, receiving input information and making control decisions. A 32-bit MCU,

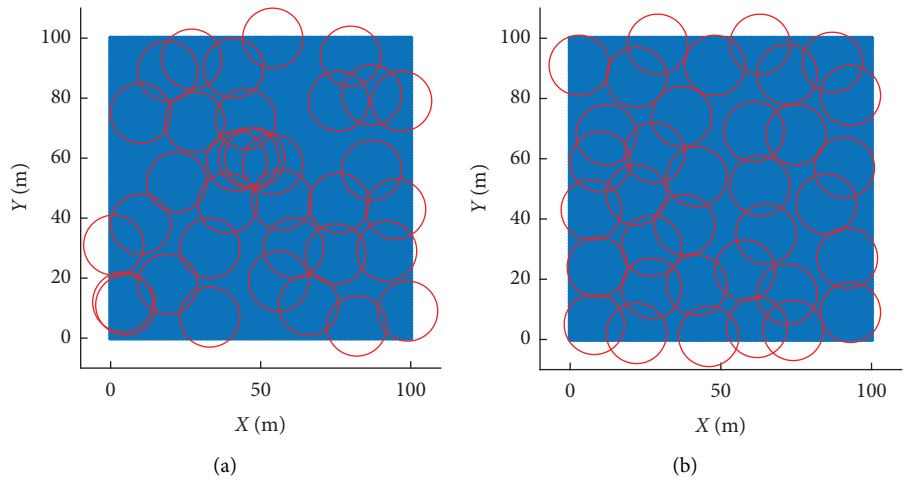

Figure 17: Deployment of clusters according to solutions obtained with GA using a roulette selection scheme and an elite selection scheme. The area with a blue background is the area of interest, and each red circle indicates the coverage range of a cluster. (a) Roulette scheme. (b) Elite scheme.

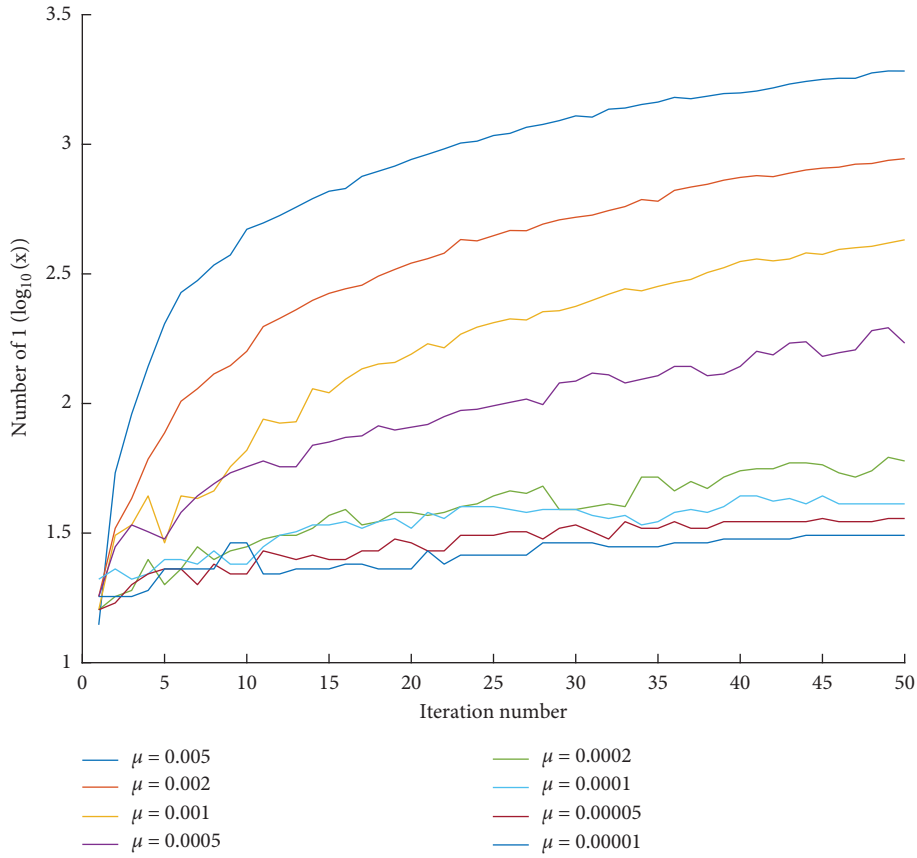

Figure 18: Change of the number of 1s in the best chromosome with different μ. As the number of 1s increases dramatically with bigger μ, use $\log_{10}(\cdot)$ to record the numbers.

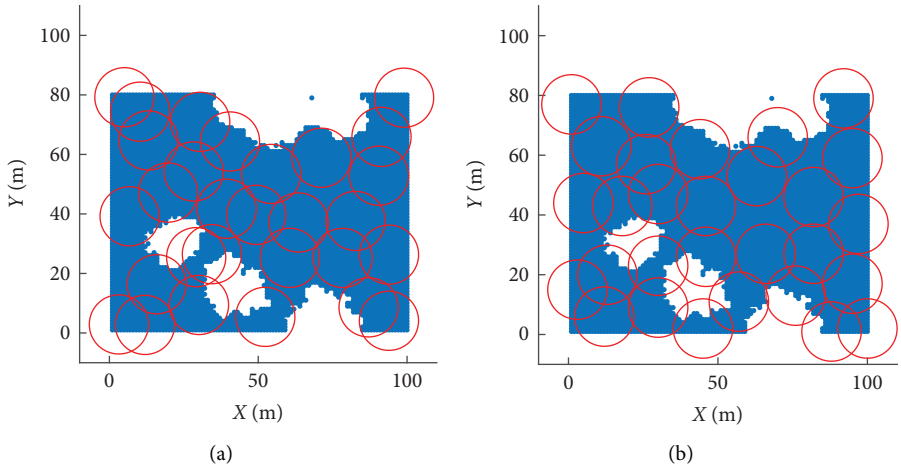

Figure 19: Deployment of clusters into an irregular area of interest according to solutions obtained with (a) the elite DPSO and (b) the elite GA.

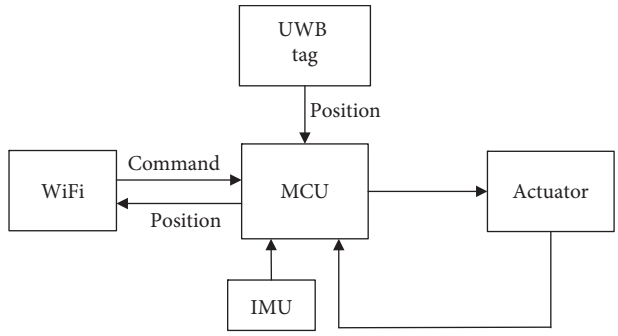

Figure 20: UGV control kernel hardware structure.

STM32H750VB, is adopted, which includes an Arm Cortex-M7 with 128 Kbytes of flash memory, 1 MB RAM, 480 MHz CPU, L1 cache, large set of peripherals. The UWB tag uses DWM1000 of Decawave. For IMU, MPU-9250 is applied, which has a 9-axis motion processing unit including a gyroscope, an accelerometer, and a digital compass. The WiFi connection is established using a low-cost WiFi microchip, ESP8266, with a full TCP/IP stack. The operation of DC motors is driven by A4950s, full-bridge DMOS PWM motor drivers. The UGV prototype is shown in Figure 21.

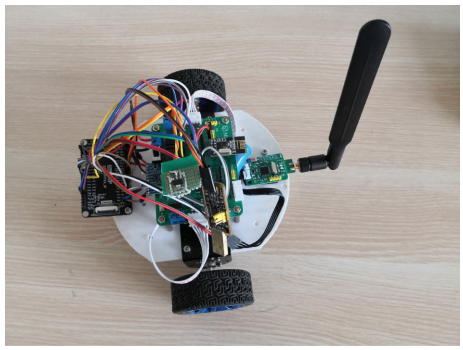

Figure 21: UGV with UWB tag.

5.2. UGV Automated Driving under UWB Positioning. In this part, experiments are carried out with UGVs using a UWB cluster positioning system. First, UGV automated driving is realized in an 8 m×9 m room, as shown in Figure 22. A UGV is programmed to follow designated trajectories shown in Figure 23. It can complete fixed-point back and forth trajectory, square trajectory, and circular trajectory. For each test, target positions or waypoints are generated. The UGV checks whether the current position is within a circle around these positions to guarantee that it follows the designated paths. The live demonstration video is provided in supplemental materials. The test results of a circular trajectory are depicted in Figure 24 and the corresponding positioning errors are provided in Figure 25. In all the experiments, the position errors are within 30 cm as shown in Figure 26, which are consistent with test results in Section 3.2.

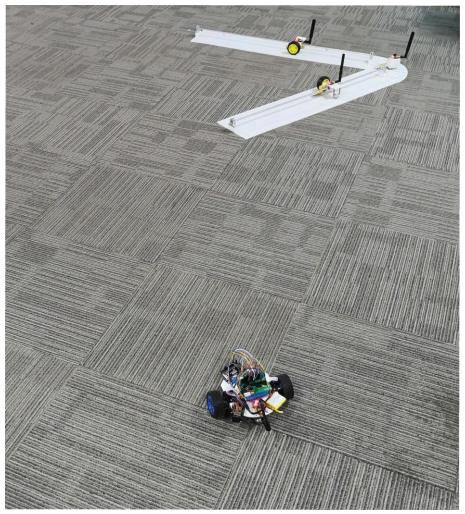

Figure 22: UWB positioning-based UGV automated driving test.

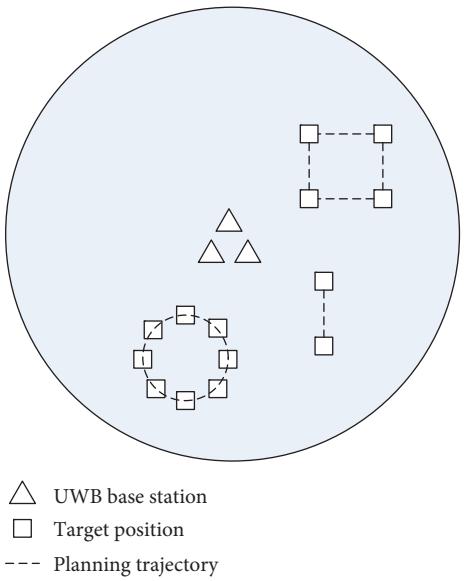

Figure 23: UWB positioning-based UGV automated driving test.

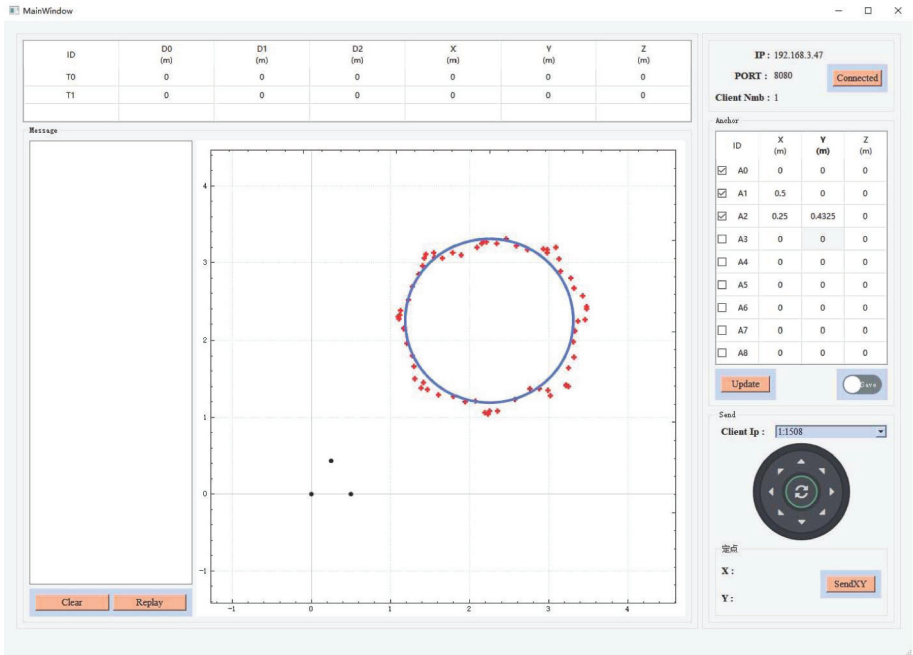

Figure 24: Field test results of a circular trajectory monitored by the Windows app.

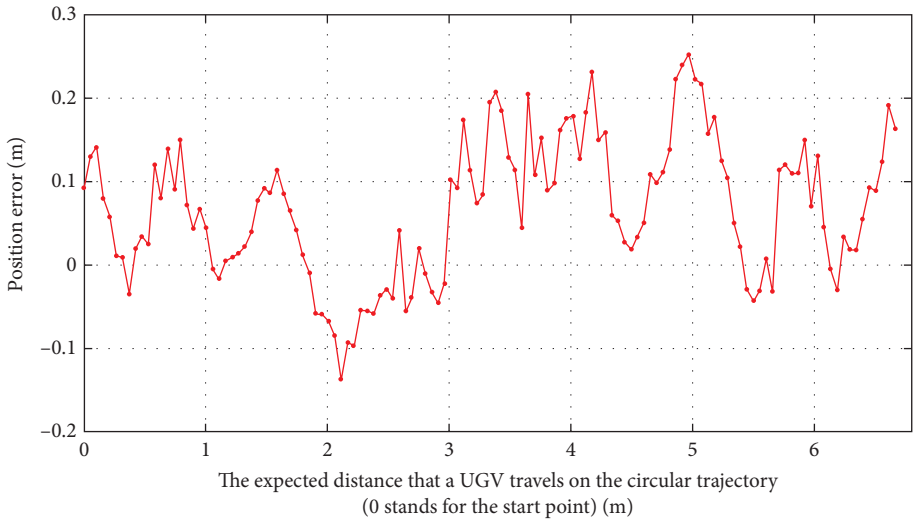

Figure 25: Position errors of a circular trajectory, corresponding to Figure 24.

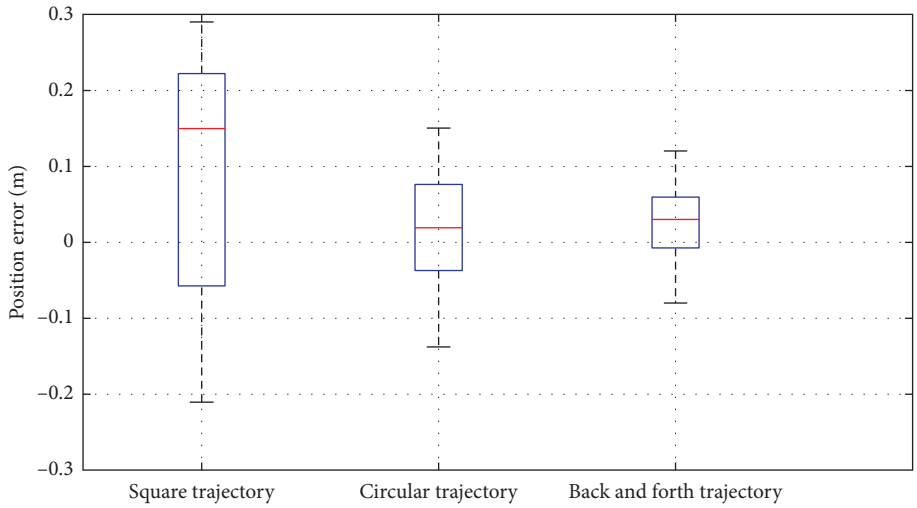

Figure 26: Position error statistics of all field tests.

6. Conclusion

In this paper, the concept of UWB clusters composing of multiple UWB base stations as one device is proposed to provide positioning service for robots. Tests in Section 3 show that this positioning system works well. Within 8 meters, the mean distance errors are less than 20 centimeters in most cases. Within 10 meters, the mean position errors are no more than 60 centimeters in most cases. By approximating the overlap area covered by all the three base stations in one group as a circle, two algorithms are proposed to obtain the best deployment scheme in an area of interest with random shape. Both algorithms can help determine the number of groups necessary. Furthermore, the UWB positioning system is verified by conducting multiple tests with UGVs. The results confirm that UGVs can follow complex trajectories precisely. The UWB cluster's ability to support multiple robots highlights its potential application in swarm robotics. Furthermore, combining multiple base stations as one positioning device, it can be equipped on a larger vehicle such as an unmanned surface vehicle or a heavy drone, providing high-accuracy positioning service for a swarm of smaller robots. Note that, in this paper, we mainly focus on the feasibility of the proposed idea, and the performance improvement of the positioning system is reserved as future work.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] K. Yu, K. Wen, Y. Li, S. Zhang, and K. Zhang, "A novel nlos mitigation algorithm for uwb localization in harsh indoor environments," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 686–699, 2019.

[2] A. R. Jiménez Ruiz and F. Seco Granja, "Comparing ubisense, bespoon, and decawave uwb location systems: indoor performance analysis," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 8, pp. 2106–2117, 2017.

[3] Z. Qiao, J. Zhang, X. Qu, and J. Xiong, "Dynamic self-organizing leader-follower control in a swarm mobile robots system under limited communication," *IEEE Access*, vol. 8, pp. 53 850–53 856, 2020.

[4] A. Tahir, J. M. Böling, M. Haghbayan, and J. Plosila, "Comparison of linear and nonlinear methods for distributed control of a hierarchical formation of uavs," *IEEE Access*, vol. 8, pp. 95 667–695 680, 2020.

[5] M. A. Maghenem, A. Loría, and E. Panteley, "Cascades-based leader-follower formation tracking and stabilization of multiple nonholonomic vehicles," *IEEE Transactions on Automatic Control*, vol. 65, no. 8, pp. 3639–3646, 2020.

[6] M. Pérez-Ruiz, D. C. Slaughter, C. J. Gliever, and S. K. Upadhyaya, "Automatic GPS-based intra-row weed knife control system for transplanted row crops," *Computers and Electronics in Agriculture*, vol. 80, pp. 41–49, JAN 2012.

[7] D. Nad, N. Miskovic, and F. Mandic, "Navigation, guidance and control of an overactuated marine surface vehicle," *Annual Reviews in Control*, vol. 40, pp. 172–181, 2015.

[8] J. Fang and J. Qin, "Advances in atomic gyroscopes: a view from inertial navigation applications," *Sensors*, vol. 12, no. 5, pp. 6331–6346, 2012.

[9] A. Sahoo, S. K. Dwivedy, and P. S. Robi, "Advancements in the field of autonomous underwater vehicle," *Ocean Engineering*, vol. 181, pp. 145–160, 2019.

[10] T. Konrad, J.-J. Gehrt, J. Lin, R. Zweigel, and D. Abel, "Advanced state estimation for navigation of automated vehicles," *Annual Reviews in Control*, vol. 46, pp. 181–195, 2018.

[11] M. Tulldahl, J. Rydell, J. Holmgren, J. Nordlof, and E. Willen, "Lidar-based positioning in forest environments," *In Electro-Optical Remote Sensing XIII*, vol. 11160, 2019.

[12] A. F. Scannapieco, A. Renga, and A. Moccia, "Compact millimeter wave fmcw insar for uas indoor navigation," *MetroAeroSpace*, vol. 14, pp. 551–556, 2015.

[13] L. Zhu, X. Li, B. Li, J. Jin, Y. Zhang, and W. Li, "A method of 3D reconstruction based on single camera," *Modern*

Technologies in Space- and Ground-based Telescopes and Instrumentation, vol. 7739, 2010.

[14] Q. Liu, J. Qiu, and Y. Chen, "Research and development of indoor positioning," *China Communications*, vol. 13, no. 2, pp. 67–79, 2016.

[15] R. F. Brena, J. P. Garcia-Vazquez, C. E. Galvan-Tejada, D. Munoz-Rodriguez, C. Vargas-Rosales, and J. Fangmeyer, "Evolution of indoor positioning technologies: a survey," *Journal of Sensors*, vol. 2017, Article ID 2630413, 21 pages, 2017.

[16] J. Kunhoth, A. Karkar, S. Al-Maadeed, and A. Al-Ali, "Indoor positioning and wayfinding systems: a survey," *Human-Centric Computing and Information Sciences*, vol. 10, no. 1, 2020.

[17] A. K. Brown, "GPS/INS uses low-cost MEMS IMU," *IEEE Aerospace and Electronic Systems Magazine*, vol. 20, no. 9, pp. 3–10, 2005.

[18] M. Aftatah, A. Lahrech, and A. Abounada, "Fusion of GPS/INS/Odometer measurements for land vehicle navigation with GPS outage," *In CloudTech*, vol. 20, pp. 48–55, 2016.

[19] Y. Chen, D. Zheng, P. A. Miller, and J. A. Farrell, "Underwater inertial navigation with long base line transceivers: a near-real-time approach," *ICDC*, vol. 7, pp. 5042–5047, 2013.

[20] P. Lee, C. Lee, S. Kim, B. Jeon, S. Hong, and T. Aoki, "Preliminary study on the inertial-Doppler localization of a deep-sea launcher hanging on a cable," in *Proceedings of the 3rd International Workshop On Scientific Use Of Submarine Cables And Related Technologies*, pp. 139–144, Tokyo, Japan, 2003.

[21] S. Duan, Y. Li, S. Chen et al., "Study of obstacle avoidance based on fuzzy planner for wheeled mobile robot," in *Proceedings of the WCICA 2011*, pp. 672–676, Taipei, Taiwan, 2011.

[22] J. Cheng and Q. Hui, "Motion planning for amigobot with line-segment-based map and voronoi diagram," in *Proceedings of the SYSCON 2016*, pp. 15–22, Tokyo, Japan, 2016.

[23] Y. Wang, Y. Zhao, X. Wang, and Z. Xu, "Vision method of vehicle localization based on the roadside landmark," in *Proceedings of the FOI 2018*, vol. vol. 10832, Taipei, Taiwan, 2018.

[24] L. Ma, Y. Lin, Y. Cui, and Y. Xu, "Vision-based positioning method based on landmark using multiple calibration lines," *Lecture Notes in Electrical Engineering*, vol. 423, pp. 471–484, 2018.

[25] X. Li, Y. Shang, A. Su et al., "Vision navigation for UAV based on scene matching," *Unattended Ground, Sea, and Air Sensor Technologies and Applications*, vol. 8388, 2012.

[26] R. G. Lins, S. N. Givigi, S.-M. Hung, and A. Noureldin, "A novel machine vision approach applied for autonomous robotics navigation," in *Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1912–1917, Hong Kong, China, 2015.

[27] J. Biswas and M. Veloso, "WiFi localization and navigation for autonomous indoor mobile robots," in *Proceedings of the ICRA 2010*, pp. 4379–4384, Anchorage, AL, USA, 2010.

[28] Z. Song, X. Wu, T. Xu, J. Sun, Q. Gao, and Y. He, "A new method of AGV navigation based on Kalman Filter and a magnetic nail localization," in *Proceedings of the ROBIO 2016*, pp. 952–957, Qingdao, China, 2016.

[29] B. Xu and D. Wang, "Magnetic locating AGV navigation based on kalman filter and PID control," in *Proceedings of the CAC 2018*, pp. 2509–2512, Xian, China, 2018.

[30] C. Zhang and Y. Zhang, "Integrated positioning and navigation for aircraft automatic taxiing," in *Proceedings of the DASC 2014*, Dalian, China, 2014.

[31] C. Liu, M. Fan, and R. Song, "A novel intelligent and agile warehouse system for energy meter storage," in *Proceedings of the ICIST 2013*, pp. 129–133, Kaunas, Lithuania, 2013.

[32] S. Gezici and H. V. Poor, "Position estimation via ultra-wide-band signals," *Proceedings of The IEEE*, vol. 97, no. 2, pp. 386–403, 2009.

[33] F. Mazhar, M. G. Khan, and B. Sällberg, "Precise indoor positioning using UWB: a review of methods, algorithms and implementations," *Wireless Personal Communications*, vol. 97, no. 3, pp. 4467–4491, 2017.

[34] A. Alarifi, A. Al-Salman, M. Alsaleh et al., "Ultra wideband indoor positioning technologies: analysis and recent advances," *Sensors*, vol. 16, no. 5, 2016.

[35] G. Li, L. Dong, H. Xu, and Y. Lin, "Research on region coverage approach with swarm robots," *Jiqiren/Robot*, vol. 39, no. 5, pp. 670–679, 2017.

[36] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[37] J. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, Cambridge, UK, 1992.