Robust Self-Deployment for a Swarm of Autonomous Mobile Robots with Limited Visibility Range

Geunho Lee, Nak Young Chong, and Xavier Defago
School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa, Japan
{geun-lee, nakyoung, defago}@jaist.ac.jp

Abstract— **In this study, we focus on a self-deployment problem for a swarm of autonomous mobile robots that can be used to build a sensor networking infrastructure with equilateral triangle lattice configurations. In order to deploy the swarm, this paper proposes a self-stabilizing distributed self-deployment algorithm under a robot model with the following features: no identification numbers, no common coordinates, no predetermined leader, no memory for past actions and implicit communication. Regardless of the restricted model, our proposed algorithm based on local interactions provides a solution for the self-deployment problem. Moreover, the algorithm provides robust capability of swarm connectivity in spite of loss of several robots. We discuss in details the features of the algorithm, including self-organization, self-stabilization, and robustness. A simulation study demonstrates the validity of the algorithm.**

I. INTRODUCTION

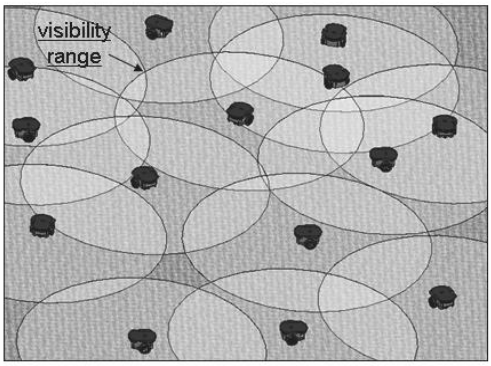
With the emergence of mobile network-enabled systems, much attention has been paid to increase availability and use of wireless sensor networks in the fields of swarm robotics [1]. The mobile sensor networking systems are regarded as large swarms of wireless sensor units mounted on spatially distributed autonomous mobile robots. The sensor network can be used in many real applications such as environmental monitoring, surveillance, and so on. In order to enable autonomous robots to perform the mentioned application examples, such as building a networking infrastructure, it is basically necessary to control their deployment.

Recently, self-deployment approaches for large-scale swarms have been reported using simple local interactions. The majority of these approaches are based on the intuition observed from an organism of animals and insects or physical phenomena in nature, as called behavior-based [2-3] or virtual physics-based [4-11] approaches. Balch and Hybinette [2] suggested the notion of social potentials to achieve robot formations mimicking the process that holds the molecules into place to form crystals. In this study each robot had several attachment site geometries having the potential to attract other robots. Each site provided a different lattice pattern. Martison and Payton [4] proposed a virtual line force method to deploy mobile robots into a regular spaced lattice. The basic concept behind the line force was to place each robot in an imaginary array of parallel lines. An artificial physics framework [5] was described to achieve a desired deployment of a swarm of mobile robots based on a sophisticated model analogous to gravitational force. Based on an acute-angle test, Shucker and Bennett [6] introduced a virtual spring force algorithm to deploy a number of robots using artificial forces such as repulsion and attraction. Howard et al. [7] proposed an electromagnetic force approach to deploy robots throughout a defined environment. Heo and Varshney [8] described a technique using the concept of force inspired from equilibrium of molecules. Zou and Chakrabarty [9] presented a potential field algorithm based on the virtual forces between robots. Cohen and Peleg [10] introduced a deployment algorithm for robot swarms to locally spread them uniformly within a given region. The common thing of self-deployment approaches introduced above is local interactions or local behaviors based on attractive and repulsive force. The behavior-based and virtual physics-based works generally derive force information to obtain a desired behavior for robots. However, their approaches require an effort to adjust parameters to perform a successful self-deployment.

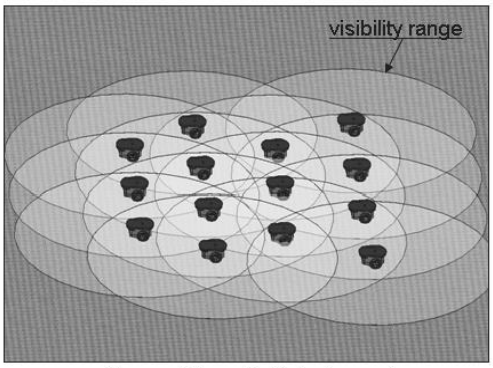
In this paper, we consider an autonomous mobile robot that freely moves on the 2-dimensional plane. The robots have a limited range of sensing (e.g., visibility). Robots can communicate with each other in close proximity, but are not allowed to communicate explicitly. Instead of the direct method, they are able to locally interact only by observing locations of other robots. Under this feature, we address the self-deployment problem of how to enable robots to deploy a swarm uniformly as shown in Fig. 1. As a desired goal, the swarm is composed of many equilateral triangular geometric lattice configurations over a 2-dimensional space. This problem is very important since deploying the swarm provides a way for mobile robots with limited visibility range to self-organize their unified networking infrastructure. Furthermore, self-deployment only based on local interactions in close proximity contains a robust capability of swarm connectivity regardless of loss of several robots. From a practical standpoint, the coordination for deploying many robots provides a first step toward flocking, i.e., allowing a swarm to maintain their connectivity while moving in a mobile ad hoc network environment [12].

The main contribution of this paper is to provide a simple distributed self-deployment algorithm such that many mobile robots can eventually configure their network infrastructure with equilateral triangle lattice configurations. Only the input arguments of the same algorithm for each robot consist of both its current position and a point set containing the positions of currently observed robots within limited visibility range with respect to its local coordinate system.

978-1-4244-1635-6/07/$25.00



(a) arbitrary distribution



(b) completion of self-deployment

Fig. 1. Illustration of self-deployment problem

Furthermore, robots at local level can self-repair the fault of a swarm due to loss of several robots. Therefore, the proposed algorithm features a self-organizing, self-stabilizing, and robust solution.

The rest of this paper is organized as follows. Section II presents the system model and definition. Section III describes a basic algorithm for building a sensor networking system, and demonstrates simulation results. Section IV discusses a problem of the basic algorithm, presents an upgraded algorithm to overcome the problem, and shows simulation results. Finally, Section V concludes the paper.

II. Problem Statement

We consider a system composed of a homogeneous set of autonomous mobile robots $\{r_1, \cdots, r_n\}$. Each robot is modeled as a point, that freely moves on a 2-dimensional space. The robots have no prior knowledge of their identification numbers, and do not share any common coordinate system nor the identity of a leader robot. Furthermore, instead of the direct communication, they are able to locally interact only by observing locations of other robots. Each robot executes the same algorithm and then acts independently and asynchronously from other robots. Time is represented as an infinite sequence of discrete time instants. At a time instant, a robot can either be idle or execute one of the following three actions:

- **Observation**: Each robot r_i takes a snapshot of the positions $\{p_1, p_2, \cdots\}$ of other robots located in its limited visibility range *VR*, and then makes an observation set O_i of positions of the observed robots with respect to its local coordinate system.
- **Computation**: r_i performs its computation in accordance with the given oblivious algorithm, resulting in a target p_t .
- **Motion**: r_i moves toward p_t and goes back to the Observation state.

Here, we assume that all robots repeat an endless activation cycle of observation, computation, and motion. At each activation, a robot computes its target position, which only requires the position of other robots, and moves toward the computed target under a given algorithm. Robots do not retain any memory of past actions. This assumption has the nice property that any algorithms developed for that model are inherently self-stabilizing [1] [13].

Based on the proposed model, the algorithm of a robot consists of a function φ that is executed at each activation. The arguments of φ consist of the current position of the robot, and a set of positions for other robots observed at the corresponding time instant. The returned value is the new target position of the robot. Notice that arguments to φ include only current observing and thus the robots are oblivious, since they are unable to remember any past actions or decisions.

The problem addressed in this paper is self-deployment by a swarm of autonomous mobile robots as illustrated in Fig. 1. We formulate the self-deployment problem as follows.

Given that a swarm of robots $r_1, \cdots, r_n$ with arbitrarily distinct positions according to our model is located in the two-dimensional plane, robots can eventually be deployed at uniform intervals.

In this paper, we intend to analyze the self-deployment problem by dropping the aforementioned assumptions such as on-board memory and communication devices and then find the solution for the problem. Furthermore, our study pursues that robots have no global or common knowledge in advance. From a practical point of view, we expect that this allows simpler, more robust and economically advantageous robots to be used.

III. Basic Algorithm

This section presents a self-deployment algorithm that can generate an equilateral triangle lattice. It is necessary to impose a condition as follows; all robots are initially located in distinct and arbitrarily distributed positions.

A. Algorithm description

The intuition behind the algorithm is simple, and we briefly describe it here (see Algorithm-1). An input is given in observed positions for other robots within *VR*

[1] Self-stabilization is the property of a system which, started in an arbitrary state, always converges toward a desired behavior [14] [15].

ALGORITHM - 1 SELF-DEPLOYMENT (code executed by a robot r_i)

constant d_u := uniform range distance

Function $\varphi(O_i, p_i)$

1 $p_{s1} := \min_{p \in O_i - \{p_i\}} [dist(p_i, p)]$

2 $p_{s2} := \min_{p \in O_i - \{p_i, p_{s1}\}} [dist(p_{s1}, p) + dist(p, p_i)]$

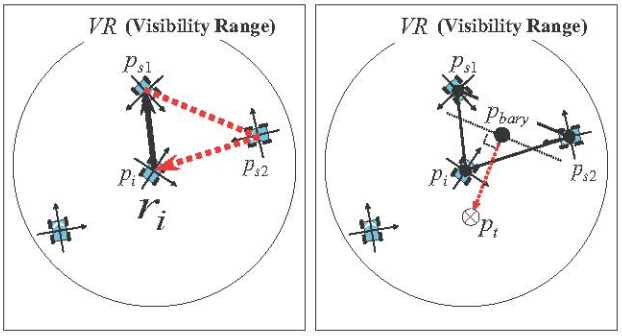
3 $(b_x, b_y) :=$ barycenter $(\{p_i, p_{s1}, p_{s2}\})$

4 $\theta :=$ angle between $\overline{p_{s1}p_{s2}}$ and r_i's local horizontal axis

5 $target_x := b_x + d_u \times \cos(\theta + \pi/2)\sqrt{3}$

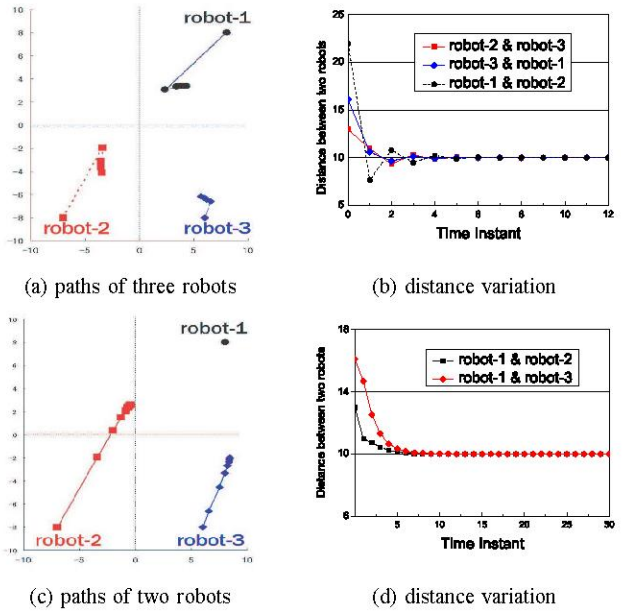
6 $target_y := b_y + d_u \times \sin(\theta + \pi/2)\sqrt{3}$

7 move toward target point$(target_x, target_y)$



(a) neighbor selection (b) target computation

Fig. 2. Illustration of ALGORITHM-1

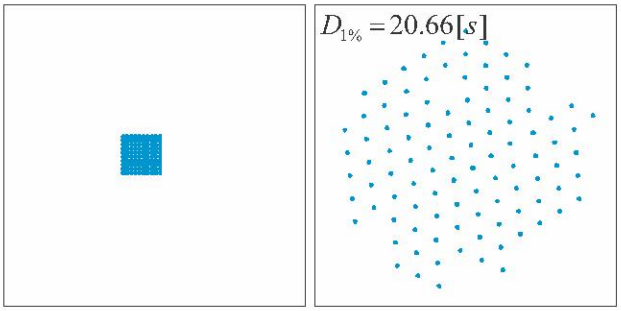


(a) paths of three robots (b) distance variation

(c) paths of two robots (d) distance variation

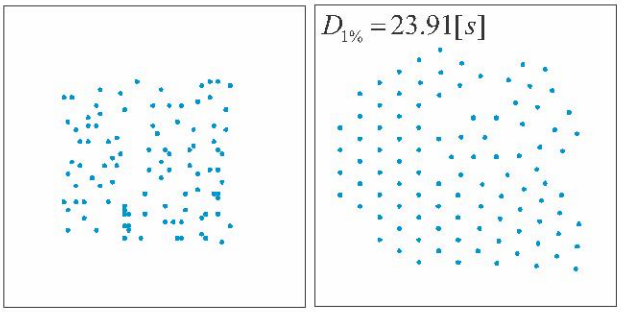
Fig. 3. Simulation results as executed by three robots ((a)-(b) case-1: agreement on the mutual neighbor, (c)-(d) case-2: disagreement on the mutual neighbor)

and the position p_i occupied by r_i. Moreover, denote the constant uniform interval as d_u. To begin with, selects the first neighbor located the shortest distance from itself, as illustrated in Fig. 2-(a). The second neighbor is selected, such that the total lateral distance is minimized. In close proximity, r_i selects two neighbors locally interacted with. Here, a neighbor set $\{p_{s1}, p_{s2}\}$ is defined as a set of positions of two neighbors selected by r_i. In Fig. 2-(b), p_i forms a geometric triangle with $\{p_{s1}, p_{s2}\}$ in the plane. Specially, a geometric configuration is a set of three distinct points which includes $\{p_{s1}, p_{s2}\}$ and p_i. Using $\{p_{s1}, p_{s2}\}$, r_i finds the center p_{bary} of the geometric configuration formed by three robots as presented in Fig. 2-(b). Next, r_i measures the angle θ between the line connecting two neighbors and the horizontal axis with respect to its local coordinate system. r_i calculates its target point p_t based on p_{bary}, and moves toward p_t. In this algorithm, r_i locally interacts with $\{p_{s1}, p_{s2}\}$ in order to generate an isosceles triangle with d_u. By doing this repeatedly, the swarm is finally deployed to have equilateral triangular lattices. Notice that under this algorithm, each robot can select dynamically neighbors with whom it interacts in close proximity. Consequently, the algorithm brings the swarm to the desired geometric configuration in which each robot forms an equilateral triangle with the selected neighbors.



(a) compressed distribution (b) self-deployment

Fig. 4. Simulation results for self-deployment from an initially compressed distribution with 100 robots



(a) random distribution (b) self-deployment

Fig. 5. Simulation results for self-deployment from a random distribution with 100 robots

B. Simulation Results

Before displaying simulation results, we define the completion condition of self-deployment for a swarm of robots in simulations. The deployment completion depends on whether each robot converges into $d_u \pm 1\%$ for two neighbors. Let $D_{1\%}$ denote the deployment time satisfying the completion condition for all robots in our simulation. In Fig. 3, as a concrete example, we investigated how three robots converge into an equilateral triangle. Fig. 3-(a) shows paths of three robots starting from arbitrary positions. Fig. 3-(b) presents

ALGORITHM - 2 SELF-DEPLOYMENT WITH SELF-REPARATION
(code executed by a robot r_i)

constant d_u := uniform range distance
Function $\varphi(O_i, p_i)$
1 $p_{s1} := \min_{p \in O_i - \{p_i\}} [dist(p_i, p)]$
2 $p_{s2} := \min_{p \in O_i - \{p_i, p_{s1}\}} [dist(p_{s1}, p) + dist(p, p_i)]$
3 **IF** $\{(d_u \cong dist(p_i, p_{s1})) \wedge (d_u \cong dist(p_i, p_{s2}))\}$ **THEN**
4 $p_{s3} := \min_{p \in O_i - \{p_i, p_{s1}, p_{s2}\}} [dist(p_i, p)]$
5 $p_{sk} := \min_{p \in \{p_{s1}, p_{s2}\}} [dist(p_{s3}, p)]$
6 **END IF**
7 (b_x, b_y) := barycenter $(\{p_i, p_{s1}, p_{s2}\})$
8 θ := angle between $\overline{p_{s3}p_{sk}}$ and r_i's local horiz. axis
9 $target_x := b_x + d_u \times \cos(\theta + \pi/2)\sqrt{3}$
10 $target_y := b_y + d_u \times \sin(\theta + \pi/2)\sqrt{3}$
11 move toward target point$(target_x, target_y)$

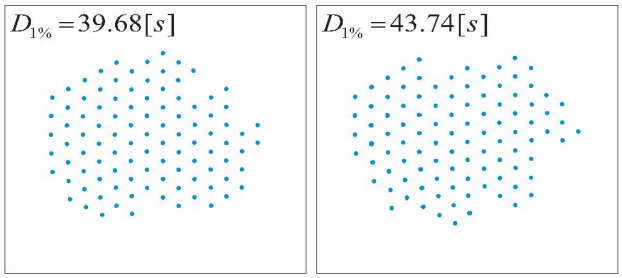
the variations of distances between each robot and neighbors for robot movement of Fig. 3-(a). From the graph, each robot could eventually converge into d_u with its neighbors. Next, to evaluate the case of disagreement on the mutual neighbor, we suppose that robot-1 stops at the occupied position in the simulation of Fig. 3-(c). In this simulation, robot-2 and robot-3 regarded robot-1 as one neighbor, and then generated the desired triangle while approaching robot-1. Regardless of whether each robot agrees on the mutual neighbor or not, its neighbors had the same behavior for the same geometric configuration as itself, and then converged toward the defined d_u. The reason for this is that each robot searches the closet adjacent robots within its *VR* and selects them as $\{p_{s1}, p_{s2}\}$. Consequently, the simple algorithm eventually brings the robots toward the desired configuration, located on each vertex of an equilateral triangle. From initially different distributions as shown in Fig. 4-(a) and Fig. 5-(a), we evaluated the self-deployment of 100 robots. From the simulation results of Fig. 4-(b) and Fig. 5-(b), a swarm of robots could be uniformly self-deployed with d_u while locally interacting. Therefore, collecting the desired geometric configurations can globally reach self-deployment of robot swarms without a centralized control scheme. Specially, each robot needs only two neighbors under ALGORITHM-1. This means fewer neighbors, or less influence from the existence of other robots within the *VR* of each robot. From a practical point of view, this effect is directly coupled with decreasing computational complexity.

IV. EXTENDED ALGORITHM

A. Problems with the basic algorithm

The algorithm given in Section III does not guarantee complete uniform distribution of robots across the swarm. In Fig. 4-(b) and Fig. 5-(b), the simulation results showed that unintended holes remained in the deployment process. In this section, we briefly discuss how to cope with this problem and provide a slightly revised method that alleviates this problem.

Specifically, ALGORITHM-1 can bring all robots to converge into each vertex of equilateral triangles in spite of selecting their neighbors dynamically at each time instant.



(a) self-reparation from Fig. 4-(b) (b) self-reparation from Fig. 5-(b)

Fig. 6. Simulation results for self-reparation

TABLE I
THE NUMBER OF NEIGHBORS WITH d_u IN THE SWARM [ROBOTS]

No. neighb. at d_u	deployment results			
	Fig. 4-(b)	Fig. 6-(a)	Fig. 5-(b)	Fig. 6-(b)
2	4	4	3	3
3	7	4	21	12
4	18	14	22	13
5	16	12	15	12
6	55	60	39	60

However, the number of robots locally interacting is only two neighbors. This is caused by the triangular geometry, as each robot has determined the direction of deployment evolving when selecting its neighbors (and by its obliviousness, since it cannot remember the previous neighbors and thus cannot compute neighbors' motion).

B. Self-reparation algorithm

The problem above can be solved by revising the neighbor selection, and we call this solution a self-reparation. The simple ALGORITHM-2 proceeds as follows. The algorithm also starts with the selection of $\{p_{s1}, p_{s2}\}$. Next, each robot checks whether each distance to each of two neighbors is equal to d_u. In detail, this condition can be regarded as the subroutine inserted between Line 2 of ALGORITHM-1 and Line 3. After checking the condition, if it is correct, each robot r_i then selects its next closest neighbor p_{s3} within its *VR*, except for other members $\{p_{s1}, p_{s2}\}$ of the geometric configuration. Otherwise, r_i does not find another neighbor p_{s3}. Then, r_i finds the center point of the triangle formed by a position p_{sk} between p_{s1} and p_{s2}, p_{s3}, and p_i in accordance with ALGORITHM-2. Finally, r_i moves toward p_t. By doing this repeatedly, the swarm distributed uniformly can be eventually deployed while filling up spontaneous holes.

C. Simulation Results

Fig. 6 presents the simulation results for the proposed self-reparation method. As compared with the results of Fig. 4-(b) and Fig. 5-(b), the snapshots of Fig. 6 present results of deployments without unintended holes. Furthermore, Table I shows comparison data for the number of neighboring robots with d_u centering around the position of each robot after the

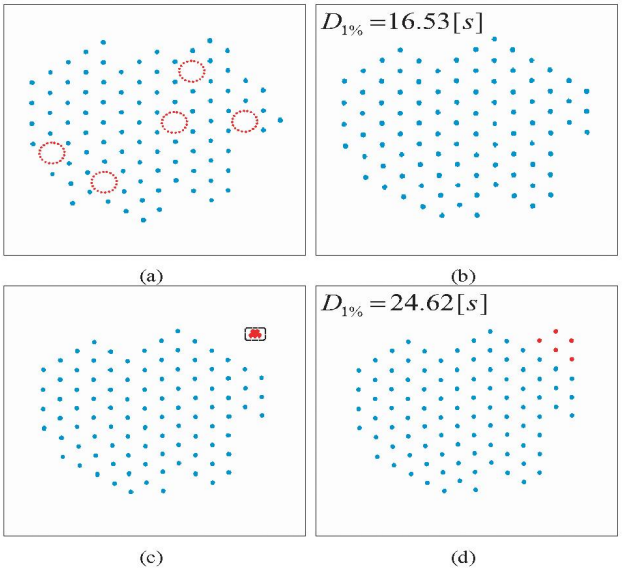


Fig. 7. Simulation results for robustness against loss of 5 robots ((a) loss of 5 robots, (b) redeployment with 95 robots, (c) replaced 5 robots, (d) redeployment with 100 robots)

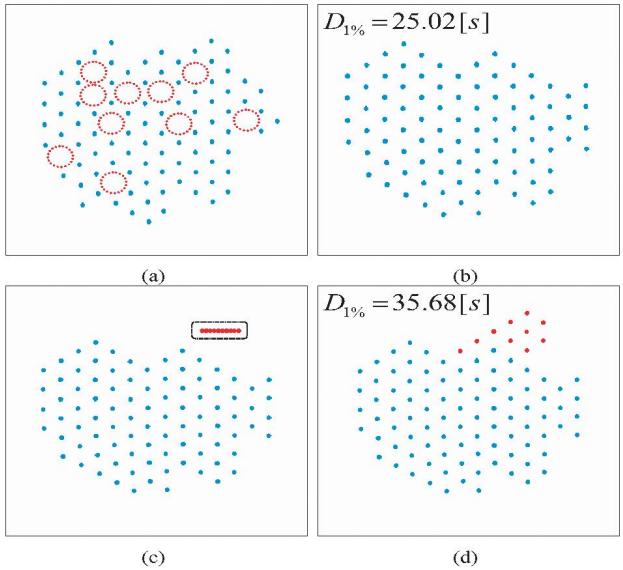


Fig. 8. Simulation results for robustness against loss of 10 robots ((a) loss of 10 robots, (b) redeployment with 90 robots, (c) replaced 10 robots, (d) redeployment with 100 robots)

completion of the self-deployment with a condition $d_u \pm 1\%$. We compared the deployment method with and without self-reparation according to initial distributions. As a result, the self-reparation method increased the number of robots at distance d_u. This means that ALGORITHM-2 improved the connectivity between neighboring robots with d_u. Although we obtain improved results to some extent, the algorithm is not sufficient to guarantee completely uniform distribution, as there exist some irregularities. In fact, a possible option would be to increase the *VR* of each robot. This question has been left unsolved here for future work.

In the next simulation, robustness is verified against disappearances of robot members. Here, we supposed that the disappeared robots have gone somewhere due to an accidental failure. The total number of robots in a swarm is 100. These simulations begin with the state of Fig. 5-(a). After the self-reparation of Fig. 6-(b), 5 robots in Fig. 7 or 10 robots in Fig. 8 disappeared without warning. Due to loss of several robot members, several holes appeared in the deployment. Each robot checks the existence of the disappeared robots within *VR*. If there is a hole around it, it executes ALGORITHM-2 repeatedly. By the algorithm, the robots approached other robots, and then the holes disappeared. Fig. 7-(b) and Fig. 8-(b) present the results of re-deployment. In addition, we performed simulations replacing the number of lost robots in Fig. 7-(c) and Fig. 8-(c). Fig. 7-(d) and Fig. 8-(d) show the results of re-deployment after replacing several robots. From the simulation results, ALGORITHM-2 with self-reparation capability is effective in improving the robustness of mobile sensor networks. Furthermore, the simple self-reparation at the local level can repair both local and global faults of a sensor networking application.

TABLE II

SIMULATION RESULT OF DEPLOYMENT TIME EXECUTING ALGORITHM-1 AND ALGORITHM-2 WITH/WITHOUT SELF-REPAIRING [SEC]

average deployment time	ALGORITHM-1 without self-reparation	ALGORITHM-2 with self-reparation
within 1 %	27.36	41.54
within 0.1 %	42.86	56.28
within 0.01 %	61.48	78.22

Finally, we examined each execution time by both ALGORITHM-1 and ALGORITHM-2. Data of Table II indicates the average deployment time for 30 kinds of initially random distributions of 100 robots. Here, we made conditions for deployment completion as three following rates: 1%, 0.1%, and 0.01%. Fig. 9 shows proportion of deployment time executing ALGORITHM-1 (red- empty box) and ALGORITHM-2 (blue- slash box). In the box-plot, boxes represent the inter-quartile range of the data, while the horizontal bars inside the boxes mark the median values. From the results, we see that ALGORITHM-2 gains the advantage of a little variation for the completion time even though the self-deployment with the self-reparation takes much time.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented a self-deployment algorithm, enabling a large-scale swarm of robots to build a sensor networking infrastructure. A swarm of robots was assumed to participate in the deployment process under simple models, with conditions such as no unique individual identifications,

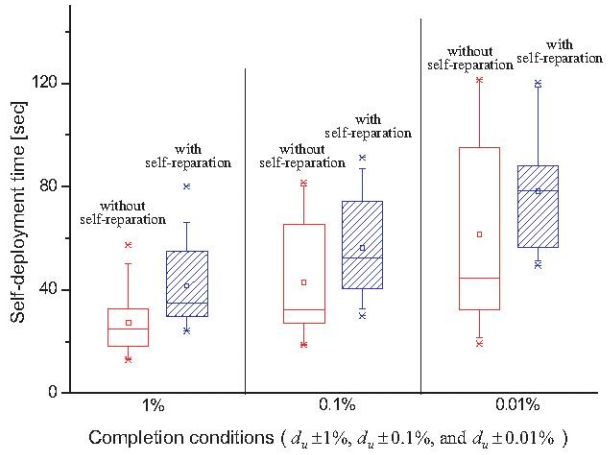


Fig. 9. Proportion of deployment time without (red) or with (blue) self-reparation

no determined leader, no common coordinates, no memory for past actions and implicit communication with each other. Just by observing the other robots in their visibility range, the robots selected dynamically neighbors with whom they interacted. By this local interaction based on the geometric approach to forming an equilateral triangle, the swarm could be uniformly self-deployed. In our proposed algorithm without adjusting parameters, large-scalae swarms of robots could yield a desirable self-deployment. In particular, the proposed deployment control featured robustness against a loss of participating members. The proposed algorithm featuring self-stabilizing and self-organizing design was evaluated by simulations. Our analysis and simulation results show that the self-deployment for a mobile sensor networking swarm is a simple, robust, and effective approach.

Self-deployment is a first step toward real-world implementations of a mobile sensor networking infrastructure. To enable the successful deployment, the algorithm presented in this paper relies on the fact that robots can exactly sense the positions of neighbor robots in close proximity, as is the case with sonar reading [16] or infrared sensor (IR) reading [5]. From the realistic point of view, it requires each robot to be able to uniquely identify other robots in its vicinity and distinguish other robots from miscellaneous objects in an environment. On the other hand, when utilizing interaction through communication, one must consider how and toward what end it is used. Moreover, it is necessary for robots to use a prior knowledge such as an individual identification number or global coordinates as occasion demands [17-18]. Like the implementation problems of observation, communication in physical robotics is not also free or reliable and can be constrained by limited bandwidth and range, and unpredictable interference. Therefore, as an additional motivation, we will intend to use this problem as a starting point for understanding the relation between the capabilities and the difficulties of several communication models described in [19]. It will be interesting to study the role and strength according to the communication models. This question is left for later works.

ACKNOWLEDGMENTS

This research was conducted program for the "Fostering Talent in Emergent Research Fields" in Special Coordination Funds for Promoting Science and Technology by Japan Ministry of Education, Culture, Sports, Science and Technology.

REFERENCES

[1] E. Sahin. "Swarm robotics: from sources of inspiration to domains of application," SAB 2004, E. Sahin and W. M. Spears (Eds.), Lecture Notes in Computer Science, 3342, Springer, pp.10-20, 2005
[2] T. Balch and M. Hybinette. "Social potentials for scalable multi-robot formations," Proc. IEEE International Conference on Robotics and Automation, pp.73-80, 2000
[3] B. Werger and M. J. Mataric. "From Insect to Internet: Situated Control for Networked Robot Teams," Annals of Mathematics and Artificial Intelligence, Vol.31, No.1-4, pp.173-198, 2001
[4] E. Martison and D. Payton. "Lattice Formation in Mobile Autonomous Sensor Arrays," SAB 2004, E. Sahin and W. M. Spears (Eds.), Lecture Notes in Computer Science, 3342, Springer, pp.98-111, 2005
[5] W. Spears, D. Spears, J. Hamann, and R. Heil. "Distributed, physics-based control of swarms of vehicles," Autonomous Robots, Vol.17, No.2-3, pp.137-162, 2004
[6] B. Shucker and J. K. Bennett. "Scalable control of distributed robotic macrosensors," Proc. 7th International Symposium on Distributed Autonomous Robotic Systems, 2004
[7] A. Howard, M. J. Mataric, and G. S. Sukhatme. "Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem," Proc. 6th International Symposium on Distributed Autonomous Robotic Systems, pp.299-308, 2002
[8] N. Heo, P. K. Varshney. "A distributed self spreading algorithm for mobile wireless sensor networks," Proc. IEEE Wireless Communication and Networking Conference, 2003
[9] Y. Zou and K. Chakrabarty. "Sensor deployment and target localization based on virtual forces," Proc. IEEE Infocom Conference, 2003
[10] R. Cohen and D. Peleg. "Local algorithms for autonomous robots systems," Sirocco 2006, P. Flocchini and L. Gasieniec (Eds.), Lecture Notes in Computer Science, 4056, Springer, pp.29-43, 2006
[11] G. Wang, G. Cao, and T. L. Porta. "Movement-assisted sensor deployment," Proc. IEEE Infocom Conference, pp.2469-2479, 2004
[12] I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis. "On the average and worst-case effciency of some new distributed communication and control algorithms for ad-hoc mobile networks," Proc. 1st ACM International Workshop on Principles of Mobile Computing, pp.1-19, 2001
[13] I. Suzuki and M. Yamashita. "Distributed anonymous mobile robot: Formation of geometric patterns," SIAM Journal of Computing, Vol. 28, No. 4, pp.1347-1363, 1999
[14] S. Dolev. Self-Stabilization, MIT Press, 2000
[15] M. Schneider. "Self-stabilization," ACM Computing Survey, Vol.25, No.1, pp.45-67, 1993
[16] G. Lee and N. Y. Chong. "Decentralized formation control for a team of anonymous mobile robots," Proc. of 6th Asian Control Conference, pp.971-976, 2006
[17] M. Lam and Y. Liu. "ISOGIRD: an efficient algorithm for coverage enhancement in mobile sensor networks," Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.1458-1463, 2006
[18] J. Nembrini, A. Winfield, and C. Melhuish. "Minimalist coherent swarming of wireless networked autonomous mobile robots," in Proc. International Conference on Simulation of Adaptive Behavior (SAB), pp.373-382, 2002
[19] R. Prakash and R. Baldoni. "Architecture for group communication in mobile systems," Proc. 17th IEEE Symposium on Relaiable Distributed Systems, pp.235-242, 1998