In Situ Visualization Inspired by Ant Colony Formation[*)]

Yan WANG, Nobuaki OHNO[1)] and Akira KAGEYAMA

Graduate School of System Informatics, Kobe University, Kobe 657-8501, Japan

[1)] *Graduate School of Information Science, University of Hyogo, Kobe 651-2197, Japan*

(Received 9 January 2023 / Accepted 10 April 2023)

A challenge in the in-situ visualization is the proper setting of the visualization's viewpoint. While visualizations should focus on a small part of the simulation region or region of interest (ROI), the appearance/disappearance and the motion of ROI are usually unknown beforehand in simulations of complex phenomena. In our previous paper, we proposed to make the visualization camera a kind of autonomous agent or a "smart camera". In this approach, the camera agent locates itself near an ROI and keeps an appropriate distance to track the motion of the ROI. The next step in our approach is to increase the number of camera agents to visualize an ROI from multiple perspectives. A problem in introducing multiple camera agents resides in the rules for the agents' motion. In this paper, we propose to use virtual materials, or pheromones, that trigger social responses in camera agents.

Keywords: in-situ visualization, agent-based visualization, ant colony algorithm, 3D cellular automaton

DOI: 10.1585/pfr.18.2401045

1. Introduction

In-situ visualization is becoming a necessary research method in high-performance computing (HPC) [1]. In this approach, researchers obtain high-quality visualization images or videos during simulation runs by adding visualization tasks to simulations without storing massive numerical data. In the post-hoc visualization approach usually adopted in HPC, the numerical data's spatial and temporal resolution are strictly restricted due to the storage capacity and I/O bandwidth.

A problem in the in-situ visualization is that it does not allow interactive control of visualization parameters because one has to specify the parameters before the simulation job. One cannot change the parameters during the simulation, such as camera position, viewing angle, applied visualization algorithms, and the related variables of the algorithm. Among them, the most severe restriction resides in the camera position. It is generally difficult to prescribe a local area, or a region of interest (ROI), in the simulation region where one should apply intensive visualizations during a simulation, especially when the simulated phenomena exhibit complex dynamics.

We proposed an in-situ visualization approach that enables interactive analysis, rather than interactive control, of in-situ visualized videos [2]. The key idea is to apply multiple in-situ visualizations from many different viewpoints at once, then to apply the interactive exploration of the video dataset produced by the in-situ visualization. A similar approach based on images to in-situ visualization is Cinema [3, 4]. Extending our video-based method, we formulated "4D Street View" [5, 6]. In this scheme, we place multiple omnidirectional cameras with the full field of view of 4π steradians. The omnidirectional cameras are scattered in the whole simulation region. The viewpoint and viewing direction can be interactively changed afterward as in the Google street view [7]. In the 4D Street View, the visualization points, or cameras, are fixed in space, and that hampers the flexibility of viewing experiences. In 2021, we proposed a complementary approach [8] in which visualization cameras can move toward ROI by themselves. The key is to make a viewpoint an independent entity, combining an agent-based model (ABM) with the in-situ visualization. The application of ABM to information visualization in general was proposed in [9] as agent-based visualization (ABV). The agent-based in-situ visualization proposed in our study is an application of ABV concept to the in-situ visualization for HPC. In our agent-based in-situ visualization, the agents are visualization cameras that autonomously identify and track ROIs by following prescribed rules to apply in-situ visualization near them during the simulation. The case of a single autonomous camera tracking the motion of an unpredictable ROI was shown in our previous study.

The aim of this paper is to propose a method to extend the agent-based in-situ visualization by a single agent to multiple agents or camera swarms. In our single-camera test case in the previous study, we set a rule of ROI tracking for the camera agent which was designed to make the camera agent autonomously finds and tracks an ROI in the simulation region. But if we apply the same rule to the multiple agents, they will be concentrated in a small spot near

author's e-mail: 190x701x@stu.kobe-u.ac.jp

[*)] This article is based on the presentation at the 31st International Toki Conference on Plasma and Fusion Research (ITC31).

an ROI because they follow the same prescription. The rule for multiple cameras should differ from that for a single camera. In addition to avoiding overcrowding, a rule is required that makes the best use of the camera swarm scattered throughout the simulation region. An efficient visualization could be achieved when some agents "sniff out" an ROI and communicate its location to other agents. The unpredictable appearance and motion of ROIs in the simulation can be effectively traced by multiple agents. In other words, we propose to introduce so-called swarm intelligence [10] to the in-situ visualization.

The key idea of this work can also be illustrated by contrasting two space-times: We consider a kind of virtual space-time, or a "ghost" world, where a group of movable viewpoints called camera agents resides. We assume that the ghost world is overlapped with the "real" world, where physical phenomena take place. Each camera agent in the ghost space applies visualization of the physics in the real world. The real world influences the camera agents in the ghost world, but the interaction is one-sided. The ghost world does not influence the real world. The size of the ghost world is generally more extensive than the real world to obtain visualizations taken from outside the real world. The time of the ghost world can also be different from that of the real world. The time step for the camera agents' motion is not necessarily the same as the time step of the simulation in general. (In the present paper, we use the same time step for both the real and ghost world.) To concentrate the visualization on some local region in the simulation, or region-of-interest (ROI), we assume a kind of virtual material (pheromone) penetrates from the real world to the ghost world that attracts the camera agents.

2. Ant Foraging Algorithm

In agent-based modeling (ABM), autonomous entities called agents have their own properties, ability, and behaviors. Agents communicate with each other and with the environment [11].

One standard means of agent-environment interaction is to use a pheromone. A pheromone, which is a function of space and time or a scalar field, is known to lead a self-organization of a group of simple agents [12]. Simulating the pheromones of insects and other animals, the pheromone is usually designed to be released by some agents, acting as a source term of the scalar field. Some released pheromones are evaporated, and some are diffusively propagated in space. Another agent at a distance can sense not only the existence of the pheromone but also its spatial gradient. When we want the agent to get closer to the source position of the pheromone, we design a rule of the agent motion so that it moves toward the direction of ascending the field gradient.

In the ant foraging algorithm [13], ant agents that find "food" tell the food's location and the path between the food and their nest by depositing pheromones into the environment. When an ant agent finds food, it picks up some of them to bring it back to the nest. The ant releases an attractive pheromone on the way back to the nest. When a randomly walking other ant agent happens to be on the pheromone trail, it tracks the trail and finds the food in the end. The release of the pheromone is repeated on the way back to the nest, and thus more other ants are attracted to the trail.

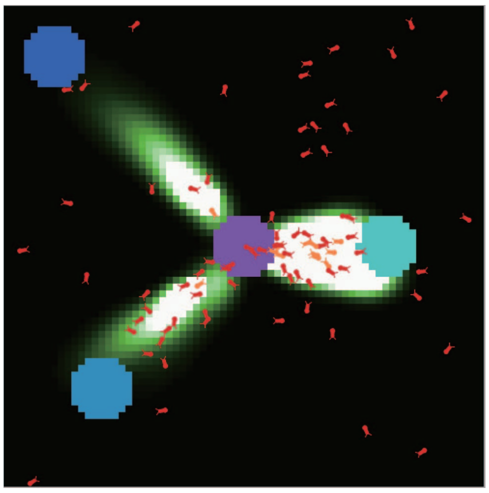
Fig. 1 The ant foraging algorithm simulated by NetLogo.

We have simulated the ant agents with the foraging algorithm with a standard ABM simulator NetLogo [14]; see Fig. 1. The purple circle is the ants' nest, the blue ones are food, and the white and green paths are the pheromones they release. The ant agents acted as expected by indirect interaction by means of pheromones: They autonomously find the blue circle representing food and release pheromones to communicate with other agents when they return to the purple circle representing the nest.

3. Agent-Based In-Situ Visualization of a Test Simulation

We incorporate the ant foraging algorithm into the in-situ visualization of 3-dimensional (3-D) simulations. Here we choose 3-D cellular automata (CA) [15–17] as a target simulation. Among myriads of possible rules for 3-D CA, we adopt the following rule [18] called Rule 4/4/5/M. Each cell has one of five cell states (state-0 to state-4) at each time step. A cell of state-0 is called empty. The state of each cell in the next time step is determined by the number of positive state cells in its 3-D Moore neighbor cells. (There are 26 Moore neighbors for every cell.) The empty cell becomes sate-4 when the neighbor number is 4. The state integer decrements for one every time step, e.g., state-4 becomes state-3, and so on. The state-1 becomes state-0 unless the neighbor number is 4.

This simple rule leads to amazingly complicated and unpredictable dynamics of the cells, which is why we picked up this CA as a test bench for our agent-based vi-

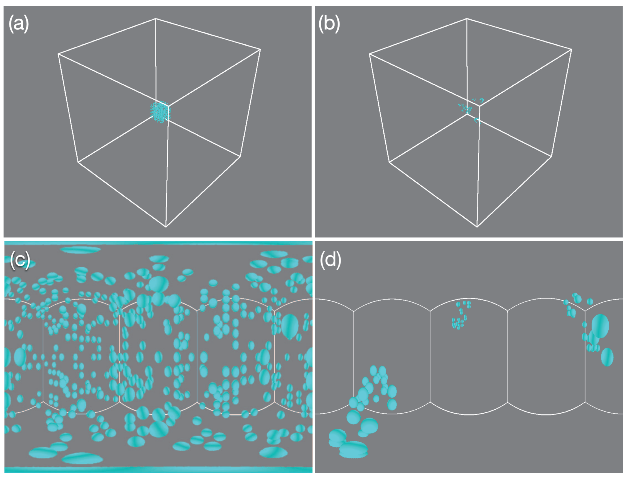

Fig. 2 3-D cellular automaton of Rule 4/4/5/M. In-situ visualized by VISMO. (a) Initial condition. (b) A snapshot after some time steps. (c) An omnidirectional view near the center of the simulation region just after the initial condition. (d) Another omnidirectional view after some time steps.

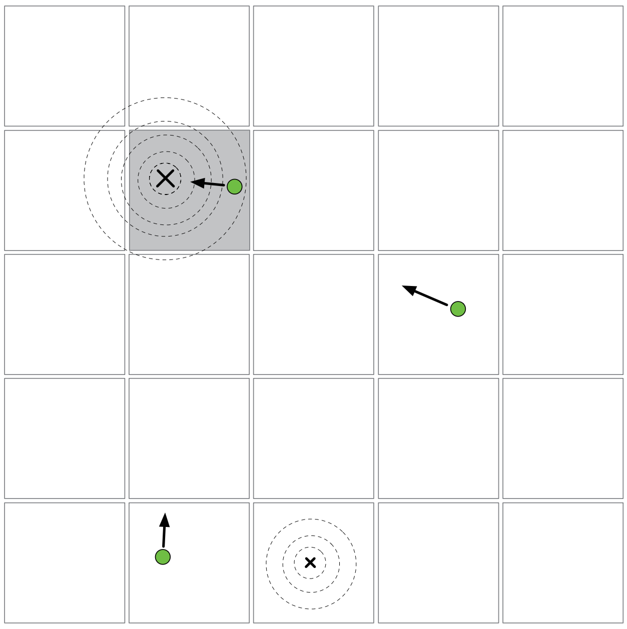

Fig. 3 A schematic view of the divided domain method for efficient tracking of the camera agents.

sualization. We have developed a 3-D CA code in Fortran 2003. The code is parallelized by the domain decomposition with MPI. The simulation region (the whole cells) is divided into subregions $M_x \times M_y \times M_z$. One MPI process is assigned to one subregion. A snapshot of the 3-D CA by this code is shown in Fig. 2 in which state-1 cells are shown by blue balls. The in-situ visualization is done by VISMO [19]. VISMO is an in-situ visualization library for the modern Fortran. We use a new feature of omnidirectional visualization of VISMO, which will be reported in another paper.

Figure 2 illustrates an overview of the 3-D CA. The panels Figs. 2 (a) and (b) are overviews visualized by the standard, i.e., directional visualization cameras, located outside the simulation region. The panels (c) and (d) are omnidirectional visualization images taken from a viewpoint. The viewpoint of (c) and (d) was manually selected so that the location is near the center of the simulation region, rather than by an automatic selection based on the pheromone algorithm described below.

In this paper, we describe our implementation strategy to combine the ant foraging algorithm into the 3-D CA code. The simulation and visualization results will be reported in a separate paper. The "food" of the camera agents corresponds to ROI in this CA.

In general, the definition of ROI depends on the simulation of the subject to be visualized. In the case of 3D-CA, the ROI was defined by the following algorithm. The state of each cell is binary, either *alive* or *non-alive*. The ROI as the visualization target is larger than the size of a single cell and smaller than the entire simulation domain. ROI in this work is a location where a group of cells, or a cell cluster, collectively exhibits intriguing variations in space and time. Lacking a rigorous definition of the intriguing cell cluster, we simply define a ROI as a cell cluster that have enough number of *alive* cells.

Each agent senses the diffused pheromone emanating from an ROI according to the original ant foraging algorithm. When the camera agent detects an ROI by means of the pheromone, the agent releases an additional pheromone. It continues to release the pheromone as it tracks the moving ROIs. The other camera agents are attracted to the trail of the pheromone. When a camera agent detects no pheromone, it randomly walks until it encounters the pheromone.

Because the amplitude of the pheromone field is too weak to sense for agents located very far from the ROI, we divide the whole simulation region into L subdomains to address the above problem. (The subdomains generally differ from the MPI subregion.) Here we call cells with state-1 to state-4 alive cells. We count the number of alive cells, A_ℓ, in ℓ-th subdomain ($1 \le \ell \le L$) and release the pheromone at the center of gravity of alive cells over the region of the ℓ-th subdomain if A_ℓ is larger than a critical value, say 100. Figure 3 schematically shows the case when $L = 5 \times 5 = 25$. The green circles are camera agents. Instead of following the gradient vector of the pheromone field, agents move toward the ℓ_{m}-th subdomain that has the maximum number of $A_{\ell_{\mathrm{m}}}$ ($A_{\ell_{\mathrm{m}}} = max\{A_\ell\}$). In Fig. 3, two subdomains contain enough number of alive cells, and the center of gravity in each subdomain is denoted by cross marks. The gray background depicts the subdomain with the largest number of alive cells A. When an agent enters into the ℓ_{m}-th subdomain (gray in Fig. 3) or the agent happens to have been inside the subdomain at the moment, the agent starts moving toward the gradient vector of the pheromone and releasing the pheromone by itself as a trail.

When there is no ROI at all in the simulation region, the agents move randomly, waiting for the appearance of the pheromone in some subdomain.

In order to avoid the overcrowding of agents, we will implement a repulsive pheromone. The repulsive pheromone acts oppositely from the attractive pheromone described above. Each agent releases the repulsive pheromone when it gets closer to an ROI. Other agents move in the opposite way as the attractive pheromone; it moves toward the negative gradient direction of the repulsive pheromone field. Another paper will also report the results of this repulsive pheromone method.

4. Summary

We propose the concept of agent-based in-situ visualization as a new approach to the practical in-situ visualization of large-scale parallel computer simulations. In this method, visualization points or cameras are autonomous entities. The camera agents fly in the simulation to find and get closer to local regions of interest (ROI) under interactions with the environment (pheromones) and other agents.

Acknowledgments

This work was supported by JSPS KAKENHI (Grant Numbers 22H03603, 22K18703).

[1] H. Childs, J.C. Bennett and C. Garth, editors, *In Situ Visualization for Computational Science* (Springer International Publishing, 2022).

[2] A. Kageyama and T. Yamada, An approach to exascale visualization: Interactive viewing of in-situ visualization, Comput. Phys. Commun. **185**(1), 79 (January 2014).

[3] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D.H. Rogers and M. Petersen, An Image-Based approach to extreme scale in situ visualization and analysis, 2014.

[4] P. O'Leary, J. Ahrens, S. Jourdain, S. Wittenburg, D.H. Rogers and M. Petersen, Cinema image-based in situ analysis and visualization of MPAS-ocean simulations, Parallel Comput. **55**, 43 (July 2016).

[5] A. Kageyama and N. Sakamoto, 4D street view: a video-based visualization method, PeerJ Comput Sci. **6**, e305 (November 2020).

[6] A. Kageyama, N. Sakamoto, H. Miura and N. Ohno, Interactive exploration of the In-Situ visualization of a magnetohydrodynamic simulation, Plasma Fusion Res. **15**, 1401065 (2020).

[7] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent and J. Weaver, Google street view: Capturing the world at street level, Computer **43**(6), 32 (June 2010).

[8] W. Yan, R. Sakai and A. Kageyama, Toward agent-based in situ visualization. In Chang By and C Choi, editors, Methods and Applications for Modeling and Simulation of Complex Systems, CCIS, 1636, October 2022.

[9] A. Grignard and A. Drogoul, Agent-Based visualization: A Real-Time visualization tool applied both to data and simulation outputs, In The AAAI-17 Workshop on Human-Machine Collaborative Learning, pages 670-675, 2017.

[10] G. Beni and J. Wang, *Swarm intelligence in cellular robotic systems*, In P Dario and others, editors, *Robots and Biological Systems: Towards a New Bionics?* (Springer-Verlag Berlin Heidelberg, 1993) pp. 703-712.

[11] U. Wilensky and W. Rand, *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo* (MIT Press, April 2015).

[12] J.-L. Deneubourg, S. Aron, S. Goss and J.M. Pasteels, The self-organizing exploratory pattern of the Argentine ant, J. Insect Behav. **3**(2), 159 (March 1990).

[13] M. Dorigo and C. Blum, Ant colony optimization theory: A survey, Theor. Comput. Sci. **344**(2-3), 243 (November 2005).

[14] S. Tisue and U. Wilensky, NetLogo: Design and implementation of a multi-agent modeling environment, In Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence, 2004.

[15] S. Wolfram, *A New Kind of Science* (Wolfram Media, 2002).

[16] A. Ilachinski, *Cellular Automata: A Discrete Universe* (World Scientific, 2001).

[17] J.L. Schiff, *Cellular Automata: A Discrete View of the World* (Wiley, December 2007).

[18] softologyblog, 3D cellular automata, https://softologyblog.wordpress.com/2019/12/28/3d-cellular-automata-3/, December 2019. Accessed: 2023-1-8.

[19] N. Ohno and H. Ohtani, Development of In-Situ visualization tool for PIC simulation, Plasma Fusion Res **9**, 3401071 (2015).