Formation Control of Multi-Agent System Based on Higher-Order Partial Differential Equations

Kaiyo Yamaguchi, Takahiro Endo *Member, IEEE*, and Fumitoshi Matsuno, *Senior Member, IEEE*

***Abstract*— We propose a method of controlling the formation in a multi-agent system using partial differential equations (PDEs). In this method, the behavior of the entire multi-agent system is modeled by second- or higher-order PDEs. We also propose a boundary controller that exponentially stabilizes the PDE model. By discretizing the PDE model under the proposed controller, the follower agents' control laws can be derived. Moreover, the boundary controller corresponds to the leader agents' control laws. The use of higher-order PDEs leads to the generation of various formations that cannot be generated by using lower-order PDEs. Finally, we conduct numerical simulations and experiments to validate the proposed method.**

***Index Terms*— stability, swarm, multiple robots, formation control, decentralized, partial differential equation, boundary control.**

I. Introduction

In recent years, research on a robotic swarm, which is a group of multiple robots, has been actively conducted. A swarm has the robustness necessary to achieve a goal even if the environment or the situation changes, the flexibility to respond to changes in the environment or the situation, and sufficient scalability to use a large number of individuals. In a robotic swarm, decentralized formation control is attracting attention as a swarm application and is expected to be applied to autonomous driving technology, mobile robot arrangement, UAV formation flight, and others [1].

Most studies on the formation control of multi-agent systems have described the behavior of an entire system in terms of ordinary differential equations (ODE), which can be roughly divided into three control methods [1]. The first is the position-based method, in which the desired formation is achieved by all agents having their own position information on a global coordinate system and controlling their positions to converge to the desired position [2]–[4]. The second method is the displacement-based method, in which each agent knows the relative position vectors with its neighbor agents and the desired formation is achieved by converging them to the desired values [5]–[11]. And the third is the distance-based method, in which the desired formation is achieved by each agent having information on the relative distance to its neighboring agents and converging them to the desired values [12]–[15]. However, in any of these methods, the control laws of each agent include information on the desired formation, such as the desired position, the desired relative position, and the desired relative distance of each agent, that differs for each agent. Thus, to apply the above methods to a huge number of agents, we need to calculate the desired formation value for each of the huge number of agents, and it requires a lot of work to implement the desired value in the controller of each of the huge number of agents becomes difficult. Furthermore, in the method using ODE, the gains of the model need to be tuned again when the number of agents is changed.

To solve these problems, formation control methods based on partial differential equations have been proposed. These methods use partial differential equations (PDEs) to represent the behavior of an entire robot group. A control method for the PDE model is also proposed. Then, the controller for each agent is derived by discretizing the PDE model. In these methods, we can place all agents in the desired formation merely by setting a few parameters given uniformly to all follower agents included in the PDEs and a few parameters given to the leader agent. The advantage of this method is that it is not necessary to retune the gain of the model itself, even when the number of agents increases. This is because a model of the multi-agent system is described by the partial differential equation independent of the number of agents.

First, G. Ferrari et al. showed that the consensus control law often used in conventional formation control was consistent with the heat equation, which is a partial differential equation, and they suggested that the heat equation could serve as a model of the multi-agent system [16]. P. Frihauf et al. proposed a method that enables multi-agent formation control by using the boundary controller of a PDE model based on the heat equation [17]. Since then, much research has been conducted on the formation control of multiple robots using the boundary control of partial differential equations. T. Meurer et al. used Burgers equations, which are nonlinear second-order partial differential equations, and proposed a method to achieve the desired formation that conventional methods could not [18]. J. Qi et al. proposed a method that can control the formation of a multi-agent system in a three-dimensional environment by using two polar coordinate heat equations [19], [20]. J. Qi et al. proposed a method that can set a time-varying desired shape by using a wave equation [21]. In addition, there have been many studies on the formation control of a multi-agent system by using boundary control of second-order

This work was supported in part by JST SICORP Grant Number JPMJSC18E4, Japan.

K. Yamaguchi, T. Endo, and F. Matsuno are with the Department of Mechanical Engineering and Science, Kyoto University, Kyoto daigaku-katsura, Nishikyo-ku, Kyoto, Japan (e-mail: kaiyo19940526@gmail.com, endo@me.kyoto-u.ac.jp, matsuno@me.kyoto-u.ac.jp)

PDEs [22]–[24]. However, the methods using boundary control proposed in these studies have second-order partial differential equations, and the number of PDE parameters that can be set is limited. Therefore, there are problems such as limitations on the desired formation that can be achieved. Furthermore, when achieving the desired shape at an arbitrary position, some PDE parameters must be set to 0, and the desired shape that can be achieved is further limited.

To address this problem, G. Freudenthaler et al. proposed a method that can move the desired shape to an arbitrary position by a method that uses an exogenous system and can achieve abundant desired shapes [25]. J. Qi et al. proposed a method that can control the trajectory by using a distributed control that gives the desired shape in advance to a closed-loop system of partial differential equations [26]. Here, note that there are two kinds of controllers for PDE: boundary controllers and distributed controllers. The control action of a boundary controller is at the boundary of the PDE system, and that of a distributed controller is distributed in the domain of the PDE system. In addition, Wei et al. proposed a distributed control method for a model based on the heat equation and proposed a method that can achieve all functions that are second-order differentiable as the desired shapes [27]. Although these methods can achieve many shapes, the controller used is a distributed controller. Thus, due to the nature of the control method, the information on the desired shape is included in the control law of each follower agent. Therefore, as the number of agents increases, it becomes difficult to design control laws for each agent, and communication with all agents is required to change the size and position of the target shape.

From these points of view, in this paper we propose a formation control method of a multi-agent system by using the boundary control method of higher-order PDEs. The proposed method has the following advantages.

(i) As in the previous studies on boundary control methods of multi-agent systems, follower agents do not need individual desired values in the implementation of the controller.

(ii) Since the order of a PDE model can be designed freely, it is possible to achieve formations that cannot be achieved with other PDE-based formation control methods using boundary control.

Our proposed method utilizes a more conceptually difficult, higher-order PDE model to synthesize controllers that can generate agent deployment to a richer class of planar curves. The added complexity from designing controllers for higher-order PDEs allows the user to assign multi-agent deployment to a much richer class of planar curves than what is achievable in the existing literature, where only second-order PDEs are considered. The additional planar curves that our proposed algorithms can achieve require additional equilibria for the PDE, which necessitates the higher-order PDEs. Further, in the previous studies, the methods were validated only by numerical simulation as far as we know, whereas in this paper experiments confirmed that the method is useful for an actual robot.

This paper is organized as follows. Section II presents the problem settings. Section III proposes a boundary controller and describes the stability analysis. Section IV and V provide the numerical and experimental results to validate the proposed method, respectively. Finally, section VI concludes the paper.

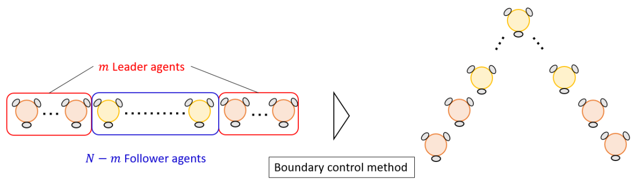


Fig. 1: Placement of N agents

II. Problem Statement

A. Problem setting

This paper discusses the placement of N agents on the desired shape as shown in Fig. 1. Here, note that Fig. 1 is an example, and it is not the goal of this paper to achieve the formation as shown in the right figure of Fig. 1. In this method, we consider a multi-agent system consisting of m leader agents corresponding to the boundary conditions of the mth-order PDE and $N-m$ follower agents that determine their own behavior based on the PDEs. The purpose of this method is to place N agents on the desired formation by designing a model of the PDEs and the boundary input that converges the agents to the desired formation.

We give an overview of formation control using a PDE-based approach. For example, let us consider a situation in which there are multiple follower agents, and they are arranged in a chain, as shown in Fig. 1. We assume that each agent is connected so that information can be exchanged with the neighbor agents. Then, let us consider the case where the behavior of entire agents, that is, the swarm, is given by the following one-dimensional heat equation:

$$\dot{x}(\alpha,t) = ax''(\alpha,t), \qquad \alpha \in (0,1), \quad t \geq 0, \tag{1}$$

$$x(0,t) = u_0(t), \quad x(1,t) = u_a(t), \tag{2}$$

where $x(\alpha,t)$ is the temperature at time t at position α, and a is a physical parameter. Here, a dot denotes the time derivative, and a prime means the spatial derivative. Further, $u_0(t)$ and $u_a(t)$ are the boundary inputs. $\alpha \in [0,1]$ is corresponding to the agent index number. Now, we assume that the control inputs $u_0(t)$ and $u_a(t)$ converge $x(\alpha,t)$ to a certain desired formation $\bar{x}(\alpha)$. We spatially discretize (1) and let $x_i(t) = x((i-1)h,t)$, $i = 1,\cdots,N$. Here, $h = 1/(N-1)$ is a spatial difference, and $x_i(t)$ is corresponding to the position of the i-th agent. Then, we obtain the following equation:

$$\dot{x}_i(t) = \sum_{j\in N_i} \{x_i(t) - x_j(t)\}, \tag{3}$$

where N_i is the set of neighbors of agent i. Here note that, in the derivation of (3), we performed the change of variables $t = h^2\tau$ and replace τ with t. Equation (3) corresponds to agent $i = 1,\cdots,N-1$ is controlled by the well-known consensus-based control. In addition, the leaders at the boundaries are position-controlled as $x_1(t) = u_0(t)$ and $x_N(t) = u_a(t)$. Therefore, by finding the control inputs $u_0(t)$ and $u_a(t)$ that converge $x(\alpha,t)$ to $\bar{x}(\alpha)$, we can converge the swarm to the desired formation. In particular, the positions of the leaders are controlled by $x_1(t) = u_0(t)$ and $x_N(t) = u_a(t)$, and the followers other than the leaders are controlled by the consensus-based control. This procedure is formation control using a PDE-based approach.

This paper focuses on boundary control and boundary sensing as the leader's controller. Here, note that the boundary control/sensing means that the control/measurement action is at the boundary of the system. This is because only the leader's controller requires information on the desired formation, and the leader's controller should be constructed using the information of the neighbors around the leader. As explained in the previous paragraph, the controllers of agents other than the leaders are determined from PDE dynamics, which does not include the information on the desired formation. Taking the one-dimensional heat equation (1) and (2) as an example, let us consider the controller that converges $x(\alpha, t)$ to $\bar{x}(\alpha)$. If the controllers consists only of the values obtained by boundary sensing, we can design the controllers as $u_0(t) = -k_{f0}\{x'(0,t) - \bar{x}'(0)\} + \bar{u}_0$, $u_a(t) = -k_{fa}\{x'(1,t) - \bar{x}'(1)\} + \bar{u}_1$, for example. Here, k_{f0} and k_{fa} are some feedback gains, $\bar{u}_0$ and $\bar{u}_1$ are constant determined from the desired formation. Further, the controllers are derived by (21) and (22). Using the spatial difference, the leaders' controllers are as follows: $u_0(t) = -k_{f0}/h\{x_2(t) - x_1(t) - \bar{x}'(0)\} + \bar{u}_0$, and $u_a(t) = -k_{fa}/h\{x_N(t) - x_{N-1}(t) - \bar{x}'(1)\} + \bar{u}_1$. Thus, by focusing on boundary control and boundary sensing, we can be seen that only the leaders' controllers have the information of the desired formation, and the leaders' controllers become the decentralized control. Here, note that the decentralized control for multi-agent systems is the control method, in which the agents act according to the locally available information. In contrast, a centralized control for multi-agent systems is the control method, in which the information of the whole agents is managed by a central computer.

On the other hand, if we use the information obtained by internal sensing in the construction of the controllers, we can design the controllers as follows: $u_0(t) = -k_{f0}\int_0^1\{x(\alpha,t) - \bar{x}(\alpha)\}\mathrm{d}\alpha + \bar{u}_0$ and $u_a(t) = -k_{fa}\int_0^1\{x(\alpha,t) - \bar{x}(\alpha)\}\mathrm{d}\alpha + \bar{u}_1$. Here, note that these controllers are just an example, and the stability of (1) by these controllers has not been examined. Further, internal sensing means that the spatially continuous $x(\alpha, t)$ value can be measured. Using the spatial difference, the leaders' controllers become as follows: $u_0(t) = -k_{f0}\sum_{j=0}^{N}\{x_j(t) - \bar{x}_j\} + \bar{u}_0$, and $u_a(t) = -k_{fa}\sum_{j=0}^{N}\{x_j(t) - \bar{x}_j\} + \bar{u}_1$, where $\bar{x}_j = \bar{x}((i-1)h)$. Thus, the leaders' controllers require all agents' information, and the leaders' controllers are no longer decentralized controllers. For this reason, boundary control and boundary sensing are adopted in this paper.

B. Proposed PDE model

We specify the behaviors of the agents by the following PDEs:

$$\dot{x}(\alpha, t) = \sum_{i=0}^{m} a_i x^{(i)}(\alpha, t), \tag{4}$$

$$\dot{y}(\alpha, t) = \sum_{i=0}^{m} b_i y^{(i)}(\alpha, t), \tag{5}$$

where $x(\alpha, t)$ and $y(\alpha, t)$ are real numbers representing the x-axis and y-axis coordinates of the agents, respectively, and they are the state variables of the PDE. a_i and b_i are constant parameters. In addition, α is a spatial variable that characterizes each agent satisfying $0 \le \alpha \le 1$, and t represents time. $x^{(i)}(\alpha, t)$ represents the i-th order derivative of the spatial variable α of $x(\alpha, t)$. Since the discussions of (4) and (5) are equivalent, we consider only the model for the x-axis (4) here. Here, note that (4) and (5) are decoupled. That is, the x-axis and y-axis coordinates of the agents are decoupled. In general, there are many dynamics in which the x-axis and y-axis coordinates are coupled. However, as we mentioned in the experiment, we envision the use of mobile robots with decoupled x-axis and y-axis, such as omnidirectional robots. Our method can be applied to an object whose x-axis and y-axis are decoupled, such as an omnidirectional robot. The design of the controller for decoupled PDEs is future work. In addition, PDEs (4) and (5) are set so that the behavior of the swarm follows these PDEs (4) and (5). In particular, these PDEs do not follow a kinematic model and are not derived from Newton's Second Law.

We set the boundary conditions of this model as follows:

$$x(0, t) = u_0(t), \tag{6}$$

$$x^{(j)}(0, t) = u_{0j}, \qquad \text{for } j = 1, ..., [m/2] - 1, \tag{7}$$

$$x(1, t) = u_a(t), \tag{8}$$

$$x^{(l)}(1, t) = u_{al}, \qquad \text{for } l = 1, ..., [(m-1)/2], \tag{9}$$

where $u_0(t)$ and $u_a(t)$ are control inputs for stabilizing the PDE model (4), and u_{0j} and u_{al} are constants set by the designer according to the desired formation. $[m/2]$ and $[(m-1)/2]$ represent the integer parts of $m/2$ and $(m-1)/2$, respectively. Here, the boundary conditions (7) and (9) are not required when $m = 2$, and the boundary condition (7) is not required when $m = 3$, so we do not define them in that case.

Next, we derive the equilibrium curve of this PDE model. In (4), $\dot{x}(\alpha, t)$ equals 0 at the equilibrium point, so that when the system converges to the equilibrium point $(x(\alpha, t) \to \bar{x}(\alpha))$, $\bar{x}(\alpha)$ is a solution that satisfies the following ODE:

$$\sum_{i=0}^{m} a_i \bar{x}^{(i)}(\alpha) = 0, \tag{10}$$

$$\bar{x}(0) = \bar{u}_0, \quad \bar{x}^{(j)}(0) = u_{0j}, \tag{11}$$

$$\bar{x}(1) = \bar{u}_a, \quad \bar{x}^{(l)}(1) = u_{al}, \tag{12}$$

where $\bar{u}_0$ and $\bar{u}_a$ are constants uniquely obtained from the desired formation. Therefore, we propose boundary controllers to converge $x(\alpha, t)$ to $\bar{x}(\alpha)$. Then, the agents can be placed on the desired shape by setting $\bar{x}(\alpha)$ to the desired formation. For example, to deploy agents in a circle, the order m and $\bar{u}_0$, $\bar{u}_a$, u_{0j} are set so that $\bar{x}(\alpha) = \sin(2\pi\alpha)$ and $\bar{y}(\alpha) = \cos(2\pi\alpha)$.

Now, let us introduce the new variable $\tilde{x}(\alpha, t) = x(\alpha, t) - \bar{x}(\alpha)$. Then, the error system of this PDE model (4) can be expressed as follows:

$$\dot{\tilde{x}}(\alpha, t) = \sum_{i=0}^{m} a_i \tilde{x}^{(i)}(\alpha, t), \tag{13}$$

$$\tilde{x}(0, t) = \Delta u_0(t), \tag{14}$$

$$\tilde{x}^{(j)}(0, t) = 0, \qquad \text{for } j = 1...[m/2] - 1, \tag{15}$$

$$\tilde{x}(1, t) = \Delta u_a(t), \tag{16}$$

$$\tilde{x}^{(l)}(1, t) = 0, \qquad \text{for } l = 1...[(m-1)/2], \tag{17}$$

where $\Delta u_0(t) = u_0(t) - \bar{u}_0$, and $\Delta u_a(t) = u_a(t) - \bar{u}_a$.

III. Controller Design and Stability Analysis

In this section, we design the boundary controllers that exponentially stabilize the PDE system (4).

A. Controller design

The control objective is to propose a boundary controller satisfying

$$x(\alpha, t) \to \bar{x}(\alpha). \tag{18}$$

This means that the agent is placed on the desired shape, which corresponds to $\tilde{x}(\alpha, t) \to 0$ in the error system (13). To achieve this control objective, we prove the exponential stability of the closed-loop system using the Lyapunov method described later. At that time, the designed control inputs when $m \geq 3$ are as follows:

$$\Delta u_0(t) = k_0 \Bigg\{ L \Big(\sum_{j=3}^{m} a_j \tilde{x}^{(j-2)}(0, t) \Big) - K \Big(\sum_{j=2}^{m} a_j \tilde{x}^{(j-1)}(0, t) \Big) \Bigg\}, \tag{19}$$

$$\Delta u_a(t) = k_a \Bigg\{ (K + L) \Big(\sum_{j=2}^{m} a_j \tilde{x}^{(j-1)}(1, t) \Big) - L \Big(\sum_{j=3}^{m} a_j \tilde{x}^{(j-2)}(1, t) \Big) \Bigg\}, \tag{20}$$

where k_0 and k_a are feedback gains and are non-zero real numbers. L and K are positive constants derived from the Lyapunov function. In the case of $m = 2$, since the first term on the right-hand side in (19) and the second term on the right-hand side in (20) cannot be defined, the following control input is designed with these terms eliminated:

$$\Delta u_0(t) = -k_0 K a_2 \tilde{x}^{(1)}(0, t), \tag{21}$$

$$\Delta u_a(t) = k_a (K + L) a_2 \tilde{x}^{(1)}(1, t). \tag{22}$$

B. Stability analysis

Now we prove the exponential stability in Theorem 1. However, the proof of Theorem 1 is a proof for the case $m \geq 3$. Even when $m = 2$, only some exponential stability conditions are changed, and the outline of the proof does not change. For the proof when $m = 2$, see Appendix II.

Theorem 1: The closed-loop system (13) is exponentially stable if the following conditions are satisfied.

- When m is an even number,

$$(-1)^{m/2} a_m \leq 0, \tag{23}$$

$$\phi_s := Ls(-1)^{s+1} a_{2s+1} - L \frac{(-1)^s a_{2s+1}}{2} + |a_{2s}| L + K(-1)^s a_{2s} \leq 0, \qquad \text{for } s = 1, \ldots, \frac{m}{2}, \tag{24}$$

$$\psi := a_0 K - a_1 \frac{L}{2} + \sum_{s=1}^{\frac{m}{2}-1} \frac{\phi_s}{2^s} + K \frac{(-1)^{\frac{m}{2}} a_m}{2^{\frac{m}{2}}} < 0, \tag{25}$$

$$\frac{(1-p)\psi + a_0 L}{L} < 0, \tag{26}$$

$$k_0 + \Bigg(\sum_{s=1}^{\frac{m}{2}-1} \frac{\phi_s}{2^{s-1}} + K \frac{(-1)^{\frac{m}{2}} a_m}{2^{\frac{m}{2}-1}} + \frac{La_2 - Ka_1}{2} \Bigg) k_0^2 \leq 0, \tag{27}$$

$$k_a + \Bigg(\frac{K + L}{2} a_1 - \frac{a_2}{2} L \Bigg) k_a^2 \leq 0. \tag{28}$$

- When m is an odd number,

$$(-1)^{\frac{m-1}{2}} a_m \geq 0, \tag{29}$$

$$\phi_s := Ls(-1)^{s+1} a_{2s+1} - L \frac{(-1)^s a_{2s+1}}{2} + |a_{2s}| L + K(-1)^s a_{2s} \leq 0, \quad \text{for } s = 1, \ldots, \frac{m-1}{2}, \tag{30}$$

$$\psi := a_0 K - a_1 L/2 + \sum_{s=1}^{\frac{m-1}{2}} \frac{\phi_s}{2^s} < 0, \tag{31}$$

$$\frac{(1-p)\psi + a_0 L}{L} < 0, \tag{32}$$

$$k_0 + \Bigg(\sum_{s=1}^{\frac{m-1}{2}} \frac{\phi_s}{2^{s-1}} + \frac{La_2 - Ka_1}{2} \Bigg) k_0^2 \leq 0, \tag{33}$$

$$k_a + \Bigg(\frac{K + L}{2} a_1 - \frac{a_2}{2} L \Bigg) k_a^2 \leq 0. \tag{34}$$

Here, p is a constant satisfying $0 < p \leq 1$.

Proof: For this proof, see Appendix I. ■

Remark 1: Our proposed method enriches the class of planar curves to which we can deploy the agent. However, as the order of PDE increases, the number of neighbor agents required for each agent's controller increases when we implement the controller. For example, the control input of the i-th follower needs information from at least $[(m+1)/2]$ agents next to the follower, where m is the order of the PDE. In addition, the controllers of the leader agents when $m \geq 3$ require information on m agents in its neighbor. For more details, please see the following subsection. Therefore, when increasing the order of PDE, enough agents are required to realize the controller.

C. Controller of each agent

To apply the above PDE to the control of a multi-agent system, it is necessary to discretize the PDE. For this, we use the discretization method used in [28]. In particular, we use the central difference for the first-order spatial derivative. On the other hand, for spatial derivatives of other orders, we discretize the partial derivative by alternately repeating forward and backward differences. Except for the first-order derivative, the terms of $x^{(s)}(\alpha,t)$ are discretized with respect to spatial variables as follows:

$$x^{(s)}(\alpha,t) = X(s) = \frac{\sum_{k=0}^{s}(-1)^k {}_sC_k x_{i+[(s+1)/2]-k}}{h^s}, \quad (35)$$

where h represents the spatial difference $1/(N-1)$, and x_i is the i -th node where $\alpha = 0$ is the node x_1 and corresponds to the x coordinate of i-th agent. Thus, the control input of the i-th follower agent $\dot{x}_i$ is as follows:

$$\dot{x}_i = \sum_{s=2}^{m} a_s X(s) + a_1 \frac{x_{i+1}-x_{i-1}}{2h} + a_0 x_i. \quad (36)$$

From (36), we can see that the controller of each follower agent contains only the common PDE parameters, a_s, a_1, and a_0, and the information of the neighboring agents.

Next, we derive the controllers of the leader agents. For the discretization of the boundary conditions, we use the forward difference for the boundary condition at $\alpha = 0$ and the backward difference for the boundary condition at $\alpha = 1$. This discretization of the boundary conditions is based on [28]. By discretizing the designed boundary controller (19) and (20), the two leaders' controllers $x_1(t)$ and $x_N(t)$ corresponding to $\alpha = 0$ and 1 become

$$x_1(t) = k_0\Big\{L\sum_{j=3}^{m} a_j X_0(j-2) - K\sum_{j=2}^{m} a_j X_0(j-1)\Big\} + \bar{u}_0, \quad (37)$$

$$x_N(t) = k_a\Big\{(K+L)\sum_{j=2}^{m} a_j X_a(j-1) - L\sum_{j=3}^{m} a_j X_a(j-2)\Big\} + \bar{u}_a, \quad (38)$$

where

$$X_0(j) = \sum_{k=0}^{j}\frac{(-1)^k {}_jC_k x_{1+j-k}}{h^j} - \bar{x}^{(j)}(0), \quad (39)$$

$$X_a(j) = \sum_{k=0}^{j}\frac{(-1)^k {}_jC_k x_{N-k}}{h^j} - \bar{x}^{(j)}(1). \quad (40)$$

Here, $\bar{x}^{(j)}(0)$ and $\bar{x}^{(j)}(1)$ are related to the desired shape $\bar{x}(\alpha)$, and thus we can see that the controls of the leader agents include information about the desired shape.

On the other hand, controllers of $m-2$ leader agents other than $i = 1$ and N are as follows:

$$x_p(t) = k_0\Big\{L\sum_{j=3}^{m} a_j X_0(j-2) - K\sum_{j=2}^{m} a_j X_0(j-1)\Big\} + \bar{u}_0 + \sum_{j=1}^{P-1} {}_{P-1}C_j u_{0j} h^j, \quad (41)$$

$$x_q(t) = k_a\Big\{(K+L)\sum_{j=3}^{m} a_j X_a(j-1) - L\sum_{j=2}^{m} a_j X_a(j-2)\Big\} + \bar{u}_a + \sum_{j=1}^{N-Q} (-1)^j {}_{N-Q}C_j u_{aj} h^j, \quad (42)$$

where P, Q are integers, and $P = 2, \cdots, [m/2]-1$, $Q = N - [(m-1)/2], \cdots, N-1$.

IV. Numerical Simulations

Here we confirm by the numerical simulations that various desired shapes can be achieved. We consider the following cases:

Case 1:
$\bar{x}(\alpha) = \sin(2\pi\alpha) + 0.5$ and $\bar{y}(\alpha) = \cos(2\pi\alpha) + 1$.
This means that the desired shape is a circle whose center is far from the origin.

Case 2:
$\bar{x}(\alpha) = \sin(3\pi\alpha) + 0.5$ and $\bar{y}(\alpha) = \sin(2\pi\alpha) + 0.5$.
The desired shape is a heart shape whose center is far from the origin.

Case 3:
$\bar{x}(\alpha) = \alpha$ and $\bar{y}(\alpha) = \left(\alpha - \frac{1}{2}\right)^2$.
The desired shape is a quadratic curve.

Case 4:
$\bar{x}(\alpha) = \alpha + 0.5$ and $\bar{y}(\alpha) = \left(\alpha - \frac{1}{2}\right)^2 + 0.25$.
The desired shape is a quadratic curve with offset.

Case 5:
$\bar{x}(\alpha) = \alpha + 0.5$ and $\bar{y}(\alpha) = 10\alpha(\alpha - 1/2)(\alpha - 1) + 0.25$.
The desired shape is a cubic curve whose center is far from the origin.

In particular, in Cases 2 to 5, the shape cannot be achieved unless the desired values are given to all agents in the previous studies. In addition, the initial positions of the agents are deployed at equal intervals on the x-axis $[0, 1]$.

A. Setting of parameters

Now we demonstrate how to set the PDE and various parameters when the desired shape is given in our method. As a simple example, we consider a case where the desired shape is given by $\bar{x}(\alpha) = \alpha$.

(*Step 1*): *Setting the order m of the PDE model and the PDE parameters.* Substituting the desired shape $\bar{x}(\alpha)$ into the equilibrium curve (10) gives the following identity for α:

$$a_1 + a_0\alpha = 0. \quad (43)$$

We set m so that the PDE parameters have nontrivial solutions when solving this identity for α. In this simple example, we need to set m to be 2 or higher. If $m = 2$, the boundary conditions (7) and (9), which correspond to the control inputs of the leader agents, are omitted. Now we describe how to determine the control inputs of the leader agents, so we set $m = 3$ here.

Finding a_i, $i = 0, \dots, 3$, that satisfies the identity (43) gives

$$a_2, a_3 : \text{arbitrary}, \tag{44}$$

$$a_0 = a_1 = 0. \tag{45}$$

We set the PDE parameters to 0 as much as possible within the range that satisfies the exponential stabilization conditions (23)–(34), that is, the conditions for the closed loop system to be exponential stable. This is to make it easier to confirm the uniqueness of the equilibrium curve solution described later. Now, we set $a_3 = -1$, $a_0 = a_1 = a_2 = 0$.

(*Step 2): Setting the desired value in the controllers of the leader agents.* From $\bar{x}(\alpha) = \alpha$, the parameters related to the leader agents can be uniquely determined as follows: $\bar{u}_0 = \bar{x}(0) = 0$, $\bar{u}_a = \bar{x}(1) = 1$, $u_{a1} = \bar{x}^{(1)}(1) \approx \frac{\bar{x}_N - \bar{x}_{N-1}}{h} = 1$, where $\bar{x}_i$ represents the desired position of agent i and is represented as $\bar{x}((i-1)h)$. In addition, $\bar{x}^{(1)}(0)$, $\bar{x}^{(2)}(0)$, and $\bar{x}^{(2)}(1)$ which are parameters included in (39) and (40), can be uniquely determined as follows: $\bar{x}^{(1)}(0) \approx \frac{\bar{x}_2 - \bar{x}_1}{h} = 1, \bar{x}^{(2)}(0) \approx \frac{\bar{x}_3 - 2\bar{x}_2 + \bar{x}_1}{h^2} = 0, \bar{x}^{(2)}(1) \approx \frac{\bar{x}_N - 2\bar{x}_{N-1} + \bar{x}_{N-2}}{h^2} = 0$.

(*Step 3): Investigating the uniqueness of a solution to the equilibrium curve.* Up to Step 2, an equilibrium curve with $\bar{x}(\alpha) = \alpha$ as the solution could be set. However, if a solution other than the desired shape exists in this ODE, the model may not converge to the desired shape due to the superposition of the solutions. Therefore, it is necessary to find that the solution to the equilibrium curve set by the user has only the function of the desired shape. In this method, since the equilibrium curve is a linear ordinary differential equation with simple boundary conditions (which are constants), we can easily derive solutions by using symbolic calculation software. The equilibrium curve for this example of $\bar{x}(\alpha) = \alpha$ is as follows:

$$\begin{cases} \bar{x}^{(3)}(\alpha) = 0, \\ \bar{x}(0) = 0, \bar{x}(1) = 1, \bar{x}^{(1)} = 1. \end{cases} \tag{46}$$

Solving this ODE gives the solution $\bar{x}(\alpha) = \alpha$. From this, it is confirmed that this ODE has only the desired shape as the solution.

(*Step 4): Setting the feedback gain.* Since the setting of each PDE parameter was completed in Step 2, we substitute those parameters into the exponential stability conditions (27) and (28), or (33) and (34). We only need to set the feedback gains k_0 and k_a to satisfy the exponential stability conditions. If the response of the system diverges from this setting, the values of the PDE parameters are large, and the discretization error may be amplified. In that case, a reduction coefficient of less than 1 is multiplied by all the PDE parameters obtained in Step 2, and the feedback gain setting in Step 5 is re-set accordingly.

Using the above method, PDE parameters can be set for each case in the simulation. The results are as follows:

Cases 1 and 2:

$$\begin{cases} (a_0, a_1, \cdots, a_m) \\ \quad = (0, 0, 4\pi^2, 38/15(4\pi^2), 1, 38/15) \times 10^{-4}, \\ (b_0, b_1, \cdots, b_m) \\ \quad = (0, 0, 4\pi^2, 38/15(4\pi^2), 1, 38/15) \times 10^{-4}, \\ (k_{0x}, k_{ax}, k_{0y}, k_{ay}) = (-0.1, -0.1, -0.1, -0.1). \end{cases}$$

Cases 3 and 4:

$$\begin{cases} (a_0, a_1, \cdots, a_m) = (0, 0, 0, -2) \times 10^{-3}, \\ (b_0, b_1, \cdots, b_m) = (0, 0, 0, -2) \times 10^{-3}, \\ (k_{0x}, k_{ax}, k_{0y}, k_{ay}) \\ \quad = (-1.25, -1, -1.25, -1) \times 10^{-2}. \end{cases}$$

Case 5:

$$\begin{cases} (a_0, a_1, \cdots, a_m) = (0, 0, 0, 0, -1, 0) \times 10^{-2}, \\ (b_0, b_1, \cdots, b_m) = (0, 0, 0, 0, -1, 0) \times 10^{-2}, \\ (k_{0x}, k_{ax}, k_{0y}, k_{ay}) = (-1, -1, -1, -1) \times 10^{-2}. \end{cases}$$

The parameters of each leader agent are uniquely obtained from each desired shape. For Cases 1, 2, and 5, the 5th-order PDE model, $m = 5$, is used, the step time Δt is set as 1ms, and the number of agents N is 11. For Cases 3 and 4, the 3-rd order PDE model, $m = 3$, is used, the step time Δt is 0.25s, and the number of agents N is 7.

B. The shapes in cases 2 to 5

We describe why the shapes in cases 2 to 5 cannot be achieved by the conventional method that uses the boundary control of PDEs. Since the existing researches consider a second-order PDE model, its equilibrium curve is defined as follows:

$$a_2 \bar{x}^{(2)}(\alpha) + a_1 \bar{x}^{(1)}(\alpha) + a_0 \bar{x}(\alpha) = 0, \tag{47}$$

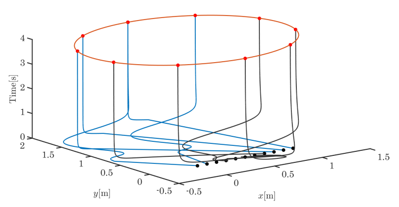
where the PDE parameters a_0, a_1, a_2 are set so that the solution of (47) becomes the desired shape. Now, let us consider case 2. Substituting the desired shape $\bar{x}(\alpha) = \sin(3\pi\alpha) + 0.5$ into (47) gives

$$-9\pi^2 a_2 \sin(3\pi\alpha) + 3\pi a_1 \cos(3\pi\alpha) + a_0 \sin(3\pi\alpha) + 0.5a_0 = 0. \tag{48}$$

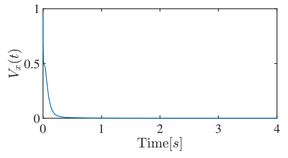
In order for (48) to hold at any $\alpha \in [0, 1]$, parameters a_0, a_1, a_2 must all be set to 0. This means that all the coefficients of the PDE model are set to 0, in which case each agent does not move from the initial position. Therefore, with the conventional method, it is not possible to set a complicated function like cases 2-5. On the other hand, in our method, since the partial derivative of the arbitrary high order is used, it is possible to use the partial derivative of the order whose target shape is the solution.

C. Results of numerical simulations

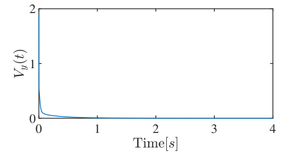
Figures 2–6 show the simulation results of Cases 1 to 5, respectively. In each figure, (a) shows the time response of each agent's position. The black line is the trajectory of each follower agent, and the blue line is the trajectory of each leader agent. The target shape is shown by an orange line, the initial



(a) Time response of each agent's position

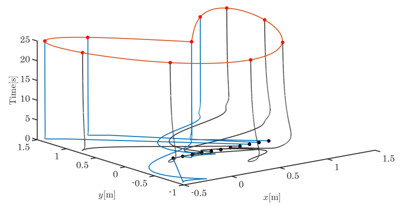


(b) Lyapunov function of x-axis

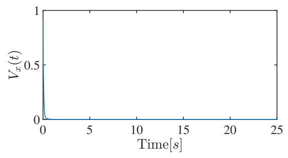


(c) Lyapunov function of y-axis

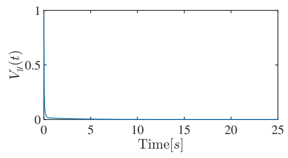
Fig. 2: Simulation results (Case 1).



(a) Time response of each agent's position

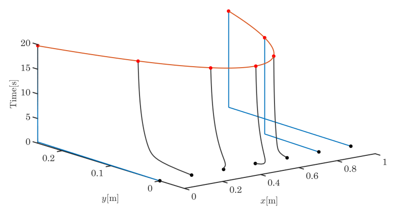


(b) Lyapunov function of x-axis

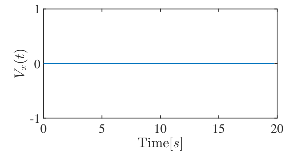


(c) Lyapunov function of y-axis

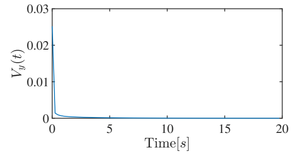
Fig. 3: Simulation results (Case 2).



(a) Time response of each agent's position

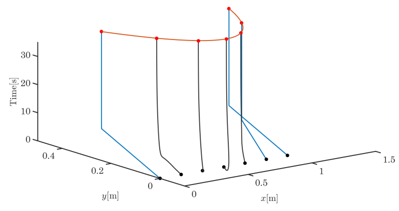


(b) Lyapunov function of x-axis

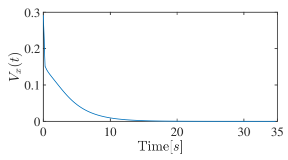


(c) Lyapunov function of y-axis

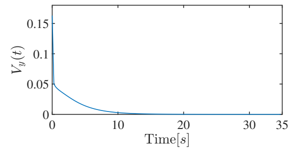
Fig. 4: Simulation results (Case 3).



(a) Time response of each agent's position



(b) Lyapunov function of x-axis



(c) Lyapunov function of y-axis

Fig. 5: Simulation results (Case 4).

position of each agent is shown as a black point, and the convergence point of each agent is shown as a red point. From these figures, we can see that all agents are deployed on the desired shapes.

On the other hand, in Figs. 2–6, (b) and (c) show the time response of the L^2 norm $V_x(t)$ and $V_y(t)$ of the error system, respectively. Here, $V_x(t)$ and $V_y(t)$ are the Lyapunov function of the model on the x and y axes, respectively. In the figures, the value of the Lyapunov function $V_i(t)$, for $i = x$ and y, is shown by a blue line. The Lyapunov function $V(t)$ in the simulation is derived as follows:

$$V(t) = \int_0^1 \tilde{x}^2 \mathrm{d}\alpha \approx \sum_{i=0}^{N} \tilde{x}^2((i-1)h)h. \tag{49}$$

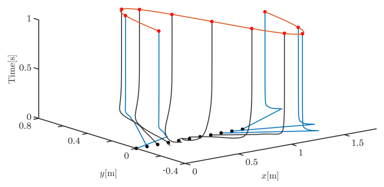
In a multi-agent system, this equation corresponds to the sum of the squared error between each agent and its desired position. From these figures, we found that all agents converge to the target shape while satisfying the exponential stability.

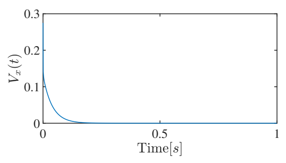
V. Experiments

Now, we consider the validity of our method using real robots.

A. Update algorithm of desired formation

An actual robot has a speed limitation, that is, the maximum speed the robot can output. However, the control input for the leader agent in our method is a position input, and thus there is a possibility that the speed of the leader agent exceeds the real robot's speed limitation. For example, Fig. 7 shows the time response of each agent's velocity in Case 5 of the numerical simulation. In this figure, the blue lines represent the speed of each leader agent and the black lines represent the speed of each follower agent. The maximum speed of the omni-wheel robot used in the experiment is 0.3m/s, and we can see that several robots are faster than that. Here, note that the reason why the speed limitation occurs is that the desired formation is far from the initial configuration, and the leader agent moves fast accordingly. Thus, we update the desired shape as follows: as shown in Fig. 8, the desired shape is gradually updated in a time-varying manner from a location close to the initial arrangement and is fixed to the desired shape that the user finally wants to reach when the time becomes $t = t_k$. By using this time-varying manner, it can be considered that all agents are finally placed on the desired shape while keeping the speed limitation. At that time,


(a) Time response of each agent's position


(b) Lyapunov function of x-axis

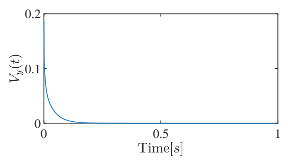

(c) Lyapunov function of y-axis

Fig. 6: Simulation results (Case 5).

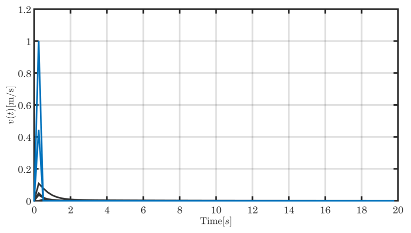


Fig. 7: Time response of each agent's velocity (Case3 in numerical simulation).

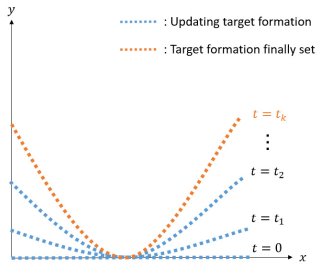


Fig. 8: Schematic diagram of updating target formation.

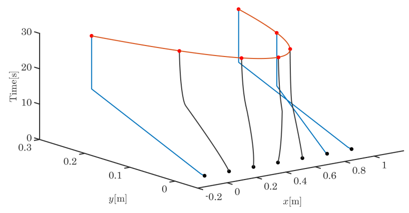


Fig. 9: Time response of each agent's position after applying the update algorithm (Case3 in numerical simulation).

only the design parameters of the leader agent are updated, and it is not necessary to update the design parameters of the follower agent.

Figure 9 shows the results of application of the update algorithm to Case 3 in the numerical simulation, and Fig. 10 shows the time response of the speed of each agent at that time. These results indicate that all agents converged to the desired shape while keeping the real robot's speed limitation 0.3m/s. In this simulation, the values of $\bar{x}(\alpha)$, $\bar{y}^t(\alpha)$, t_k, and t_{end} were set to α, $\frac{2t}{t_{end}}(\alpha-1/2)^2$, $\frac{t_{end}}{2}$, and 30s, respectively.

B. Experiment environment

In this experiment, we used a motion capture system. Figure 11 shows a schematic diagram of the experimental system. Although control of the system itself was centralized, each agent used only local information when calculating its control inputs on the control PC, and the inputs were sent to each agent via Bluetooth from the control PC.

We performed the following cases in the experiment:

Case E1:

$$\bar{x}(\alpha) = 1.8\alpha \text{ and } \bar{y}(\alpha) = 2.5(\alpha - 1/2)^2.$$

Case E2:

$$\bar{x}(\alpha) = 1.8\alpha + 0.3 \text{ and } \bar{y}(\alpha) = 2.5(\alpha - 1/2)^2 + 0.3.$$

In both cases, the desired formations were quadratic curves, and the desired formation in Case E1 is generated near the origin while that in Case E2 is generated at a position distant from the origin. Setting parameters, such as PDE parameters and feedback gains, took the same values as in Cases 3 and 4 in the numerical simulation. To update the desired formation, we set $\bar{y}^t = 5.0t(\alpha - 1/2)^2/t_{end}$ on Case E1, and $\bar{x}^t = 1.8\alpha + 0.6t/t_{end}$, $\bar{y}^t = 5.0t(\alpha - 1/2)^2/t_{end} + 0.6t/t_{end}$ on Case E2. In both cases, the desired shape is finally fixed at $t_{end}/2$s. The end time t_{end} is set to 30s for Case E1 and 48s for Case E2. The initial state in both cases is a state in which agents are arranged at equal intervals in the $[0, 1.8]$ section on the x-axis, and the total number of agents N is set to 7.

C. Experimental results

The experimental results for Cases E1 and E2 are shown in Fig. 12 and 13, respectively. In the figures, (a) is a snapshot of the experiment and shows the appearance of the experiment at 0s, 10s, 20s, and t_{end} after the start time. In both cases, agents 1 to 7 were set as agents 1 to 7 from the left robot to the right robot, and agents 1, 6, and 7 were set as the leader agents. (b) shows the behavior of each agent by using different colors. The initial position is represented by a black point, and the convergence position is represented by a black dotted line. (c) and (d) show the time responses of the x and y coordinates, respectively. Information on the positions of these agents was obtained from motion capture data.

From Fig. 12 and 13, we can see that the proposed method makes it is possible to generate quadratic curves in a real robot system near the origin and at points away from the origin. We calculated the sum of squared error between the convergence position of each agent and the desired position of each agent for each axis. In Case E1, the error of the x axis was $4.39 \times 10^{-8}\text{m}^2$ and that of the y axis was $1.83 \times 10^{-7}\text{m}^2$. In Case E2, the error of the x axis was $2.89 \times 10^{-9}\text{m}^2$ and the error of the y axis was $3.37 \times 10^{-9}\text{m}^2$. Therefore, in both cases, it was confirmed that all agents converged to the desired formation. The reason why a little error remains in some agents is that the input to the robot decreases as the position approaches

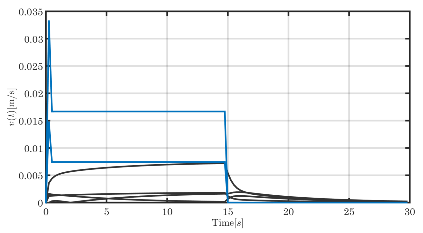


Fig. 10: Time response of each agent's velocity after applying the update algorithm (Case 3 in numerical simulation).

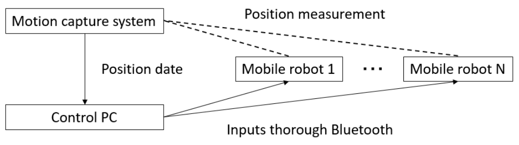
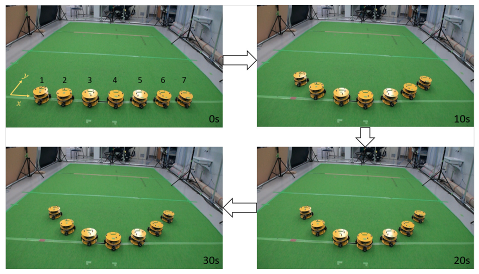
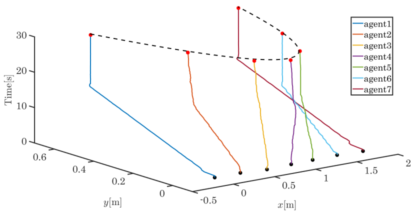


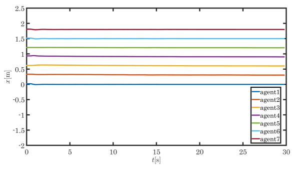
Fig. 11: Schematic diagram of the experimental system



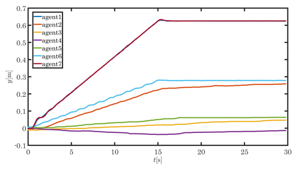
(a) Snapshot of experiment



(b) Time response of each agent's position



(c) x coordinate of each agent



(d) y coordinate of each agent

Fig. 12: Experimental results (Case E1).

the desired position, and the applied voltage drops below the minimum applied voltage.

VI. Conclusion

We have proposed a method that uses higher-order PDEs to control formation in a multi-agent system to achieve abundant formations that the previous methods could not achieve. In this method, only the leader agent has the relative position information of the desired shape, and the controller of the follower agent contains only constants of the PDE. Therefore, even if we apply this method to a huge number of robot groups, the number of parameters to be set does not change.

We derived a boundary controller that stabilizes a PDE having any m-order and showed that each agent converges exponentially to the desired shape by using this control method. Furthermore, we showed that the proposed method can be applied to a real robot system by proposing an update algorithm of the desired formation.

In our future work we will prove the well-posedness of PDEs in this method and collision avoidance between agents, and we will propose a method that can be applied even when the desired shape is time-varying. Further, we did not consider the loss of a leader agent in this paper. We hypothesize, however, that such fragility could be solved by a moving agent who could replace the leader when a leader agent is lost, with the leader as an escort runner. The robustness to the loss of a leader agent will be addressed in future research.

Appendix I
Proof of Theorem 1

We consider $L^2 := L^2(0,1)$ as the state space and consider a candidate for the Lyapunov function as follows: That is, we consider $L^2(0,1)$ space as the state space of closed-loop system, and we consider the Lyapunov function consisting of the norm in $L^2(0,1)$ space:

$$V(t) = \frac{K}{2}\int_0^1 \tilde{x}^2 \mathrm{d}\alpha + \frac{L}{2}\int_0^1 \alpha\tilde{x}^2 \mathrm{d}\alpha, \tag{50}$$

where K and L are positive constants. Differentiating (50) with respect to time and employing (13) gives

$$\dot{V}(t) = KI_1 + LI_2, \tag{51}$$

where

$$I_1 = \int_0^1 \Big(\sum_{i=0}^m a_i\tilde{x}^{(i)}\Big)\tilde{x}\mathrm{d}\alpha, \quad I_2 = \int_0^1 \alpha\Big(\sum_{i=0}^m a_i\tilde{x}^{(i)}\Big)\tilde{x}\mathrm{d}\alpha. \tag{52}$$

For clarity, we divide the proof into three steps.
(*1st step*): First, we consider I_1, which is rewritten as follows:

$$I_1 = \int_0^1 \Big(\sum_{i=2}^m a_i\tilde{x}^{(i)}\Big)\tilde{x}\mathrm{d}\alpha + a_1\int_0^1 \tilde{x}^{(1)}\tilde{x}\mathrm{d}\alpha + a_0\int_0^1 \tilde{x}^2\mathrm{d}\alpha. \tag{53}$$

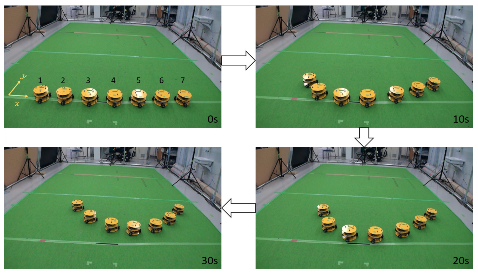
When $m \geq 4$, using integration by parts for the first term on the right side, I_1 can be written as

$$I_1 = U_1 - \int_0^1 \Big(\sum_{i=4}^m a_i\tilde{x}^{(i-1)}\Big)\tilde{x}^{(1)}\mathrm{d}\alpha + J_3 + J_2 + J_1 + J_0, \tag{54}$$

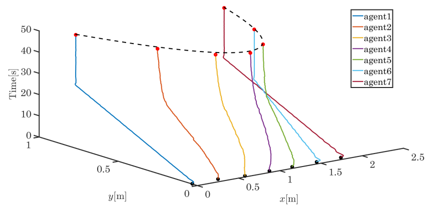
where

$$\begin{cases} U_1 = \left[\Big(\sum_{i=2}^m a_i\tilde{x}^{(i-1)}\Big)\tilde{x}\right]_0^1, \\ J_0 = a_0\int_0^1\tilde{x}^2\mathrm{d}\alpha, \quad J_1 = a_1\int_0^1\tilde{x}^{(1)}\tilde{x}\mathrm{d}\alpha, \\ J_2 = -a_2\int_0^1(\tilde{x}^{(1)})^2\mathrm{d}\alpha, \quad J_3 = -a_3\int_0^1\tilde{x}^{(1)}\tilde{x}^{(2)}\mathrm{d}\alpha. \end{cases} \tag{55}$$

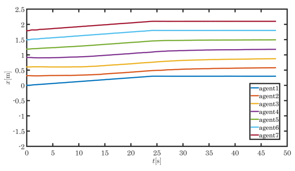
Furthermore, we can perform the integration by parts on the second term of (54). This allows us to transform (54) just



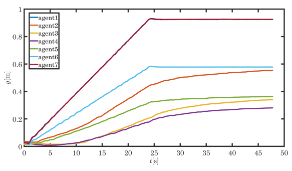
(a) Snapshot of experiment



(b) Time response of each agent's position



(c) x coordinate of each agent



(d) y coordinate of each agent

Fig. 13: Experimental results (Case E2).

as (53) has been transformed into (54). By repeating this procedure, I_1 can finally be rewritten as follows:

$$I_1 = \sum_{i=1}^{[m/2]} U_i + \sum_{i=0}^{m} J_i, \tag{56}$$

where

$$\begin{cases} U_i = (-1)^{i+1}\Big[\Big(\sum_{j=2i}^{m} a_j \tilde{x}^{(j-i)}\Big)\tilde{x}^{(i-1)}\Big]_0^1, \\ J_{2s} = (-1)^s a_{2s} \int_0^1 \tilde{x}^{(s)}\tilde{x}^{(s)} \mathrm{d}\alpha, \quad \text{for } s = 0, 1, 2, \cdots, \\ J_{2s+1} = (-1)^s a_{2s+1} \int_0^1 \tilde{x}^{(s+1)}\tilde{x}^{(s)} \mathrm{d}\alpha. \end{cases} \tag{57}$$

For $i = 2, \dots, [m/2]$, $\tilde{x}^{(i-1)}(0) = \tilde{x}^{(i-1)}(1) = 0$, and thus $U_i = 0$. So, the first term on the right side of (56) becomes

$$\sum_{i=1}^{[m/2]} U_i = \Big(\sum_{j=2}^{m} a_j \tilde{x}^{(j-1)}(1)\Big)\tilde{x}(1) - \Big(\sum_{j=2}^{m} a_j \tilde{x}^{(j-1)}(0)\Big)\tilde{x}(0). \tag{58}$$

For J_{2s+1}, using the integration by parts gives

$$J_{2s+1} = \frac{(-1)^s a_{2s+1}}{2}\{(\tilde{x}^{(s)}(1))^2 - (\tilde{x}^{(s)}(0))^2\}. \tag{59}$$

In $s = 1, \dots, [(m-1)/2]$, if m is an even number, $\tilde{x}^{(s)}(1) = \tilde{x}^{(s)}(0) = 0$, and thus we obtain

$$\sum_{s=0}^{[(m-1)/2]} J_{2s+1} = \frac{a_1}{2}(\tilde{x}^2(1) - \tilde{x}^2(0)). \tag{60}$$

On the other hand, if m is an odd number, $\tilde{x}^{([(m-1)/2])}(0) \neq 0$, and thus we obtain

$$\sum_{s=0}^{[(m-1)/2]} J_{2s+1} = \frac{a_1}{2}(\tilde{x}^2(1) - \tilde{x}^2(0)) - \frac{(-1)^{[(m-1)/2]}a_m}{2}(\tilde{x}^{([(m-1)/2])}(0))^2. \tag{61}$$

Therefore, when the condition (29) holds, J_{2s+1} can be evaluated as follows:

$$\sum_{s=0}^{[(m-1)/2]} J_{2s+1} \leq \frac{a_1}{2}(\tilde{x}^2(1) - \tilde{x}^2(0)). \tag{62}$$

Thus, we obtain the following inequality for I_1 regardless of whether m is even or odd.

$$I_1 \leq \Big(\sum_{j=2}^{m} a_j \tilde{x}^{(j-1)}(1)\Big)\tilde{x}(1) - \Big(\sum_{j=2}^{m} a_j \tilde{x}^{(j-1)}(0)\Big)\tilde{x}(0) + \frac{a_1}{2}(\tilde{x}^2(1) - \tilde{x}^2(0)) + \sum_{s=0}^{[m/2]} (-1)^s a_{2s} \int_0^1 (\tilde{x}^{(s)})^2 \mathrm{d}\alpha. \tag{63}$$

(2nd step): Next, we consider I_2, the second term on the right side of (51). Using the integration by parts gives

$$I_2 = \Big[\alpha\Big(\sum_{i=2}^{m} a_i \tilde{x}^{(i-1)}\Big)\tilde{x}\Big]_0^1 - \int_0^1 \Big(\sum_{i=2}^{m} a_i \tilde{x}^{(i-1)}\Big)\tilde{x}\mathrm{d}\alpha - \int_0^1 \alpha\Big(\sum_{i=2}^{m} a_i \tilde{x}^{(i-1)}\Big)\tilde{x}^{(1)}\mathrm{d}\alpha + a_1 \int_0^1 \alpha \tilde{x}^{(1)}\tilde{x}\mathrm{d}\alpha + a_0 \int_0^1 \alpha \tilde{x}^2 \mathrm{d}\alpha. \tag{64}$$

By repeating the same procedure as the expansion of I_1 using the integration by parts, we obtain

$$I_2 = \sum_{i=1}^{[m/2]} \bar{U}_i + \sum_{i=1}^{[m/2]} S_i + \sum_{i=0}^{m} \bar{J}_i, \tag{65}$$

where

$$\begin{cases} \bar{U}_i = \Big[\alpha\Big(\sum_{j=2i}^{m} a_j \tilde{x}^{(j-i)}\Big)\tilde{x}^{(i-1)}\Big]_0^1, \\ S_i = (-1)^i \int_0^1 \Big(\sum_{j=2i}^{m} a_j \tilde{x}^{(j-1)}\Big)\tilde{x}^{(i-1)}\mathrm{d}\alpha, \\ \bar{J}_{2s} = (-1)^s a_{2s} \int_0^1 \alpha(\tilde{x}^{(s)})^2 \mathrm{d}\alpha, \quad \text{for } s = 0, 1, 2, \cdots, \\ \bar{J}_{2s+1} = (-1)^s a_{2s+1} \int_0^1 \alpha \tilde{x}^{(s+1)}\tilde{x}^{(s)} \mathrm{d}\alpha. \end{cases} \tag{66}$$

Since $\tilde{x}^{(i-1)}(1) = 0$ at $i = 2, \cdots, [m/2]$, we obtain $\bar{U}_i = 0$. Thus, the first term on the right side of (65) can be written as

$$\sum_{i=1}^{[m/2]} \bar{U}_i = \Big(\sum_{j=2}^{m} a_j \tilde{x}^{(j-1)}(1)\Big)\tilde{x}(1). \tag{67}$$

For $\bar{J}_{2s+1}$, using the integration by parts leads to

$$\bar{J}_{2s+1} = \frac{(-1)^s a_{2s+1}}{2}\Big\{(\tilde{x}^{(s)})^2(1) - \int_0^1 (\tilde{x}^{(s)})^2 \mathrm{d}\alpha\Big\}. \tag{68}$$

For $\bar{J}_{2s}$, we have the following inequality for $s = 1, \dots, m/2$ when m is even and for $s = 1, \dots, (m-1)/2$ when m is odd:

$$\bar{J}_{2s} = (-1)^s a_{2s} \int_0^1 \alpha (\tilde{x}^{(s)})^2 d\alpha \le |a_{2s}| \int_0^1 (\tilde{x}^{(s)})^2 \mathrm{d}\alpha. \quad (69)$$

For S_i, simple calculations give

$$\begin{aligned} S_i = (-1)^i &\Big[\Big(\sum_{j=2i+1}^m a_j \tilde{x}^{(j-i-1)}\Big)\tilde{x}^{(i-1)}\Big]_0^1 + S_{i+1} \\ &+ (-1)^{i+1} a_{2i+1} \int_0^1 (\tilde{x}^{(i)})^2 \mathrm{d}\alpha \\ &+ \frac{(-1)^i a_{2i}}{2} \Big\{(\tilde{x}^{(i-1)})^2(1) - (\tilde{x}^{(i-1)})^2(1)\Big\}, \end{aligned} \quad (70)$$

and thus we obtain

$$\begin{aligned} \sum_{i=1}^{[m/2]} S_i = [m/2] S_{[m/2]} &+ \sum_{i=1}^{[m/2]-1} i(-1)^{i+1} a_{2i+1} \int_0^1 (\tilde{x}^{(i)})^2 \mathrm{d}\alpha \\ &- \Big(\sum_{j=3}^m a_j \tilde{x}^{(j-2)}(1)\Big)\tilde{x}(1) \\ &+ \Big(\sum_{j=3}^m a_j \tilde{x}^{(j-2)}(0)\Big)\tilde{x}(0) - \frac{a_2}{2}\left(\tilde{x}^2(1) - \tilde{x}^2(0)\right). \end{aligned} \quad (71)$$

Here, if m is even, we can easily obtain $S_{[m/2]} = 0$, which leads to

$$\begin{aligned} \sum_{i=1}^{[m/2]} S_i = &\sum_{i=1}^{[m/2]-1} i(-1)^{i+1} a_{2i+1} \int_0^1 (\tilde{x}^{(i)})^2 d\alpha \\ &- \Big(\sum_{j=3}^m a_j \tilde{x}^{(j-2)}(1)\Big)\tilde{x}(1) \\ &+ \Big(\sum_{j=3}^m a_j \tilde{x}^{(j-2)}(0)\Big)\tilde{x}(0) - \frac{a_2}{2}\left(\tilde{x}^2(1) - \tilde{x}^2(0)\right). \end{aligned} \quad (72)$$

Thus, when m is even, we obtain the following estimation about I_2 by substituting (23), (67)–(69), and (72) into (65):

$$\begin{aligned} I_2 \le &\Big[\Big(\sum_{j=2}^m a_j \tilde{x}^{(j-1)}(1)\Big) - \Big(\sum_{j=3}^m a_j \tilde{x}^{(j-2)}(1)\Big)\Big]\tilde{x}(1) \\ &+ \frac{a_1 - a_2}{2}\tilde{x}^2(1) + \Big(\sum_{j=3}^m a_j \tilde{x}^{(j-2)}(0)\Big)\tilde{x}(0) + \frac{a_2}{2}\tilde{x}^2(0) \\ &+ \sum_{s=1}^{m/2-1} p_s \int_0^1 (\tilde{x}^{(s)})^2 \mathrm{d}\alpha - \frac{a_1}{2}\int_0^1 \tilde{x}^2 \mathrm{d}\alpha + a_0 \int_0^1 \alpha \tilde{x}^2 \mathrm{d}\alpha, \end{aligned} \quad (73)$$

where $p_s = s(-1)^{s+1} a_{2s+1} - \frac{(-1)^s a_{2s+1}}{2} + |a_{2s}|$.

On the other hand, when m is odd, we obtain

$$S_{[m/2]} = (-1)^{[m/2]+1} a_m \int_0^1 (\tilde{x}^{([m/2])})^2 \mathrm{d}\alpha. \quad (74)$$

Thus, using the same procedure as when m is even, the following equation holds for I_2 when m is odd:

$$\begin{aligned} I_2 = &\Big[\Big(\sum_{j=2}^m a_j \tilde{x}^{(j-1)}(1)\Big) - \Big(\sum_{j=3}^m a_j \tilde{x}^{(j-2)}(1)\Big)\Big]\tilde{x}(1) \\ &+ \frac{a_1 - a_2}{2}\tilde{x}^2(1) + \Big(\sum_{j=3}^m a_j \tilde{x}^{(j-2)}(0)\Big)\tilde{x}(0) \\ &+ \frac{a_2}{2}\tilde{x}^2(0) + \sum_{s=1}^{(m-1)/2} p_s \int_0^1 (\tilde{x}^{(s)})^2 \mathrm{d}\alpha \\ &- \frac{a_1}{2}\int_0^1 \tilde{x}^2 \mathrm{d}\alpha + a_0 \int_0^1 \alpha \tilde{x}^2 \mathrm{d}\alpha. \end{aligned} \quad (75)$$

(3rd step): We evaluate the time derivative of the Lyapunov function.

First, let us consider the case where m is even. By substituting (63) and (73) into (51), the following estimation can be obtained if condition (24) holds:

$$\begin{aligned} \dot{V}(t) \le &\Big[(K+L)\Big(\sum_{j=2}^m a_j \tilde{x}^{(j-1)}(1)\Big) \\ &\qquad - L\Big(\sum_{j=3}^m a_j \tilde{x}^{(j-2)}(1)\Big)\Big]\tilde{x}(1) \\ &+ \Big(\frac{K+L}{2}a_1 - \frac{a_2}{2}L\Big)\tilde{x}^2(1) \\ &+ \Big[L\Big(\sum_{j=3}^m a_j \tilde{x}^{(j-2)}(0)\Big) - K\Big(\sum_{j=2}^m a_j \tilde{x}^{(j-1)}(0)\Big)\Big]\tilde{x}(0) \\ &+ \frac{La_2 - Ka_1}{2}\tilde{x}^2(0) + a_0 L \int_0^1 \alpha \tilde{x}^2 \mathrm{d}\alpha \\ &+ \Big(a_0 K + \frac{a_1}{2}L + \sum_{s=1}^{m/2-1} \frac{\phi_s}{2^s} \\ &\qquad + K\frac{(-1)^{m/2} a_m}{2^{m/2}}\Big)\int_0^1 \tilde{x}^2 \mathrm{d}\alpha \\ &+ \Big(\sum_{s=1}^{m/2-1} \frac{\phi_s}{2^{s-1}} + K\frac{(-1)^{m/2} a_m}{2^{m/2-1}}\Big)\tilde{x}^2(0). \end{aligned} \quad (76)$$

In this derivation, we used (78). Here, from the equality $\tilde{x}^{(i-1)} = \int_0^\alpha \tilde{x}^{(i)} \mathrm{d}\alpha + \tilde{x}^{(i-1)}(0)$ and Young's inequality, we obtain

$$(\tilde{x}^{(i-1)})^2 \le 2\int_0^1 (\tilde{x}^{(i)})^2 \mathrm{d}\alpha + 2(\tilde{x}^{(i-1)}(0))^2. \quad (77)$$

Since $\tilde{x}^{(i)}(0) = 0$ for $i = 1, \cdots, m/2-1$, we obtain the following estimation:

$$\begin{aligned} \int_0^1 (\tilde{x}^{(i)})^2 \mathrm{d}\alpha &\ge \frac{1}{2}\int_0^1 (\tilde{x}^{(i-1)})^2 \mathrm{d}\alpha - \Big(\tilde{x}^{(i-1)}(0)\Big)^2 \\ &\ge \frac{1}{2^i}\int_0^1 \tilde{x}^2 \mathrm{d}\alpha - \frac{1}{2^{i-1}}\tilde{x}^2(0). \end{aligned} \quad (78)$$

Further, substituting the control inputs (19) and (20) into (76) gives

$$\begin{aligned} \dot{V}(t) \le &\Big\{\Big(\frac{K+L}{2}a_1 - \frac{a_2}{2}L\Big)k_a^2 + k_a\Big\}A^2 \\ &+ (k_0 + qk_0^2)O^2 + \psi \int_0^1 \tilde{x}^2 \mathrm{d}\alpha + a_0 L \int_0^1 \alpha \tilde{x}^2 \mathrm{d}\alpha, \end{aligned} \quad (79)$$

where

$$\begin{cases} q = \sum_{s=1}^{m/2-1} \frac{\phi_s}{2^{s-1}} + K\frac{(-1)^{m/2}a_m}{2^{m/2-1}} + \frac{La_2 - Ka_1}{2}, \\ A = (K+L)\sum_{j=2}^{m} a_j\tilde{x}^{(j-1)}(1) - L\sum_{j=3}^{m} a_j\tilde{x}^{(j-2)}(1), \\ O = L\sum_{j=3}^{m} a_j\tilde{x}^{(j-2)}(0) - K\sum_{j=2}^{m} a_j\tilde{x}^{(j-1)}(0). \end{cases} \tag{80}$$

Therefore, when the conditions (27) and (28) hold, $\dot{V}(t)$ satisfies the following by using a real number p satisfying $0 < p \le 1$.

$$\begin{aligned} \dot{V} &\le \psi\int_0^1 \tilde{x}^2 \mathrm{d}\alpha + a_0L\int_0^1 \alpha\tilde{x}^2\mathrm{d}\alpha \\ &\le p\psi\int_0^1 \tilde{x}^2\mathrm{d}\alpha + \{(1-p)\psi + a_0L\}\int_0^1 \alpha\tilde{x}^2\mathrm{d}\alpha \\ &\le 2\rho V(t), \end{aligned} \tag{81}$$

where $\rho = \max\{\frac{p\psi}{K}$ and $\frac{(1-p)\psi + a_0L}{L}\}$. If conditions (25) and (26) are satisfied, $\rho < 0$. Therefore, the following inequality is derived.

$$\frac{K}{2}\int_0^1 \tilde{x}^2\mathrm{d}\alpha \le \exp(2\rho t)\frac{K+L}{2}\int_0^1 \tilde{x}^2(\alpha, 0)\mathrm{d}\alpha \tag{82}$$

This inequality means

$$\|\tilde{x}(\alpha,t)\|_{L^2} \le \sqrt{\frac{K+L}{K}}\exp(\rho t)\|\tilde{x}(\alpha,0)\|_{L^2}, \tag{83}$$

where $\|\cdot\|_{L^2} = \sqrt{\int_0^1 (\cdot)^2 \ \mathrm{d}\alpha}$. From the above, when m is even, it is shown that the L^2 norm decreases exponentially with time, indicating that the closed-loop system is exponentially stable.

Next, let us consider the case where m is odd. Using the same procedure in the case where m is even, we obtain the following estimation for $\dot{V}$:

$$\begin{aligned} \dot{V}(t) \le \Big(a_0K - \frac{a_1}{2}L + \sum_{s=1}^{(m-1)/2}\frac{\phi_s}{2^s}\Big)\int_0^1 \tilde{x}^2\mathrm{d}\alpha \\ + a_0L\int_0^1 \alpha\tilde{x}^2\mathrm{d}\alpha \le 2\rho V(t). \end{aligned} \tag{84}$$

If the conditions (31) and (32) are satisfied, $\rho < 0$. Therefore, the following inequality is derived.

$$\frac{K}{2}\int_0^1 \tilde{x}^2\mathrm{d}\alpha \le \exp(2\rho t)\frac{K+L}{2}\int_0^1 \tilde{x}^2(\alpha,0)\mathrm{d}\alpha. \tag{85}$$

From the above, when m is odd, it is shown that the L^2 norm decreases exponentially with time, indicating that the closed-loop system is exponentially stable.

Appendix II
Proof of stability when $m = 2$

Here, we prove the stability of the closed-loop system for $m = 2$ described in III-B. The closed-loop system (13) is exponentially stable when $m = 2$ if the following conditions hold:

$$-a_2 \le 0, \tag{86}$$

$$\psi = a_0K - a_1L/2 - K\frac{a_2}{2} < 0, \tag{87}$$

$$\frac{(1-p)\psi + a_0L}{L} < 0, \tag{88}$$

$$k_0 + \Big(-Ka_2 + \frac{La_2 - Ka_1}{2}\Big)k_0^2 \le 0, \tag{89}$$

$$k_a + \Big(\frac{K+L}{2}a_1 - \frac{a_2}{2}L\Big)k_a^2 \le 0. \tag{90}$$

Let us consider the state space as $L^2 = L^2(0,1)$, as in the case of $m \ge 3$, and consider the following function as the Lyapunov function candidate:

$$V(t) = \frac{K}{2}\int_0^1 \tilde{x}^2\mathrm{d}\alpha + \frac{L}{2}\int_0^1 \alpha\tilde{x}^2\mathrm{d}\alpha, \tag{91}$$

where K and L are positive constants. The time derivative of $V(t)$ can be calculated as follows by employing (13):

$$\dot{V}(t) = KI_1 + LI_2, \tag{92}$$

where

$$I_1 = \int_0^1\Big(\sum_{i=0}^{2} a_i\tilde{x}^{(i)}\Big)\tilde{x}\mathrm{d}\alpha, \quad I_2 = \int_0^1 \alpha\Big(\sum_{i=0}^{2} a_i\tilde{x}^{(i)}\Big)\tilde{x}\mathrm{d}\alpha. \tag{93}$$

Now, I_1 can be expanded as

$$I_1 = a_2\int_0^1 \tilde{x}^{(2)}\tilde{x}\mathrm{d}\alpha + a_1\int_0^1 \tilde{x}^{(1)}\tilde{x}\mathrm{d}\alpha + a_0\int_0^1 \tilde{x}^2\mathrm{d}\alpha. \tag{94}$$

Using the integration by parts on the first and second terms on the right side of the above equation, and performing the same procedure for $m \ge 3$, we obtain

$$\begin{aligned} I_1 = a_2\tilde{x}^{(1)}(1)\tilde{x}(1) - a_2\tilde{x}^{(1)}(0)\tilde{x}(0) - a_2\int_0^1 (\tilde{x}^{(1)})^2\mathrm{d}\alpha \\ + \frac{a_1}{2}(\tilde{x}^2(1) - \tilde{x}^2(0)) + a_0\int_0^1 \tilde{x}^2\mathrm{d}\alpha. \end{aligned} \tag{95}$$

Next, following the same procedure for I_2, we obtain

$$\begin{aligned} I_2 = a_2\tilde{x}^{(1)}(1)\tilde{x}(1) - \frac{a_2}{2}(\tilde{x}^2(1) - \tilde{x}^2(0)) - a_2\int_0^1 \alpha(\tilde{x}^{(1)})^2\mathrm{d}\alpha \\ + \frac{a_1}{2}\tilde{x}^2(1) - \frac{a_1}{2}\int_0^1 \tilde{x}^2\mathrm{d}\alpha + a_0\int_0^1 \alpha\tilde{x}^2\mathrm{d}\alpha. \end{aligned} \tag{96}$$

Substituting the obtained I_1, I_2, and the conditions (86)–(90) into (92) leads to

$$\dot{V}(t) \le 2\max\Big\{\frac{p\psi}{K}, \frac{(1-p)\psi + a_0L}{L}\Big\}V(t). \tag{97}$$

From conditions (87) and (88), $\max\{p\psi/K, ((1-p)\psi + a_0L)/L\}$ is negative. Therefore, the closed-loop system is exponentially stable as in the case of $m \ge 3$.

References

[1] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[2] W. Dong and J. A. Farrell, "Cooperative control of multiple nonholonomic mobile agents," *IEEE Trans. Autom. Control*, vol. 53, no. 6, pp. 1434–1448, 2008.

[3] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, 2002.

[4] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control: Collective group behavior through local interaction," *IEEE Control Syst. Mag.*, vol. 27, no. 2, pp. 71–82, 2007.

[5] S. Coogan and M. Arcak, "Scaling the size of a formation using relative position feedback," *Automatica*, vol. 48, no. 10, pp. 2677–2685, 2012.

[6] D. V. Dimarogonas and K. J. Kyriakopoulos, "A connection between formation infeasibility and velocity alignment in kinematic multi-agent systems," *Automatica*, vol. 44, no. 10, pp. 2648–2654, 2008.

[7] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control.*, vol. 48, no. 6, pp. 988–1001, 2002.

[8] Z. Lin, Zhiyun, B. Francis, and M. Maggiore, "Necessary and sufficient graphical conditions for formation control of unicycles," *IEEE Trans. Autom. Control.*, vol. 50, no. 1, pp. 121–127, 2005.

[9] R. Olfati-Saber and R. Murray, "Consensus Problems in Networks of Agents With Switching Topology and Time-Delays," *IEEE Trans. Autom. Control.*, vol. 49, no. 9, pp. 1520–1533, 2004.

[10] A. Roza, M. Maggiore, and L. Scardovi, "A Smooth Distributed Feedback for Formation Control of Unicycles," *IEEE Trans. Autom. Control.*, vol. 64, no. 12, pp. 4998–5011, 2019.

[11] G. Wen, Z. Duan, W. Ren, and G. Chen, "Distributed consensus of multi-agent systems with general linear node dynamics and intermittent communications," *Int. J. Robust Nonlin.*, vol. 24, no. 5, pp. 2438–2457, 2014.

[12] M. Cao, C. Yu, and B. D. O. Anderson, "Formation control using range-only measurements," *Automatica*, vol. 47, no. 4, pp. 776–781, 2011.

[13] K.-K. Oh and H.-S. Ahn, "Formation control of mobile agents based on inter-agent distance dynamics," *Automatica*, vol. 47, no. 10, pp. 2306–2312 ,2011.

[14] D. V. Dimarogonas and K. H. Johansson, "Stability analysis for multi-agent systems using the incidence matrix: Quantized communication and formation control," *Automatica*, vol. 46, no. 4, pp. 695–700 ,2010.

[15] K.-K. Oh and H.-S. Ahn, "Distance-based undirected formations of single-integrator and double-integrator modeled agents in n-dimensional space," *Int. J. Robust Nonlin.*, vol. 24, no. 5, pp.1809–1820 ,2014.

[16] G. Ferrari-Trecate, A. Buffa, and M. Gati, "Analysis of Coordination in Multi-Agent Systems Through Partial Difference Equations," *IEEE Trans. Autom. Control.*, vol. 51, no. 6, pp. 1058–1063, 2006.

[17] P. Frihauf and M. Krstic, "Leader-Enabled Deployment Onto Planar Curves : A PDE-Based Approach," *IEEE Trans. Autom. Control.*, vol. 56, no. 8, pp. 1791–1806, 2011.

[18] T. Meurer and M. Krstic, "Finite-time multi-agent deployment : A nonlinear PDE motion planning approach," *Automatica*, vol. 47, no. 11, pp. 2534–2542, 2011.

[19] J. Qi, R. Vazquez, and M. Krstic, "Multi-Agent Deployment in 3-D via PDE Control," *IEEE Trans. Autom. Control.*, vol. 60, no. 4, pp. 891–906, 2015.

[20] J. Qi, S. X. Tang, and C. Wang, "Parabolic PDE-based multi-agent formation control on a cylindrical surface," *Int. J. Control*, vol. 92, no.1, pp. 77–99, 2019.

[21] J. Qi, J. Zhang, and Y. Ding, "Wave Equation-Based Time-Varying Formation Control of Multiagent Systems," *IEEE Trans. Control Syst. Technol.*, vol. 26, no.5, pp. 1578–1591, 2018.

[22] X. Dong, Y. Zhou, Z. Ren, and Y. Zhong, "Time-Varying Formation Tracking for Second-Order Multi-Agent Systems Subjected to Switching Topologies With Application to Quadrotor Formation Flying," *IEEE Trans. Ind. Electron.*, vol. 64, no. 6, pp. 5014–5024, 2017.

[23] H. Sano, "Feedback stabilization of one-dimensional parabolic systems related to formations," *Bull, Pol, Ac:Tech*, vol. 63, no. 1, pp. 295–303, 2015.

[24] J.-W. Wang, "A PDE-based Approach to Formation Control Design for a Large Vehicular Platoon," *Chinese Automation Congress*, pp. 1129–1134, 2015.

[25] G. Freudenthaler and T. Meurer, "PDE-based tracking control for multi-agent deployment," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 582–587, 2016.

[26] Q. I. Jie, and P. Feng, and Q. I. Jinpeng, "A PDE approach to formation tracking control for multi-agent systems," in *Proc. of Chinese Control Conference*, 2015, pp. 7136–7141.

[27] J. Wei, E. Fridman, and K. H. Johansson, "A PDE approach to deployment of mobile agents under leader relative position measurements," *Automatica*, vol. 106, pp. 47–53, 2019.

[28] S. Marx and E. Cerpa, "Output feedback stabilization of the Korteweg-de Vries equation," *Automatica*, vol. 87, pp. 210–217, 2018.