Stochastic Self-organizing Control for Swarm Robot Systems

Daisuke Inoue(✉), Daisuke Murai, and Hiroaki Yoshida

Toyota Central R&D Labs., Inc., Nagakute, Aichi, Japan
{daisuke-inoue,Daisuke-Murai,h-yoshida}@mosk.tytlabs.co.jp
https://www.tytlabs.com/

Abstract. In swarm robot systems, forming a target shape with autonomously moving robots is an important task. Considering cost and scalability, it is desirable that the observation information required by the robots to form patterns be minimal, whereas the patterns themselves can be as complicated as needed. In this paper, we propose a method of achieving this task under the situation that a scalar value representing a clue to its position is the only information that each robot can observe. We adopted the optimization method proposed by Mesquita et al. [International workshop on hybrid systems: Computation and control, pp. 358 (2008)] as a control method for the swarm robot systems. This method requires neither centralized controllers nor position identification of each robot, and we thus refer to it as "self-organizing control." Compared with existing control methods, the proposed method reduces memory usage and computational complexity. By means of both numerical simulations and experiments with actual robots, we quantitatively confirmed that self-organization was achieved.

Keywords: Swarm robotics · Self-organization · Multi-agent systems

1 Introduction

Swarm robot systems aim at performing complicated tasks with large groups of robots equipped with relatively simple hardware [2,3,13]. Such systems are attracting much attention because of their flexibility, environmental adaptability, and robustness against failure [5,15]. One task proposed for robot swarms is the formation of a target shape while each robot moves autonomously. The task includes several fundamental activities, including *consensus*, in which robots gather in one place, and *coverage*, in which robots are scattered all over a certain field [8,12,17]. In this task, the complexity of the shape that the robots can form depends on the amount of observations made by each robot. For example, when each robot is able to directly obtain its position and orientation using GPS and a gyro sensor, controlling movement towards the target coordinates is easy. However, when a large number of robots is used, it is costly to equip each robot

Y. Tan et al. (Eds.): ICSI 2019, LNCS 11655, pp. 405–416, 2019.
https://doi.org/10.1007/978-3-030-26369-0_38

with sensors, and the lack of sensors makes shape formation very difficult. To circumvent such problems, we studied a method for pattern formation without any individual robot knowing its own position directly; that is, a *self-organizing control* method. Instead of having robots obtain information about their positions, we assumed a situation where each robot can obtain a scalar value representing "the desirability of the current position." One simple example of the scalar value is a light intensity that a robot can observe. In this example, robots gather more densely where the light is strong and more sparsely where the light is weak. Although such a problem setting is similar to a global control problem, in that scalar values are transmitted to the entire group, the situation considered here is more realistic when we consider a massive number of robots, because sending position and orientation directly become less obvious.

In this context, the paper by Mesquita et al. [10] reports a remarkable study. They proposed a method to maximize a certain class of the function $Q : \mathbb{R}^d \to \mathbb{R}$ ($d = 2$ or 3) while letting a large number of virtual agents search in $\mathbb{R}^d$-space according to a certain rule R. As a result of running the agent with this rule, it was shown that the density distribution of the agents converged to the given function Q. Then, the user can know the maximum value of the function, at which the agents exist most densely, after some time elapses. Therefore, in Ref. [10], the rule of each robot's motion was introduced as a means of optimization.

In this study, we applied the method proposed in Ref. [10] to impart self-organizing control to a swarm robots in real space. In other words, by applying the agent paradigm to robots, Q becomes the position clue and R becomes the rule for robot movement, and the density distribution formed by the robots is expected to converge to Q. Because the rule R is probabilistic, we call this method *stochastic self-organizing control*. As compared with conventional methods often used for coverage control in multi-agent systems, such as the ones in Refs. [1,6,9], the present stochastic self-organizing control has the following features:

(1) The robots stochastically remain moving, even in a steady state.
(2) The simple algorithms have low computational complexity.
(3) The only necessary information is the value of Q (without self-positions and orientation).

Although (1) is seemingly problematic from the viewpoint of energy consumption, it can provide performance superior to that of deterministic control when there are differences in robot abilities, because the continuous motion keeps robots exchanging their positions within a certain area. Items (2) and (3) mean that the robots can be controlled effectively even when they have small memory and few sensors, respectively; in other words, the proposed method is applicable to a simple, small robot.

The rest of this paper is organized as follows. In Sect. 2, we formulate a shape formation problem and describe the stochastic self-organizing control method. In Sects. 3 and 4, we perform a large-scale simulation and an actual machine

experiment. In Sect. 5, we evaluate the performance of the proposed method and discuss future directions for algorithm improvement.

2 Stochastic Self-organizing Control

2.1 Problem Formulation

Consider N mobile robots on $\mathbb{R}^2$-space. Assume that each robot has a unique identifier from 1 to N. The position and velocity of robot i at time $t \in \mathbb{R}_+$ ($\mathbb{R}_+ := \{a \mid a \in \mathbb{R}, a \geq 0\}$) are denoted by $x_i(t) \in \mathbb{R}^2$ and $v_i(t) \in \mathbb{V}$, respectively, where $\mathbb{V}$ denotes the velocity space of each robot. We make the following assumptions for each robot:

1. $\mathbb{V} = \rho\mathbb{S}$, where $\rho \in \mathbb{R}_+$ is a constant and $\mathbb{S}$ is the unit sphere equipped with a Lebesgue measure $d\mu$. This means that each robot always moves at a constant speed of $\rho > 0$.
2. The robot continuously selects either straight-ahead motion or rotational motion. This method of movement is called *run and tumble*.
3. We define the set $\mathcal{D}$ as $\mathcal{D} := \{f \in L^1(\mathbb{R}^2) \mid f > 0,\ \int_{\mathbb{R}^2} f(x)dx = 1\}$, and consider $Q \in \mathcal{D}$. Each robot is *not* able to obtain its own position $x_i(t)$ and orientation, but it can measure the value of the function $Q(x_i(t))$.
4. All robots are assumed to move according to the same algorithms. It is also assumed that robots do not communicate with each other, and their density is sufficiently small to avoid collisions. At this time, because the motion of each robot can be regarded as independent, in the following description, the position and speed of the robot are represented by $x(t)$ and $v(t)$, respectively; that is, index i is dropped.

We define $p(x, v, t)$ as the probability density of finding a robot at position x with velocity v at time t. We assume that for each fixed time t, $p(x, v, t) \in L^1(\mathbb{R}^2 \times \rho\mathbb{S})$ holds, when $\mathbb{R}^2 \times \rho\mathbb{S}$ is provided with the product measure $dx \otimes d\mu$. The purpose of the considered control problem is to ensure that

$$\lim_{t\to\infty} \int_{\rho\mathbb{S}} p(x, v, t)d\mu(v) = Q(x). \tag{1}$$

is satisfied by designing rule R. This rule determines the speed $v(t)$ depending on the history of values of the scalar function $\{Q(x(\tau)) \mid \tau \in [s, t],\ s \leq t\}$ observed by each robot.

2.2 Controller Design

Mesquita et al. [10] proposed a method for solving an optimization problem by finding the maximum of a function, say $Q(x)$. This method relies on stochastically selecting the rotational motion of an agent. Here we apply it to achieve our goal in Eq. (1). The probability that a tumble does *not* occur between the time instants t and s is given by

$$\exp\left(-\int_s^t \lambda(x+\tau v, v)d\tau\right). \tag{2}$$

where $\lambda : \mathbb{R}^2 \times \rho\mathbb{S} \to \mathbb{R}$ is the *tumbling rate*, λ is a design parameter, and $\lambda \in L^\infty(\mathbb{R}^2 \times \rho\mathbb{S})$ holds. At each tumble, the velocity changes to a random value $\bar{v} \in \rho\mathbb{S}$ with probability density T_{v^-}, which may depend on the velocity v^- just before the tumble. This is represented as $\bar{v} \sim T_{v^-}$. T_{v_-} is also a design parameter. In summary, we obtain

$$\begin{aligned}
&\dot{x}(t) = v(t),\\
&\dot{v}(t) = 0,\\
&v(t) = \begin{cases} \bar{v},\ \bar{v} \sim T_{v^-} & \text{w. p. } 1 - \exp\left(-\int_s^t \lambda(x+\tau v, v)d\tau\right),\\ v^- & \text{w. p. } \exp\left(-\int_s^t \lambda(x+\tau v, v)d\tau\right). \end{cases}
\end{aligned} \tag{3}$$

where the third equation corresponds to rule R.

In Ref. [14], an equivalent expression of Eq. (3), which is a partial differential equation describing the time evolution of the density distribution of the variables, is derived as follows:

$$\frac{\partial p}{\partial t} + v \cdot \nabla_x p(x, v, t) = -\lambda(x, v)p(x, v, t) + \int_{\rho\mathbb{S}} T_{v'}\lambda(x, v')p(x, v', t)d\mu(v'). \tag{4}$$

Equation (4) is interpreted intuitively as follows: on the left-hand side, we find a drift term $v\cdot\nabla_x p(x, v, t)$ corresponding to straight robot movement. On the right-hand side, we find a loss term $-\lambda p(x, v, t)$ that corresponds to robots leaving state (x, v) and a gain term that corresponds to robots transitioning to the next state (x, v).

In [10], λ and T_{v^-} are designed for solving an optimization problem of maximizing $Q(x)$ by means of analyzing Eq. (4). This is expressed as the following theorem:

Theorem 1. **([10]).** *Suppose that Q satisfies $Q \in \mathcal{D}$ and $\|\nabla_x \ln Q(x)\| \in L^\infty(\mathbb{R}^2)$. If $T_{v'}$ is designed as a uniform distribution as*

$$T_{v'}(v) = \frac{1}{\mu(\rho\mathbb{S})}. \tag{5}$$

and λ is designed as

$$\lambda(x, v) = \eta(x) - v \cdot \nabla_x \ln Q(x). \tag{6}$$

where $\eta(x) = \int_{\rho\mathbb{S}} T_{v'}(v)\lambda(x, v')d\mu(v')$, then, for any $p(x, v, t)$ that satisfies $p(x, v, 0) \in \mathcal{D}$ and Eq. (4),

$$p(x, v, t) \to Q(x), \quad (t \to \infty). \tag{7}$$

holds in norm.

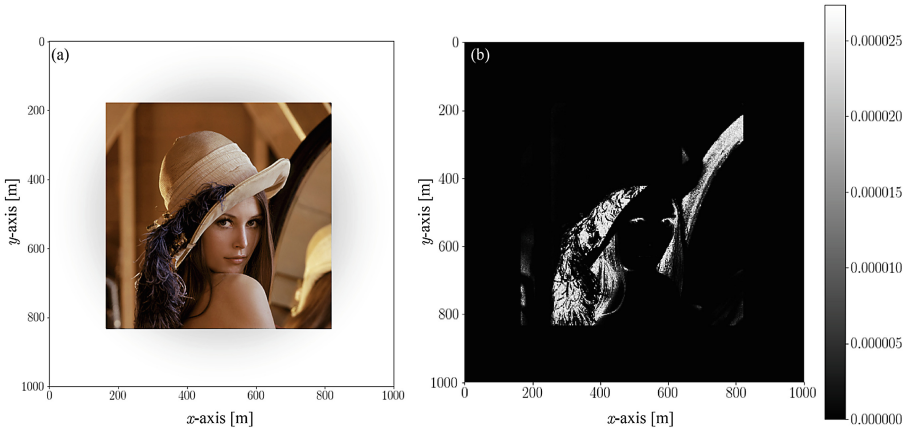

Fig. 1. Scalar value function Q used for numerical calculation. (a): Image of Lena placed in the space $[0,\ 1,000]^2\,\mathrm{m}^2$. (b): Image converted to scalar function Q after processing by grayscaling, high contrasting, and normalization.

In Theorem 1, we regard $\eta(\cdot)$ in Eq. (6) as a design parameter and let it be a constant $\bar{\eta} \in \mathbb{R}_+$. This means that the designer chooses the average tumbling rate of each robot in advance. Note that because of this the convergence of the distribution, which is described in Theorem 1, will *not* rigorously hold. The probability of Eq. (2) is then calculated as

$$\begin{aligned} \exp\left(-\int_s^t \lambda(x(\tau), v(\tau))d\tau\right) &= \exp\left(-\int_s^t \bar{\eta} - v \cdot \nabla_x \ln Q(x(\tau))d\tau\right) \\ &= \exp(-\bar{\eta}(t-s))\frac{Q(x(t))}{Q(x(s))}. \end{aligned} \tag{8}$$

In Eq. (8), the size of eta serves as a parameter for controlling the rotation probability as time passes. By using Eq. (8), we can control each robot with the following steps:

1. Set $s \leftarrow t_0$ at the initial time $t = t_0$.
2. Calculate the value $\bar{p} \in [0, 1)$ of Eq. (8) at each time t.
3. Generate a uniform random number in the $[0, 1]$ section. If it is larger than $\bar{p}$, rotate with an angle sampled from a uniform distribution of $\rho\mathbb{S}$ and reset the value of s ($s \leftarrow t$). If it is smaller than $\bar{p}$, continue traveling straight ahead (no turning).
4. Repeat steps 2 and 3.

As guaranteed by Theorem 1, the distribution $p(x, v, t)$ of the robot is expected to converge to the given function $Q(x)$ using the above steps.

3 Numerical Simulation

We carried out numerical simulation using a large number of robots, namely $40,000$. Each robot moves according to the dynamics of Eq. (3), where the speed

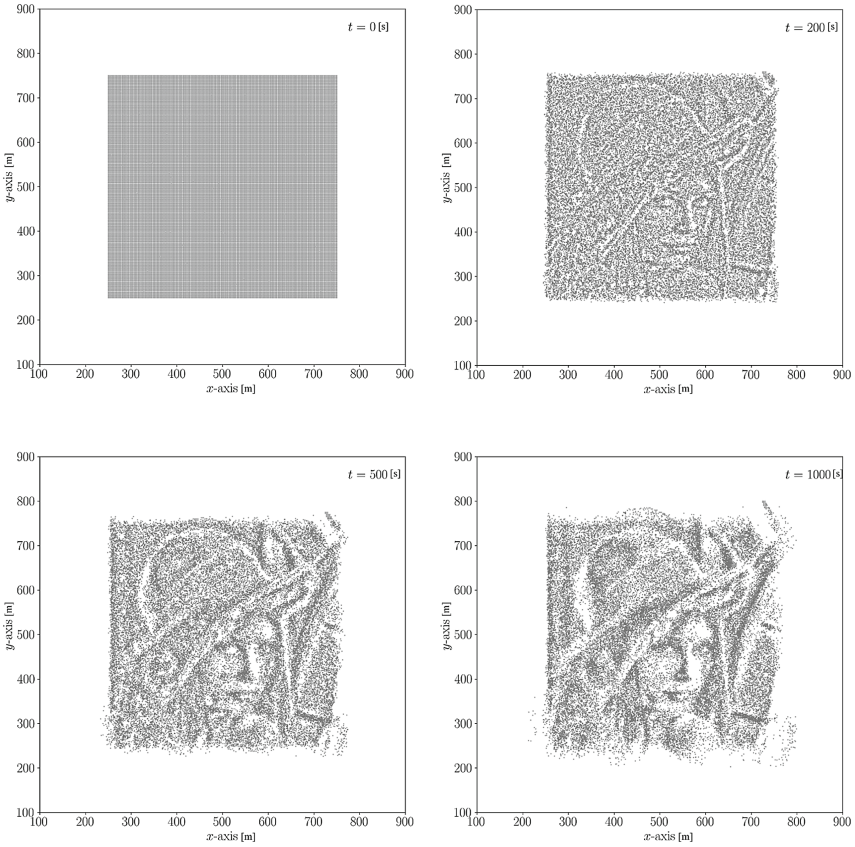

Fig. 2. Trajectory of the movement of 40,000 robots.

of the robot ρ is 0.05 m/s. In the actual computation, Eq. (3) is discretized using Euler approximation at a sampling interval of 1.0 s. The value of $\bar{\eta} = 0.1\,\mathrm{s}^{-1}$ is used as a parameter representing the average rotation rate in Eq. (8). The initial positions of the robot are assumed to be equally spaced in the section $[250,\ 750]^2\,\mathrm{m}^2$. For the scalar value function Q, we used the often used Lena [16] test image, to correspond to the section $[0,\ 1000]^2\,\mathrm{m}^2$, as shown in Fig. 1(a). To convert the original image into a suitable density distribution, we performed the following procedure:

1. Grayscaling: We converted the image to monochrome, and expressed it as scalar value function $\psi : \mathbb{R}^2 \to \mathbb{R}$.

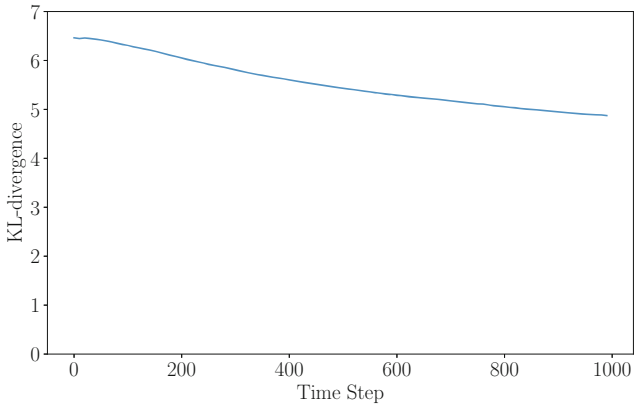

Fig. 3. KL divergence obtained by numerical integration at each time step.

2. High contrasting: To increase the kurtosis of the distribution, we applied the following nonlinear transformation (for details, see Ref. [7]):

$$\phi(q) = \exp(30(\psi(q) - 1)). \tag{9}$$

where $q \in \mathbb{R}$ represents each pixel value of the image.
3. Normalization: The value of each pixel was normalized by the sum of the values of all the pixels, yielding a density distribution.

Figure 1(b) shows a plot of the scalar function Q after performing these operations.

We used the procedure described above to simulate the movement of 40,000 robots, and their resulting trajectories are plotted in Fig. 2. We confirmed that the robots, which were uniformly aligned at the initial time, tended to self-organize into a form of the scalar value function Q representing the image. To evaluate the convergence of the distribution quantitatively, we introduced the following Kullback-Leibler (KL) divergence $d_{\mathrm{KL}}(t; p, Q)$:

$$d_{\mathrm{KL}}(t; p, Q) := \int_{\mathbb{R}^2} \int_{\rho\mathbb{S}} \left(p(x, v, t) \ln \frac{p(x, v, t)}{Q(x)} \right) dx d\mu(v). \tag{10}$$

This is a measure of how the probability distribution p is different from the reference probability distribution Q. Because $d_{\mathrm{KL}}(t; p, Q) \to 0$ $(t \to \infty)$ holds *iff* $p \to Q$ holds, we can regard the smallness of $d_{\mathrm{KL}}(t; p, Q)$ as an evaluation of the proximity of the distributions. Figure 3 shows a time evolution of the value of d_{KL}, obtained by numerically integrating Eq. (10) within the space $[0,\ 1,000]^2\ \mathrm{m}^2$ discretized into a grid with 10,000 cells. The KL divergence, which was 6.46 at the initial time $t = 0\,\mathrm{s}$, decreased to 4.87 at $t = 10\,\mathrm{s}$, which means that the density distribution $p(x, v, t)$ formed by the robots approached the scalar value function $Q(x)$.

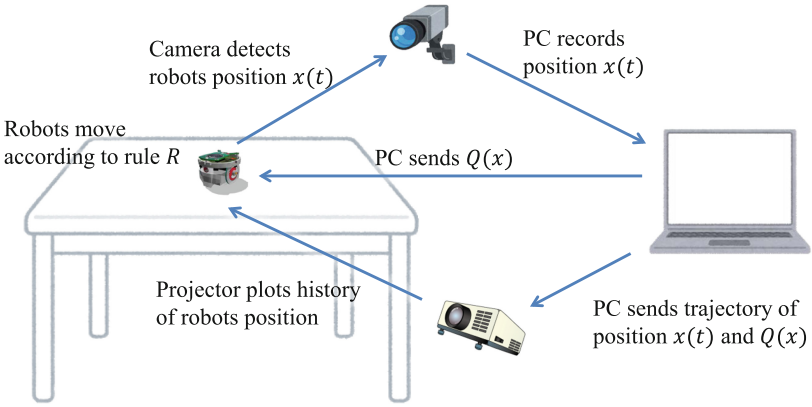

Fig. 4. Experimental setup.

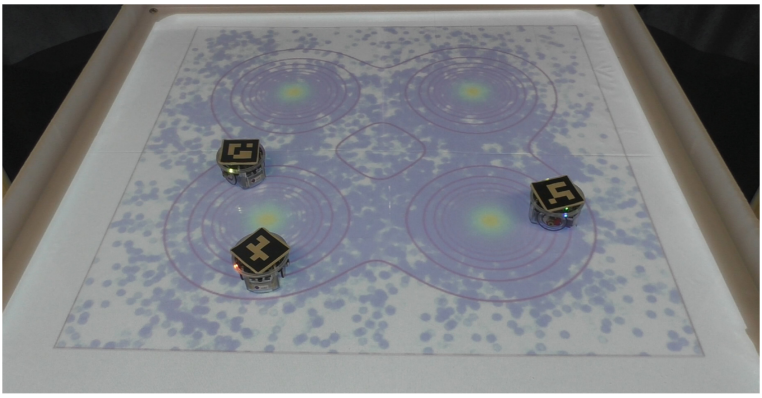

Fig. 5. Robots moving on the stage along with their projected trajectory.

4 Experiment with Robots

In addition to the numerical simulations, we also carried out experiments using actual robots. We employed e-puck 2, a research and educational robot developed by Michael and Francesco [11]. The robot's diameter is as small as 7 cm, and it has a programmable microcontroller. It moves by operating two stepping motors independently, and it carries many sensors, such as infrared proximity sensor, inertial measurement unit, time-of-flight distance sensor, and camera. In addition, these robots can communicate with each other robot or a master PC via Wifi or Bluetooth interface.

Our experimental setup is shown in Fig. 4. Each robot moved on a stage measuring 1×1 m according to the steps described in Sect. 2.2 with the value of Q at its own position. Also, for visualization, a projector displayed the locus of the position of the robots. Figure 5 shows the robots moving on the stage and the

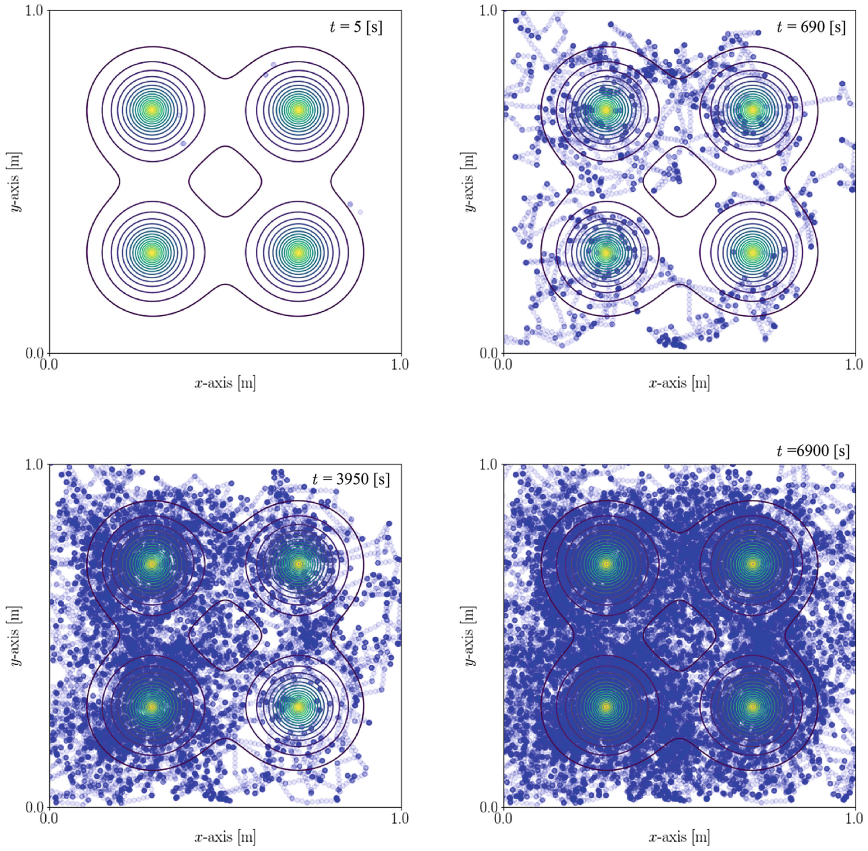

Fig. 6. Trajectory of movement of three e-puck robots.

displayed trajectories. In this experiment, the robots did not sense the function Q; instead, the master PC sent the information to each robot. More specifically, the master PC used the camera above the stage to detect markers applied to the top of each robot and then calculated the value of Q from the marker position. The value of Q was then transmitted to each robot via Bluetooth communication. The reason for employing this procedure to obtain the value of Q, rather than directly sensing it, was to separate the evaluation of the performance of the proposed control method from the evaluation of the sensing performance of the robot. Because the number of Bluetooth simultaneous communication devices was limited, three robots were used in the experiment. In the numerical simulation described in Sect. 3, convergence of the distribution was evaluated by means of a joint distribution of many robots. In this experiment, convergence of the distribution was verified by observing the distribution formed by the trajectory of the position of each robot. This means that we assumed that Ergodicity is established in the system.

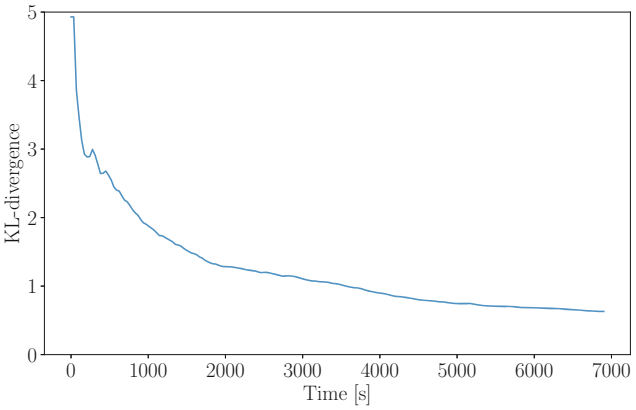

Fig. 7. KL divergence obtained by numerical integration at each time step.

The speed of movement of each robot ρ was 0.02 m/s, and control was performed with the sampling period 0.6 s. For the parameter $\bar{\eta}$, we used the value $0.6\,\mathrm{s}^{-1}$. Unlike the numerical simulation, each robot can collide with walls or other robots because the robots have a finite size. To avoid collisions, we implemented an avoidance function that detects an obstacle using infrared sensors and changes the traveling direction in the direction opposite to the obstacle. Consequently, strictly speaking, the movement algorithm is different from the one described in Sect. 2.2. It was thus expected that the collision avoidance negatively effects the distribution convergence. We discuss this in more detail in Sect. 5.

In Fig. 6, the multimodal function used as Q and the trajectories of the position of robots are visualized. We confirmed that the trajectories of the three robots approached the shape of the given scalar value function Q as time proceeded. Figure 7 shows the plot of the KL divergence (10) at each time step t, which was obtained by dividing the 1×1 m space of the stage into 14,400 cells and numerically integrating the values on them. The KL divergence, which was 4.92 at the initial time $t = 0$ s, decreased to 0.63 at $t = 6,900$ s, and it was quantitatively confirmed that the density distribution $p(x, v, t)$ formed by the robots approached the scalar value function $Q(x)$ after sufficient time had elapsed.

5 Discussion and Concluding Remarks

Here, we compare our proposed method with the method proposed by Cortés et al. [6], which is often used for coverage control. Cortés proposed the following method for moving the robots to minimize an evaluation function:

$$\dot{x}_i(t) = -K\left[x_i(t) - g(\mathcal{V}_i(x(t)))\right]. \tag{11}$$

$$g(\mathcal{V}_i(x(t))) := \frac{\int_{\mathcal{V}_i(x(t))} zQ(z)dz}{\int_{\mathcal{V}_i(x(t))} Q(z)dz}. \tag{12}$$

where $K \in \mathbb{R}_+$ is the gain of the controller and $\mathcal{V}_i(x)$ is called the *Voronoi region*, defined as $\mathcal{V}_i(x) := \{z \in \mathbb{R}^2 | \|z - x_i\| \leq \|z - x_i\| \ \forall j \text{ s.t. } j \neq i\}$. Here, the evaluation function represents the closeness between the distribution formed by the robot swarm and the target shape.

Compared with [6], the computational cost of the proposed method is small. In Ref. [6], integration of the function on the Voronoi region was necessary to obtain the value of g in Eq. (12). In contrast, our method is implemented by only calculating Eq. (8) and a uniform random number at each time step. In addition, in the proposed method, less information is necessary than in the method of Cortés, where the robots need to know their own position and those of neighboring robots to compute their Voronoi regions at each time step. Finally, while Eq. (11) assumes that the robots are able to go straight to the target coordinates, which usually requires them to have their own orientation information, in the proposed method, only the value of $Q(x)$ is used to carry out run and tumble with only the random angle rotation at tumble phases; robots do not need information on their own positions and orientations.

In the proposed method, however, because of its stochastic nature, the probability of collision between robots is expected to be higher than that in the Cortés' method. It is thus necessary for each robot to have a function to avoid collision with other robots, as implemented in Sect. 4. The collision avoidance is regarded as an interaction force acting on the robot. At this time, a new term representing the proximity interaction is added to Eq. (3) as follows:

$$\begin{aligned} \dot{x}_i(t) &= v_i(t), \\ \dot{v}_i(t) &= f_{ij}(t), \\ v_i(t) &= \begin{cases} \bar{v},\ \bar{v} \sim T_{v_i^-} & \text{w. p. } 1 - \exp\left(-\int_s^t \lambda(x_i + \tau v_i, v_i) d\tau\right), \\ v_i^- & \text{w. p. } \exp\left(-\int_s^t \lambda(x_i + \tau v_i, v_i) d\tau\right). \end{cases} \end{aligned} \tag{13}$$

where $f_{ij}(t)$ is the interaction term expressing the repulsive force between robot i and robot j. As a result of this term, Eq. (3) is *not* independent for each robot, so that the dimension of the corresponding partial differential equation (4) is N. In the field of statistical mechanics, *mean field approximation*, which averages interactions with surrounding particles and performs approximation, is often applied to decrease the dimension of the system, by means of an appropriate enclosure. However, for proximity interaction, as in our case, the mean field approximation may not be valid, and a different approach is necessary to decrease the dimension. Recently, Bruna et al. [4] proposed an approximation method for systems with proximity interactions, and we are now working on developing a new algorithm incorporating this approximation approach.

Acknowledgement. The authors would like to thank Dr. Yuji Ito for the useful discussions.

References

1. Bandyopadhyay, S., Chung, S.J., Hadaegh, F.Y.: Probabilistic and distributed control of a large-scale swarm of autonomous agents. IEEE Trans. Rob. **33**(5), 1103–1123 (2017)
2. Barca, J.C., Sekercioglu, Y.A.: Swarm robotics reviewed. Robotica **31**(3), 345–359 (2013). https://doi.org/10.1017/S026357471200032X
3. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intell. **7**(1), 1–41 (2013)
4. Bruna, M., Chapman, S.J., Robinson, M.: Diffusion of particles with short-range interactions. SIAM J. Appl. Math. **77**(6), 2294–2316 (2017)
5. Cao, Y.U., Fukunaga, A.S., Kahng, A.: Cooperative mobile robotics: antecedents and directions. Auton. Robots **4**(1), 7–27 (1997)
6. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. Robot. Autom. **20**(2), 243–255 (2004)
7. Izumi, S., Azuma, S., Sugie, T.: Distributed hybrid controllers for multi-agent mass games by a variable number of player agents. Asian J. Control **17**(3), 762–774 (2015). https://doi.org/10.1002/asjc.930
8. Martinez, S., Cortes, J., Bullo, F.: Motion coordination with distributed information. IEEE Control Syst. **27**(4), 75–88 (2007)
9. Meng, Y., Guo, H., Jin, Y.: A morphogenetic approach to flexible and robust shape formation for swarm robotic systems. Robot. Auton. Syst. **61**(1), 25–38 (2013). https://doi.org/10.1016/j.robot.2012.09.009
10. Mesquita, A.R., Hespanha, J.P., Åström, K.: Optimotaxis: a stochastic multi-agent optimization procedure with point measurements. In: Egerstedt, M., Mishra, B. (eds.) HSCC 2008. LNCS, vol. 4981, pp. 358–371. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78929-1_26
11. Mondada, F., et al.: The E-puck, a robot designed for education in engineering. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, vol. 1, pp. 59–65. IPCB: Instituto Politécnico de Castelo Branco (2009)
12. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. Proc. IEEE **95**(1), 215–233 (2007)
13. Şahin, E.: Swarm robotics: from sources of inspiration to domains of application. In: Şahin, E., Spears, W.M. (eds.) SR 2004. LNCS, vol. 3342, pp. 10–20. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30552-1_2
14. Stroock, D.W.: Some stochastic processes which arise from a model of the motion of a bacterium. Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete **28**(4), 305–315 (1974). https://doi.org/10.1007/BF00532948
15. Tan, Y.: Handbook of Research on Design, Control, and Modeling of Swarm Robotics, 1st edn. IGI Global, Hershey (2015)
16. Wakin, M.: Lena (2005). https://www.ece.rice.edu/~wakin/images/
17. Xie, G., Wang, L.: Consensus control for a class of networks of dynamic agents. Int. J. Robust Nonlinear Control: IFAC-Affiliated Journal **17**(10–11), 941–959 (2007)