Received December 9, 2020, accepted December 18, 2020, date of publication December 30, 2020, date of current version January 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3048157

Mass Game Simulator: An Entertainment Application of Multiagent Control

SHINSAKU IZUMI, (Member, IEEE), YUTO SHIOMOTO, AND XIN XIN, (Senior Member, IEEE)
Faculty of Computer Science and Systems Engineering, Okayama Prefectural University, Okayama 719-1197, Japan

Corresponding author: Shinsaku Izumi (izumi@cse.oka-pu.ac.jp)

This work was supported in part by the Japan Society for the Promotion of Science, Grants-in-Aid for Scientific Research, Japan, under Grant 19K15016.

ABSTRACT This paper proposes and develops a simulator of multiagent mass games in which agents achieve formations displaying given grayscale images via distributed control. One potential application of mass games is entertainment for ordinary users. However, it is difficult for nonexpert users to understand the existing control methods for mass games, and the implementation requires considerable time and effort. The proposed mass game simulator developed to overcome these difficulties is provided as a MATLAB application. With this simulator, we can import a reference image, simulate a mass game for the reference image, confirm the simulation result, and save the result through simple mouse and keyboard inputs without the need for expert knowledge or programming effort. In addition, the application of the proposed simulator to entertainment and digital signage is discussed and evaluated. In particular, in the latter application, by playing an animation of the simulation results obtained using the simulator on digital signage, we can construct a novel type of signage whose contents are shown by the motion of individual dots corresponding to agents. These results indicate that control of multiagent systems can be used for nonengineering applications, which have not been considered in most existing studies.

INDEX TERMS Distributed control, entertainment, formation control, mass games, multiagent systems, simulators.

I. INTRODUCTION

A. MOTIVATION AND CONTRIBUTIONS

In the systems and control field, there has been considerable interest in the control of multiagent systems to allow multiple agents interconnected through a network to achieve a global objective using only local information.

The interest in multiagent systems is motivated by the swarm behavior of living things. For example, a flock of birds and a school of fish travel as groups by observing their neighbors for defense against predators and the protection of their territory [1]. Another motivation for studying multiagent systems is the variety of potential engineering applications, *e.g.,* formation flying of aerial vehicles [2], [3], sensor networks [4], [5], and smart grids [6], [7]. Moreover, in recent years, the Internet of things (IoT), *i.e.,* the growing set of devices connected to the Internet, has garnered considerable attention [8]. This scenario is also an application example of multiagent systems, where the devices correspond to agents.

In addition to such engineering applications, multiagent systems have the following potential advantages. First, the systems can perform tasks that cannot be achieved by single-agent systems. Examples include environmental monitoring [9] and cooperative transportation [10] using multiple agents, state estimation via data fusion [11], [12], and distributed optimization over networks [13]. Second, multiagent systems are robust against failures. Even if the performance of some agents degrades due to failures, the entire system can continue to operate [14].

In this work, we focus on formation control in multiagent systems to enable agents to achieve a desired formation in a distributed manner. Specifically, the problem considered here is to find distributed controllers for conducting *mass games* in which agents achieve formations displaying given *grayscale images*, as illustrated in Fig. 1. One potential application of mass games is *entertainment*, where we can observe agent formations displaying objects shown in pictures. Such an

The associate editor coordinating the review of this manuscript and approving it for publication was Choon Ki Ahn.

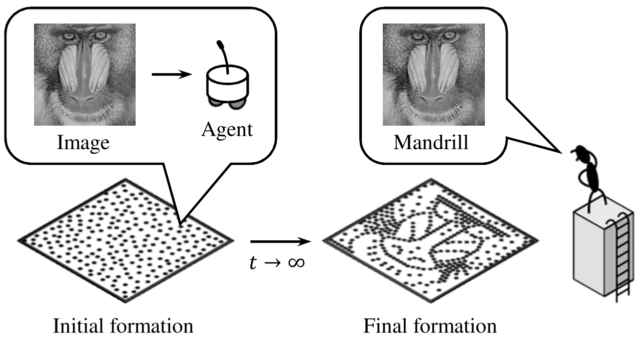


FIGURE 1. **A multiagent mass game.**

application is not considered in most existing studies on multiagent systems because these studies have been motivated by biological interest and engineering applications, as explained above. In addition, the behavior of agents in constructing formations with interesting appearances can attract the attention of ordinary users to multiagent control and can enhance their motivation in learning its theory.

The mass game problem was originally proposed by Azuma *et al.* [15] (including the first author). They developed distributed controllers for conducting mass games by combining coverage control[1] [16] and halftone image processing[2] [17]. Subsequently, in [18], the authors considered an additional mass game problem in which there is a variable number of agents participating in the mass game, and they extended the controllers developed in [15] to this problem. However, the extension of mass games to entertainment applications for ordinary users faces two major problems. First, it is difficult for nonexpert users to understand the theory of mass games developed in [15], [18]. As mentioned above, the central idea underlying mass game controllers is the combination of coverage control and halftone image processing, *i.e.,* techniques used in the fields of control theory and image processing, respectively. Few nonexpert users will be familiar with both of these distinct fields. Second, the implementation of the algorithm for simulating a mass game requires considerable time and effort. For example, it requires the computation of an integral of a function over a region, with the function and region being specified by the given grayscale image and the current configuration of agents, respectively. However, typical programming languages are not equipped with dedicated functions for such computation; thus, the computation must be implemented by each user.

Given these problems, this work was conducted to develop a *mass game simulator (MGS)*, *i.e.,* an application allowing nonexpert users to easily simulate a mass game without the need for expert knowledge or programming effort. Our main contributions are summarized as follows.

(i) Based on the control method given in [15], we develop the MGS as a MATLAB application. The MGS can be used to simply simulate a mass game by means of mouse and keyboard operations. In addition to this basic functionality, we introduce a zoom function to confirm the detailed behavior of agents and a save function to save the simulation results as image and movie files. The former aids users in understanding the idea underlying the mass game controllers, and the latter broadens the application range of the MGS, as demonstrated later in this paper.

(ii) We present two application examples of the MGS: an entertainment application for ordinary users and a digital signage application. In the former application, users can not only see the behavior of agents in constructing formations with interesting appearances but also know how to control the agents. In the latter application, by applying an animation of the simulation results obtained using the MGS to digital signage, we can construct a novel type of signage whose contents are shown by the motion of individual dots corresponding to agents. This approach is illustrated through an example of digital signage placed at the entrance of a room. These results indicate that the control of multiagent systems is applicable to nonengineering fields, which have not been considered in most existing studies.

(iii) In the digital signage application, we evaluate the quality of the contents displayed in the signage via comparison with an existing method [28] for the formation control of multiagent systems. The results demonstrate that the displayed contents generated by the MGS are better than those generated by the existing method in terms of both visual quality and time spent producing the desired formation. One may think that this paper simply implements the control method proposed in [15], but the choice of the algorithm to implement is important in the development of simulators. The results indicate that our choice is suitable for the digital signage application.

B. RELATED WORK

To stress the novelty of this paper, we discuss the differences between this work and related work from the perspective of theory (*i.e.,* control methods of agents) and simulators. A summary of the following discussion is given in Table 1.

Many researchers have studied the formation control of multiagent systems. For example, Sugihara and Suzuki [19] proposed heuristic distributed algorithms for forming geometric patterns, and Suzuki and Yamashita [20] characterized the class of achievable patterns in a theoretical manner. Fax and Murray [21] analyzed the stability of the formation dynamics of agents using graph theory and proposed a method for information exchange among agents.

[1] A distributed control technique used to place agents so that their density distribution becomes the desired one.

[2] An image processing technique used to transform grayscale images into binary images while retaining as much of the visual quality of the original image as possible. In this process, the degree of gray coloring is expressed by the density distribution of black pixels.

TABLE 1. Relation between this work and related works.

	Formation control of multiagent systems			Others
	Desired formations			
Main focus	Simple formations specified by lines	Displaying images with a few colors	Displaying grayscale images	
Theory	[19]–[26]	[27], [28]	[15], [18]	$\cdots$
Simulators	[32]–[35]		This work	[29]–[31], [36]–[39]

Ren and Sorensen [22] developed a distributed formation control method based on a combination of a leader-follower approach and a consensus algorithm. Montijano *et al.* [23] proposed a vision-based formation control method for agents in three-dimensional space under the assumption that there is no external positioning system to globally localize the agents. Yang *et al.* [24] proposed a control method to achieve V-shaped formations inspired by the flocking behavior of flying geese. Wang *et al.* [25] and Liu *et al.* [26] addressed robust formation control problems for multiagent systems subject to external disturbances. However, the control methods proposed in these works are not applicable to the mass game problem because these works focused primarily on simple formations specified by lines, *e.g.,* circles, polygons, and V-shapes, and did not consider formations displaying grayscale images.

Concepts similar to mass games are discussed in [27], [28]. Alonso-Mora *et al.* [27] proposed a display in which mobile robots with controllable colors can be regarded as individual pixels and images and animations are shown via the positioning and motion of the robots. Wang and Rubenstein [28] developed a distributed control method to achieve formations that display letters and shapes given as images. However, it is difficult to apply the control methods proposed in these works to the mass game problem because the methods apply to either binary images or images with only a few colors, whereas it is necessary to represent the grayscale information (*i.e.,* various degrees of gray coloring) of images in the mass game problem. This difference is manifest in the digital signage application in Section IV-B.

Meanwhile, various simulators have been developed. A typical example is vehicle driving simulators [29], [30]. Another example is robot simulators for a single robot [31] and multiple robots [32]–[35]. There are also simulators of satellite attitude dynamics [36], tourist urban routes [37], stand-alone photovoltaic systems [38], and real-time pricing in smart grids [39]. However, these simulators cannot replace our MGS because the MGS handles not only the control method for mass games but also images and animations, whereas the existing simulators do not.

C. NOTATION

The following notation is used throughout this paper. Let $\mathbb{R}$ and $\mathbb{R}_+$ be the real number field and the set of positive real numbers, respectively. For the vectors $x_1, x_2, \ldots, x_n \in \mathbb{R}^2$ and the set $\mathbb{I} := \{i_1, i_2, \ldots, i_m\} \subseteq \{1, 2, \ldots, n\}$, we define $[x_i]_{i\in\mathbb{I}} := [x_{i_1}^\top \; x_{i_2}^\top \; \cdots \; x_{i_m}^\top]^\top \in \mathbb{R}^{2m}$. The cardinality of the set $\mathbb{S}$ is expressed as $|\mathbb{S}|$. Next, consider the convex set $\mathbb{Q} \subset \mathbb{R}^2$ and the distinct vectors $x_1, x_2, \ldots, x_n \in \mathbb{Q}$. For $x := [x_1^\top \; x_2^\top \; \cdots \; x_n^\top]^\top$ and each $i \in \{1, 2, \ldots, n\}$, $\mathbb{V}_i(x)$ denotes the Voronoi cell for x_i, *i.e.,*

$$\mathbb{V}_i(x) := \left\{ q \in \mathbb{Q} \,\middle|\, \|q - x_i\| \leq \|q - x_j\| \;\forall j \in \{1, 2, \ldots, n\} \right\}, \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm of a vector. In addition, we use $G(x)$ to represent the Delaunay graph for $x_1, x_2, \ldots, x_n$; that is, $G(x)$ is the undirected graph comprising the node and edge sets $\{1, 2, \ldots, n\}$ and $\left\{(i, j) \in \{1, 2, \ldots, n\}^2 \,\middle|\, \mathbb{V}_i(x) \cap \mathbb{V}_j(x) \neq \emptyset, i \neq j\right\}$, respectively.

II. MULTIAGENT MASS GAMES

Before presenting the proposed MGS, we briefly summarize the existing results [15] on mass games.

A. MASS GAME PROBLEM

Consider the multiagent system Σ in two-dimensional space depicted in Fig. 2. The system comprises n agents and the local controllers implemented in them.

The model of agent i ($i \in \{1, 2, \ldots, n\}$) is given by

$$\dot{x}_i(t) = u_i(t), \tag{2}$$

where $x_i(t) \in \mathbb{R}^2$ and $u_i(t) \in \mathbb{R}^2$ denote the position and control input of agent i, respectively. This is a model of an omnidirectional mobile robot.

We suppose that each agent operates in a distributed manner based on local information. The information available to each agent is specified as follows. Let $x(t) \in \mathbb{R}^{2n}$ be the positions of all agents, *i.e.,* $x(t) := [x_1^\top(t) \, x_2^\top(t) \; \cdots \; x_n^\top(t)]^\top$. Assuming that for every $t \in \mathbb{R}_+ \cup \{0\}$, all agents exist in a given bounded convex set $\mathbb{Q} \subset \mathbb{R}^2$, as shown in Fig. 2, we can represent the Voronoi cell for $x_i(t)$ ($i \in \{1, 2, \ldots, n\}$) and the corresponding Delaunay graph by $\mathbb{V}_i(x(t))$ and $G(x(t))$, respectively. Using this notation, we assume that for every $t \in \mathbb{R}_+ \cup \{0\}$, each agent i can obtain information on (i) its own position $x_i(t)$ and (ii) the positions of the neighboring agents on the Delaunay graph $G(x(t))$.

The local controller for agent i is then described by

$$u_i(t) = f(x_i(t), [x_j(t)]_{j\in\mathbb{N}_i(t)}), \tag{3}$$

where $x_i(t)$ and $[x_j(t)]_{j\in\mathbb{N}_i(t)} \in \mathbb{R}^{|\mathbb{N}_i(t)|}$ are the inputs, $u_i(t)$ is the output, $f : \mathbb{R}^2 \times \mathbb{R}^{2|\mathbb{N}_i(t)|} \to \mathbb{R}^2$ is a function determining

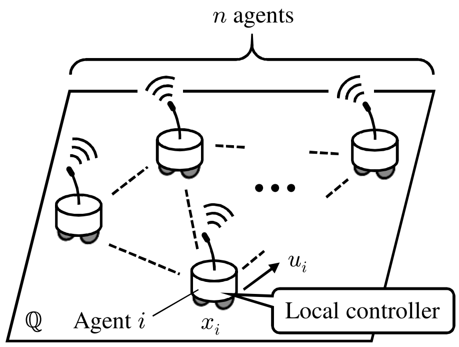


FIGURE 2. **Multiagent system Σ.**

the controller structure, and $\mathbb{N}_i(t) \subset \{1, 2, \dots, n\}$ is the index set of the neighboring agents on the Delaunay graph $G(x(t))$. This local controller is distributed in the sense that its inputs are restricted to the information described in (i) and (ii) above. Notably, the subscript i is not attached to the function f; this leads to the constraint that all local controllers must be common, which ensures the scalability of the overall system.

For the multiagent system Σ, we suppose that a reference grayscale image, which is displayed as a formation of the agents, as illustrated in Fig. 1, is given as the function $\varphi : \mathbb{Q} \to [0, 1]$. The function φ assigns the respective pixel values of the image to individual points q within the set $\mathbb{Q}$ introduced above, where we note that $\mathbb{Q}$ is the area used to produce the formation displaying the image. More precisely, $\varphi(q) = 1$, $\varphi(q) = 0$, or $\varphi(q) \in (0, 1)$ indicate that the pixel at point q is white, black, or gray, respectively. By means of this notation, the mass game problem addressed in [15] can be defined as follows.

Problem 1: For the multiagent system Σ, suppose that a reference grayscale image φ is given. Then, find distributed controllers (*i.e.,* a common function f) such that for every initial formation $x(0) \in \mathbb{Q}^n$, the final formation $x(\infty)$ displays the image φ, as illustrated in Fig. 1.

We make two remarks regarding Problem 1. First, it might be assumed that this problem can easily be solved by calculating the desired positions of all agents based on the reference image and then assigning each desired position to a corresponding local controller, but this is not true. In fact, it is impossible to satisfy the constraint that all local controllers are common because different information is given for each local controller. Second, the specification to be satisfied in Problem 1 is based on the visual quality of the agent formation. A typical approach for multiagent control is to describe specifications using mathematical functions and then design local controllers to optimize these functions (see, *e.g.,* [1]). However, since the visual quality of the agent formation depends on human perception and gaps exist between the true and perceived feelings of humans [40], quantifying the specification with a mathematical function is generally difficult. These two facts make the problem challenging.

B. DISTRIBUTED CONTROLLERS FOR MASS GAMES

The idea proposed in [15] to solve Problem 1 involves representing the reference (grayscale) image as a density distribution of the agents. Specifically, we regard each agent as a black pixel and place more agents at locations within the space $\mathbb{Q}$ corresponding to the darker parts of the reference image. This idea is inspired by halftone image processing (see footnote 2 in Section I-A); thus, the grayscale information of the reference image is represented.

Based on this idea, Azuma *et al.* [15] proposed the following solution to Problem 1:

$$f(x_i(t), [x_j(t)]_{j\in\mathbb{N}_i(t)}) := -k(x_i(t) - c(\mathbb{V}_i(x(t)))), \tag{4}$$

where $k \in \mathbb{R}_+$ is the controller gain and $c(\mathbb{V}_i(x(t))) \in \mathbb{Q}$ is the weighted centroid of the Voronoi cell $\mathbb{V}_i(x(t))$, *i.e.,*

$$c(\mathbb{V}_i(x(t))) := \frac{\displaystyle\int_{\mathbb{V}_i(x(t))} q\phi(q)dq}{\displaystyle\int_{\mathbb{V}_i(x(t))} \phi(q)\,dq}. \tag{5}$$

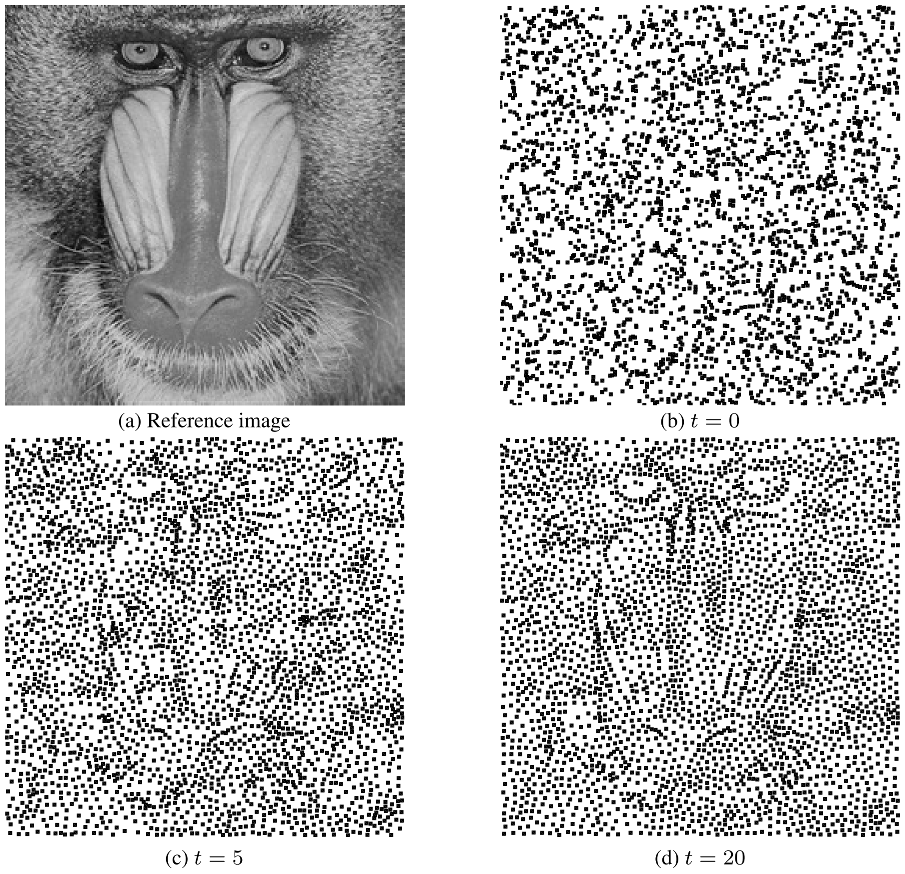
In (5), $\phi : \mathbb{Q} \to \mathbb{R}_+ \cup \{0\}$ is the weighting function defined as

$$\phi(q) := e^{-\kappa\varphi(q)}, \tag{6}$$

where $\kappa \in \mathbb{R}_+$ is a parameter determining the contrast of the image of the resulting formation. The local controller given by (3)–(5), which is called the coverage controller [16], enables the placement of more agents at locations within space $\mathbb{Q}$ with higher values of the weighting function ϕ. Furthermore, according to the definition of φ, $\phi(q)$ in (6) takes higher values at points q corresponding to the darker parts of the reference image. Therefore, the local controllers given by (3)–(6) steer the agents such that more agents are placed at locations within $\mathbb{Q}$ corresponding to the darker parts of the reference image. This process yields a density distribution of agents that reflects the grayscale information of the reference image.

The abovementioned mass game controllers are illustrated through an example. Consider the multiagent system Σ with $n := 3000$ and $\mathbb{Q} := [0, 100]^2$. The reference image φ shown in Fig. 3(a) is given in [41] as one of the standard images in the image processing field. This is an eight-bit grayscale image, *i.e.,* $\varphi(q) \in \{0, 1/255, 2/255, \dots, 1\}$. We apply the local controllers constructed by (3)–(6), where $k := 0.5$ and $\kappa := 10$. Figs. 3(b)–(d) show the time series of the resulting formation, where the black dots indicate agents and the initial formation (*i.e.,* Fig. 3(b)) is randomly chosen based on a uniform probability distribution on $\mathbb{Q}^n$. The formation converges to that displaying the reference image over time. Moreover, in the final formation shown in Fig. 3(d), the grayscale information of the reference image is represented by the spatial variations in the agent density.

Remark 1: The local controller given by (3)–(6) can be intuitively explained as follows. On the basis of (2)–(4), the local controller steers agent i to the weighted centroid



(a) Reference image

(b) $t = 0$

(c) $t = 5$

(d) $t = 20$

FIGURE 3. Example of a mass game.

$c(\mathbb{V}_i(x(t)))$ of the Voronoi cell $\mathbb{V}_i(x(t))$. The weighted centroid $c(\mathbb{V}_i(x(t)))$ is a location within $\mathbb{V}_i(x(t))$ specified as a darker point of the corresponding part of the reference image according to the definition of φ, (5), and (6). Thus, the local controller steers agent i to that location.

Remark 2: Although the positions $[x_j(t)]_{j \in \mathbb{N}_i(t)}$ of neighboring agents do not appear directly on the right-hand side of (4), the local controllers given by (3)–(6) are distributed. This is because the weighted centroid $c(\mathbb{V}_i(x(t)))$ (*i.e.,* the Voronoi cell $\mathbb{V}_i(x(t))$) can be calculated based on only the positions of agent i and its neighbors on the Delaunay graph $G(x(t))$ [16].

C. ISSUES TO BE ADDRESSED

As explained in Section I-A, one potential application of mass games is entertainment, but the following issues must be resolved to facilitate such an application. First, it is difficult for nonexpert users to understand the abovementioned theory of mass games. Understanding mass game theory requires knowledge of, for example, halftone image processing, coverage control, Voronoi cells, and Delaunay graphs. However, this knowledge comes from distinct fields—specifically, image processing, control theory, and computational geometry—thus, relatively few nonexpert users will be familiar with all of the required knowledge. Second, the implementation of the abovementioned method takes considerable time and effort. For instance, for (4)–(6), it is necessary to compute the weighted centroid $c(\mathbb{V}_i(x(t)))$ of the Voronoi cell $\mathbb{V}_i(x(t))$ for the weighting function ϕ determined by the reference image φ. However, as conventional programming languages such as MATLAB are not equipped with dedicated functions to compute the integral of a given function over a Voronoi cell, this calculation must be implemented by each user.

III. DEVELOPMENT OF MGS

To overcome these issues, we developed the MGS that allows the simulation of mass games without the need for expert knowledge or programming effort. This section introduces the MGS.

A. MGS

An overview of the MGS is shown in Fig. 4. The MGS was developed using the MATLAB App Designer, *i.e.,* a development environment in MATLAB used to create applications, which can be downloaded as a MATLAB application installer from the website [42]. Running the MGS requires MATLAB version R2019a or later and no toolboxes.

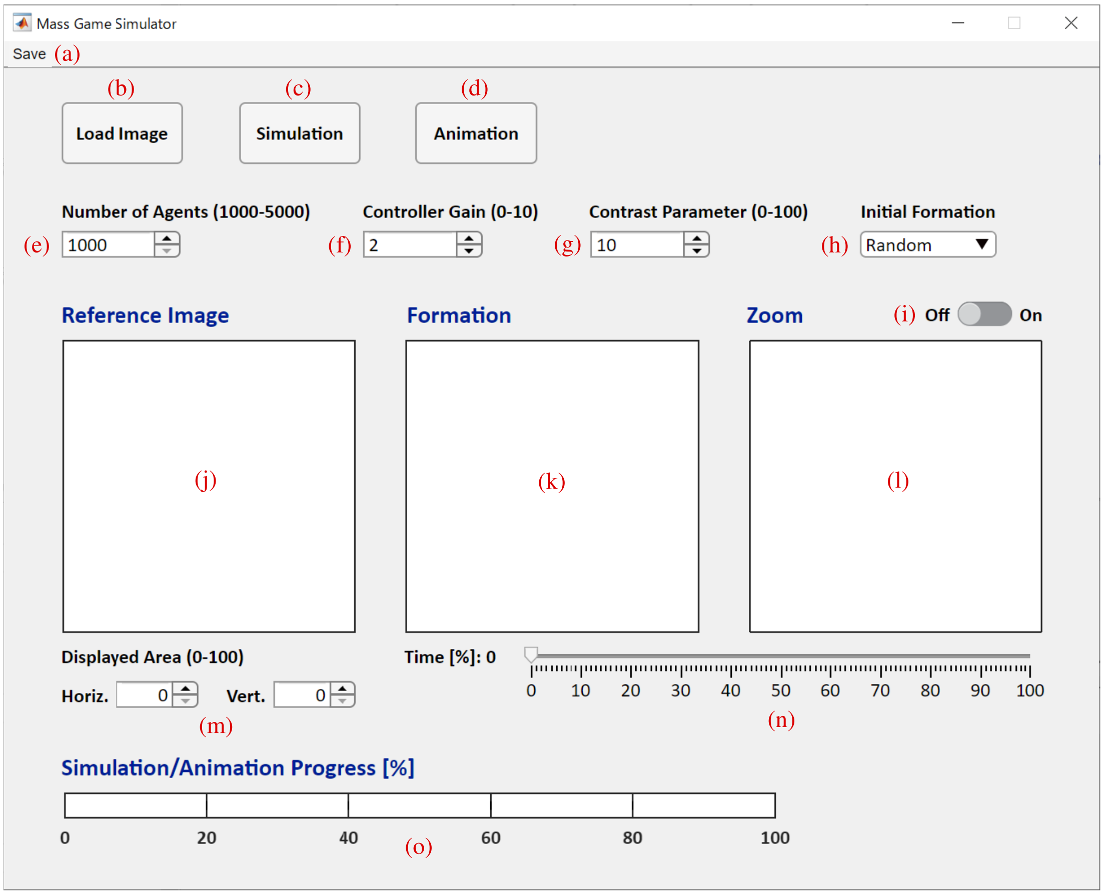


FIGURE 4. Overview of the MGS.

The components of the MGS, shown by (a)–(o) in Fig. 4, are described in detail as follows:

(a) *"Save" menu:* By selecting "Save as Image" from this menu, the image of the resulting formation shown in the Formation window (k) can be saved as the image file "formation.jpg." Moreover, by selecting "Save as Movie," the time series data of the resulting formation can be saved as the movie file "massgame.avi." These files are generated in the current directory.
(b) *"Load Image" button:* This button is used to import an image file as the reference image, where the supported file formats are jpg and bmp.
(c) *"Simulation" button:* Clicking this button starts the simulation of the mass game for the reference image shown in the Reference Image window (j).
(d) *"Animation" button:* Clicking this button following the simulation runs an animation of the time evolution of the resulting formation in the Formation window (k).
(e) *"Number of Agents" spinner:* This spinner is used to set the number n of the agents, where $n \in [1000, 5000]$.
(f) *"Controller Gain" spinner:* This spinner is used to set the gain k in (4), where $k \in (0, 10]$.
(g) *"Contrast Parameter" spinner:* This spinner is used to set the parameter κ in (6), where $\kappa \in (0, 100]$.
(h) *"Initial Formation" menu:* This menu is used to set the initial formation $x(0)$ as either "Random" or "Same as before." Under the former setting, $x(0)$ is randomly determined according to a uniform probability distribution on $\mathbb{Q}^n$. Under the latter setting, the final formation obtained by the previous simulation is set as $x(0)$.
(i) *"Zoom" switch:* Turning this switch "On" shows a magnification of the Formation window (k) in the Zoom window (l).
(j) *"Reference Image" window:* This window shows the reference image imported using the Load Image button (b). If the imported image is rectangular, a trimmed square image is shown.
(k) *"Formation" window:* This window shows the resulting formation. By moving the Time slider (n), the formation at the corresponding time is displayed. In addition, by clicking the Animation button (d), an animation of the time evolution of the formation is shown, as explained in (d) above.
(l) *"Zoom" window:* If the Zoom switch is "On," the window shows a magnification of the Formation window (k) and the weighting function ϕ in (6). The weighting function ϕ is shown as a grayscale background,

in which the darker areas correspond to locations with higher values of ϕ.

(m) *"Displayed Area" spinners:* If the image imported using the Load Image button (b) is rectangular, we can use these spinners to move the area displayed in the Reference Image window (j). The settings 0 and 100 indicate the sides of the image.

(n) *"Time" slider:* This slider is used to specify the time (in units of percentage) of the formation shown in the Formation window (k); on the slider, the settings 0 and 100 indicate the start and end of the simulation, respectively.

(o) *"Simulation/Animation Progress" bar:* This bar shows the progress (in units of percentage) of the simulation and animation initiated by clicking the Simulation and Animation buttons (c) and (d), respectively.

Remark 3: We can generalize the initial formation setting by allowing users to determine the initial position of each agent. However, such a generalization substantially increases the effort required by users because the number n of agents is at least 1000. Hence, to reduce user effort, we simplified the settings in the Initial Formation menu as "Random" and "Same as before."

Remark 4: Although the MGS assumes square reference images, the method described in Section II can be applied to rectangular reference images because they satisfy the condition that the space $\mathbb{Q}$ corresponding to the reference image is a convex set. The assumption of square reference images is imposed to simplify the development of the MGS and to increase its reliability. In fact, there are various sizes of rectangular images depending on their aspect ratio, which makes the implementation of the Reference Image, Formation, and Zoom windows difficult and causes bugs.

B. USAGE

The usage of the MGS is explained as follows, in which (a)–(o) correspond to the labels in Fig. 4.

Step 1 Import a file of the reference image using the Load Image button (b). The imported image is shown in the Reference Image window (j).

Step 2 If necessary, move the area displayed in the Reference Image window (j) using the Displayed Area spinners (m) so that the desired reference image is shown in the window.

Step 3 Set the number n of agents, the gain k, and parameter κ using spinners (e)–(g).

Step 4 Set the initial formation $x(0)$ using the Initial Formation menu (h), in which "Same as before" is available only after the mass game has been simulated at least once.

Step 5 Click the Simulation button (c) to begin the mass game simulation. The progress of the simulation is shown on the Simulation/Animation Progress bar (o). Once the simulation is complete, go to Step 6.

Step 6 If necessary, turn the Zoom switch (i) "On" (see item (i) in Section III-A for the details of the Zoom switch).

Step 7 The Time slider (n) can be moved to display the resulting formation at the specified time in the Formation window (k).

Step 8 By clicking the Animation button (d), an animation of the simulation result is shown in the Formation window (k). The progress of the animation is shown on the Simulation/Animation Progress bar (o). No operation should be performed until the animation ends.

Step 9 If necessary, the simulation result can be saved as an image and/or movie file using the Save menu (a) (see item (a) in Section III-A for details on saving files).

The following points regarding the use of the MGS should be noted. First, we can select a color image in Step 1 even though the method described in Section II assumes grayscale reference images. When a color image is selected in Step 1, it is automatically converted into its grayscale version. Second, Step 1 can be skipped when performing a simulation for a previously used reference image (or its variations obtained using the Displayed Area spinners) and different values of n, k, and κ.

C. ILLUSTRATIVE EXAMPLE

An illustrative example of the MGS in operation is provided in Fig. 5, where $n := 2500$, $k := 1.5$, $\kappa := 15$, and the steps correspond to those in Section III-B. Using MGS, we can simulate the mass game for a selected reference image, confirm the simulation result, and save it by means of simple mouse and keyboard operations (see also a demonstration video of the MGS provided as supplementary material). In addition, examples for changing the simulation settings using the Number of Agents and Contrast Parameter spinners are provided in Fig. 6, in which (a) and (b) correspond to cases of increasing n and κ, respectively. Compared to Fig. 5(d), the image of the resulting formation in Fig. 6(a) is darker and that in Fig. 6(b) has higher contrast. These results demonstrate that the spinners allow us to change the simulation settings.

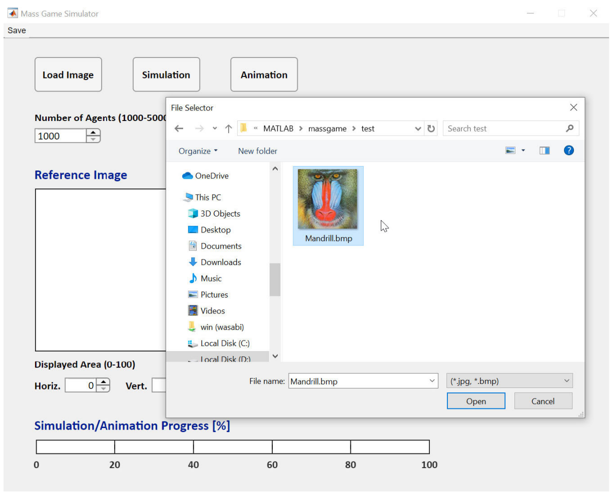
Remark 5: While the MGS is provided as a MATLAB application, it is possible to develop a similar application using another programming language if it can handle not only numerical computations but also images and animations, as seen in this subsection.

IV. APPLICATIONS

This section presents two application examples of the MGS. These applications have not been discussed in most existing studies on multiagent control.

A. ENTERTAINMENT

We start by introducing an example of a method in which beginners can use the MGS for entertainment. As a first step, based on the usage explained in Section III-B, we run the

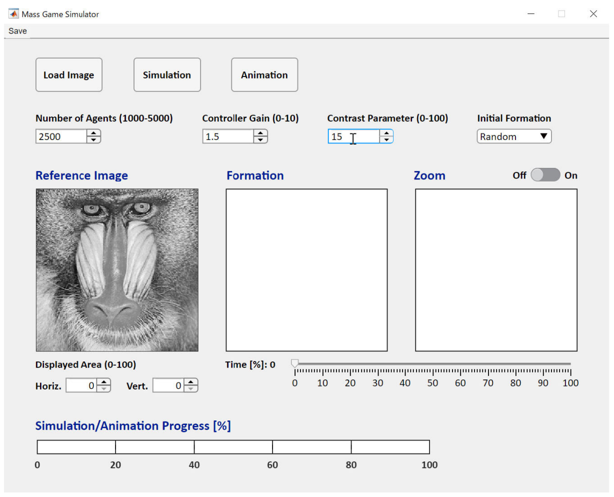


Number of Agents (1000-5000): 1000

Displayed Area (0-100) Horiz.: 0

File name: Mandrill.bmp (*.jpg, *.bmp)

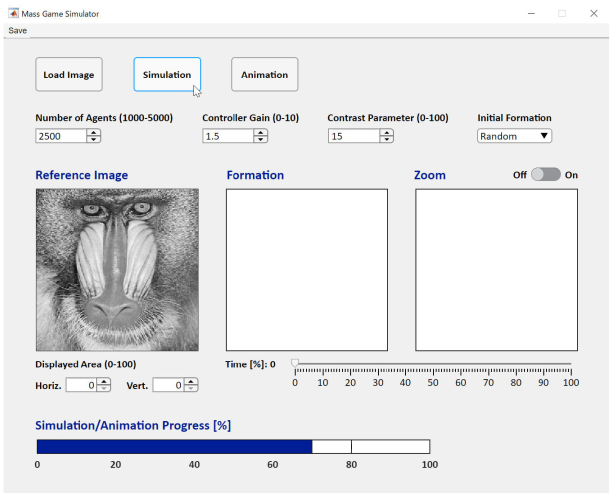
(a) Step 1



Number of Agents (1000-5000): 2500 Controller Gain (0-10): 1.5 Contrast Parameter (0-100): 15 Initial Formation: Random

Displayed Area (0-100) Horiz.: 0 Vert.: 0

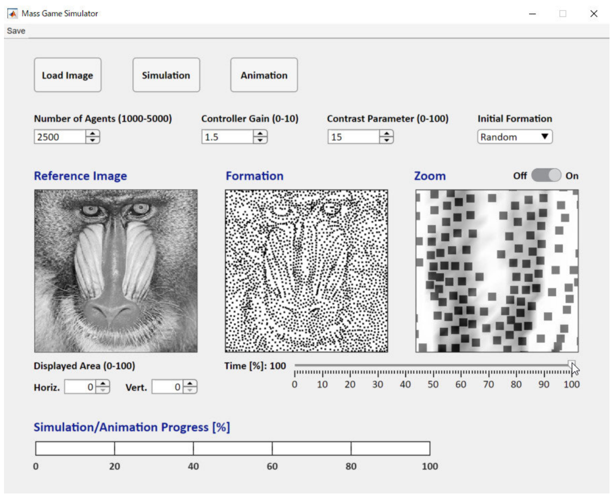
(b) Step 3



Number of Agents (1000-5000): 2500 Controller Gain (0-10): 1.5 Contrast Parameter (0-100): 15 Initial Formation: Random

Displayed Area (0-100) Horiz.: 0 Vert.: 0

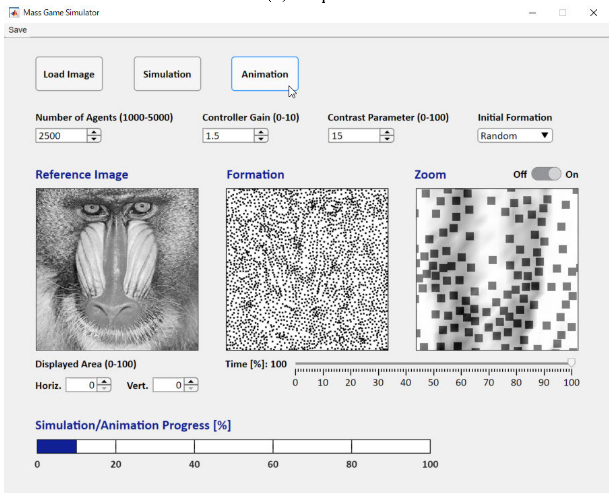
(c) Step 5



Number of Agents (1000-5000): 2500 Controller Gain (0-10): 1.5 Contrast Parameter (0-100): 15 Initial Formation: Random

Displayed Area (0-100) Horiz.: 0 Vert.: 0

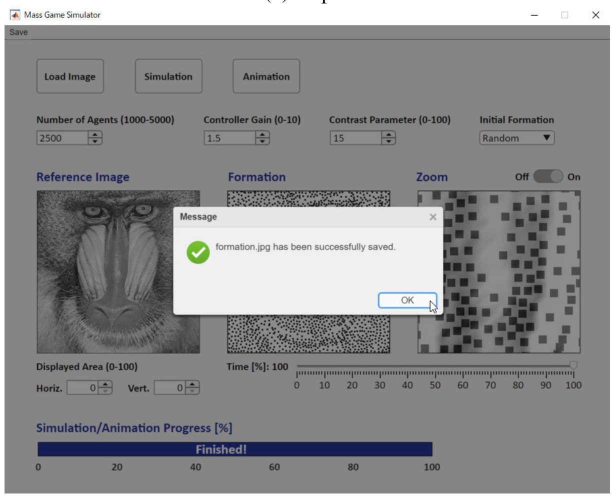
(d) Step 7



Number of Agents (1000-5000): 2500 Controller Gain (0-10): 1.5 Contrast Parameter (0-100): 15 Initial Formation: Random

Displayed Area (0-100) Horiz.: 0 Vert.: 0

(e) Step 8



formation.jpg has been successfully saved.

Finished!

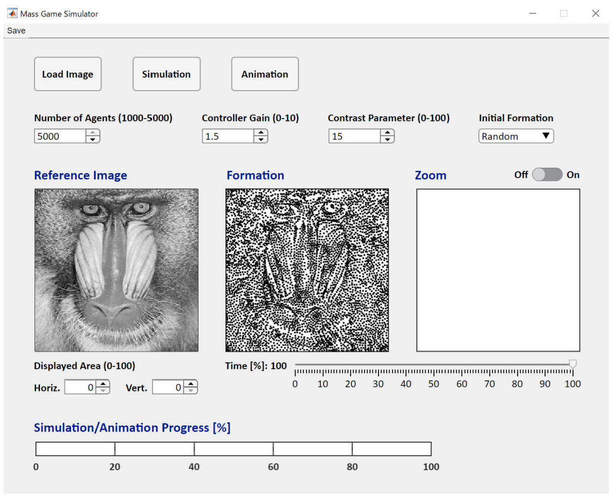
Number of Agents (1000-5000): 2500 Controller Gain (0-10): 1.5 Contrast Parameter (0-100): 15 Initial Formation: Random

Displayed Area (0-100) Horiz.: 0 Vert.: 0

(f) Step 9

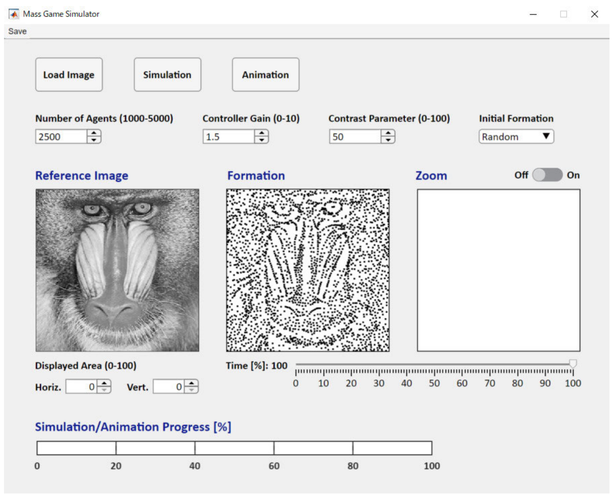
FIGURE 5. Use of the MGS.

MGS once with the Zoom switch set to "Off." Then, by turning the Zoom switch "On" and inspecting the Zoom window, we can confirm that the agents move to darker areas of the window, *i.e.*, the locations with higher values of the weighting function ϕ, as shown in Fig. 7. This exercise is helpful in understanding the underlying idea of mass game



Number of Agents (1000-5000): 5000
Controller Gain (0-10): 1.5
Contrast Parameter (0-100): 15
Initial Formation: Random
Displayed Area (0-100) Horiz.: 0 Vert.: 0
Time [%]: 100

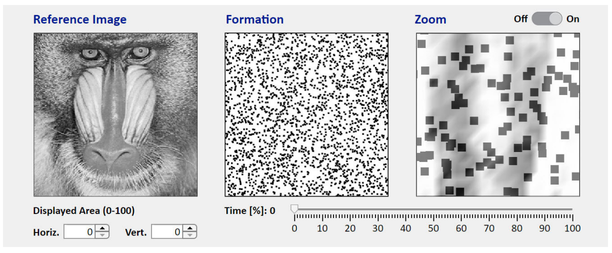
(a) $n := 5000$



Number of Agents (1000-5000): 2500
Controller Gain (0-10): 1.5
Contrast Parameter (0-100): 50
Initial Formation: Random
Displayed Area (0-100) Horiz.: 0 Vert.: 0
Time [%]: 100

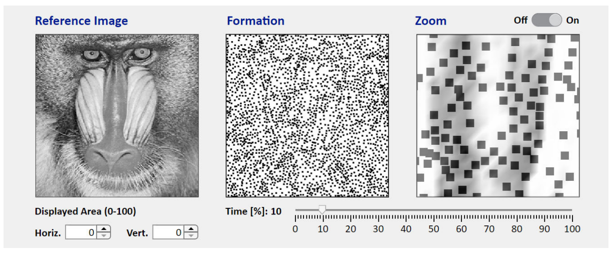
(b) $\kappa := 50$

FIGURE 6. Changing the simulation settings.



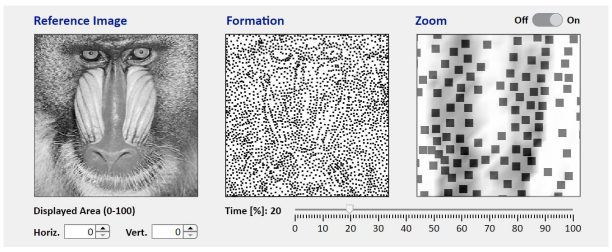
Displayed Area (0-100) Horiz.: 0 Vert.: 0
Time [%]: 0

(a) Time: 0%



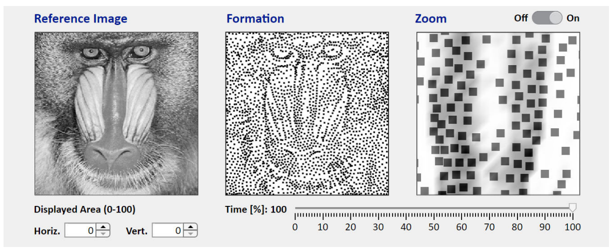
Displayed Area (0-100) Horiz.: 0 Vert.: 0
Time [%]: 10

(b) Time: 10%



Displayed Area (0-100) Horiz.: 0 Vert.: 0
Time [%]: 20

(c) Time: 20%



Displayed Area (0-100) Horiz.: 0 Vert.: 0
Time [%]: 100

(d) Time: 100%

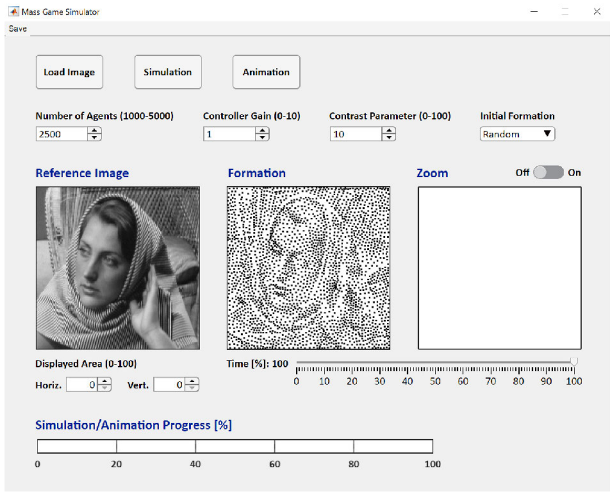
FIGURE 7. Confirmation of the behavior of agents using the Zoom window.

controllers that is described in Section II-B. This process also helps us to focus not only on the appearance of the formation but also on how to control the agents. Introducing the Zoom switch and window provides these advantages. Subsequently, we can create our own formations by changing the reference image and the simulation settings (*i.e.*, the values of n, k, and κ). For example, by tuning n and κ, we can create an optimal formation in terms of subjective visual quality for a specific reference image.

B. DIGITAL SIGNAGE

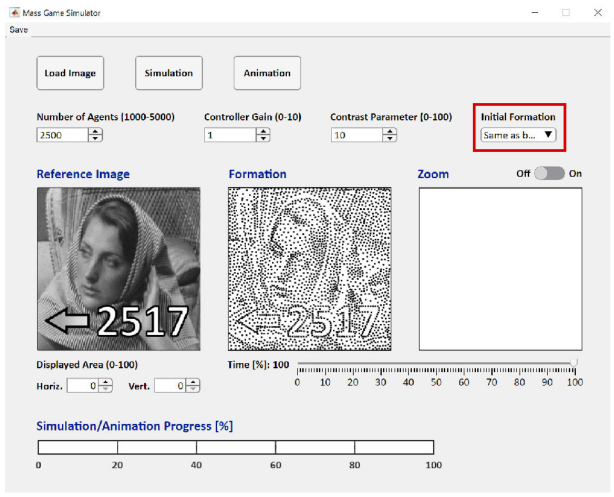
Digital signage, which provides information using display devices, has been widely used in recent years. It can be found in, for example, transportation systems, stadiums, and corporate buildings. Here, we consider the application of the MGS to digital signage. Specifically, we create an animation of the mass game using the MGS and play it on digital signage. By choosing a reference image containing appropriate information, we can construct a novel type of signage where the information is represented by the motion of individual dots corresponding to agents.

As an example, we created an animation for the application to digital signage placed at the entrance of the authors' room (room number 2517). We performed simulations for two reference images based on another standard image using MGS, as shown in Fig. 8, where $n := 2500$, $k := 1$, $\kappa := 10$, and "Same as before" was chosen from the Initial Formation menu in Fig. 8(b) to combine the resulting two animations. Fig. 9 shows a photo of playing the combination of the resulting two animations on digital signage using a DSM-32L7X display (MITSUBISHI ELECTRIC Corporation). Fig. 10 shows snapshots of the displayed contents, in which the first animation was switched to the second animation at approximately $t = 10$ s. As the dots move, the displayed contents change smoothly as a random formation



Number of Agents (1000-5000): 2500
Controller Gain (0-10): 1
Contrast Parameter (0-100): 10
Initial Formation: Random
Displayed Area (0-100): Horiz. 0 Vert. 0
Time [%]: 100

(a) First animation



Number of Agents (1000-5000): 2500
Controller Gain (0-10): 1
Contrast Parameter (0-100): 10
Initial Formation: Same as b...
Displayed Area (0-100): Horiz. 0 Vert. 0
Time [%]: 100

(b) Second animation

FIGURE 8. **Creation of animations for the application to digital signage.**

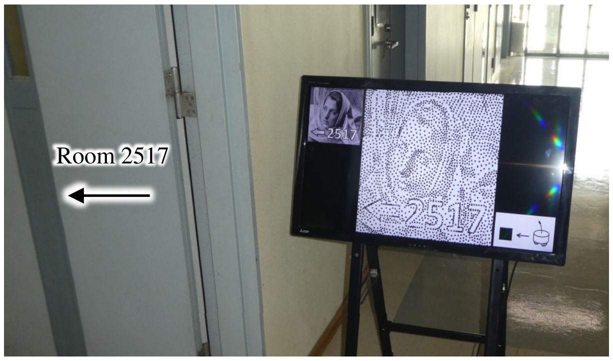
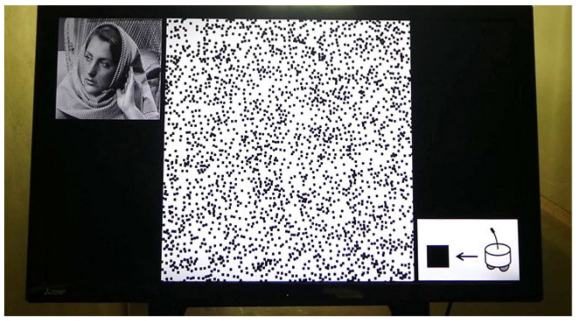
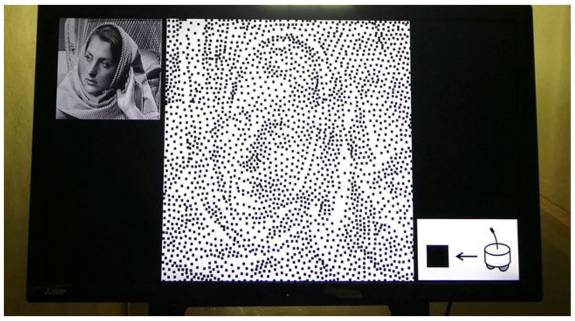


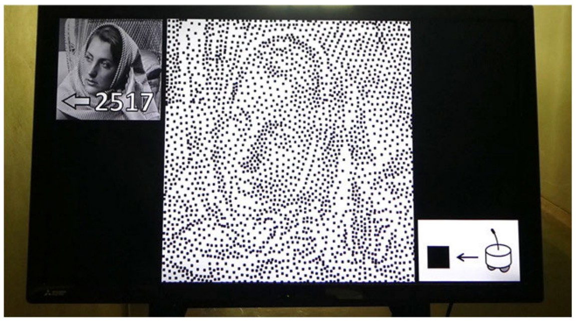
FIGURE 9. **Playing the animation on digital signage.**



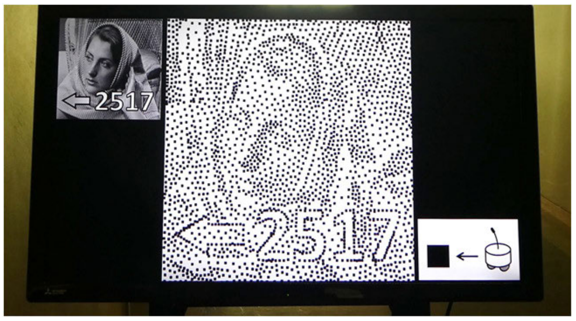
(a) $t = 0\,\mathrm{s}$



(b) $t = 7\,\mathrm{s}$



(c) $t = 10\,\mathrm{s}$

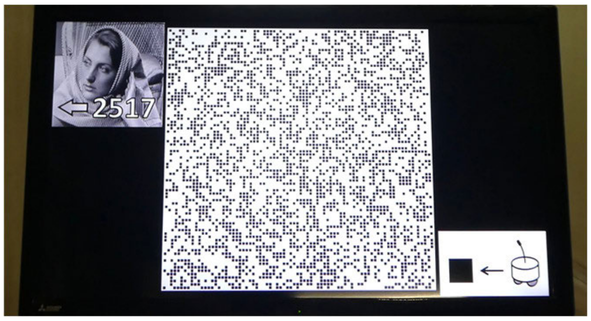


(d) $t = 15\,\mathrm{s}$

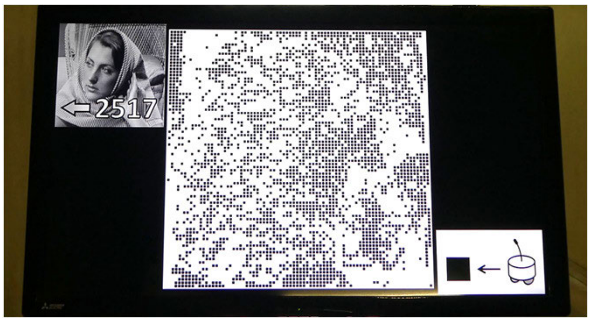
FIGURE 10. **Snapshots of the displayed contents (the brightness of the images has been adjusted to improve their visibility).**

→ a woman → both a woman and the room number. The corresponding video is provided as supplementary material. The introduction of the Save menu and the Initial Formation menu allows such an application.

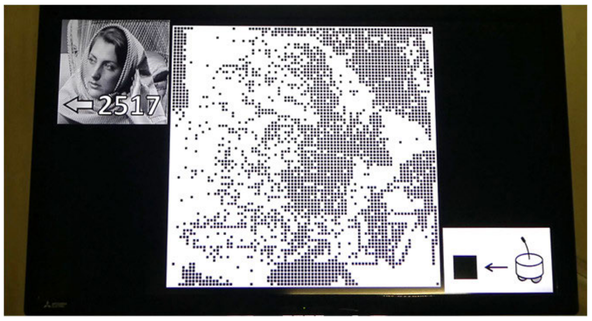
We then evaluated the quality of the displayed contents via comparison with an existing method [28] for the formation control of multiagent systems. We created an animation of a simulation result by applying the existing method to the reference image in Fig. 8(b). Here, to apply the existing method, the reference (grayscale) image was binarized using a threshold, and the desired positions of the respective agents were set as the locations of the individual black pixels in the resulting binary image. The threshold of the binarization was



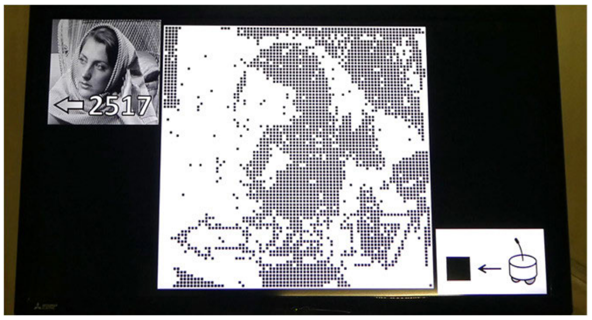
(a) $t = 0\,\mathrm{s}$



(b) $t = 10\,\mathrm{s}$



(c) $t = 30\,\mathrm{s}$



(d) $t = 60\,\mathrm{s}$

FIGURE 11. **Snapshots of the displayed contents produced by an existing method [28] (the brightness of the images has been adjusted to improve their visibility).**

set such that the number of black pixels in the resulting binary image was approximately 2500, *i.e.*, the number of agents in the preceding example; as a result, $n := 2891$ was obtained. We ran the resulting animation on digital signage in a manner similar to that in the preceding example; snapshots of the displayed contents are shown in Fig. 11. Compared to Fig. 10, it is difficult to obtain the information on the reference image (in particular, the left half) from the formation because the grayscale information of the reference image is lost as a result of the binarization. In contrast, the method we employ (*i.e.*, the method described in Section II) represents the grayscale information as a density distribution of agents without binarization, thereby producing a formation whose appearance is closer to the reference image. Furthermore, the time spent producing the desired formation is longer than that for Fig. 10 because the method proposed in [28] randomly assigns each desired position to a corresponding agent at the beginning of the process without taking the distances between the agent positions and the desired positions into account, whereas the method we employ steers each agent to a favorable position close to the current position in a distributed manner.

V. CONCLUSION

In this work, we proposed and developed a simulator of multiagent mass games for practical use. The proposed simulator can be operated through simple mouse and keyboard inputs without the need for expert knowledge or programming effort. The simulator also includes zoom and save functions to extend its application range. Using these functions, we applied the proposed simulator to entertainment and digital signage. In the digital signage application, we demonstrated that the displayed contents generated by the proposed simulator are better than those generated by an existing method in terms of both visual quality and time spent producing the desired formation. The results presented in this paper will lead to new applications of the control of multiagent systems.

Several directions for future research exist. One example is to make the proposed simulator available to non-MATLAB users. Another example is to incorporate the framework of another type of mass game, given in [18], into the proposed simulator. We also plan to develop other applications for the proposed simulator.

REFERENCES

[1] S. Martinez, J. Cortes, and F. Bullo, "Motion coordination with distributed information," *IEEE Control Syst. Mag.*, vol. 27, no. 4, pp. 75–88, Aug. 2007.

[2] X. Wang, Q. Yang, T. Cai, and Y. Wei, "Distributed formation flight control with translational and rotational maneuvering," *IEEE Access*, vol. 7, pp. 159565–159574, Oct. 2019.

[3] Y. Li, M. Jiu, Q. Sun, and Q. Dong, "An adaptive distributed consensus control algorithm based on continuous terminal sliding model for multiple quad rotors' formation tracking," *IEEE Access*, vol. 7, pp. 173955–173967, Dec. 2019.

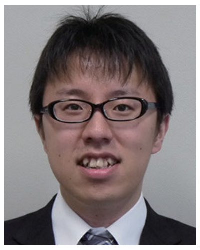
[4] S. Izumi, S.-I. Azuma, and T. Sugie, "Analysis and design of multi-agent systems in spatial frequency domain: Application to distributed spatial filtering in sensor networks," *IEEE Access*, vol. 8, pp. 34909–34918, Feb. 2020.

[5] S. Izumi, R. Katayama, X. Xin, and T. Yamasaki, "Distributed spatial filtering over networked systems," *IEEE Control Syst. Lett.*, vol. 5, no. 2, pp. 617–622, Apr. 2021.

[6] M. Hasanuzzaman Shawon, S. M. Muyeen, A. Ghosh, S. M. Islam, and M. S. Baptista, "Multi-agent systems in ICT enabled smart grid: A status update on technology framework and applications," *IEEE Access*, vol. 7, pp. 97959–97973, Jul. 2019.

[7] H. Li, Z. Wang, G. Chen, and Z. Y. Dong, "Distributed robust algorithm for economic dispatch in smart grids over general unbalanced directed networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4322–4332, Jul. 2020.

[8] S. He, Q. Tang, C. Q. Wu, and X. Shen, "Decentralizing IoT management systems using blockchain for censorship resistance," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 715–727, Jan. 2020.
[9] M. Yan, Y. Guo, L. Zuo, and P. Yang, "Information-based optimal deployment for a group of dynamic unicycles," *Int. J. Control, Autom. Syst.*, vol. 16, no. 4, pp. 1824–1832, Jul. 2018.
[10] G. A. Cardona, D. Tellez-Castro, and E. Mojica-Nava, "Cooperative transportation of a cable-suspended load by multiple quadrotors," *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 145–150, Sep. 2019.
[11] S. Izumi and S.-I. Azuma, "Real-time pricing by data fusion on networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 1175–1185, Mar. 2018.
[12] S. P. Talebi and S. Werner, "Distributed Kalman filtering and control through embedded average consensus information fusion," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 4396–4403, Oct. 2019.
[13] Y. Kajiyama, N. Hayashi, and S. Takai, "Distributed subgradient method with edge-based event-triggered communication," *IEEE Trans. Autom. Control*, vol. 63, no. 7, pp. 2248–2255, Jul. 2018.
[14] C. Deng and G.-H. Yang, "Distributed adaptive fault-tolerant control approach to cooperative output regulation for linear multi-agent systems," *Automatica*, vol. 103, pp. 62–68, May 2019.
[15] S.-I. Azuma, S. Izumi, and T. Sugie, "Halftone mass games by fixed number of mobile robots," (in Japanese), *Trans. Inst. Syst., Control Inf. Eng.*, vol. 25, no. 4, pp. 94–100, 2012.
[16] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.
[17] R. Ulicheney, *Digital Halftoning*. Cambridge, MA, USA: MIT Press, 1987.
[18] S. Izumi, S.-I. Azuma, and T. Sugie, "Distributed hybrid controllers for multi-agent mass games by a variable number of player agents," *Asian J. Control*, vol. 17, no. 3, pp. 762–774, May 2015.
[19] K. Sugihara and I. Suzuki, "Distributed algorithms for formation of geometric patterns with many mobile robots," *J. Robotic Syst.*, vol. 13, no. 3, pp. 127–139, Mar. 1996.
[20] I. Suzuki and M. Yamashita, "Distributed anonymous mobile robots: Formation of geometric patterns," *SIAM J. Comput.*, vol. 28, no. 4, pp. 1347–1363, Jan. 1999.
[21] 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, Sep. 2004.
[22] W. Ren and N. Sorensen, "Distributed coordination architecture for multi-robot formation control," *Robot. Auto. Syst.*, vol. 56, no. 4, pp. 324–333, Apr. 2008.
[23] E. Montijano, E. Cristofalo, D. Zhou, M. Schwager, and C. Sagues, "Vision-based distributed formation control without an external positioning system," *IEEE Trans. Robot.*, vol. 32, no. 2, pp. 339–351, Apr. 2016.
[24] J. Yang, X. Wang, and P. Bauer, "V-shaped formation control for robotic swarms constrained by field of view," *Appl. Sci.*, vol. 8, no. 11, p. 2120, Nov. 2018.
[25] L. Wang, J. Xi, M. He, and G. Liu, "Robust time-varying formation design for multiagent systems with disturbances: Extended-state-observer method," *Int. J. Robust Nonlinear Control*, vol. 30, no. 7, pp. 2796–2808, Mar. 2020.
[26] H. Liu, T. Ma, F. L. Lewis, and Y. Wan, "Robust formation control for multiple quadrotors with nonlinearities and disturbances," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1362–1371, Apr. 2020.
[27] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple mobile robots," *Int. J. Robot. Res.*, vol. 31, no. 6, pp. 753–773, May 2012.
[28] H. Wang and M. Rubenstein, "Shape formation in homogeneous swarms using local task swapping," *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 597–612, Jun. 2020.
[29] B. A. Guvenc and E. Kural, "Adaptive cruise control simulator: A low-cost, multiple-driver-in-the-loop simulator," *IEEE Control Syst. Mag.*, vol. 26, no. 3, pp. 42–55, Jun. 2006.
[30] L. Nehaoua, H. Mohellebi, A. Amouri, H. Arioui, S. Espie, and A. Kheddar, "Design and control of a small-clearance driving simulator," *IEEE Trans. Veh. Technol.*, vol. 57, no. 2, pp. 736–746, Mar. 2008.
[31] R. Gonzalez, C. Mahulea, and M. Kloetzer, "A MATLAB-based interactive simulator for mobile robotics," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2015, pp. 310–315.
[32] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "USARSim: A robot simulator for research and education," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 1400–1405.
[33] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intell.*, vol. 6, no. 4, pp. 271–295, Dec. 2012.
[34] M. Casini and A. Garulli, "MARS: A MATLAB simulator for mobile robotics experiments," *IFAC-PapersOnLine*, vol. 49, no. 6, pp. 69–74, Jul. 2016.
[35] L. Pannocchi, C. Di Franco, M. Marinoni, and G. Buttazzo, "Integrated framework for fast prototyping and testing of autonomous systems," *J. Intell. Robotic Syst.*, vol. 96, no. 2, pp. 223–243, Nov. 2019.
[36] B. N. Agrawal and R. E. Rasmussen, "Air-bearing-based satellite attitude dynamics simulator for control software research and development," *Proc. SPIE*, vol. 4366, pp. 204–214, Aug. 2001.
[37] I. García-Magariño, "ABSTUR: An agent-based simulator for tourist urban routes," *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5287–5302, Jul. 2015.
[38] A. Mellit, H. Mekki, A. Messai, and H. Salhi, "FPGA-based implementation of an intelligent simulator for stand-alone photovoltaic system," *Expert Syst. Appl.*, vol. 37, no. 8, pp. 6036–6051, Aug. 2010.
[39] M. Miura, Y. Tokunaga, and K. Sakurama, "Graphical and scalable multi-agent simulator for real-time pricing in electric power grid," *Artif. Life Robot.*, vol. 21, no. 2, pp. 181–187, Jun. 2016.
[40] Y. Shang, "Resilient consensus for expressed and private opinions," *IEEE Trans. Cybern.*, vol. 51, no. 1, pp. 318–331, Jan. 2021.
[41] University of Southern California. *USC-SIPI Image Database*. Accessed: May 9, 2020. [Online]. Available: http://sipi.usc.edu/database/
[42] S. Izumi. (2020). *Mass Game Simulator*. [Online]. Available: https://github.com/ShinsakuIzumi/Mass_Game_Simulator



SHINSAKU IZUMI (Member, IEEE) received the M.S. and Ph.D. degrees in informatics from Kyoto University, Japan, in 2012 and 2015, respectively. He is currently an Assistant Professor with the Faculty of Computer Science and Systems Engineering, Okayama Prefectural University, Japan. His research interests include networked control systems and multiagent systems.

YUTO SHIOMOTO received the B.E. degree from Okayama Prefectural University, Japan, in 2020. His research interests include control of multiagent systems and its application.

XIN XIN (Senior Member, IEEE) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1987, the Ph.D. degree from Southeast University, Nanjing, China, in 1993, and the doctorate degree in engineering from the Tokyo Institute of Technology, in 2000. From 1991 to 1993, he did his Ph.D. studies in Osaka University as a Co-Advised Student of China and Japan with the Japanese Government Scholarship. From 1993 to 1995, he was a Postdoctoral Researcher and then became an Associate Professor at Southeast University. From 1996 to 1997, he was with New Energy and Industrial Technology Development, Japan, as an Advanced Industrial Technology Researcher. From 1997 to 2000, he was an Assistant Professor with the Tokyo Institute of Technology. Since 2000, he has been with Okayama Prefectural University as an Associate Professor, where he is currently a Professor since 2008. He has over 210 publications in journals, international conferences, and book chapters. His current research interests include robotics, dynamics and control of nonlinear and complex systems. He received the Division Paper Award of SICE 3rd Annual Conference on Control Systems, in 2004.