

Experimental investigation of distributed navigation and collision avoidance for a robotic swarm

Takumi Shibuya · Takahiro Endo · Fumitoshi Matsuno

Received: date / Accepted: date

Abstract In this study, we focus on the navigation of a robotic swarm and perform an experimental investigation. A decentralized control method using only local information has been proposed in many previous studies. However, because global coordinates are used and control input is calculated on a single PC in experiments, the system is regarded as partially centralized control. Thus, we first realize machines that are suitable for the distributed navigation control method. In conducting experiments using these machines, we encountered two problems. The first was loss of connectivity caused by the target's coordinates being momentarily out of the sensing range. The second was collisions between robots caused by the target being in the blind spot of the camera's field of view and loss of sight of the target. Note that each robot had a target and moved by following it. To solve these problems, we propose a method that removes unexpectedly output coordinates and adds rotational motion to each robot. The experimental results demonstrated that we achieved distributed navigation control with collision avoidance in an actual machine experiment.

Keywords Swarm robot · Multi-agent systems · Motion planning and navigation · Mobile robots

1 Introduction

A robotic swarm is a system in which multiple robots cooperate to perform tasks. A robotic swarm is useful because of its robustness, flexibility, and scalability [1, 2]. Thus, it is expected to be applied to tasks such as cooperative exploration [3] and cooperative transportation [4]. The most basic and essential function that results from performing these tasks is navigating the swarm to the destination while maintaining the swarm. Therefore, in this paper, we focus on the navigation control of the robotic swarm and investigate it experimentally.

Two types of control methods exist for a robotic swarm. One is a centralized control method in which a central computer manages the information of the entire swarm and calculates the control input for each robot. The other is a distributed control method in which each robot calculates its own control input according to locally available information. Centralized control uses the information of the entire swarm. It has the disadvantage that the computational load of the control PC increases as the number of robots increases, and in some cases, the system may fail. By contrast, distributed control can easily cope with changes to the number of robots and is robust to failures; thus, it is desirable to propose a control law for a robotic swarm in a distributed manner [5].

Furthermore, in a swarm robot system, as the number of robots increases, the possibility of collisions between robots increases. If a collision occurs, not only are the robots unable to perform the expected task but they may also fail. Therefore, it is essential to consider collision avoidance in the navigation of robot swarm systems.

Two studies have been conducted on the decentralized navigation control of a robotic swarm with collision avoidance [6, 7]. Both of them use the leader-follower scenario, in which a single leader navigates multiple followers. In [6], each follower calculated its control input using only local in-

Takumi Shibuya
Kyoto University, Kyoto, Japan
Tel./Fax.: +8175-383-3596
E-mail: s.takuumii@gmail.com

Takahiro Endo
Kyoto University, Kyoto, Japan
E-mail: endo@me.kyoto-u.ac.jp

Fumitoshi Matsuno
Kyoto University, Kyoto, Japan
E-mail: matsuno@me.kyoto-u.ac.jp

formation about its surroundings, and the swarm moved to the destination by maintaining an appropriate distance from its own target. Note that each robot had a target robot and followed it to maintain connectivity in the theory of Maeda et al. In [6], collision avoidance with its own target was considered, but collision avoidance with any other robot was not considered. Moreover, the distributed control has not been achieved in actual machine experiments. In particular, the global coordinates of all robots are acquired using external devices, such as motion capture. Then the local information around each robot is acquired by collecting this information on a central computer. The use of external devices in this manner makes the entire system very expensive. Additionally, the centralized control of some parts of the system makes it difficult to respond to changes to the number of robots, and this prevents the swarm robot from fully demonstrating its usefulness. This is a common problem in research on a robotic swarm, which can also be seen in [8, 9]. The study in [8] is about the surrounding target problem, and in [9], the researchers considered the formation control problem of a robotic swarm. Both of these methods are decentralized control in that the control input is calculated using only the local information of the robot, but they are not distributed control in that the control input of all the robots is calculated by one control PC in an actual experiment. To solve these problems, it is important to realize an actual robot that can achieve distributed control. In particular, in previous studies [7], researchers considered collision avoidance with all robots, but could not perform experiments with actual machines.

In this study, we focus on distributed navigation control with collision avoidance [7], and realize control using actual machines using a distributed control manner. Then, we propose two methods to solve some problems that arise from the characteristics of the actual machines. To summarize, the main contribution in [7] was the proposal of a control law that navigates the robotic swarm to its destination without collisions, whereas our main focus in this paper is to develop an actual machine suitable for investigating the controller in [7] experimentally, propose a method to solve the new problems that were identified in the experiments using these machines, and investigate the effectiveness of the proposed method in experiments.

2 Problem setting and control input

We introduce the problem setting and previously proposed control input [7].

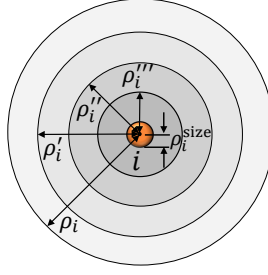


Fig. 1 Various distances of agent i .

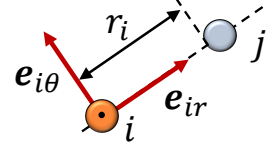


Fig. 2 Unit vector of agent i .

2.1 Robot model

We consider a robotic swarm that consists of multiple followers and a single leader. The robots move on a two dimensional plane with no obstacles. Robot i , $i = 1, 2, \dots, n$, is a follower, and robot $n + 1$ is the leader. $\mathcal{A} = \{1, 2, \dots, n + 1\}$ represents the set of all robots and $\mathcal{F} = \{1, 2, \dots, n\}$ represents the set of followers. Robot i is a circle of radius ρ_i^{size} , and we determine the center vector as $\mathbf{x}_i(t)$. The followers have a sensing range that is a circle of radius ρ_i . Follower $i \in \mathcal{F}$ can measure the relative position $\mathbf{x}_j^i(t) = \mathbf{x}_j^{\text{imin}}(t) - \mathbf{x}_i(t)$ of agent $j \in \mathcal{A}$ in the sensing range, where $\mathbf{x}_j^{\text{imin}}(t)$ is the closest point on agent j from follower i . At time t , the control input is expressed as $\mathbf{u}_i(t)$. The followers' equation of motion is given by

$$\dot{\mathbf{x}}_i(t) = \mathbf{u}_i(t). \quad (1)$$

Each follower has a target and moves by following the target. We define the positive constants ρ_i' , ρ_i'' , and ρ_i''' . Note that $\rho_i^{\text{size}} \leq \rho_i''' < \rho_i'' < \rho_i' < \rho_i$, as shown in Fig. 1, ρ_i' and ρ_i'' are the values for switching control input, and ρ_i''' is the distance that is set to prevent the target from getting closer.

2.2 Robot constraints

Follower i has the following physical constraints:

$$\|\mathbf{u}_i(t)\| \leq U_i, \quad (2)$$

$$\mathbf{u}_i(t) \text{ is continuous}, \quad (3)$$

where U_i denotes the upper limit of the velocity of follower $i \in \mathcal{F}$.

By contrast, for the swarm to maintain connectivity, the leader $n + 1$ is required to move under the following speed constraint, which is determined by the capabilities of the follower:

$$\|\mathbf{u}_{n+1}\| \leq U_{n+1} \leq \min_{i \in \mathcal{F}} U_i, \quad (4)$$

where U_{n+1} is the leader's maximum speed.

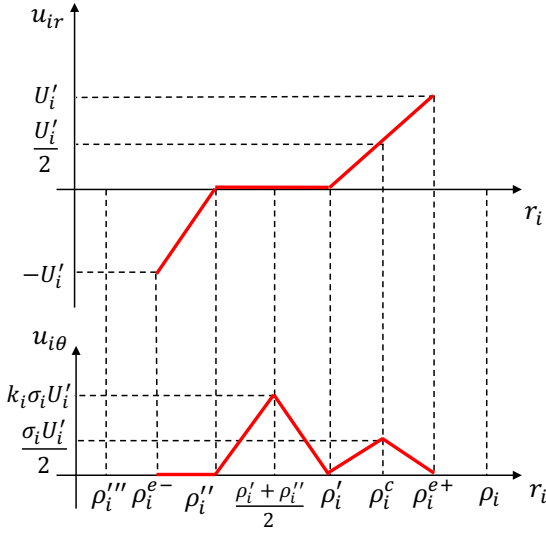


Fig. 3 Control input.

2.3 Control input

We briefly introduce previously proposed control input [7]. This control input realizes the navigation of the robotic swarm. In particular, a single leader, which moves under the constraint (4), can guide a robotic swarm whose followers have constraints ((2) and (3)) and do not collide with each other.

The control input of follower $i \in \mathcal{F}$ is determined by

$$\mathbf{u}_i(t) = u_{ir}(t)\mathbf{e}_{ir}(t) + u_{i\theta}(t)\mathbf{e}_{i\theta}(t), \quad (5)$$

where follower i has target j . Additionally, $r_i(t) := \|\mathbf{x}_j^i(t)\|$, $\mathbf{e}_{ir}(t) := \mathbf{x}_j^i(t)/r_i(t)$, and $\mathbf{e}_{i\theta}(t)$ is the unit vector obtained by rotating $\mathbf{e}_{ir}(t)$ by $\pi/2$ counterclockwise (Fig. 2). For robot k with the smallest distance to robot i other than the target, we define $d_i(t) := \|\mathbf{x}_k^i(t)\|$. Based on the relative positioning of the target, calculating control input (Fig. 3) enables the robotic swarm to move to the destination while maintaining connectivity. Moreover, when $d_i(t) \leq \rho''$, robot i changes its target to k and avoids collision. See the Appendix for the specific equations for the control input, and see [7] for more details. In particular, in [7], the researchers examined the navigation problem of the robotic swarm considering collision avoidance when the follower has the acceleration constraint in addition to constraints (2) and (3).

3 Machine configuration

We describe the configuration of the machines that perform the distributed navigation control discussed in Section 2.

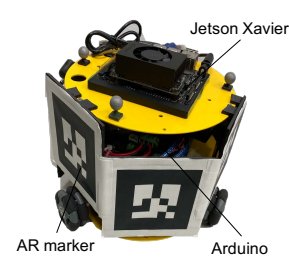


Fig. 4 Leader robot.

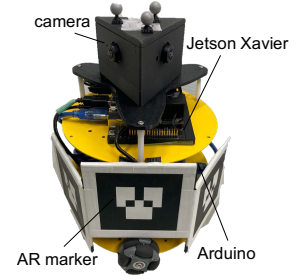


Fig. 5 Follower robot.

3.1 Leader robot

An overall view of the leader robot is shown in Fig. 4. We use a three-wheel drive (3WD) omni-wheel robot made by Nexus. This robot is equipped with Arduino as a microcontroller for motor control. Because the leader robot is tele-operated by an operator, we construct the leader robot system as follows: First, the Jetson Xavier NX, an embedded PC from NVIDIA, receives commands sent via Wi-Fi. Then, it sends the commands to Arduino to control the motors. Additionally, because the follower robots need to obtain the coordinates of the leader, we attach six AR markers around it. We use AR markers because they are lightweight and inexpensive, and their coordinates can be obtained easily.

3.2 Follower robots

An overall view of the follower robot is shown in Fig. 5. We use a 3WD omni-wheel robot. To achieve a 360-degree sensing range for the follower, we use two Sanwa Supply web cameras with a 150-degree horizontal viewing angle and one SignSmart CSI camera with a 160-degree viewing angle. As shown in Fig. 5, we place each camera on the side of an equilateral triangle. We place AR markers around all followers, as for the leader robot. Additionally, we equip each follower with NVIDIA's Jetson Xavier NX so that we can calculate the control input from the coordinates.

3.3 Process performed by the follower robots

We use ROS to perform an actual machine experiment using the previously proposed controller [7]. ROS is open-source software for robots, which has the advantage of facilitating communication and allowing advanced development using existing packages. In [7], each follower needed to obtain the coordinates of its surrounding robots. To achieve this, we use the ROS package `ar_track_alvar` to obtain the coordinates of the AR marker in the camera coordinate system. Thus, using a coordinate transformation, we obtain the coordinates of the surrounding robots in the robot coordinate

system. Using these coordinates, the robot calculates its control input on Jetson Xavier and sends the value to Arduino, thus creating a system in which a swarm moves to the destination while achieving distributed control.

4 Problems caused by the machine configuration and its improvements

In this section, we describe the problems caused by our machine configuration and improved methods for the actual machine experiments.

4.1 Navigation failure caused by loss of connectivity to the target

We had to place six identical AR markers around the robot to acquire the coordinates from any angle (Fig. 5). In [7], each robot followed a robot as its target, and it was necessary to record the ID of the target. Therefore, we used identical AR markers for each robot, and used the ID of the AR marker as the ID of each robot. This led to a scenario in which two AR markers were captured simultaneously by one camera, as shown in Fig. 6. At this time, our system randomly selected one of the markers and obtained its coordinates. Thus, a scenario arose in which the coordinates of the marker on the right-hand side of Fig. 6 were obtained in the current time step, and the coordinates of the marker on the left-hand side were obtained in the next time step. When this switch occurred, the distance to the target suddenly increased, and the target was out of the sensing range, which resulted in a loss of connectivity and a failure to achieve navigation.

4.2 Measurement of coordinates with position compensation

We describe the process that we used to solve the problem discussed in Section 4.1. Because the coordinates of each robot change continuously, when the change in distance in a one-time step is within a certain threshold ε , we use the acquired coordinates as they are, as in (6). When the change exceeds the threshold ε , we assume that the AR marker is switching, and we attempt to solve the problem by increasing the influence of the coordinates in the previous step, as in (7):

1. When $|r_i(t_{N+1}) - r_i(t_N)| \leq \varepsilon$,

$$\mathbf{x}_j^i(t_{N+1}) = \mathbf{x}_j^i(t_{N+1}). \quad (6)$$

2. When $|r_i(t_{N+1}) - r_i(t_N)| > \varepsilon$,

$$\mathbf{x}_j^i(t_{N+1}) = s\mathbf{x}_j^i(t_{N+1}) + (1-s)\mathbf{x}_j^i(t_N), \quad (7)$$

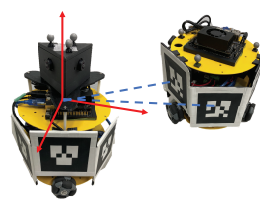


Fig. 6 Scenario of obtaining two AR markers.

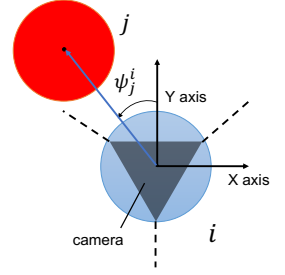


Fig. 7 Definition of ψ_j^i .

where t_N is the time at the N th time step, t_{N+1} is the time at the $N+1$ th time step for $n \in \mathbb{Z}^+$, s is a constant that satisfies $0 \leq s \leq 1$, and we set $s = 0.1$ in the experiments. We determined this value by trial and error through multiple experiments.

In [7], the researchers confirmed that each follower moves at speed below the velocity constraint of the leader; that is, if follower i moves closer to target j with $\|\mathbf{u}_i\| = U_{n+1}$ and target j moves closer to follower i with $\|\mathbf{u}_j\| = U_{n+1}$ or they move away from each other with similar speed, the change in distance in one time step is $|r_i(t_{N+1}) - r_i(t_N)| = 2U_{n+1}(t_{N+1} - t_N)$, and for other cases, the change is smaller than the above value. Based on this idea, we set the threshold values in (6) and (7) to $\varepsilon = 2U_{n+1}(t_{N+1} - t_N)$.

By adding the above process, we expect that each follower will stably follow the target.

4.3 Navigation failure caused by collision

To realize a circular sensing range, we placed a camera at the top of the robot, as shown in Fig. 5. Thus, when another robot was near the follower, the lower part of the AR marker could not be seen, and the coordinates could not be obtained. Additionally, because the field of view of the edge of the camera was characterized by a lower detection accuracy of the AR markers than that of the center, the problem of the loss of sight of the target and collision occurred when the target approached the follower from the edge of the camera's field of view.

4.4 Measurement of coordinates using rotary motion

Now, we describe the process that we used to solve the problem discussed in Section 4.3. We consider that the problem could be solved if we pointed the center of the camera at the target. Since in the control method [7], only the translation of the robot is considered, and the rotational motion is not used. We added rotational motion to the robot so that it can always follow the target with the same camera.

We define $\psi_j^i(t) \in [-\pi, \pi)$ as the angle to the center of the target $j \in \mathcal{A}$ relative to the center of follower i at time t

with respect to the Y -axis of the robot coordinate system (Fig. 7). In this case, the rotational control input at time t is $\omega_i(t)$, and the equation of motion for the rotation of the robot is defined as

$$\dot{\psi}_j^i(t) = \omega_i(t). \quad (8)$$

We define the time when the target is determined as t_i and the time when the target switch occurs as t_i^{start} . At $t = t_i$ or $t = t_i^{\text{start}}$, we divide the case according to which camera recognizes the target, and determine the desired angle ψ_i^{des} using (9)–(11): case 1 is the case in which the target is captured by the camera installed on the lower right-hand side of the equilateral triangle in Fig. 7, case 2 is the case in which the target is captured on the upper side of the equilateral triangle in Fig. 7, and case 3 is the case in which the target is captured on the lower left-hand side of Fig. 7:

case 1. When $-\pi \leq \psi_j^i < -\pi/3$,

$$\psi_i^{\text{des}} = -\frac{2\pi}{3}. \quad (9)$$

case 2. When $-\pi/3 \leq \psi_j^i < \pi/3$,

$$\psi_i^{\text{des}} = 0. \quad (10)$$

case 3. When $\pi/3 \leq \psi_j^i < \pi$,

$$\psi_i^{\text{des}} = \frac{2\pi}{3}. \quad (11)$$

Using the above angle ψ_i^{des} , we determine the rotational angular velocity input $\omega_i(t)$ of follower i at time $t > t_i$ as

$$\omega_i(t) = -k_p(\psi_i^{\text{des}} - \psi_j^i(t)), \quad (12)$$

where k_p is the feedback gain, and we set $k_p = 0.6$ in the experiments.

By adding the above inputs, even when a target approaches the vicinity of the robot, the robot can be captured by the center of the camera; thus, the target is not lost. Additionally, when a robot other than the target approaches from the edge of the camera's field of view, the camera's center points in the direction of the robot when the target is switched, thus enabling stable collision avoidance.

5 Experiments

In this section, we describe the experiments and demonstrate that it is possible to navigate four followers with one leader using previous control input [7]. We set each parameter of the followers as shown in Table 1.

5.1 Experiment 1: Navigation failure caused by loss of connectivity to the target

In experiment 1, the leader meandered forward with $\|\mathbf{u}_{n+1}\| = 0.225\text{m/s}$. We expected the leader's meandering to cause the

Table 1 Specifications of four followers in the experiments.

i	1	2	3	4
ρ_i (m)	1.30	1.30	1.30	1.30
ρ_i' (m)	1.00	1.00	1.00	1.00
ρ_i'' (m)	0.52	0.59	0.66	0.68
ρ_i''' (m)	0.29	0.32	0.34	0.36
ρ_i^{size} (m)	0.09	0.09	0.09	0.09
U_i (m/s)	0.25	0.25	0.25	0.25

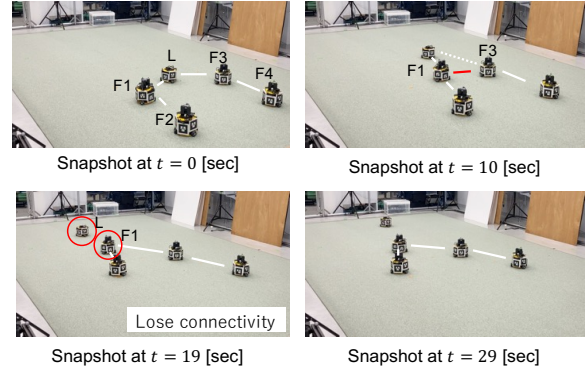


Fig. 8 Snapshots of experiment 1.

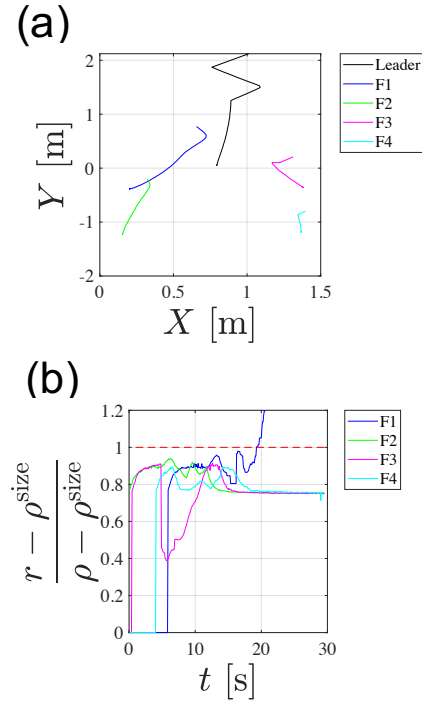


Fig. 9 Results of experiment 1: (a) path of each agent and (b) distances from the targets.

switching of the AR marker, as described in Section 4.1. We experimentally demonstrated that the problem described in Section 4.1 arose. Snapshots of experiment 1 are shown in Fig. 8, the path of each robot is shown in Fig. 9-(a), and the distance to the target of each robot is shown in Fig. 9-(b). In the legend, F1, ..., F4 denote the followers.

In Fig. 8, the white line indicates the connectivity with the own target, the red line at $t = 10$ s indicates connectivity with a new target when the target is switched, and the white dotted line indicates connectivity with the previous target. If the value of $\frac{r-\rho^{\text{size}}}{\rho-\rho^{\text{size}}}$ in Fig. 9-(b) is greater than 1, the target of each follower was out of the sensing range, whereas a value of 0 indicates that a collision occurred. From the snapshot of $t = 19$ s in Fig. 8 and $\frac{r-\rho^{\text{size}}}{\rho-\rho^{\text{size}}} > 1$ for follower 1 in Fig. 9-(b), connectivity between follower 1 and its target was lost, which resulted in navigation failure because the distance to the target could not be measured accurately, as described in Section 4.1.

5.2 Experiment 2: Navigation failure caused by collision

In experiment 2, the leader's movement involved a rapid change of direction. The velocity of the leader was $\|\mathbf{u}_{n+1}\| = 0.085\text{m/s}$. To cause target switching as described in Section 4.3, we chose the leader's movement as shown above. Snapshots of the experiment are shown in Fig. 10, the path of each robot is shown in Fig. 11-(a), and the distance to the target is shown in Fig. 11-(b). From the snapshot of $t = 34$ s in Fig. 10 and $\frac{r-\rho^{\text{size}}}{\rho-\rho^{\text{size}}} < \frac{r-\rho'''}{\rho-\rho^{\text{size}}}$ in Fig. 11-(b), navigation was not achieved because follower 2 lost sight of the target and a collision occurred.

5.3 Experiment 3: Effectiveness of the coordinates with position compensation

In experiment 3, the leader meandered forward with $\|\mathbf{u}_{n+1}\| = 0.225\text{m/s}$, as in experiment 1. In the control input for each follower, we used position compensation as described in Section 4.2. Snapshots of the experiment are shown in Fig. 12 and the experimental results are shown in Fig. 13. In Fig. 13, (a) shows the path of each agent, (b) shows the distance to the target of each robot, (c) shows the distance to the nearest robot other than the target of each robot, and (d) shows the velocity of the followers. The time to find the target for each follower was different; thus, we set the distance to the target for the follower that had no target to 0 in Fig. 13-(b). When the coordinates were not corrected, as shown in experiment 1, the AR markers were switched and the target was lost, and the followers lost connectivity. By contrast, the improved method in experiment 3 achieved navigation by maintaining connectivity between each robot, as shown in Figs. 12 and 13-(b). Additionally, Fig. 13-(c) shows that the followers did not collide with each other, as shown in Fig. 13-(d) shows that the velocity constraint was satisfied.

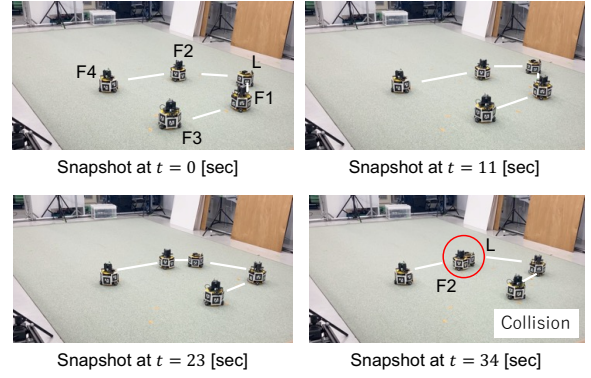


Fig. 10 Snapshots of experiment 2.

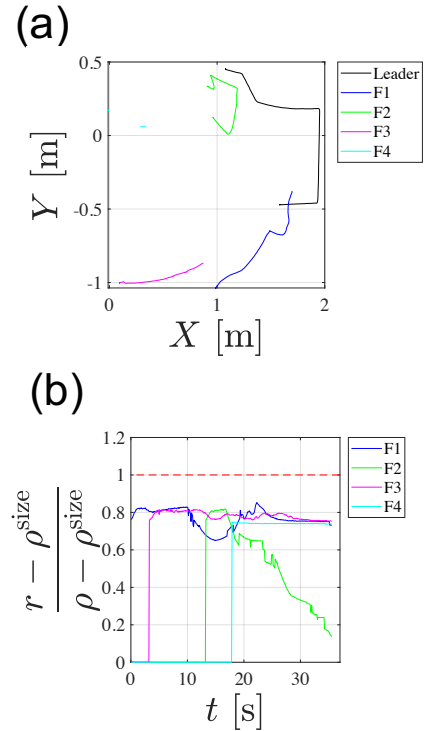


Fig. 11 Results of experiment 2: (a) path of each agent and (b) distances from the targets.

5.4 Experiment 4: Effectiveness of rotational motion

In experiment 4, the leader's movement involved a rapid change of direction, as in experiment 2. The velocity of leader was $\|\mathbf{u}_{n+1}\| = 0.085\text{m/s}$. We used the improved method discussed in Section 4.4. Fig. 14 shows snapshots of the experiment and Fig. 15 shows the experimental results. In Fig. 15, (a) shows the path of each agent, (b) shows the distance to the target of each robot, (c) shows the distance to the nearest robot other than the target of each robot, and (d) shows the velocity of the followers. In the case in which we added no rotational motion to the follower, a collision between robots occurred, as described in experiment 2. By contrast, from the snapshot of $t = 29$ s in Fig. 14 and Fig. 15-(b), the follower

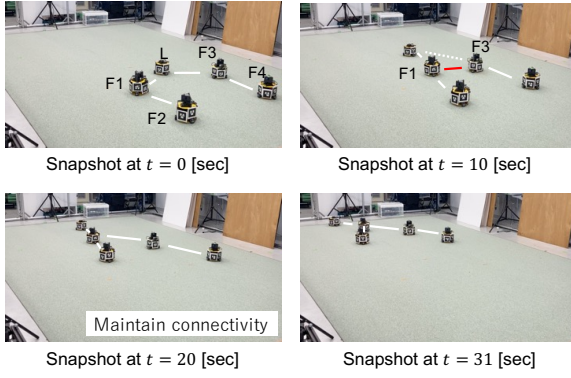


Fig. 12 Snapshots of experiment 3.

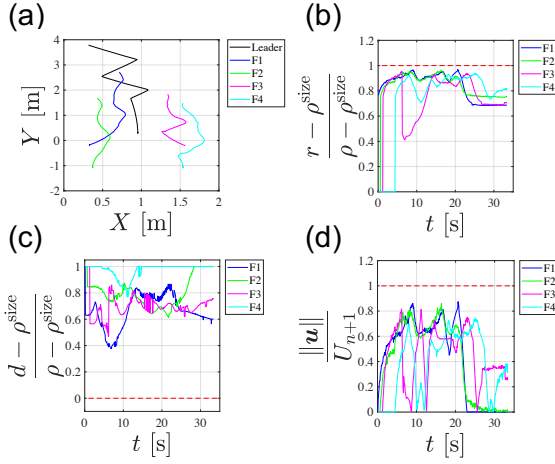


Fig. 13 Results of experiment 3: (a) path of each agent, (b) distances from the targets, (c) distances from other agents, and (d) velocity of the followers.

avoided collisions with the target by recognizing the target at the center of the camera with rotational motion, even when the target approached the follower. From the screenshot of $t = 32$ s in Fig. 14 and Fig. 15-(c), the followers also avoided collisions, even with objects other than their target. Furthermore, from Fig. 15-(d), the velocity constraint was satisfied.

6 Conclusion

In this study, we investigated navigation control experimentally, which is an essential function of a robotic swarm. To realize distributed navigation and collision avoidance control for a robotic swarm experimentally, we developed an actual machine that is suitable for distributed control theory. With the realization of these actual machines, we solved the problem that the amount of computation also increased as the number of robots increased. Additionally, we proposed methods to solve the new problems that we identified in the experiments using these machines. The experiments demonstrated the effectiveness of these methods. At present, we

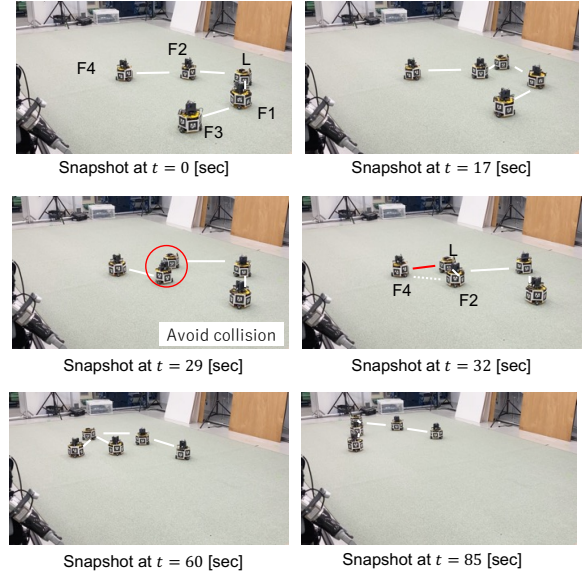


Fig. 14 Snapshots of experiment 4.

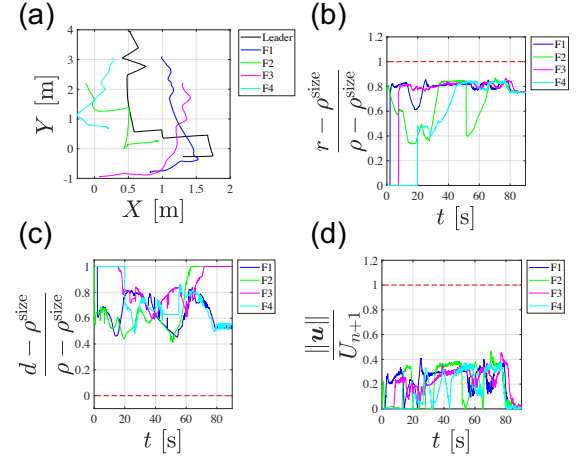


Fig. 15 Results of experiment 4: (a) path of each agent, (b) distances from the targets, (c) distances from other agents, and (d) velocity of the followers.

have only verified the effectiveness of the proposed method for our problem setting. However, the proposed method is effective for robots that have translational and rotational inputs, and whose motion is represented by a single integrator system. Future work will include applying the proposed method to other robots or environments and ensuring that this method works well in other robotic systems. We also consider that confirming that navigation is achieved in the same manner, even when the number of robots is increased, is important.

Appendix

We briefly describe the control input (5). In the control input, we determine $u_{ir}(t)$ and $u_{i\theta}(t)$ as follows. We divide the control input into three cases: (i) the control input before the target is determined, (ii) the

control input for maintaining connectivity during navigation, and (iii) the control input for collision avoidance.

- Case (i) $t < t_i$:

$$u_{ir}(t) = 0, \quad u_{i\theta}(t) = 0. \quad (13)$$

- Case (ii) $t \geq t_i \cap d_i(t) := \|\mathbf{x}_k^j(t)\| > \rho_i''$:

- (a) When $\rho_i^{e-}(t) \leq r_i(t) := \|\mathbf{x}_j^i(t)\| \leq \rho_i''$,

$$u_{ir}(t) = \frac{U_i(r_i(t) - \rho_i'')}{\rho_i'' - \rho_i''}, \quad u_{i\theta}(t) = 0. \quad (14)$$

- (b) When $\rho_i'' < r_i(t) \leq (\rho_i'' + \rho_i')/2$,

$$u_{ir}(t) = 0, \quad u_{i\theta}(t) = \frac{2\sigma_i(t)U_i'(t)}{\rho_i' - \rho_i''}(r_i(t) - \rho_i''). \quad (15)$$

- (c) When $(\rho_i'' + \rho_i')/2 < r_i(t) < \rho_i'$,

$$u_{ir}(t) = 0, \quad u_{i\theta}(t) = \frac{2\sigma_i(t)U_i'(t)}{\rho_i' - \rho_i''}(\rho_i' - r_i(t)). \quad (16)$$

- (d) When $\rho_i' \leq r_i(t) \leq \rho_i^c(t)$,

$$u_{ir}(t) = \frac{U_i(r_i(t) - \rho_i')}{\rho_i' - \rho_i'}, \quad u_{i\theta}(t) = \sigma_i(t)u_{ir}(t). \quad (17)$$

- (e) When $\rho_i^c(t) < r_i(t) \leq \rho_i^{e+}(t)$,

$$u_{ir}(t) = \frac{U_i(r_i(t) - \rho_i')}{\rho_i' - \rho_i'}, \quad u_{i\theta}(t) = \sigma_i(t)(U_i'(t) - u_{ir}(t)), \quad (18)$$

where

$$U_i'(t) := \max_{0 \leq \tau \leq t} \|u_{ir}(\tau)\|, \quad (19)$$

$$\begin{cases} \rho_i^{e-}(t) := \rho_i'' - \frac{U_i'(t)}{U_i}(\rho_i'' - \rho_i'''), & \rho_i^c(t) := \rho_i' + \frac{U_i'(t)}{2U_i}(\rho_i - \rho_i'), \\ \rho_i^{e+}(t) := \rho_i' + \frac{U_i'(t)}{U_i}(\rho_i - \rho_i') \end{cases} \quad (20)$$

and $\sigma_i(t) \in [-1, 1]$ is a parameter used to change the shape of the swarm. Note that, when $r_i(t) \leq \rho_i''$, $r_i(t) \geq \rho_i^{e-}$ always holds from (14) and (19). Furthermore, when $r_i(t) \geq \rho_i'$, $r_i(t) \leq \rho_i^{e+}$ always holds from (17)–(19). Thus, it is sufficient to design the control input in the range $\rho_i^{e-}(t) < r_i(t) < \rho_i^{e+}(t)$.

- Case (iii) $t \geq t_i \cap d_i(t) \leq \rho_i''$:

$$\mathbf{u}_i(t) = \mathbf{u}_i^j + \frac{t - t_i^{\text{start}}}{T_i}(\mathbf{u}_i^k(t) - \mathbf{u}_i^j), \quad (21)$$

where $T_i = h_i/(\zeta U_i)$, $h_i = \min\{\rho_i - \rho_i', \rho_i'' - \rho_i'''\}$, ζ is a positive constant that satisfies $2\pi(1 - e^{-1/\zeta})/\zeta \leq 1$ (we used $\zeta = 2.253$ in this paper). Furthermore, let t_i^{start} be the time at which $d_i(t) = \rho_i''$, and agent k be the new target at $t = t_i^{\text{start}}$. Note that if another agent comes into the area closer than ρ_i'' , it becomes a new target. Additionally, $\mathbf{u}_i^j := \mathbf{u}_i(t_i^{\text{start}})$. From the implementation point of view, it is difficult to switch targets instantly. Thus, the input is changed from the previous target reference to the new target reference over T_i .

We designed the control input $u_{ir}(t)$ to guarantee that followers maintain LSC. By contrast, we designed $u_{i\theta}(t)$ to widen the swarm shape and prevent it from becoming a chain structure. As a result, the follower behaves as follows in each case: In Case (II)-(a), the follower moves away from the target to avoid colliding with it. In Case (II)-(b) and (c), the follower moves away from the other agents to avoid approaching them. In Case (II)-(d) and (e), the follower moves away from the other agents while approaching the target to maintain connectivity. An outline of the control input in case (ii) is shown in Fig. 3.

For more details, see [7].

References

1. Bayindir L (2016) A review of swarm robotics tasks, *Neurocomputing* 172: 292-321
2. Taheri H, Zhao, CX (2020) Omnidirectional mobile robots, mechanisms and navigation approaches, *Mech. Mach. Theory* 153: Article 103958
3. Suarez P, Iglesias A, Galvez A (2019) Make robots be bats: specializing robotic swarms to the Bat algorithm, *Swarm Evol. Comput.* 44: 113-129
4. Tuci E, Alkilabi MH and Akanyeti O (2018) Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art, *Front. Robot. AI* 5: 59
5. Ota J (2006) Multi-agent robot systems as distributed autonomous systems, *Adv. Eng. Inform.*, 20(1): 59-70
6. Maeda R, Endo T, Matsuno F (2017) Decentralized navigation for heterogeneous swarm robots with limited field of view, *IEEE Robot. Autom. Lett.*, 18: 904-911
7. Y. Sano, T. Endo, T. Shibuya, and F. Matsuno, “Decentralized Navigation and Collision Avoidance for Robotic Swarm with Heterogeneous Abilities,” *Advanced Robotics*, accepted.
8. Ovchinnikov K, Semakova A, Matveev A (2015) Cooperative surveillance of unknown environmental boundaries by multiple nonholonomic robots, *Rob. Auton. Syst.*, 72: 164-180
9. Zhou Y, Zhou D, Wang Z, Schwager M (2018) Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures, *IEEE Trans. Robot.*, 34: 916-923

Acknowledgements This research is supported, in part, by the Japan Science and Technology Agency, the Strategic International Collaborative Research Program (JST SICORP), e-ASIA Joint Research Program (Grant Number JPMJSC18E4), Project No. 18065977.