Mechatronics 19 (2009) 85–105

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

Decentralized formation control for small-scale robot teams with anonymity

Geunho Lee *, Nak Young Chong

School of Information Science, Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan

ARTICLE INFO

Article history:
Received 9 October 2006
Accepted 6 June 2008

Keywords:
Decentralized coordination
Self-organizing robot teams
Leader-referenced formation control
Neighbor-referenced formation control
Agreement on common coordinates and ID allocations

ABSTRACT

This paper presents decentralized formation controls for a team of anonymous mobile robots performing a task through cooperation. Robot teams are required to generate and maintain various geometric patterns adapting to environmental changes in many cooperative robotics applications. In particular, all robots must continue to strive toward achieving the team's mission even if some members fail to perform their role. Toward this end, formation control approaches are proposed under the conditions that robot teams are initially not allowed to have individual identification numbers (IDs), a predetermined leader, and agreement on coordinate systems. Therefore, all members are required first to reach agreement on their coordinate system and obtain unique IDs for role allocations in a self-organizing way. Then, employing IDs within a common coordinate system, two formation control approaches can be realized: leader-referenced and neighbor-referenced formations. Both approaches are verified using an in-house simulator and physical mobile robots. We detail and evaluate each formation control approach, whose common features include self-organization, robustness, and flexibility.

1. Introduction

Recently, the coordination of multiple robots has been gaining increasing attention, since robots which can perform cooperative tasks as a team offer many advantages over a single high performance robot in efficiency, low per robot cost, fault-tolerance, generality, and so on. Therefore, robot teams are expected to be deployed in a wide variety of applications including surveillance-and-security [1], object transportation [2], object manipulation [3,4], search-and-rescue [5,6], intelligent transportation systems (ITS) [7,8], and exploration [9,10]. To enable a team of multiple robots to successfully perform the assigned tasks, it is often required to generate and/or maintain geometric patterns adapting to environmental changes. Thus, this paper presents the formation control architecture and algorithm needed to coordinate multiple robot movements within a team. Specifically, formation control includes such functions as pattern generation, flocking,[1] and pattern switching. In practice, real-world applications require all robots to continue to strive toward achieving the team's mission even if some members fail to function properly. In addition, every robot needs to move from one position to another position as quickly as possible according to the task [11]. Our goal is to develop a software framework for supporting general purpose applications of cooperative robots running the same algorithm.

Formation control of robot teams can be divided into centralized or decentralized approaches. The centralized approach relies on a specific robot to supervise the movement of the robots through a communication channel. Egerstedt and Hu [12] employed a virtual reference on the desired trajectory controlled from a remote host with which individual robots maintain their predefined positions. Belta and Kumar [13] generated smooth interpolating motion for individual robots, so that the total kinetic energy is minimized while certain constraints are satisfied. In general, a heavy computation burden is imposed on the supervising robot, which also requires tight communication with other robots. In contrast, the decentralized formation control is the coordination achieved through individual robot's decisions.

Most research in decentralized control mainly focuses on (1) how to achieve a specific formation pattern [14–17], (2) how to keep the formation pattern while flocking [1,18,20–25], or (3) how to switch between formation patterns in order to adapt to an environment [26,27]. For the first problem, Suzuki and Yamashita [14] studied the problem of generating regular polygonal shapes based on a non-oblivious algorithm with an unlimited amount of memory. To achieve the shapes, robots were required to utilize their past experience or memory. This algorithm was modified to an oblivious (or memoryless) algorithm and applied to circle formation by Defago and Konagaya [15]. Ikemoto et al. [16] proposed a biologically-inspired algorithm which enabled a robot team to form various geometric patterns. This study required

* Corresponding author.
E-mail address: geun-lee@jaist.ac.jp (G. Lee).

[1] The terminology is based on [18] implying that a team of robots follows a leader robot while maintaining formation. This problem is called "flocking" throughout this paper.

0957-4158/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mechatronics.2008.06.005

robots to be initially lined up before generating a pattern. Fujibayashi et al. [17] proposed a probabilistic formation rule that controlled the number of connections between robots. However, it is generally difficult to choose the probability parameters according to the pattern and the number of robots. For the problem of flocking, two methods were implemented, the leader–follower method and the leaderless method. In the leader–follower method, a robot is selected as the moving reference point. Gervasi and Prencipe [18] proposed a computational solution based on CORDA [19] with weak assumptions such as asynchrony, anonymity, no memory and a simple behavior cycle. In their study, all followers generate a geometric pattern symmetrically with respect to the pre-selected leader. Balch and Arkin [20] studied a new paradigm of reactive behaviors for four formation patterns, where the robots were assigned roles such as leader or follower with unique IDs. Carpin and Parker [21] similarly introduced a cooperative leader following approach that could handle a heterogeneous team with different types of sensors using broadcast communication. As an extension of this approach, Parker et al. [22] introduced a tightly-coupled navigation assistance approach by a leader with rich sensing capability as the central figure of a robot team. Such strategies [21,22] make the leader more costly and the team becomes less robust to the failure of the leader. Additional leader–follower approaches are introduced in [1,23,24]. An alternative approach uses no leader. Balch and Hybinette [25] proposed a physics-based flocking approach without a leader, inspired by crystal generation processes. Each robot had several local attachment sites that are attracted to other robots. Finally, for the problem of pattern switching, a graph theoretic approach was proposed by Desai [26] for switching to another geometric pattern. The approach used a control graph, which is a set of assigned targets, to define behaviors of multiple robots. Kurabayashi et al. [27] presented an adaptive transition technique to enable a team of robots to change formation by varying the phase gaps among artificial non-linear oscillators. General functionality of team organization, team maintenance, and team adaptation was addressed in [28], where Fredslund and Mataric used robots equipped with color helmets indicating their ID. When robots generate a formation, robot IDs and corresponding target points were predetermined in a particular class of formation. The leader may change according to the type of formation, and the followers must find a new neighbor in order to switch to other patterns. Lemay et al. [29] proposed a similar approach that assigned the position of the robots based on their IDs.

In contrast to most previous works, our approach begins with the following assumptions: (1) the team members do not have an external mark or ID; (2) the leader is not a priori selected; and (3) the team members are located at arbitrary distinct positions with no coordinate system agreement. Based on these assumptions, this paper presents a self-organizing team formation. Specifically, our proposed approach to formation control is divided

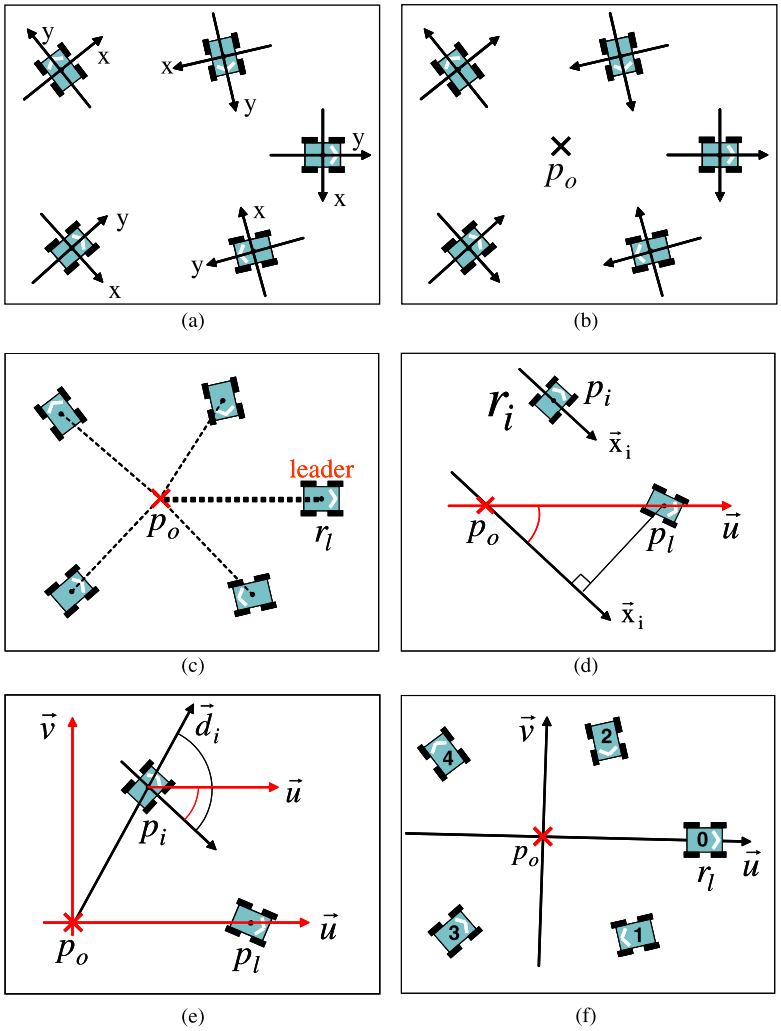

Fig. 1. Agreement on common coordinates and ID allocations: (a) initial distribution, (b) common origin, (c) leader selection, (d) common direction, (e) common coordinates and (f) ID allocations.

into two strategies, the leader-referenced approach and the neighbor-referenced approach. Two potential contributions are: (1) the team is enabled to generate a variety of formations adapting to the given conditions and (2) the same or similar formations can be recovered in spite of a lack of some participating members. These features improved flexibility and robustness.

The remainder of this paper is organized as follows. Section 2 presents a self-organizing team formation definition and strategy for a small-scale team of multiple robots. In Sections 3 and 4, the leader-referenced and neighbor-referenced approaches are proposed and then verified by simulations. Section 5 compares the two proposed approaches and introduces the hybrid control approach. Section 6 gives the experimental results with four physical robots based on the leader-referenced approach. Finally, the conclusion of this paper is explained in Section 7.

2. Self-organizing robot teams

2.1. Coordinate agreement and ID allocation

Robots are modeled as planar points and are assumed to be located at arbitrary, distinct positions without *a priori* coordinate system agreement, as illustrated in Fig. 1a. In addition, robots are anonymous and are able to detect the positions of other robots. Let r_i and p_i denote any robot and its position. Then r_i can measure the position p_j of the other robot r_j with respect to the coordinate system of r_i (denoted by $(L_i[x_j], L_i[y_j])$). A *configuration* means a set of positions which a team of n robots $r_1, \ldots, r_n$ occupies in the 2-dimensional plane. Namely, the configuration $C_i = \{L_i[p_k] \mid 1 \leqslant k \leqslant n\}$ is the representation of all of the robots' distributions with respect to the local coordinate system of r_i. Furthermore, we call the set of target positions a *formation pattern*, and denote $F = \{f_k | 0 \leqslant k \leqslant n-1\}$, where f_i indicates a target point to be occupied by r_i. The distance between p_i and p_j is denoted as $\mathrm{dist}(p_i, p_j)$. Given two arbitrary vectors $\vec{n}$ and $\vec{m}$, let $\mathrm{ang}(\vec{n}, \vec{m})$ be an angle between $\vec{n}$ and $\vec{m}$. The center point for C_i is obtained by dividing the sum of all points by the number n, as shown Fig. 1b. The center point is called the *common origin* p_o of C_i and denoted by

$$p_o = (L_i[x_c], L_i[y_c]) = \left(\frac{\sum L_i[x_j]}{n}, \frac{\sum L_i[y_j]}{n}\right). \quad (1)$$

In C_i, each robot defines as the *leader robot*, r_l positioned farthest away from p_o (see Fig. 1c). The position p_l of r_l indicates the leader coordinates with respect to each robot. Next, a *common direction* is defined by connecting from p_o to p_l as illustrated in Fig. 1d. We denote the *common direction* as $\vec{u}$ and define the angle between the local coordinate $\vec{x}$-axis of r_i and $\vec{u}$ by

$$\mathrm{ang}(\vec{x}_i, \vec{u}) = \cos^{-1}\left(\frac{L_i[x_l] - L_i[x_c]}{\mathrm{dist}(p_l, p_o)}\right). \quad (2)$$

Moreover, $\vec{u}$ defines the horizontal axis of a common coordinate system. It is straightforward to decide the vertical axis $\vec{v}$ by rotating the horizontal axis 90° counterclockwise. Therefore, every robot can specify their position in the *common coordinate system* with $\vec{u}$ and $\vec{v}$ given by

$$\begin{aligned} u_i &= \mathrm{dist}(p_i, p_o) \times \cos(\mathrm{ang}(\vec{x}_i, \vec{d}_i) - \mathrm{ang}(\vec{x}_i, \vec{u})) \\ v_i &= \mathrm{dist}(p_i, p_o) \times \sin(\mathrm{ang}(\vec{x}_i, \vec{d}_i) - \mathrm{ang}(\vec{x}_i, \vec{u})) \end{aligned} \quad (3)$$

where $\vec{d}_i$ is a vector passing through p_i from p_o as presented in Fig. 1e. Using (3), r_i can be assigned new common coordinates (u_i, v_i) with respect to $(\vec{u}, \vec{v})$, and acquire the other robots' coordinates $p_j = (u_j, v_j)$ by

$$\begin{aligned} u_j &= u_i + \mathrm{dist}(p_i, p_j) \times \cos(\mathrm{ang}(\vec{x}_i, \vec{d}_j) - \mathrm{ang}(\vec{x}_i, \vec{u})) \\ v_j &= v_i + \mathrm{dist}(p_i, p_j) \times \sin(\mathrm{ang}(\vec{x}_i, \vec{d}_j) - \mathrm{ang}(\vec{x}_i, \vec{u})) \end{aligned} \quad (4)$$

where $\vec{d}_j$ is a vector passing through p_j from p_i.

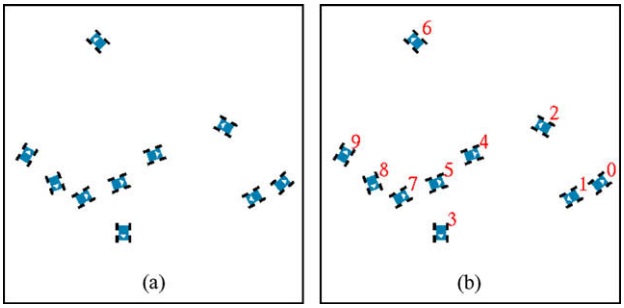

Fig. 2. Simulation results of ID allocations with 10 robots from an arbitrary distribution.

Finally, given $(\vec{u}, \vec{v})$, IDs are assigned to all the robots, starting from r_l numbered 0, by sorting their $\vec{u}$-coordinates in an increasing order (see Fig. 1f). Specifically, they are assigned an odd ID if they have a negative-coordinate of $\vec{v}$-axis, or an even ID if they have a positive coordinate of $\vec{v}$-axis, by turns, until the numbering is completed in either half plane. Remaining members in the other half plane are assigned their IDs consecutively, beginning with the number after the last number assigned. Fig. 2 displays the result of ID allocation with ten robots.

Algorithm 1

Agreement and ID allocations (code executed by r_i)

INPUT: $\{p_1, \ldots, p_n\}$
$p_o :=$ common origin
$p_l := \max_{p \in C_i}[\mathrm{dist}(p, p_o)]$
$\vec{u} :=$ common direction
$\mathrm{ang}(\vec{x}_i, \vec{u}) :=$ angle between $\vec{x}_i$ and $\vec{u}$
$\vec{v} :=$ vertical axis of $\vec{u}$
$(u_i, v_i) := r_i$'s common coordinates
$(u_j, v_j) :=$ other robots' common coordinates
IF $\{p_i = p_l\}$ **THEN**
  $ID_i := 0$ (*leader*)
ELSE $\{p_i \neq p_l\}$
  IF $\{v_i \geqslant 0\}$ **THEN**
    $ID_i := 2\times$ (ranking by increasing order of $\vec{u}$)
  ELSE $\{v_i < 0\}$
    $ID_i := 2\times$ (ranking by increasing order of $\vec{u}$) - 1
  END IF
END IF
OUTPUT: ID_i

If two or more robots are located at the same distance from p_o, the robot team cannot decide r_l uniquely. In this case, the leader selection is repeated after all positions of the leader candidates $r_{l.c}$ are slightly perturbed off the circle having a radius equal to the distance from p_o. Here the following condition of $\mathrm{dist}(p_{l.c}, p_o) < \mathrm{dist}(p'_{l.c}, p_o)$ holds, where $p_{l.c}$ indicates the current positions of $r_{l.c}$ and $p'_{l.c}$ means the new perturbed position of $r_{l.c}$. The other robots remain stationary until a single leader is selected. In Fig. 3, the team has four leader candidates (represented by black triangles), and can select one leader by giving a slight perturbation. The black crosses indicate p_o of each robot. The team reached agreement on $(\vec{u}, \vec{v})$ after the perturbation.

2.2. Problem statement

Based on the coordinate agreement and ID allocations, the *formation control problem* is defined for small-scale mobile robot teams as follows:

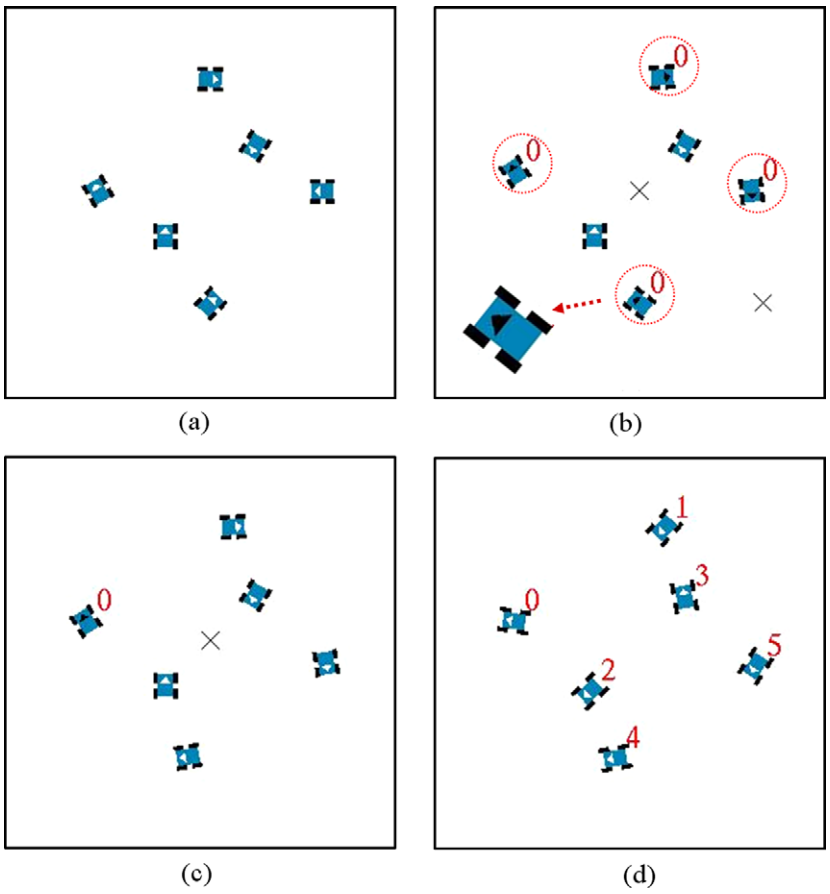

Fig. 3. Simulation results of leader selection from four leader candidates in the six robot teams: (a) initial distribution, (b) four leader candidates, (c) one leader selection, and (d) ID allocations.

Let $r_i, \ldots, r_n$ be anonymous robots at distinct positions, and r_l be the leader of the robot team. Also let $(\vec{u}, \vec{v})$ be the common coordinates, and ID_i be the allocated robot ID. The robots are able to find a solution for **Formation Control Problem** if the self-organizing robot team can have the following three functions, pattern generation, flocking, and pattern switching, in order to perform a cooperative task.

We decompose the formation control problem into three functions, as illustrated in Fig. 4. Function 1 enables a robot team to generate geometric shapes in a distributed control manner. Function 2 maintains the generated shape while robots are moving. Function 3 enables the team to adapt its current shape.

- *Function 1.* (PATTERN GENERATION) *Given a common coordinate system and ID allocations, robots at distinct positions form any geometric formation pattern.*
- *Function 2.* (FLOCKING) *Given a formation pattern, the robot team maintains the pattern while navigating toward a goal.*
- *Function 3.* (PATTERN SWITCHING) *Given a formation pattern, the robot team adapts the pattern according to the task and/or environmental changes.*

As mentioned above, no global coordinate system is given. Therefore, each robot is first required to reach agreement on the coordinate system. Then, robots obtain their IDs for task allocation to achieve the team mission. This is done in a self-organizing way.

3. Leader-referenced formation control

This section is concerned with the integration possibility of each function for formation control on the basis of the leader-referenced approach [30]. By simulations, we verify the features of the proposed approach, including self-organization, flexibility, and robustness.

3.1. Two types of formation patterns

We divide the formation patterns into the *uni-line type* and the *multi-line type*. If the pattern is single-lined, it is considered to be the uni-line type, otherwise the multi-line type. For the uni-line type, robots are positioned symmetrically with respect to the $\vec{u}$. This type includes circle, polygons, wedge, vertical line, horizontal line, arc, and fan-shape.

All patterns are set to keep a uniform interval d_u between robots. For the wedge, vertical line, and horizontal line in the uni-line and multi-line types, the interval means the same distance between robots. However, the interval indicates uniform circumference in the circle-type patterns that include circle, arc, fan-shape, and regular polygons. For instance, in the polygon-type patterns, the same arc length is maintained between robots along the circumscribed circle. Moreover, the *span* represents the size of the pattern, such that this argument is to multiply d_{in} by the number of $n-1$ robots.

Table 1 shows the parameters of the uni-line family. In Table 1, the parameter T represents the value of translation on the negative $\vec{u}$-axis direction from p_l. The parameter L is length from p_l to a target point f_i of F for r_i. The depicted notation, $\lfloor\ \rfloor$, means a solution after division rounds off a remainder. The m variable in the arc and the fan-shape patterns represents multiplicity such that the virtually increased number of robots generates a circle pattern. In implementation, we substituted the number 3 for m.

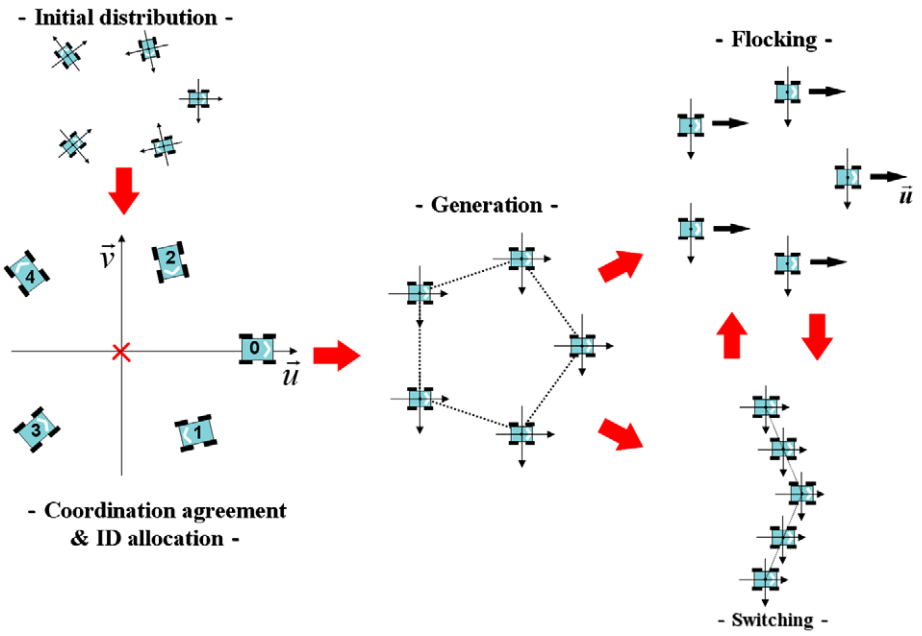

Fig. 4. Illustrating the concept of decentralized formation control.

Algorithm 2

Uni-line pattern generation (code executed by r_i)

INPUT: $\{ID_i, p_l, \mathrm{F}\}$
1 **IF** $\{ID_i$ is even$\}$ **THEN** $mark := 1$
2 **ELSE** $\{ID_i$ is odd$\}$ $mark := -1$
3 **END IF**
4 $u_{t,i} := cos\theta \times L + T$
5 $v_{t,i} := sin\theta \times L \times mark$ **OUTPUT**: $f_i = (u_{t,i}, v_{t,i})$

On the one hand, the multi-line type in this paper requires virtual links between lines. This means that the type cannot show consecutive ID distributions. Although an arrow or cross pattern may be generated, the multi-line type is limited to only n columns or n rows.

3.2. Pattern generation

Based on the above defined types, we explain two pattern generation algorithms. A robot r_i at distinct positions computes its f_i with respect to p_l. First, the uni-line type pattern generation is provided in Algorithm 2. By the *mark* variable, if r_i has even ID_i, then it is located in the left half plane of $\vec{u}$-axis. Otherwise robots with odd ID_i are located on the other side. Subsequently, r_i computes the three parameters, θ (angle), L (length), and T (translation) (see Table 1), and then obtains $f_i = (u_{t,i}, v_{t,i})$ determined according to computing parameters of the target pattern. Importantly, a team of robots can generate various geometrical uni-line type patterns by changing these three parameters of each pattern.

Table 1
Three pattern parameters for uni-line type

Pattern	θ (angle)	L (length)	T (translation)
Wedge	$(-1) \times \alpha^*$	$(-1) \times d_o$	u_l
Horizontal line	$\alpha_c = 90^\circ$	d_o	u_l
Vertical line	$\alpha_c = 0^\circ$	$d_o \times mark$	u_l
$0^\circ < \alpha^* < 90^\circ, \quad d_o = d_u \times \lfloor \frac{ID_i+1}{2} \rfloor$			
Circle	α	d_c	$u_l - L$
Arc	$\frac{\alpha}{m}$	$\frac{d_c}{m}$	$u_l - L$
Fan-shape	$(180^\circ - \frac{\alpha}{m})$	$\frac{d_c}{m}$	$u_l - L$
$\alpha = \frac{360^\circ}{n} \times \lfloor \frac{ID_i+1}{2} \rfloor, \quad d_c = \frac{360^\circ \times d_u}{2\pi \times n}$			
k-polygon $k_m = \frac{\cos\left(\frac{360^\circ}{2 \times k}\right)}{\cos\left(\frac{180^\circ}{k} - \frac{360^\circ}{n} \times \left(\lfloor \frac{ID_i+1}{2} \rfloor - \frac{n}{k} \times \lfloor \frac{(ID_i+1)/2}{n/k} \rfloor\right)\right)}, \quad d_c = \frac{360^\circ \times d_u}{2\pi \times n}$	α	$k_m \times d_c$	$u_l - L$

Algorithm 3

Multi-line pattern generation (code executed by r_i)

INPUT: $\{ID_i, p_l, \mathrm{F}\}$
1 **IF** $\{\mathrm{F}$ is column type$\}$ **THEN**
2 **IF** $\{n\%k = 0\}$ **THEN**
3 $NoLine := \lfloor n/k \rfloor$
4 **ELSE** $\{n\%k \neq 0\}$
5 $NoLine := \lfloor n/k \rfloor + 1$
6 **END IF**
7 **ELSE** $\{\mathrm{F}$ is row type$\}$
8 $NoLine := k$
9 **END IF**
10 **IF** $\{NoLine\%2$ is even$\}$ **THEN**
11 $L := \lfloor ((ID_i\%NoLine) + 1)/2 \rfloor \times d_u$
12 **ELSE** $\{NoLine\%2$ is odd$\}$
13 **IF** $\{\lfloor ID_i/NoLine \rfloor$ is even$\}$ **THEN**
14 $L := \lfloor ((ID_i\%NoLine) + 1)/2 \rfloor \times d_u$
15 **ELSE** $\{\lfloor ID_i/NoLine \rfloor$ is odd$\}$
16 **IF** $\{ID_i$ is even$\}$ **THEN**
17 $L := \lfloor ((ID_i\%NoLine) + 2)/2 \rfloor \times d_u$
18 **ELSE** $\{ID_i$ is odd$\}$
19 $L := \lfloor (ID_i\%NoLine)/2 \rfloor \times d_u$
20 **END IF**
21 **END IF**
22 **END IF**
23 **IF** $\{ID_i$ is even$\}$ **THEN** $mark := 1$
24 **ELSE** $\{ID_i$ is odd$\}$ $mark := -1$
25 **END IF**
26 $u_{t,i} := u_l - \lfloor ID_i/NoLine \rfloor \times d_u$
27 $v_{t,i} := L \times mark$
OUTPUT: $f_i = (u_{t,i}, v_{t,i})$

Secondly, Algorithm 3 explains how to generate multi-line types. Here, according to whether the consecutive ID distribution is parallel to $\vec{u}$-axis or $\vec{v}$-axis, we call the generated pattern row type or column type, respectively. Specifically, the number of lines *NoLine* can be defined as the number of the parallel lines with respect to $\vec{u}$-axis. To generate a multi-line type, r_i first checks whether its F is column type or row type. Then, r_i computes *NoLine* as presented in Algorithm 3 where the notation % represents a remainder of a division operation. Next, L described in Table 1 is obtained using *NoLine* and ID_i. Finally, r_i obtains its $f_i = (u_{t,i}, v_{t,i})$ in F. Practically, under Algorithm 3, the team of robots can generate *NoLine* lines only if the number of team members is more than *NoLine*.

Fig. 5 displays how to generate the diamond pattern using 12 robots. Fig. 6 shows that the robot team generated eight different formation patterns from the same initial distribution of Fig. 5. In simulation results shown in Figs. 5d and 6f, the robot teams can generate different diamond patterns by changing θ parameter. Fig. 7 presents the same pattern generated by 5, 10, and 15 robots, respectively. Regardless of the number of robots and initial states, the robots could build their team in a self-organizing way and generate their F through cooperation. In the two generation strategies, r_l becomes the reference point for each robot, since r_l remains stationary. More specially, the generated patterns in uni-line types are symmetrically arranged with respect to the $\vec{u}$-axis. The robots with even ID_i are located in the left half plane of the $\vec{u}$-axis, and the remaining robots with odd ID_i on the other side. Robots positioned closer to r_l are assigned higher IDs (from ID_1 to ID_4).

3.3. Flocking

Flocking in the leader-reference approach means that r_l navigates a path toward achieving a goal while the follower robots keep pace with r_l. As shown in Fig. 8, followers uniformly maintain the distance $\text{dist}(p_i, p_l)$ to r_l from r_i and the angle $\text{ang}(\vec{x}_i, \vec{c}_i)$ between the local $\vec{x}_i$- axis of r_i and $\vec{c}_i$ which indicates the distance vector connecting to p_l from p_i. All followers maintain $\text{dist}(p_i, p_l)$ and $\text{ang}(\vec{x}_i, \vec{c}_i)$ with r_l, which remain unchanged during flocking. Moreover, the followers are not allowed to move until r_l starts moving. We performed a simulation with the arc pattern in Fig. 9 where the leader conducted the followers to the goal point, at a distance of 100 units and at the 30° angle from the start point.

3.4. Pattern switching

For pattern switching, r_l remains stationary and all follower robots move to new positions. The difference from pattern generation is that the leader and existing robot IDs remain unchanged, as illustrated in Fig. 10. Specifically, the current p_l is set to p'_o of the new common coordinate system $(\vec{u}', \vec{v}')$. The new common direction $\vec{u}'$ is equal to the $\vec{u}$. Next, r_i updates its coordinates (u'_i, v'_i) with respect to $(\vec{u}', \vec{v}')$, computes the new target coordinates $(u'_{t,i}, v'_{t,i})$ according to the new assigned pattern F′, and moves to $(u'_{t,i}, v'_{t,i})$ in F′. The top row of Fig. 11 shows pattern switching from the circle to two column, and the bottom row of Fig. 11 shows results of changing from the two row to the triangle. From simulation results, r_l remains stationary to help the followers generate F′ as a reference point.

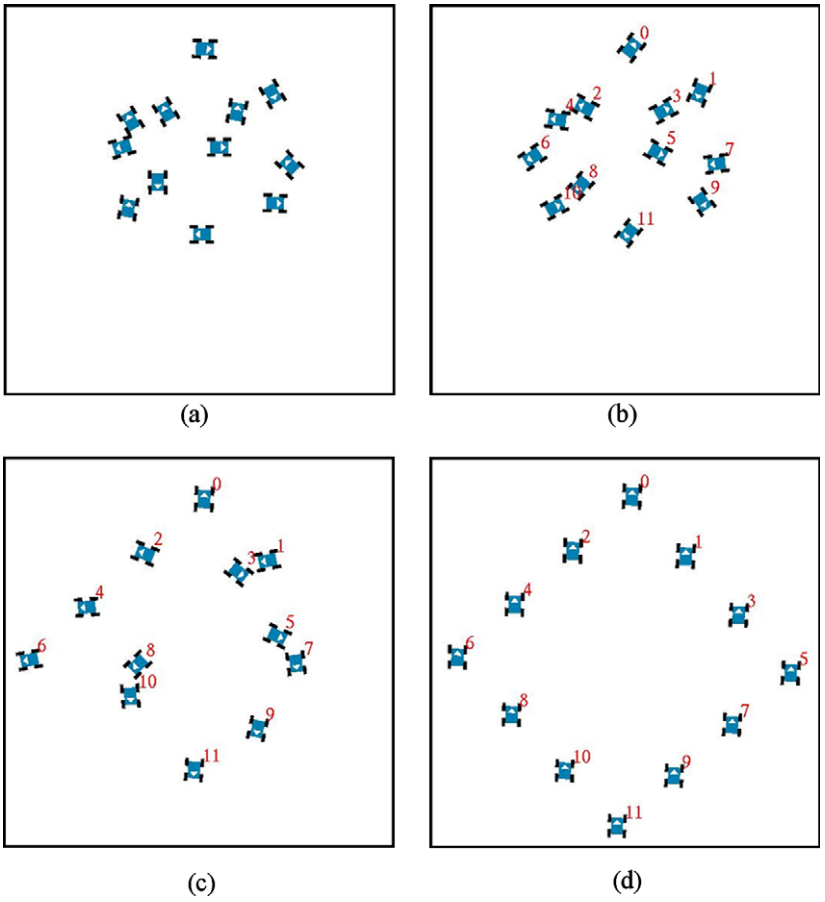

Fig. 5. Simulation results of diamond pattern generation by leader-referenced formation control: (a) initial distribution, (b) ID allocations, (c) moving to each target point, and (d) diamond pattern generation.

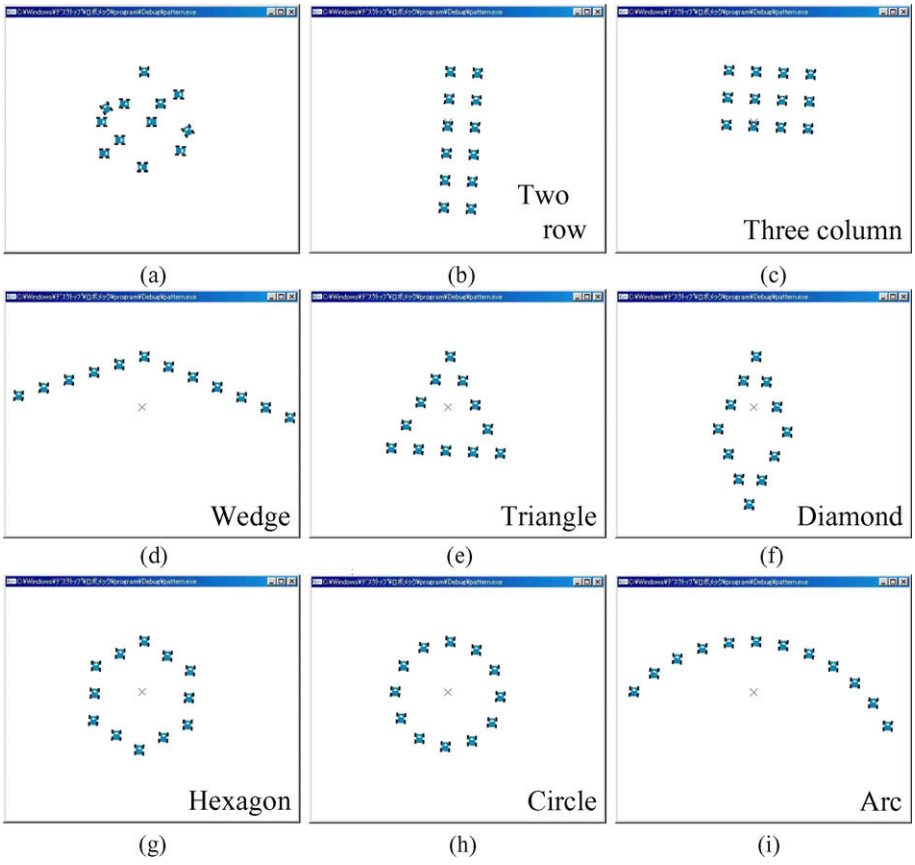

Fig. 6. Simulation results of eight patterns with 12 robots (multi-line type patterns: (b and c); uni-line type patterns: (d–i)): (a) initial distribution, (b) 11.15 (s), (c) 5.97 (s), (d) 9.28 (s), (e) 9.58 (s), (f) 6.2 (s), (g) 6.09 (s), (h) 6.44 (s), and (i) 9.35 (s).

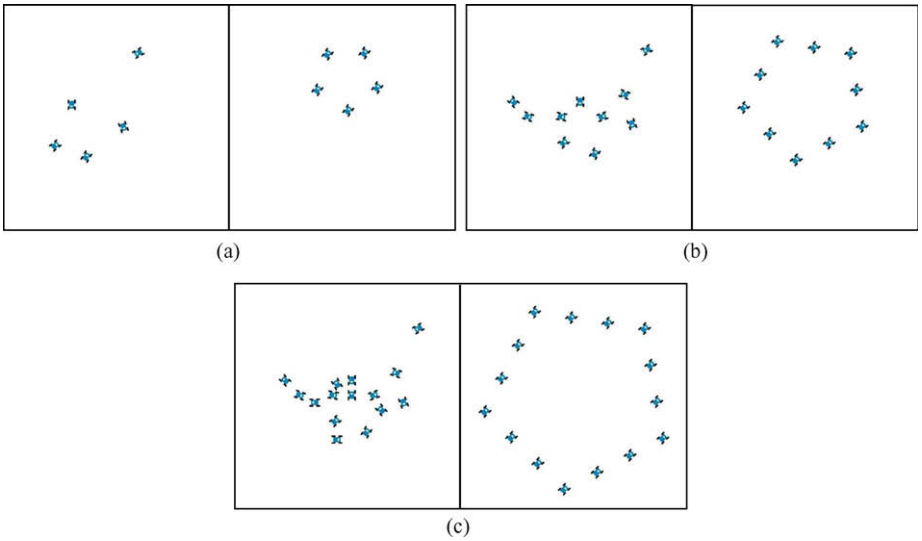

Fig. 7. Pentagon pattern generation with different numbers of team members: (a) pentagon generation with 5 team members, (b) pentagon generation with 10 team members, and (c) pentagon generation with 15 team members.

3.5. Robustness

Robot teams are often required to perform an assigned task continuously, regardless of accidental loss of a team member. If r_i becomes incapable of participating in the task, it broadcasts its ID_i. Other members recognize the loss of the member and regenerate the same or similar formation pattern as closely as possible.

We consider how to recover team formation after the loss of r_l or a follower. To begin, if a follower r_i fails to function, the team of robots can achieve the same or similar pattern by resetting the common coordinates, so that p_l turns into a new p'_o, and then reissuing new ID_i. Next, if r_l fails, the team is not able to immediately organize the same or similar pattern due to the loss of the p_o. Therefore the rest of the team members must re-organize the robot

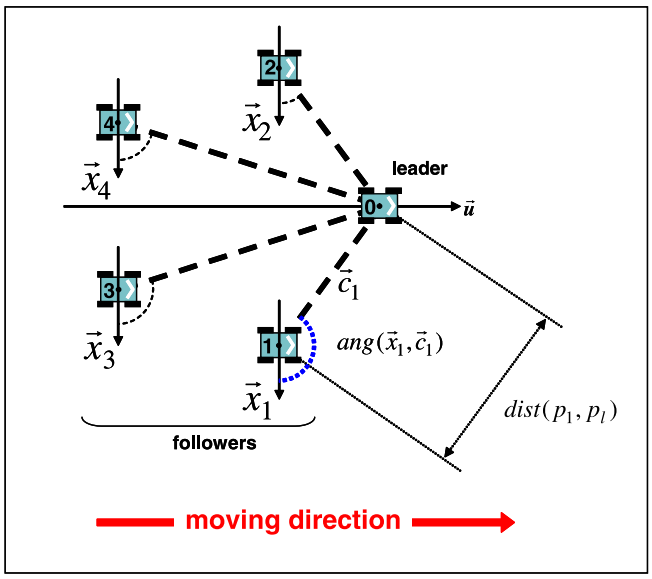

Fig. 8. Illustration of flocking approach.

teams from the current distribution, excluding the failed r_l. They repeat computation of p'_o, selection of a new r'_l, setting $\vec{u}'$ according to p'_l, acquisition of p'_i $(= (u'_i, v'_i))$ with respect to new agreed $(\vec{u}', \vec{v}')$, and reissuing new ID'_i. Note that, after the process of the new ID allocations, d_u may vary in both the number of participating robots and *span*. In order to maintain the *span* of a regenerated pattern in spite of the variation of d_u due to the loss of a team member, the robot team can self-adjust a uniform interval according to the number of participating robots and *span*.

Fig. 12 presents the simulation snapshots for the results of the same pattern generation with the same *span* after r_l failure. While flocking in the wedge pattern, r_l stops, and immediately the remaining robots select a new leader r'_l and reform another wedge pattern. The rest of the members gradually converge into the target pattern in Fig. 12e while r'_l navigates toward the goal. More specially, by using a predetermined *span* and the number of actively participating *n* robots, the team is able to preserve the size of a formation pattern even though the team lacks some members. Fig. 13 shows the snapshots of pattern recovery when a follower fails. Using these simulations, the robustness is verified against the accidental failure of a robot team member.

3.6. Formation control based on leader-reference

Robot team formation needs to have flexibility because robots are expected to be deployed in an unknown task and environments. Thus, F should be capable of switching from one pattern to another. Fig. 14 indicates how the robot team flocks adapting to an environment. In this simulation, the team navigates toward a stationary goal (the cross) at a distance of 75 units at the 15° angle, after forming the circle pattern. On the way to the goal, the team encounters an obstacle, forcing them to switch into the two column pattern, reducing the width of the team pattern (from Fig. 14b and c). Then the team re-flocks in the circle pattern after passing through the passageway (from Fig. 14c and d). Therefore, it is possible to enable a team of robots to form different patterns, flock, and change patterns adapting to the situations.

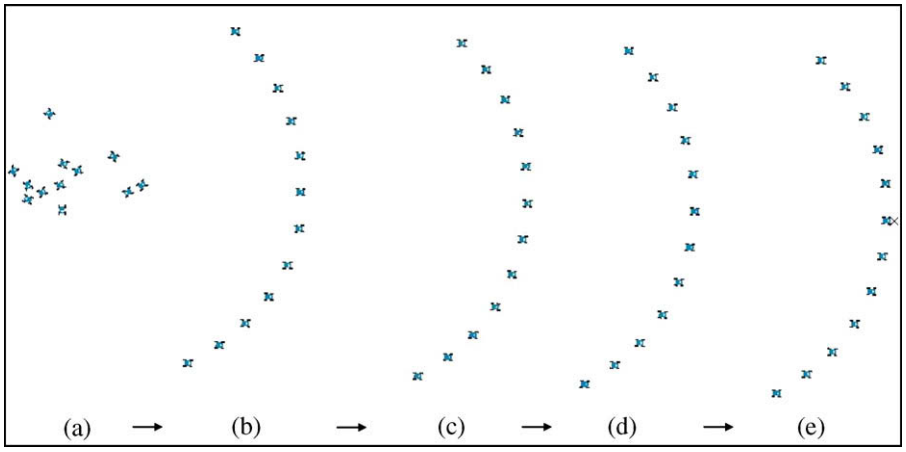

Fig. 9. Simulation results of flocking with arc pattern – task completion: 37.84 (s).

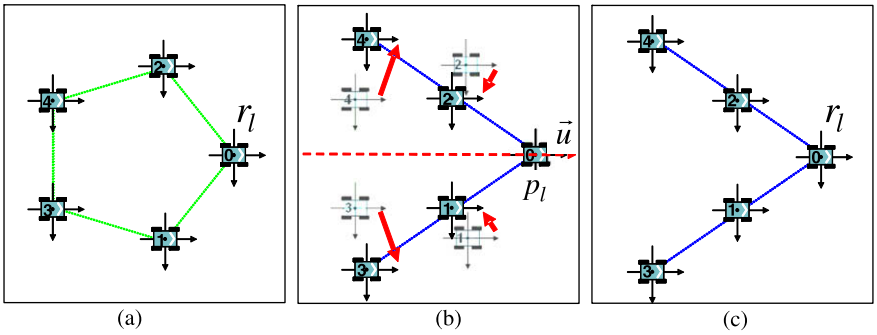

Fig. 10. Illustration of pattern switching approach. (a) Initial pen, (b) switching process and (c) final wedge pattern.

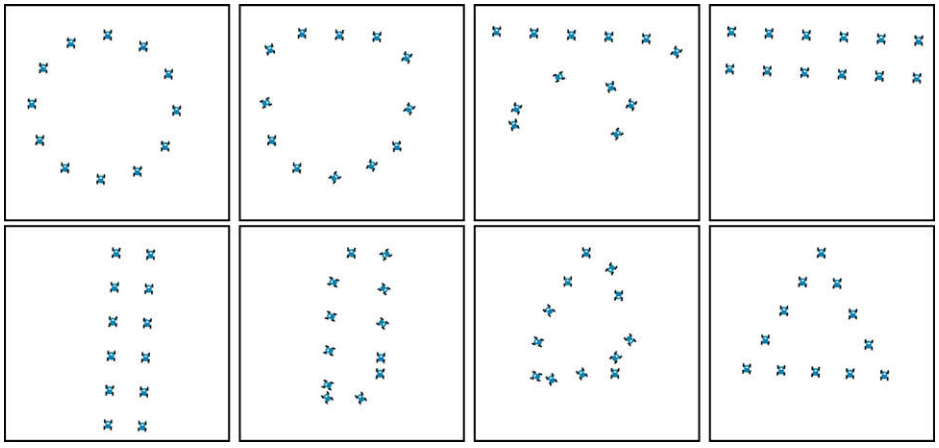

Fig. 11. Simulation results of pattern switching between uni-line type and multi-line type (top row: from circle pattern to two columns pattern, 18.25 (s); bottom row: from two rows pattern to triangle pattern, 23.71 (s)).

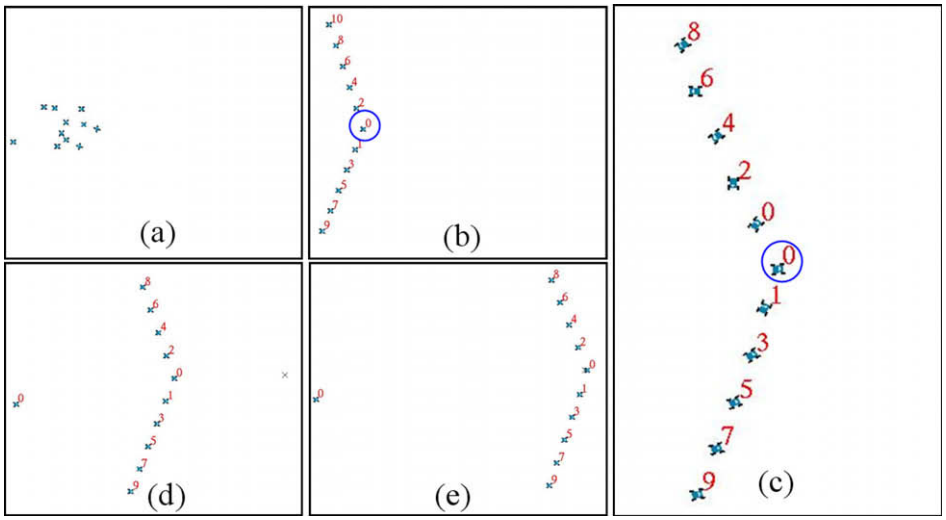

Fig. 12. Simulation results of robustness against loss in team population (Case I: failure of team leader).

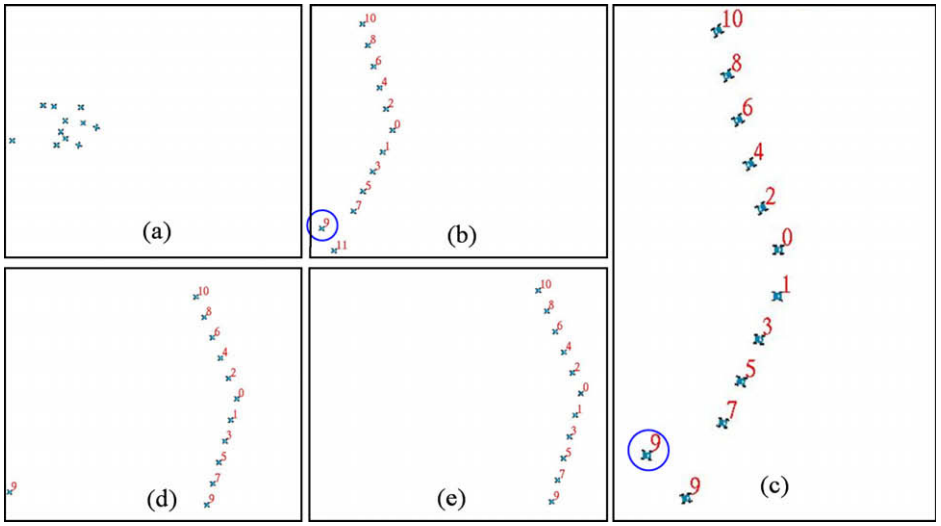

Fig. 13. Simulation results of robustness against loss in team population (Case II: failure of a follower).

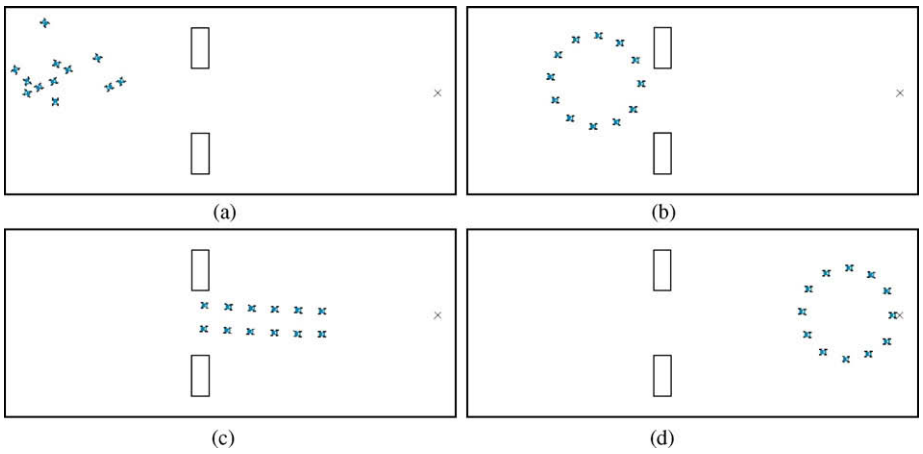

Fig. 14. Simulation results of leader-referenced formation control (example: adapting to an environment): (a) initial distribution, (b) 9.26 (s), (c) 19.52 (s) and (d) 33.93 (s).

Next, the robots first generate a circle pattern (shown in Fig. 15b) from the initial distribution shown in Fig. 15a, and then change into 6 different geometric patterns consecutively (see Fig. 15). r_l remains stationary to help the followers generate a pattern by sending messages for target patterns in the following order: hexagon, arc, two columns, diamond, circle, and wedge. Using 12 robots, the robot team could generate and switch to six different patterns.

3.7. Analysis

This part compares the team, labeled *A-team*, formed by the proposed self-organizing ID allocation, and another team, labeled *F-team*, composed of robots with initial fixed IDs. When the teams of 12 robots generated eight patterns, we examined the total distance moved and the time to complete pattern generation. The simulations were performed on 5 testbeds with each different robot distribution. In these simulations, we set and performed the same motions.

Fig. 16 illustrates the simulation results for each testbed, where the total distance moved (summing distance moved by all team members) and total time required by all robots till the completion of pattern generation are presented by solid bars, and those of any testbed-2 chosen arbitrarily are presented by lines. Gray bars indicate the results of A-team, and striped bars indicate results of F-team. The solid red line indicates data performed at testbed-2 by the A-team, and the blue dashed line is for the F-team. A-team exhibits less distance moved for all patterns. As we expected, leader selection and ID allocation determined by initial robot distribution is more efficient for A-team, since each robot can appropriately assign its task (or position) according to a task conditions or an environment. A-team also requires less pattern generation time for all patterns. We can see that leader-referenced formation control based on the self-organizing strategy provides improved efficiency for team pattern generation. The raw data are given in Tables 2 and 3.

4. Neighbor-referenced formation control

In this section, the neighbor-referenced approach is introduced to control formations [31]. We verify the features of the proposed approach, including self-organization, flexibility, and robustness by simulations.

4.1. Neighbor determination according to pattern types

In contrast to the leader-referenced approach, robots must be able to find their position with respect to their neighbors. Thus, each robot needs to be able to determine its own neighbor according to target pattern types. We describe how to determine the neighbor of each robot in this part.

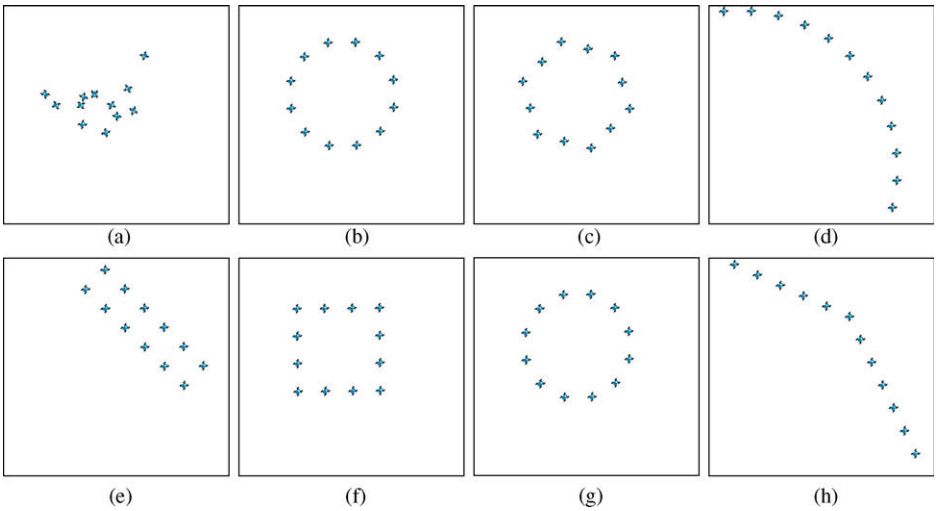

Fig. 15. Simulation results of continuous task switching: (a) 12 team members, (b) 4.38 (s), (c) 17.78 (s), (d) 32.57 (s), (e) 45.66 (s), (f) 67.97 (s), (g) 78.80 (s), and (h) 94.14 (s).

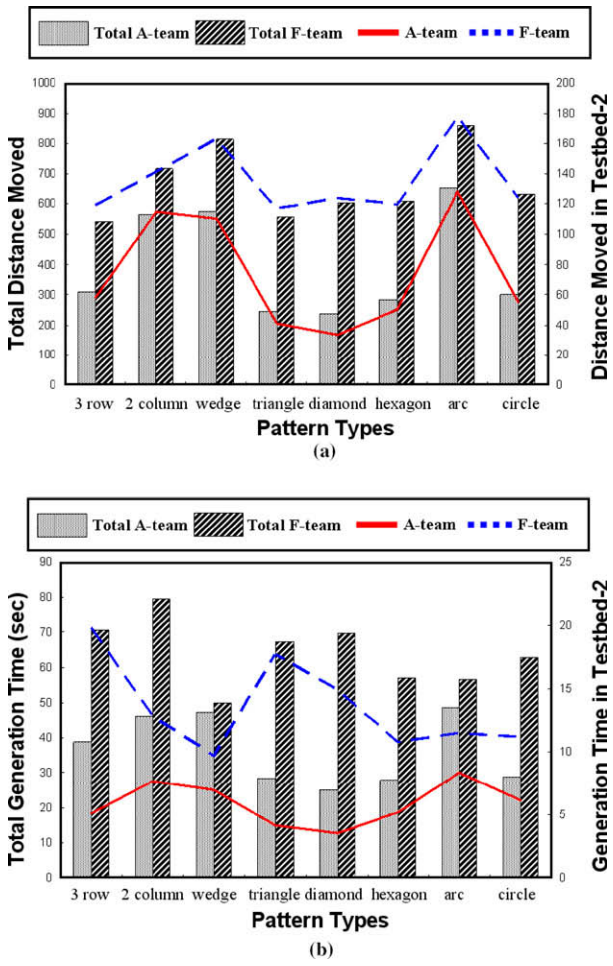

Fig. 16. Two comparison data graphs for pattern generation in leader-reference approach: (a) comparison of distance moved and (b) comparison of generation time.

By robot's ID_i and F, r_i must be able to determine its neighbor and generate a predetermined pattern as detailed in Algorithm 4. The main key to the neighbor-referenced approach is how to find a neighbor according to F. Patterns are largely divided into the two kinds, uni-line type and multi-line type, mentioned in Section 3. Therefore, we present two neighbor determination methods classified from F, and define the determined neighbor ID as N_{id}. For the uni-line patterns illustrated in Fig. 17a, the robots are symmetrically located in their target positions with respect to their $\vec{u}$. For those patterns, r_i with ID_i indicates the robot with ID_{i-2} as its neighbor. Especially, if the ID_i is 1 or 2, their neighbor is r_l and defined N_0.

Algorithm 4

Neighbor Determination (code executed by r_i)

INPUT: $\{ID_i, \mathrm{F}\}$
1 **IF** $\{\mathrm{F} : uni-line\ pattern\}$**THEN**
2 **IF** $\{ID_i > 2\}$ **THEN**
3 $N_{\mathrm{id}} := ID_i - 2$
4 **ELSE** $\{ID_i \leqslant 2\}$
5 $N_{\mathrm{id}} := 0$
6 **ELSE** $\{\mathrm{F} : multi-line\ pattern\}$
7 $NoLine := k$
8 $LiOrder := ID_i \% NoLine$
9 **IF** $\{LiOrder = 0\}$ **THEN**
10 $N_{\mathrm{id}} := ID_i - NoLine$
11 **ELSE** $\{LiOrder \neq 0\}$
12 $N_{\mathrm{id}} := ID_i - (LiOrder + 1)/2$
13 **END IF**
14 **END IF**
OUTPUT: N_{id}

Next, for multi-line type patterns, the determination of the neighbor is more complex. To begin with, r_i divides its ID_i by *NoLine* (see Algorithm 3). The remainder from the division indicates the order of multiple lines (denoted by *LiOrder*). For example, if

Table 2
Distance data in each testbed for A-team and F-team based on the leader-reference approach

Patterns	Testbed-1		Testbed-2		Testbed-3		Testbed-4		Testbed-5	
	A	F	A	F	A	F	A	F	A	F
Three rows	32.7	93.1	58.4	118.9	65.5	109.6	62.3	110.6	89.9	112.1
Two columns	71.4	124.1	115.1	141.1	118.9	167	117.4	132.3	144.4	152.2
Wedge	90.9	157.5	110.4	163.2	113.2	170.5	109.1	154.7	151.3	169.5
Triangle	49.3	107.8	41	116.7	37.7	113.5	38.1	111.8	80.9	107.9
Diamond	67.5	120.8	33.6	123.5	33.2	127.9	37.2	119.1	64.1	113.6
Hexagon	46.8	120.7	50.3	119.6	49.3	124	46.6	124.8	89.6	119.3
Arc	101.2	147.6	128.1	178.3	132.1	182.7	127.2	168.2	164.9	185.6
Circle	48.9	120.2	55.6	124.5	52.4	139.8	49.8	128.2	93	121.5

Table 3
Time data in each testbed for A-team and F-team based on the leader-reference approach (s)

Patterns	Testbed-1		Testbed-2		Testbed-3		Testbed-4		Testbed-5	
	A	F	A	F	A	F	A	F	A	F
Three rows	11.9	15.27	5.08	19.81	4.47	13.68	5.45	12.24	11.9	9.66
Two columns	11.61	20.12	7.64	12.82	6.52	27.08	8.85	11.66	7.81	11.61
Wedge	12.15	12.11	6.98	9.64	8.3	12.31	9.6	7.39	10.15	8.93
Triangle	6.66	17.13	4.17	17.69	3.54	11.86	7.36	11.47	6.66	9.32
Diamond	6.31	16.86	3.53	14.94	4.24	15.27	4.6	13.15	6.31	9.68
Hexagon	7.3	10.99	5.21	10.79	3.52	12.89	4.56	9.26	7.3	13.17
Arc	11.46	14.45	8.36	11.45	8.15	12.44	9.37	7.89	11.46	10.34
Circle	7.43	9.54	6.14	11.22	3.49	23.02	4.35	9.35	7.43	9.72

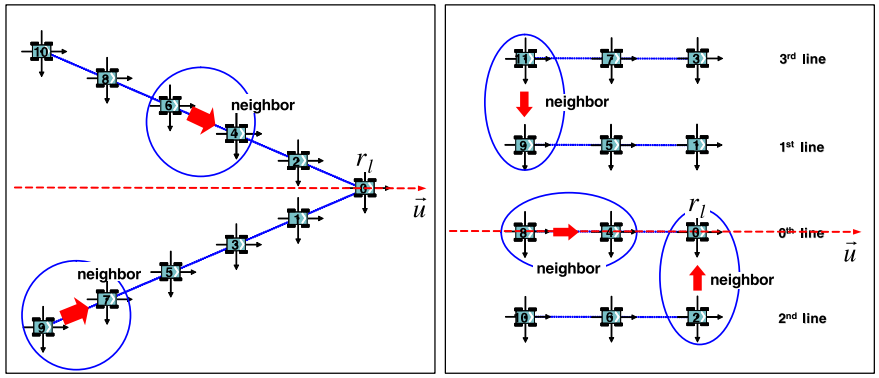

Fig. 17. Illustrating neighbor determination according to pattern types: (a) uni-line pattern and (b) multi-line pattern: four column.

the remainder is zero, the robot is located in the *zero*th line that includes r_l, as illustrated in Fig. 17b. Then, r_i finds its neighbor in the same line. In the *zero*th line, the neighbor can be found by subtracting *NoLine* from ID_i. Unless *LiOrder* is zero, the neighbor can be found by the following rule. First, r_i computes the quotient from the division of numerator, *LiOrder* plus one, by denominator, 2. Secondly, the quotient is subtracted from ID_i. Now N_{id} for the case of *LiOrder*≠0 can be rewritten as the equation: $N_{id} = ID_i - (LiOrder + 1)/2$. For instance, we display that the team of 12 robots forms four lines in Fig. 17b. Here, the robots positioned in the *third* line find the neighbor located in the *first* line.

4.2. Pattern generation

In order to generate a desired F from arbitrary distributions based on neighbor-reference, the follower r_i is required to be positioned accurately to meet the distance and angle constraints with respect to its N_{id}. In detail, r_i should determine a different local angle relative to N_{id} according to F. Therefore, the angle computation rule, termed *n-Vertex*, is applied to maintain the local angles. Let *n-Vertex* mean ID_i which is positioned on the vertex of F. For instance, the hexagon pattern has each *two-Vertices* in the even and the odd *ID*'s, respectively except for the vertex occupied by r_l. The triangle and diamond patterns are *one-Vertex* each other. For the wedge and one row patterns, no *n-Vertex* exists, which means that all robots must maintain the same local angle with their neighbors. For the triangle pattern as shown in Fig. 18, there exists *one-Vertex* each among the even and the odd *ID*'s, respectively. The robots having higher ID_i (e.g. from ID_1 to ID_6) than *one-Vertex* maintain the same local angle of 30° with respect to $\vec{u}$, while the remaining robot having lower ID_i than *one-Vertex* keeps at an angle of zero degrees with respect to $\vec{v}$. With the computation of the local angles, r_i follows their neighbor with d_u and local angle until the neighbor stops completely at the target position. After arriving at f_i, robots reach agreement on the heading direction along $\vec{u}$ by adjusting their orientation.

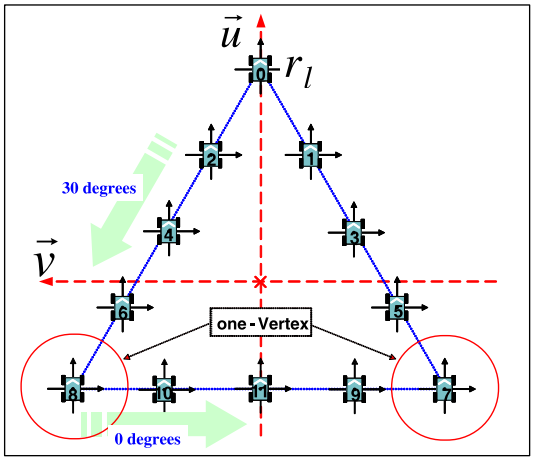

Fig. 18. Illustration of local angle computation in the triangle pattern based on *one-Vertex*.

We performed simulations of pattern generation from the same conditions as in Fig. 5. Fig. 19 displays how to generate a diamond pattern using 12 robots. Note that robots are aware of N_{id} according to their ID_i and F, but do not know how to go where, since robots follow their neighbors using the virtual linkage constraint (d_u and local angle). The followers kept pace with their neighbors while maintaining the pattern. Moreover, from the same conditions as the simulation results in Fig. 6, the team could generate eight patterns, and the elapsed times for completion of each pattern were as follows: two row 27.42 s, three column 38.44 s, wedge 25.86 s, triangle 28.5 s, diamond 28.77 s, hexagon 33.47 s, circle 32.21 s, and arc: 31.63 s (all counted times start from 0 s).

4.3. Flocking and pattern switching

Flocking in neighbor-reference approach means that r_l navigates a path toward achieving a goal while the follower robot r_i maintains its neighbor $r_{N_{id}}$ with N_{id} after F is obtained. Each robot uniformly maintains the distance $\mathrm{dist}(p_i, p_{N_{id}})$ to $r_{N_{id}}$ from r_i, and keeps the angle constraint $\mathrm{ang}(\vec{x}_i, \vec{c}_{N_{id}})$ between the local $\vec{x}_i$-axis of r_i and $\vec{c}_{N_{id}}$ which indicates the distance vector connecting to $p_{N_{id}}$ from p_i. During flocking, r_l and ID_i remain unchanged. Note that, compared with the flocking approach in the leader-reference approach, the flocking in the neighbor-reference approach is based on $r_{N_{id}}$ as a reference point. Fig. 20 presents the snapshots of the simulation results of flocking in an arc pattern from the same initial conditions as in Fig. 9. The leader conducted the followers to the target point located a distance of 100 units away at the 30° angle. The followers kept pace with their neighbors while maintaining the pattern.

Next, while switching patterns, r_l and ID_i remain unchanged. The new common direction $\vec{u}'$ is the same as the current $\vec{u}$. However, the followers had to establish a new geometric relation with their neighbors according to F′. We tested pattern switching between uni-line patterns and multi-line patterns. The top row of Fig. 21 shows switching from the uni-line circle to the two column pattern, and the bottom row of Fig. 21 shows the change from the two row pattern to the uni-line triangle, respectively. From simulation results, r_l remains stationary to help the followers generate F′.

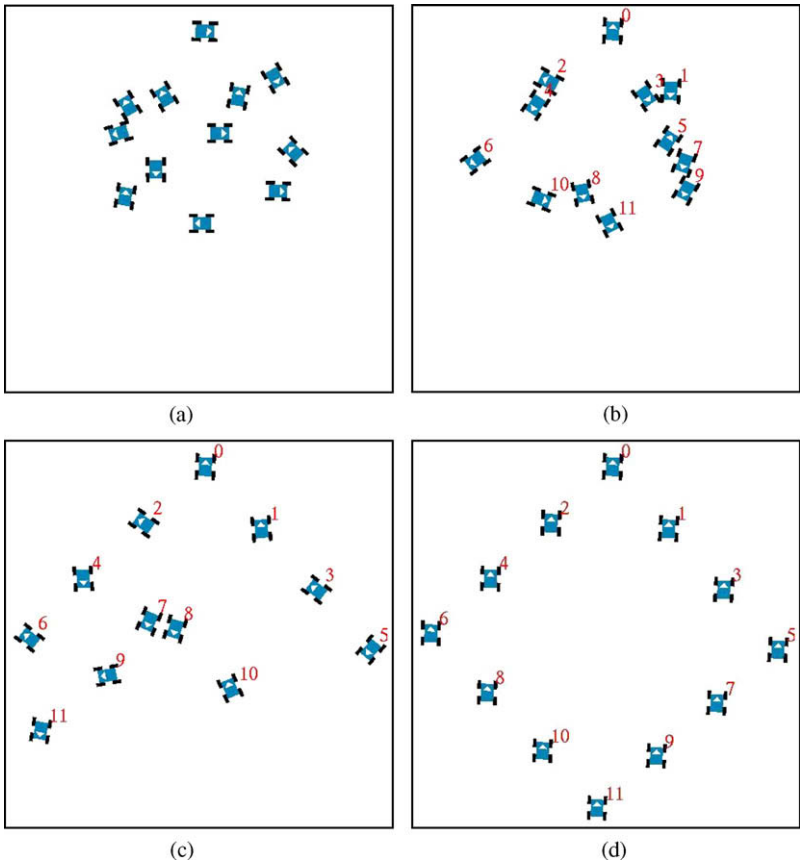

Fig. 19. Simulation results of diamond pattern generation by neighbor-referenced formation control: (a) initial distribution, (b) neighbor determination by ID, (c) neighbor tracking, and (d) diamond pattern generation.

4.4. Robustness

The robustness is verified against the accidental failure of team members, as shown in Fig. 22, where the simulation results of replacement pattern generation (with an equal *span*) after a follower fails to move are presented. While flocking in the wedge pattern, a robot stops, and immediately the remaining robots attempt to regenerate another wedge pattern by just re-issuing a new ID_i'. Fig. 22c shows that the replacement pattern has reissued ID_i. By both a predetermined *span* and the number of participating *n* robots, the team was able to preserve the size of F and re-flocks toward the goal. Note that d_u should vary with the number of remaining robots, if we want to keep *span* of the pattern unchanged.

4.5. Formation control based on the neighbor-reference

Fig. 23 shows how the team adapts patterns in a variety of circumstances using the neighbor-referenced approach. In this simulation, the robot team navigates toward a target located 75 units away, at the 15° angle. On the way to the goal, the robot team encounters an obstacle that forces the team to switch into another

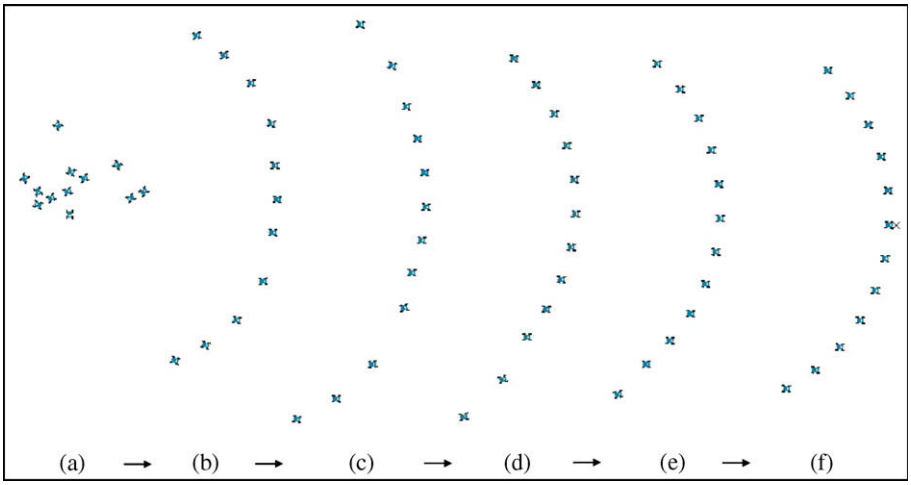

Fig. 20. Simulation results of flocking with arc pattern – task completion: 44.02 (s).

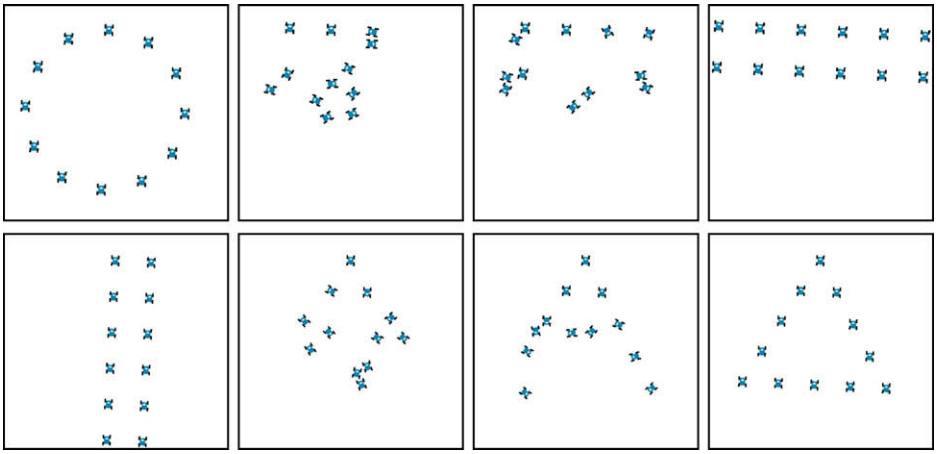

Fig. 21. Simulation results of pattern switching between uni-line type and multi-line type (top row: from circle pattern to two column pattern, 63.39 (s); bottom row: from two row pattern to triangle pattern, 63.95 (s)).

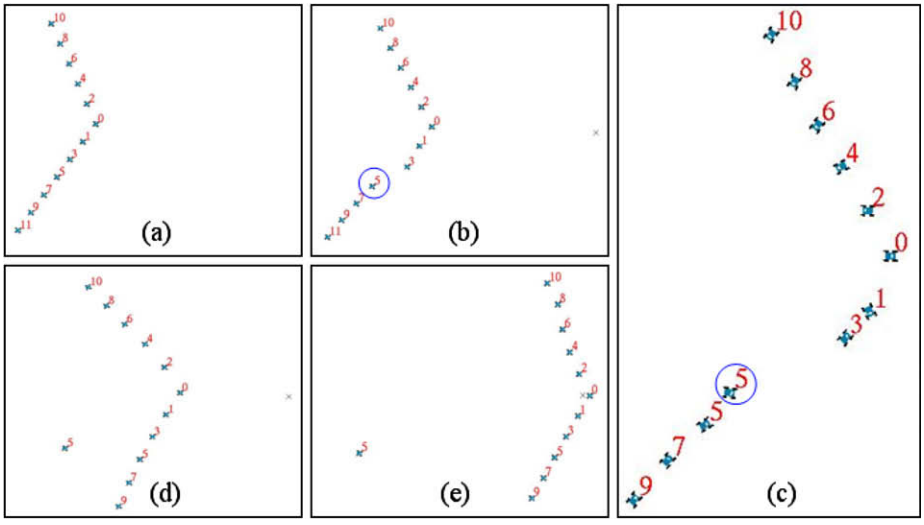

Fig. 22. Simulation results of robustness against loss in team population (failure of a follower).

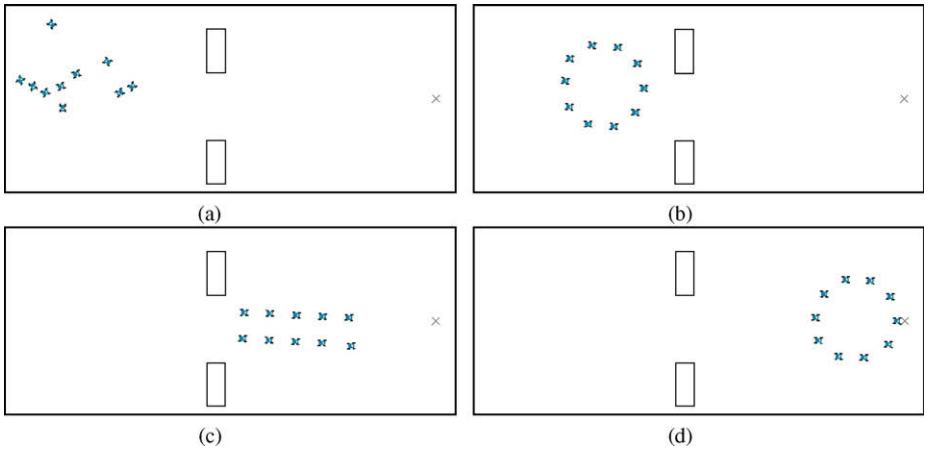

Fig. 23. Simulation results of neighbor-referenced formation control (example: adapting to an environment). (a) initial distribution, (b) 18.49 (s) (c) 41.02 (s), and (d) 62.11 (s).

pattern, which reduces the width of the team pattern, to pass through the passageway (from Fig. 23b and c). r_l decided an appropriate F′ and remained as the stationary post for formation switching. Then, the team could arrive at the goal with the circle pattern after passing through the passageway with the two row pattern (from Fig. 23c and d). Therefore, employing the proposed tech-

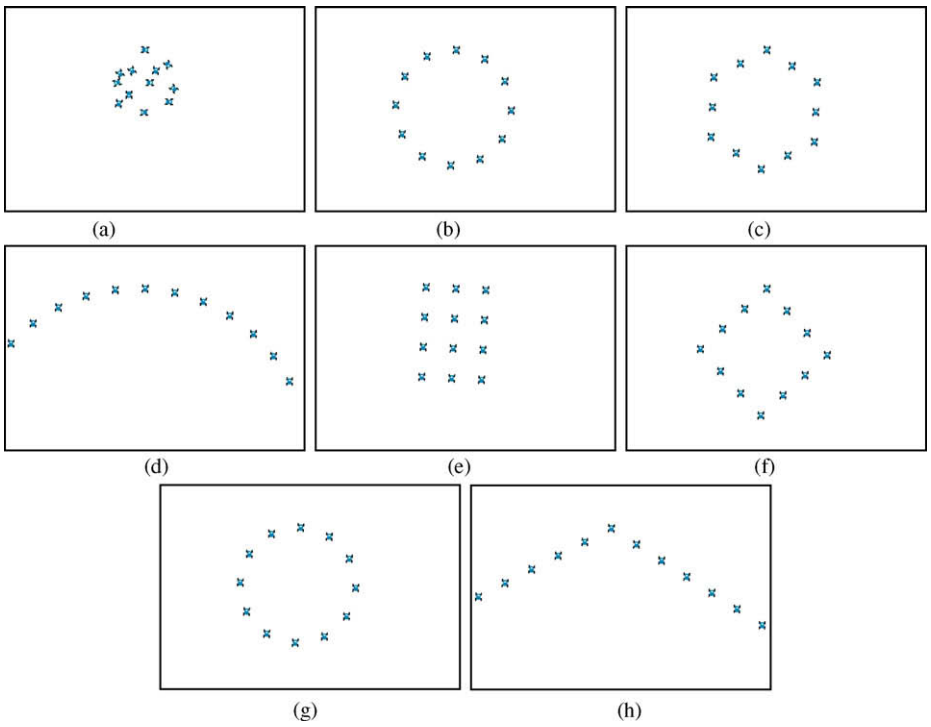

Fig. 24. Simulation results of continuous task switching. (a) 12 team members, (b) 29.3 (s) (c) 63.2 (s), (d) 106 (s), (e) 149.9 (s), (f) 216.62 (s), (g) 250.13 (s), and (h) 281 (s).

nique, it is possible for a team of robots to generate different geometric shapes, navigate by forming a team, and change formations by adapting to circumstances. Moreover, from an initial distribution in Fig. 24a, the 12 robots generate a circle pattern and change patterns into six different shapes consecutively. Note that, in Fig. 24, the team generates the circle pattern twice, which demonstrates the reliability of pattern switching from any given situation.

4.6. Analysis

This part compares the team, labeled *A-team*, formed by the proposed self-organizing approach, and another team, labeled *F-team*, composed of robots with initial fixed ID. We repeated the same simulation performed in Section 3.7 under the same conditions.

Fig. 25 illustrates the simulation results for each testbed, where the total distance moved and time are presented by bars, and those of testbed-2 are presented by lines. Gray bars indicate the result of A-team and striped bars indicate that of F-team. As a result, the proposed A-team exhibits less distance moved and less generation time than the F-team for all patterns. The solid red line indicates data from testbed-2 by the A-team and the blue dashed line is by the F-team. Both A-team and F-team show the trend of a regular generation time except for the hexagon pattern. The hexagon pattern has *two-Vertices*. Since the team generates this pattern from six local angle relationships, the generation time is relatively long. Even though the data are different, the fluctuations of two lines and two bars have similarities. The raw data for the two graphs are given in Tables 4 and 5.

5. Comparison between leader-reference and neighbor-reference

In this section, we present and discuss a comparison between leader-referenced formation control and neighbor-referenced formation control. Based on the comparison data, a hybrid approach is introduced.

5.1. General comparison between two approaches

We discuss the difference in robots' behavior between the two approaches. Fig. 26a shows the total distance moved and pattern generation time for eight patterns. This graph is the rearrangement of the A-team data shown in Figs. 16 and 25. We denote the total distance moved by bars, and the total generation time by lines. White empty bars indicate the results of the leader-reference approach and slashed bars indicate the results of the neighbor-reference approach. The red line indicates the time required by the leader-reference approach and the blue dashed line indicates the variation of the elapsed time by the neighbor-reference approach. For each pattern, the distance moved and the generation time with the neighbor-reference approach were much longer than with the leader-reference approach. In the process of pattern generation in the neighbor-reference approach, r_i finds its N_{id}, and then keeps tracking $r_{N_{\mathrm{id}}}$ until $r_{N_{\mathrm{id}}}$ stops completely. Unlike the neighbor-reference approach, that in the leader-reference approach enables robot teams to compute and move toward $(u_{t,i}, v_{t,i})$ with respect to p_l in order to form an assigned F. Therefore, leader-referenced formation control seems to be more efficient than the neighbor-referenced approach. The leader-referenced approach requires all follower robots to acquire p_l.

Secondly, we investigate the distance moved by the robots with high ID_i and low ID_i for the leader-reference and neighbor-reference approaches, according to each pattern, as shown in Fig. 26b. Here, the high ID_i indicates IDs 1–4 and the low ID_i indicates IDs 8–11 among 12 robots. From the result, the low ID_is move a longer distance than the high ID_is in the leader-reference approach based on our proposed self-organizing strategy, but move a shorter distance in the neighbor-reference approach. The reason is that the movement of each robot does not depend on other robots in the leader-reference approach. In contrast, the movement of robots coordinated by the neighbor-reference approach is affected by their $r_{N_{\mathrm{id}}}$. Consequently, the leader-referenced approach is superior in terms of the distance moved for pattern generation.

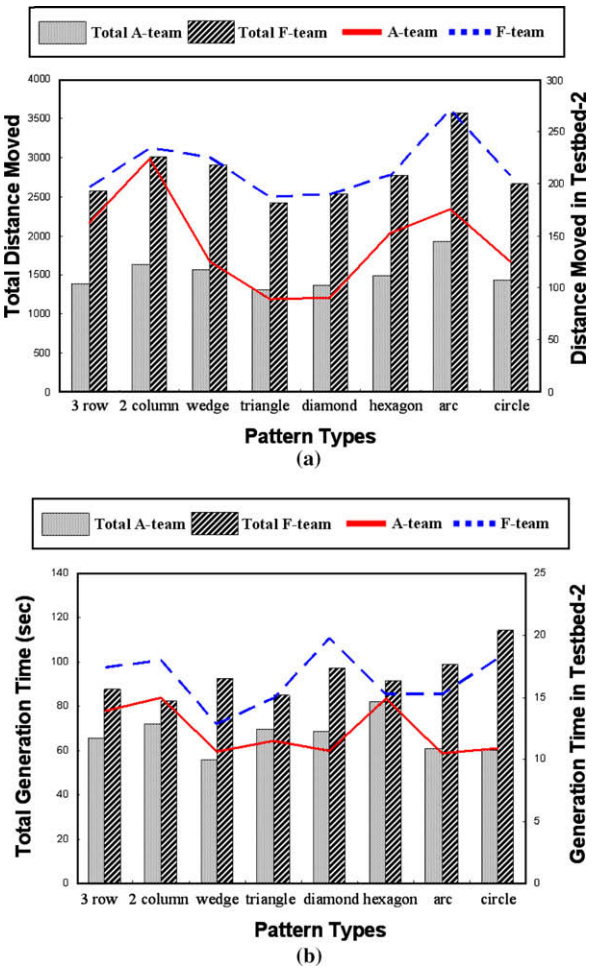

Fig. 25. Two comparison data graphs for pattern generation in neighbor-reference approach: (a) comparison by distance moved and (b) comparison by generation time.

Thirdly, we compare these two approaches regarding flocking. In Fig. 9, r_l navigated a path and the follower robots kept pace with it. In Fig. 20, r_i could also keep pace with its $r_{N_{id}}$. Although the behavior of r_i is similar, there exists a slight difference between the two approaches. For turning toward the target, the leader-reference robots tried to rotate their heading simultaneously. The neighbor-referenced robots rotated their heading in relation to their neighbors sequentially (see Fig. 20a–f). Even though it is difficult to conclude which is better, we can expect that the neighbor-referenced team may not exactly maintain F until it travels a long distance.

Finally, Figs. 13 and 22 presented the simulation snapshots of the same pattern generation with an equal task range after the failure of one robot. Only a failure robot stopped in the leader-referenced approach in Fig. 13, but, Fig. 22 showed that the robots with lower odd ID_i related to the failed robot all stopped. As a result, the neighbor-referenced approach has a greater influence on the behavior of each robot.

5.2. Comparison of positioning error

We investigated the cases of robot positioning error in pattern generation. The positioning error is considered to be an observation error in observing other robots, and/or a computing error for $(u_{t,i}, v_{t,i})$. We assume that the error does not last all through the task, but may happen at one time.

For the evaluations, we set the positioning error for the robots of ID_3, ID_6, and ID_9. Figs. 27 and 28 show the simulation results for generating the hexagon and wedge, respectively. Figs. 27a–d and 28a–d demonstrated the leader-referenced generation. Increasing the number of robots having errors, the patterns become distorted gradually. Similarly, in Figs. 27e–h and 28e–h generated by the neighbor-referenced approach, the pattern deviates from the designed one according to the number of robots having errors. Comparing these two approaches, a critical problem exists in the leader-referenced approach. For instance, the robots ID_3 and ID_5 do not keep d_u as shown in Fig. 28b. In Fig. 28c and d, the robot with ID_9 occupied the position for the robot ID_{11} that

Table 4
Distance data in each testbed for A-team and F-team based on the neighbor-reference approach

Patterns	Testbed-1		Testbed-2		Testbed-3		Testbed-4		Testbed-5	
	A	F	A	F	A	F	A	F	A	F
Three rows	152.9	159	162.7	197.2	141.7	207.3	182.9	197.8	174.1	218.2
Two columns	173.9	208.6	223.9	234.7	239	234.7	217	250.8	224.9	225.5
Wedge	102.4	206.1	125.2	225	142.3	227.6	126.6	218.5	179.7	235.2
Triangle	101.9	167.4	88.9	187.3	109.4	200	95.9	186.9	147.6	187.4
Diamond	82.7	189.3	90.7	189.7	121.3	209.8	94.6	203.2	152	187
Hexagon	102.2	204.3	152.4	208.1	121.3	220.7	137.4	211.7	160.8	219.6
Arc	150.4	255.4	175.5	271.3	184.9	273.9	164.1	287.2	229.1	278.5
Circle	109.9	179.5	125.1	208	144.5	213.4	110.3	209.9	170.5	207.1

Table 5
Time data in each testbed for A-team and F-team based on the neighbor-reference approach (s)

Patterns	Testbed-1		Testbed-2		Testbed-3		Testbed-4		Testbed-5	
	A	F	A	F	A	F	A	F	A	F
Three rows	12.29	20.91	13.91	17.38	13.23	14.67	12.79	16.34	13.32	18.15
Two columns	10.83	15.65	14.96	18.01	14.86	12.24	14.93	18.79	16.28	17.48
Wedge	9.17	15.78	10.63	12.83	9.34	13.96	10.63	25.09	16.15	24.94
Triangle	12.26	18.59	11.47	14.92	12.72	18.15	15.34	17.75	17.66	15.62
Diamond	13.8	24.47	10.68	19.78	17.73	20.17	12.24	16.05	14.05	17.02
Hexagon	13.8	24.39	14.92	15.24	11.2	14.45	16.75	13.95	25.23	23.18
Arc	10.73	21.65	10.52	15.27	12.61	23.78	13.38	17.45	13.96	20.56
Circle	11.23	22.64	10.91	18.12	13.79	15.43	8.74	24.08	15.57	34.06

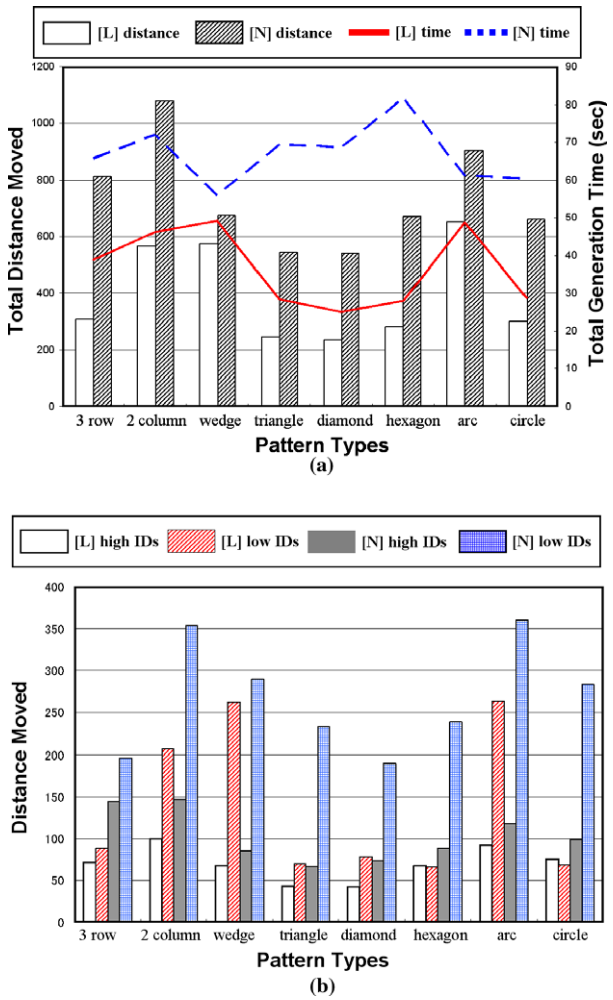

Fig. 26. Comparison between the leader-reference approach and neighbor-reference approach according to pattern generation: (a) comparison of distance moved and generation time and (b) comparison between high IDs and low IDs.

moved back and forth around the position. On the contrary, the neighbor-referenced approach could keep a uniform interval between robots in spite of deformed shapes, and could provide higher pattern maintenance stability.

5.3. Hybrid formation control

In the previous two parts, the leader-referenced approach showed superior performance in terms of the distance moved and the total time. On the contrary, the neighbor-referenced approach demonstrated stability against robot positioning errors. Basically, the leader-referenced approach requires all follower robots to keep observing the leader. However, it would be difficult for robots positioned near the trailing edge to keep their line of sight to the leader in a team in a one row pattern. The neighbor-referenced approach does not suffer from this put limitation.

This part presents hybrid formation control to overcome the line of sight problem in leader-referenced approach. The hybrid approach employs leader-referenced control that can be changed to the neighbor-reference control according to circumstances, and vice versa. Fig. 29 illustrates the simulation results of the hybrid formation control approach, where the leader-referenced six robot team encounters a narrow path and changes shape into a one row pattern, which should be controlled by the neighbor-referenced approach (from Fig. 29b and c). After the robot team exists the narrow passageway, the team re-switches the formation pattern to

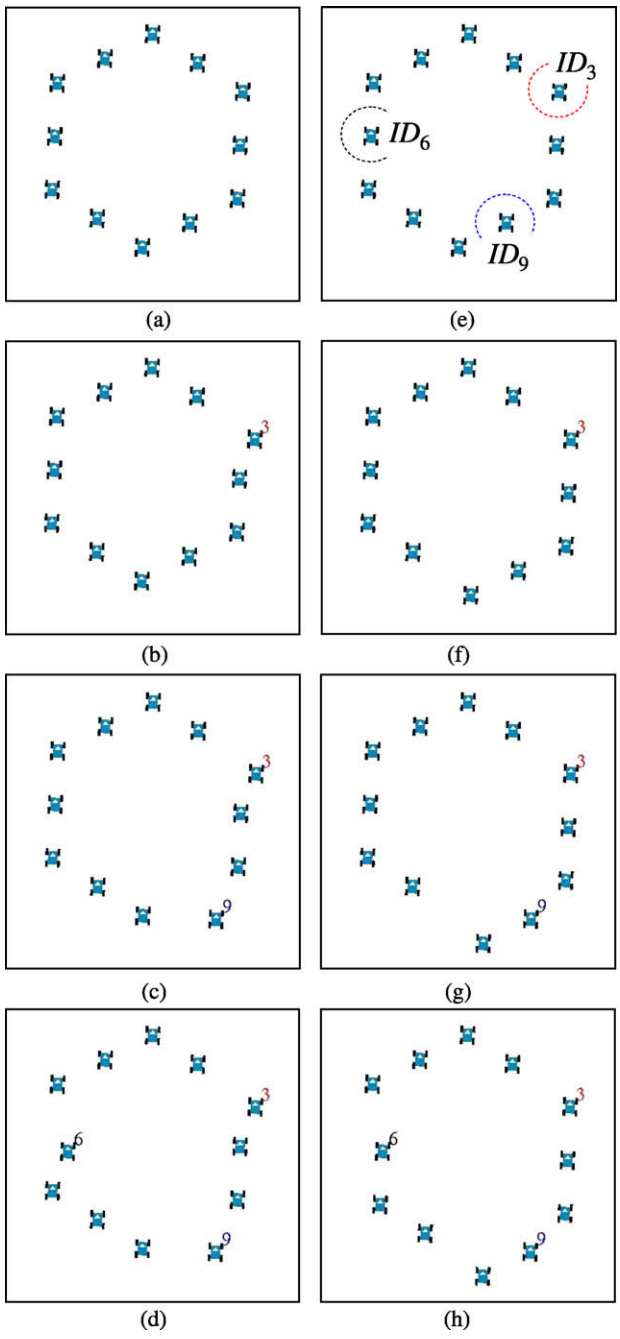

Fig. 27. Positioning error test for hexagon pattern generation by the leader-referenced formation control approach and by the neighbor-referenced formation control approach.

complete the original mission (from Fig. 23c and d). The proposed formation control approaches are composed of such activation cycles as sensing, computation, and motion. The successful completion of these steps depends upon the exact observation of other robots. The sensing capability of all robots is assumed to be unlimited and errorless, which is practically infeasible. Our physical robots in the next section were positioned initially within the boundaries of other robots' observation, with clear lines of sight.

6. Application to a small-scale team of mobile robots

We have developed a real mobile robot team of four Pioneer3-DXs (ActivMedia Inc.) in order to verify the leader-referenced

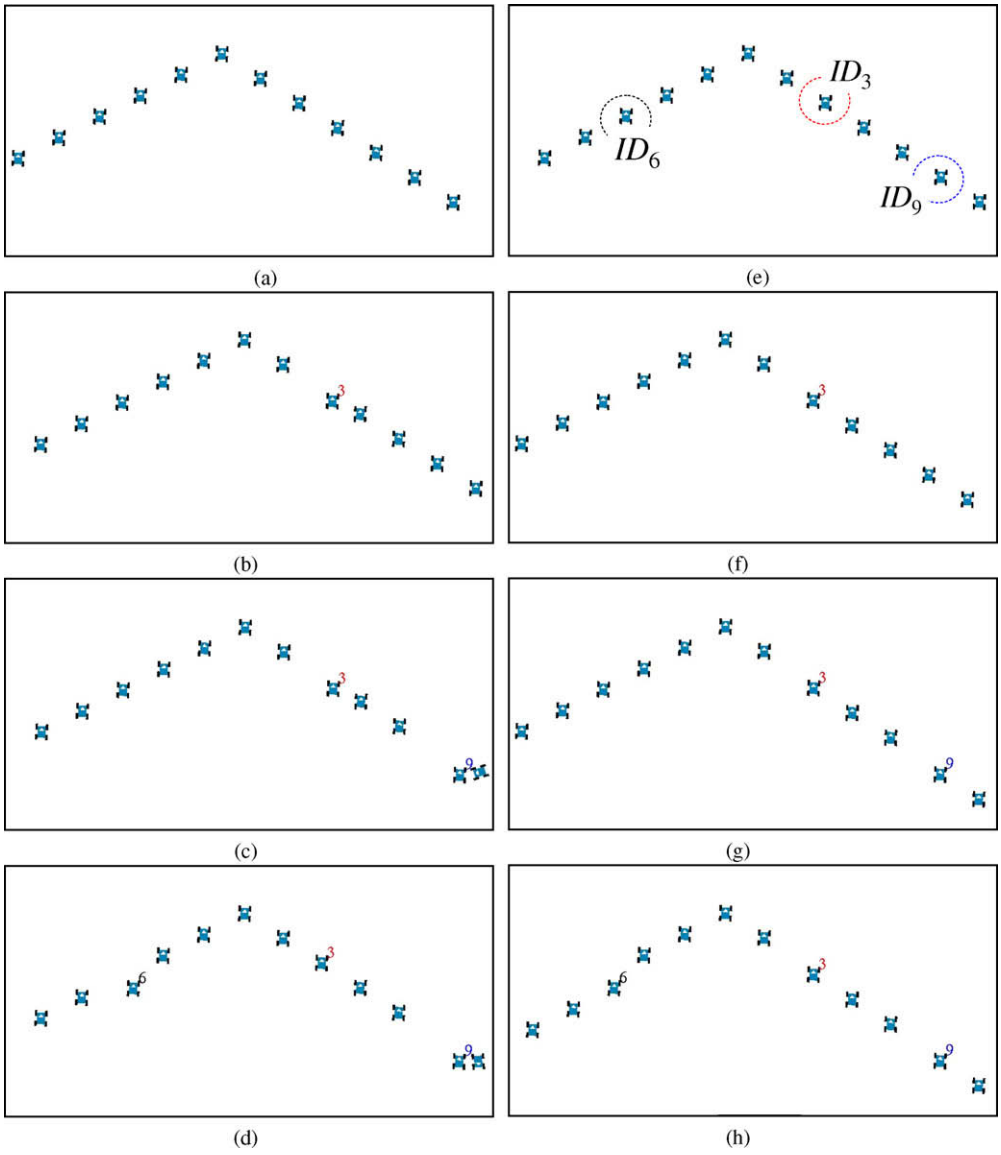

Fig. 28. Positioning error test for wedge pattern generation by the leader-referenced formation control approach and by the neighbor-referenced formation control approach: (a) no error (leader), (b) one robot error (leader), (c) two robot error (leader), (d) three robot error (leader), (e) no error (neighbor), (f) one robot error (neighbor), (g) two robot error (neighbor) and (h) three robot error (neighbor).

formation control approach. Practically, a physical robot is equipped with 16 sonar sensors, and control programs run on a laptop computer over the robot. All robots allow communication with each other by wireless LAN in order to broadcast their ID, and to send or receive formation commands.

6.1. Implementation method

The key to the applicability of the proposed approach lies in obtaining reliable estimates of the center points of the other robots with respect to each other. One problem is that real robots might have an elliptic shape. According to the robot heading, the distance between an edge and the center point would vary. Moreover, the robot might have an unequal interval of sonar sensors. Thus, the blind range would not be uniform, and the observed edge of the real robot may not be smoothly connected.

In this paper, image processing techniques are employed to recognize the centers of each robot using only sonar sensors. To begin, we made a 5000 × 5000 mm 2D-grid with 50 × 50 mm unit cells. In the searching step, first, robots detect other robots using 16 sonar sensors by rotating 180° at intervals of 10°. Robots read data from all sonar sensors three times consecutively at each interval. These distance data are recorded and updated as an integer intensity value in the corresponding cell that represents the relative distance from the observing robot. Specifically, the Canny algorithm [32] eliminates a low intensity cell within the grid, which is then run through the Sobel algorithm [33]. These methods are applied to find the edge of a robot using the gradient of discrete information which appeared within the boundary of a robot. Finally, each robot executes the histogram equalization processing [33] that generates a histogram with a uniform intensity distribution to improve edge detection. By the equalization, the grid can overcome the distortion problem resulting from an unequal interval between sensors.

Next, in the checking step, robots compare the normalized grid with a 500 × 500 mm checking mask around the estimated center point, while turning 30°. As illustrated in Fig. 30, robots collect the cells with the maximum intensity value in the checking mask. Each robot finally puts the adjacent cell together and makes a virtual half circle on the grid from which they compute the center coordi-

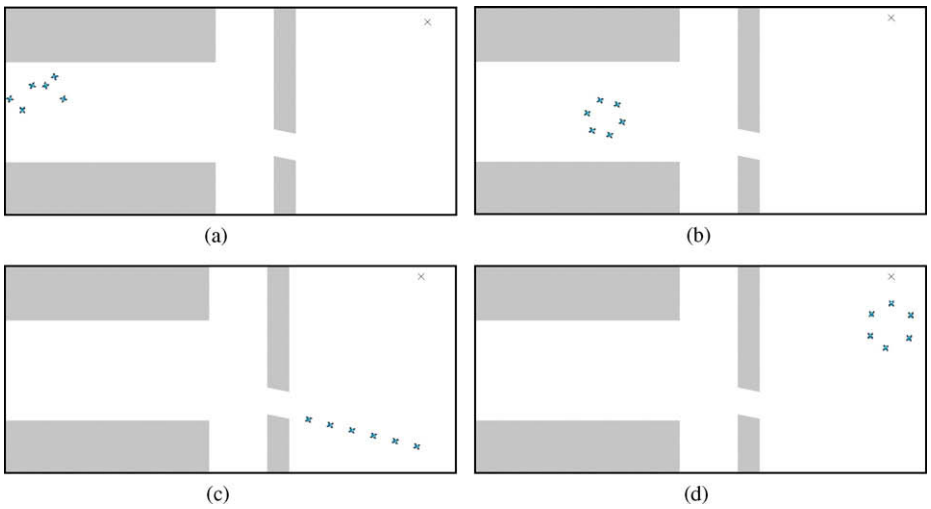

Fig. 29. Simulation results of hybrid formation control: (a) initial distribution, (b) flocking after generation, (c) switching into one row pattern, and (d) task completion.

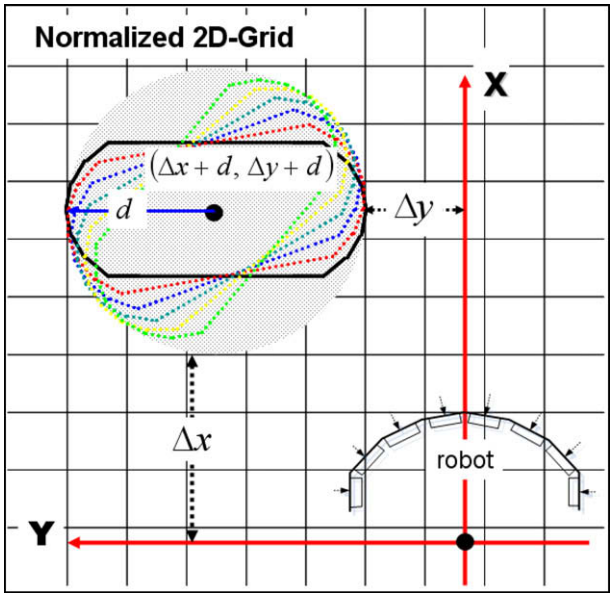

Fig. 30. Illustration of estimating center coordinates from edge trajectories.

nates of the other robots. Each robot finds the minimum distance of Δx and Δy to the half circle with respect to their local coordinate system. Then, the center coordinates are easily obtained by adding the distance of semi-major axis d of the elliptic robot edge to Δx and Δy, respectively. Using this estimation, each robot establishes a common coordinate system within an acceptable error range. Note that, however, this method requires robots to be initially positioned a minimum distance of 600 mm apart, with a clear line of sight. Practically, the time required for recognizing the positions of the robots with respect to each other is about 1 min. Note that the studies performed from the computational standpoint [14,15,18] assumed robots to be as points, or a circle equipped with unlimited sensors. In contrast, this observing algorithm can overcome the problem of the elliptic geometry of the robots with arbitrary heading directions.

Sonar sensors do not provide any information about detection point. From the non-uniform shape of the robots with only sonar sensors, it was difficult to estimate the center points of other robots. Because of these difficulties, the followers cannot estimate exactly the position of the leader in real-time, as it varies with time. It is therefore difficult to make the robot teams flock in exact patterns. We define an acceptable level of flocking accuracy. The ith follower must keep its relative position with respect to the leader using the distance $\mathrm{dist}(p_i, p_l)$ and the angle $\mathrm{ang}(\vec{x}_i, \vec{c}_i)$ (see Fig. 8). In real experiments, the distance was controlled within $0 \leqslant \mathrm{dist}(p_i, p_l) \leqslant 100$ mm and the angle was within $0 \leqslant \mathrm{ang}(\vec{x}_i, \vec{c}_i) \leqslant 10°$, respectively.

6.2. Experiments

In the first experiment, the robot team generates and adapts formation patterns from an arbitrary position and heading direction. Robots are aware of their target positions according to the formation pattern, but do not know who goes where. As shown in Fig. 31, the robot team generated six different formation patterns with the parameters of uniform interval 1000 mm, velocity 200 mm/s, and angular velocity 150°/s. Moreover, formations could be switched continuously from one pattern to another with the same leader. The leader remains stationary to help the followers generate a pattern, by sending messages for target patterns consecutively in the following order: arrow, diamond, two rows, fan-shape, arrow, one row, and one column. The team generates the arrow pattern twice, which demonstrates the reliability of pattern switching from any given formations.

In the second experiment, the robustness is verified against the accidental failure of robot members. While flocking in a fan-shape pattern, one robot stops, and immediately the remaining robots reform a similar triangle pattern to continue the mission. The replacement pattern is generated by reissuing IDs before the team navigates toward the target. Fig. 32 shows the snapshots of this formation recovery.

The third experiment demonstrates how the robot team flocks flexibly adapting to an environment. After forming a two row pattern, the robot team navigates toward a stationary target located 8 m away. On the way to the target, the team encounters an obstacle forcing them to switch into a diamond pattern that shifts the center point of the formation away from the obstacle. Then the team flocks to the target point while maintaining the diamond pattern. Fig. 33 shows the snapshots of this experiment. The leader decided an appropriate formation, acted as a stationary post for formation switching, and guided the team.

In conclusion, we demonstrated that the team based on the proposed observation method accomplished the assigned mission

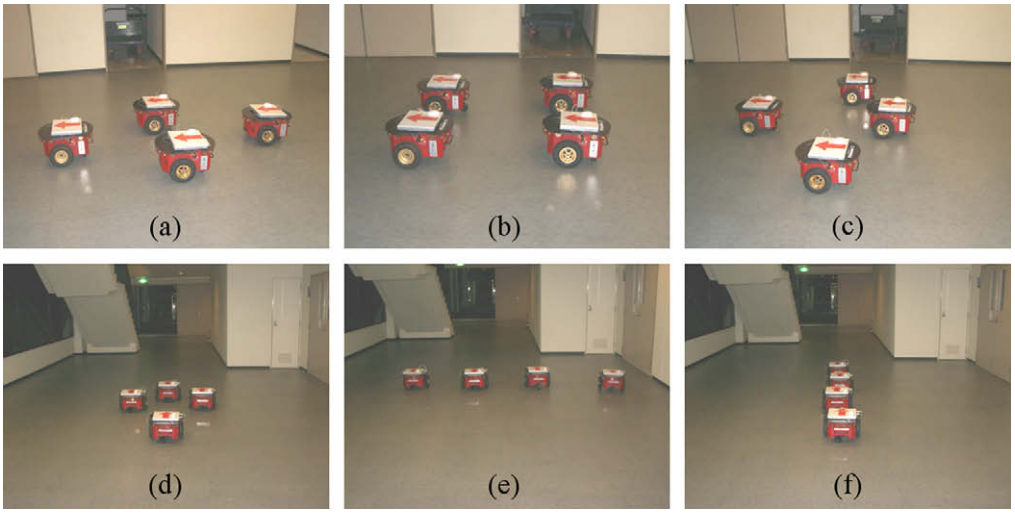

Fig. 31. Experimental results for six pattern generations using four Pioneer3-DX robots: (a) diamond, (b) two rows, (c) fan-shape, (d) arrow, (e) one row, and (f) one column.

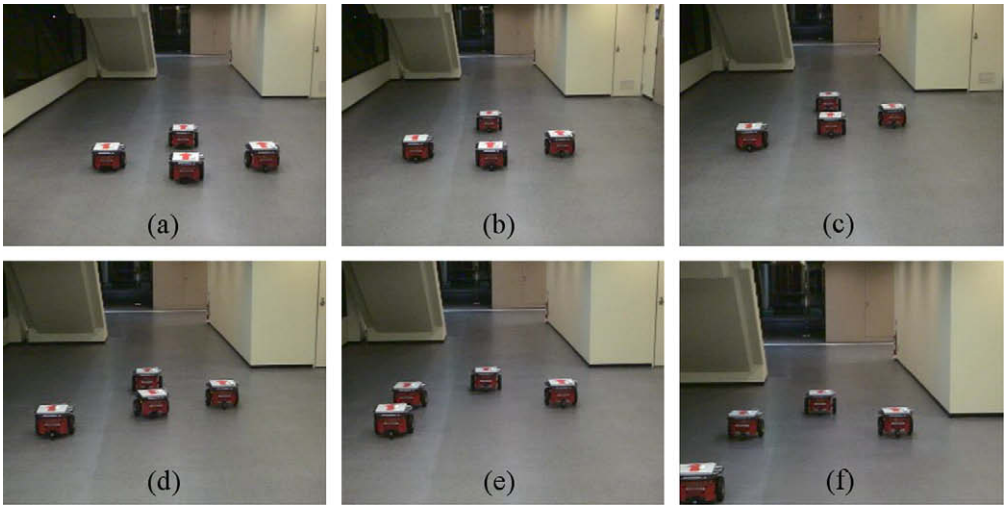

Fig. 32. Experimental results for robustness against loss in team population (similar pattern regeneration: (a) fan-shape generation by four robots, (b) flocking with the fan-shape pattern, (c) loss of a team member, (d) moving toward each target point before completion of regeneration, (e) regenerating a triangle pattern, and (f) flocking with triangle pattern by three robots).

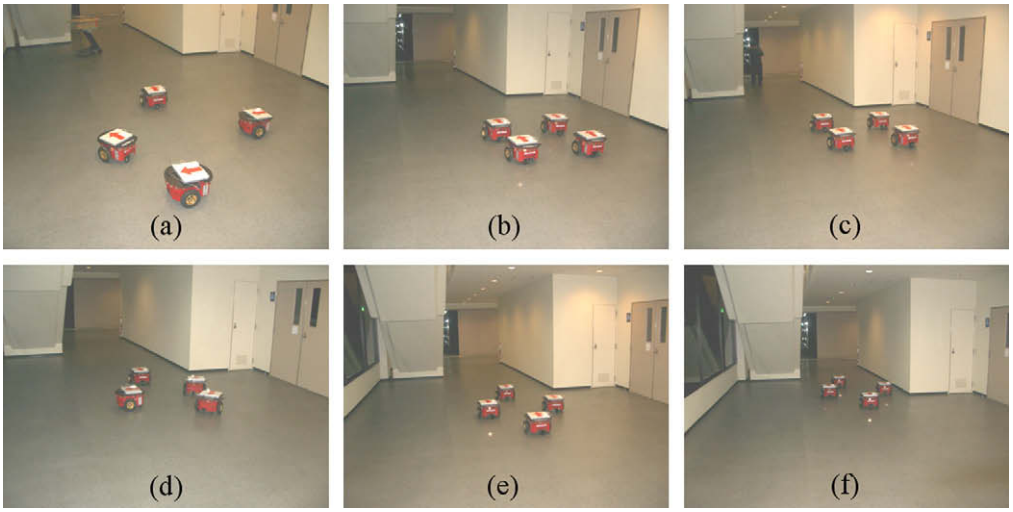

Fig. 33. Experimental results of formation control adapting to a corridor environment using four real mobile robots ((a) initial distribution, (b) generation of the two row pattern, (c) flocking with the two row pattern, (d) switching to the diamond pattern, (e) regeneration of the diamond pattern, and (f) flocking with the diamond pattern).

without high quality sensors and equipment. This allows us to organize a team with simple, economical units which we can easily deploy even in hazardous environments. As a first step toward real-world implementation, a self-organizing robot team would be applicable to *ad hoc* sensor network deployment [34].

7. Conclusion

This paper was devoted to developing a formation control framework for small-scale mobile robot teams that could adjust their formation to adapt to various situations. We proposed the self-organizing strategy, built on the following assumptions; anonymity, disagreement on common coordinate systems, no pre-selected leader, and minimal communication. Given arbitrarily distributed states of unknown robots, the proposed framework facilitated a self-organized movement of the team through five phases, including computation of common origin, leader selection, setting common direction, acquiring common coordinates, and issuing IDs. Based on these features, we decomposed the problem of formation control into three functions, pattern generation, flocking, and pattern switching. Specifically, we proposed two formation control approaches. The leader-referenced approach used the selected leader as the reference point for the position of the remaining followers. In contrast, the neighbor-referenced approach enabled each robot to maintain position with respect to their neighbor. We also proposed hybrid formation control, in which the advantages of each method could be applied to specific situations. These approaches were verified by extensive simulations. We demonstrated leader-referenced formation control using four physical robots equipped only with sonar sensors by applying image processing techniques.

Our formation control approaches for a self-organizing robot team offered robustness against individual failures and flexibility in adapting to changing environments. In addition, the movement of individual robots could converge toward their target position. Two fundamental contributions of this work are: (1) a wide variety of formations can be made in a decentralized way, adapting to an environment only by observing other robots that are anonymous and (2) the same or similar formations can be recovered in spite of a lack of some participating members resulting from individual failures. Implementation on real robots could be accomplished without high quality sensors and equipment. This allows us to organize a team with simple, economical units which we can easily deploy even in hazardous environments.

References

[1] Feddema JT, Lewis C, Schoenwald DA. Decentralized control of cooperative robotic vehicles: theory and application. IEEE Trans Robo Auto 2002;18(5):852–64.

[2] Yamashita A, Arai T, Ota J, Asama H. Motion planning of multiple mobile robots for cooperative manipulation and transportation. IEEE Trans Robo Auto 2003;19(2):223–37.

[3] Fukazawa Y, Chomchana T, Ota J, Yuasa H, Arai T, Asama H, et al. Realizing the exploration and rearrangement of multiple unknown objects by an actual mobile robot. Adv Robot 2005;19(1):1–20.

[4] Kosuge K, Oosumi T, Chiba K. Load sharing of decentralized-controlled multiple mobile robots handling a single object. In: Proceedings of IEEE International Conference on Robotics and Automation; 1997. p. 3373–8.

[5] Castillo-Effen M, Alvis W, Castillo C, Valavanis KP, Moreno WA. Modeling and visualization of multiple autonomous heterogeneous vehicles. In: Proceedings of the IEEE international conference on systems man and cybernetics; 2005. p. 2001–7.

[6] Long M, Gage A, Murphy R, Valavanis K. Application of the distributed field robot architecture to a simulated demining task. In: Proceedings of the IEEE International Conference on Robotics and Automation; 2005. p. 3193–200.

[7] Baber J, Kolodko J, Noel T, Parent M, Vlacic L. Cooperative autonomous driving. IEEE Robot Autom Mag 2005;12(March):44–9.

[8] Yi S-Y, Chong K-T. Impedance control for a vehicle platoon system. Mechatronics 2005;15(5):627–38.

[9] Bugard W, Moors M, Stachniss C, Schneider FE. Coordinated multi-robot exploration. IEEE Trans Robo Auto 2005;21(3):376–86.

[10] Hougen DF et al. Autonomous mobile robots and distributed exploratory missions. In: Parker LE, Bekey G, Barhen J, editors. Distributed autonomous robotics systems, vol. 4. Springer-Verlag; 2000. p. 221–30.

[11] Kim M, Chong NY. RFID-based mobile robot guidance to a stationary target. Mechatronics 2007;17(4–5):217–29.

[12] Egerstedt M, Hu X. Formation constrained multi-agent control. IEEE Trans Robo Auto 2001;17(6):947–51.

[13] Belta C, Kumar V. Trajectory design for formations of robots by kinetic energy shaping. In: Proceedings of the IEEE international conference on robotics and automation; 2002. p. 2593–8.

[14] Suzuki I, Yamashita M. Distributed anonymous mobile robots: formation of geometric patterns. SIAM J Comput 1999;28(4):1347–63.

[15] Defago X, Konagaya A. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In: Proceedings of the 2nd ACM international work principles of mobile computing; 2002. p. 97–104.

[16] Ikemoto Y, Hasegawa Y, Fukuda T, Matsuda K. Graduated spatial pattern formation of robot group. Inform Sci 2005;171(4):431–45.

[17] Fujibayashi K, Murata S, Sugawara K, Yamamura M. Self-organizing formation algorithm for active elements. In: Proceedings of the 21st IEEE symposium on reliable distributed systems; 2002. p. 416–21.

[18] Gervasi V, Prencipe G. Coordination without communication: the case of the flocking problem. Discrete Appl Math 2003;143(3):203–23.

[19] Prencipe G. CORDA: distributed coordination of a set of autonomous mobile robots. In: Proceedings of the 4th European research seminar on advances in distributed systems; 2001. p. 185–90.

[20] Balch T, Arkin RC. Behavior-based formation control for multi-robot teams. IEEE Trans Robo Auto 1998;14(6):926–39.

[21] Carpin S, Parker LE. Cooperative leader following in a distributed multi-robot system. In: Proceedings of the IEEE International Conference on Robotics and Automation; 2002. p. 2994–3001.

[22] Parker LE, Kannan B, Tang F, Bailey M. Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems; 2004. p. 1016–22.

[23] Lindhe M, Ogren P, Johansson KH. Flocking with obstacle avoidance: a new distributed coordination algorithm based on Voronoi partitions. In: Proceedings of the IEEE international conference on robotics and automation; 2005. p. 1785–90.

[24] Vidal R, Shakernia O, Sastry S. Following the flock. IEEE Robot Autom Mag 2004;11(December):14–20.

[25] Balch T, Hybinette M. Social potentials for scalable multi-robot formations. In: Proceedings of the IEEE international conference on robotics and automation; 2000. p. 73–80.

[26] Desai JP. A graph theoretic approach for modeling mobile robot team formations of robots. J Robot Syst 2002;19(11):511–25.

[27] Kurabayashi D, Okita K, Funato T. Obstacle avoidance of a mobile robot group using a nonlinear oscillator network. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems; 2006. p. 186–91.

[28] Fredslund J, Mataric MJ. A general algorithm for robot formations using local sensing and minimal communication. IEEE Trans Robo Auto 2002;18(5): 837–46.

[29] Lemay M, Michaud F, Letourneau D, Valin J-M. Autonomous initialization of robot formations. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems; 2004. p. 3018–23.

[30] Lee G, Chong NY. Decentralized formation control for a team of anonymous mobile robots. In: Proceedings of the 6th Asian control conference; 2006. p. 971–6.

[31] Lee G, Chong NY. Neighbor-referenced formation control for a team of mobile robots. In: Proceedings of the 4th international conference on ubiquitous robots and ambient intelligence; 2007. p. 163–8.

[32] Canny JF. A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 1986;8(6):679–98.

[33] Gonzalez RC, Woods RE. Digital image processing. second ed. Prentice Hall; 2002.

[34] Ghosh S, Basu K, Das SK. An architecture for next-generation radio access networks. IEEE Network 2005;19(5):35–42.