Toward Agent-Based In Situ Visualization

Yan Wang[1(✉)], Ren Sakai[2], and Akira Kageyama[3]

[1] Graduate School of System Informatics, Kobe University, Kobe 657-8501, Japan
190x701x@stu.kobe-u.ac.jp
[2] Faculty of Engineering, Kobe University, Kobe 657-8501, Japan
[3] Graduate School of System Informatics, Kobe University, Kobe 657-8501, Japan

Abstract. In situ visualization is becoming an essential method used for high-performance computing. For effective in situ visualization, a viewpoint should be placed close to a key spot or volume-of-interest (VOI). In order to track unpredictable motions of VOI in simulations, we propose to introduce agent-based modeling to the in-situ visualization, in which agents are autonomous cameras, and their environment is the simulation. As a demonstration experiment of the agent-based in situ visualization, we put a camera agent to 3D cellular automata. The camera agent successfully tracks a VOI of cells in highly complex time development.

Keywords: HPC · In situ visualization · Agent-based model · Agent-based visualization · Cellular automata

1 Introduction

In situ visualization is becoming an important research topic in high-performance computing (HPC), because it enables the analysis of simulation data without reducing the spatiotemporal resolution [7]. One challenge with in situ visualization is the method used to identify a local critical region in the whole simulation space, or volume of interest (VOI), where intensive visualizations are to be applied. In large-scale computer simulations of complex phenomena, however, it is almost impossible to know in advance when and where essential phenomena will occur.

In 2014, we proposed an in situ visualization approach that enables interactive analysis of VOI after simulation [14]. The key idea is to apply multiple in situ visualizations from fixed viewpoints at once before applying the interactive exploration of video dataset on PCs. (We focus on 3D simulations with time development.) The visualization cameras for recording of the video dataset were assumed to be primarily placed on 2D surfaces such as a sphere. Similar approach based on images to in situ visualization is Cinema [1,20].

By generalizing our video-based method, we proposed "4D Street View" [12,13], where we placed omnidirectional cameras using a full (=4π steradians) field of view. The omnidirectional cameras are placed in various forms in the simulation region such as on curves (1D), on surfaces (2D), or in the whole simulation region (3D). The viewpoint and viewing direction can be interactively changed

B.-Y. Chang and C. Choi (Eds.): AsiaSim 2021, CCIS 1636, pp. 3–10, 2022.
https://doi.org/10.1007/978-981-19-6857-0_1

afterward as in the Google street view [2] using an application program for PC, called 4D street viewer.

This study proposes another complementary approach to in situ visualizations to focus on VOI. It enables automatic tracking of the unpredictable behavior of VOI, such as sudden emergence/disappearance and random motion. This is achieved by integrating the agent-based model (ABM) into in situ visualizations. In this agent-based in situ visualization, agents are visualization cameras, and they autonomously identify and track VOI by following prescribed rules and applying in situ visualizations.

Our long-term goal is to implement the agent-based in situ visualization as a set of visualization cameras or "camera swarm". Toward the goal, this study presents a single camera as an element of the autonomous camera agent.

2 Related Work

Multiple in situ visualization approaches for HPC have been proposed. Temporal caching [9] is to temporarily store simulation outputs in a fast storage system for later events triggered based on the stored data. The particle data approach [15] saves view-independent particle data for the later application of particle-based rendering [21]. Proxy image [26,27,32] is a method that uses the intermediate representation of data.

Several libraries and frameworks for in situ HPC visualization have been developed, including ParaView Catalyst [3], VisIt libsim [30], ISAAC [17], Embree [29], OSPray [28], and VISMO [18,19]. ADIOS [16] is an adaptable data I/O framework, enabling asynchronous communication between simulation and visualization. SENSEI [4] is a generic in situ interface, providing a portable interface for other in situ infrastructures such as Catalyst, libsim, and OSPray.

The application of ABM to information visualization in general was proposed by [11]. They coined the term agent-based visualization (ABV). The agent-based in situ visualization proposed in this study is an application of ABV to in situ visualization for HPC.

In computer graphics, the automatic setting of camera path is an important topic having a long history [8,10,25]. Here, a relatively simple algorithm for the camera agent motion is used because the camera agent is required to autonomously respond to ever-changing simulation data.

3 Camera Agent

In general, an ABM consists of two components; environment and autonomous entities called agents [31]. Each agent interacts with the environment and other agents, following simple rules. For our proposed agent-based visualization, an agent is a visualization camera that autonomously changes its position and viewing direction. Unlike the omnidirectional cameras scattered in the 4D street view, the agent camera is directional one with a smaller field-of-view than 4π steradians. The environment is the simulation space and the physical variables

distributed there. The camera is designed to track VOI and visualize the phenomena therein. Here we focus on the behavior of a single agent.

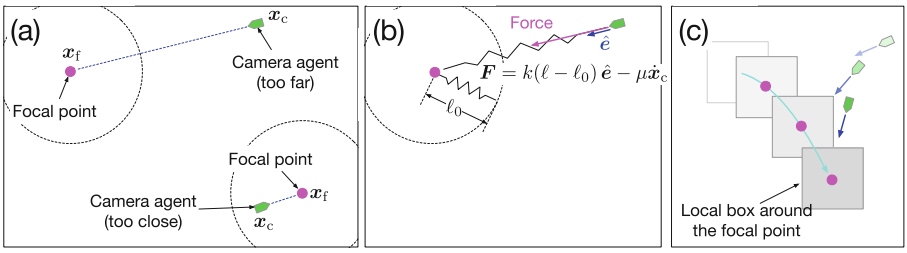

Fig. 1. (a) The agent is pulled or pushed as per the distance to its (fixed) focal point. (b) The agent moves following the mass-spring-damper model. (c) The focal point in the next time step is re-calculated from local environment around it (local box).

Figure 1 shows the rules for camera-agent motion: (i) First, the agent calculates the center of VOI called focal point from the environment [the red points in (a)]; (ii) If the distance to the focal point is larger than a reference length ℓ_0, the camera agent is then pulled to the focal point; (iii) Otherwise, the agent is pushed away from the focal point.

To implement rules (ii) and (iii), we adopt a simple mass-spring-damper model with dual time stepping. The camera agent follows an equation of motion with its intrinsic time τ, which is independent from simulation's time t. Assuming that the mass of the agent $m = 1$, we adopt the equation of motion for the position vector of the camera agent $\boldsymbol{x}_c$ as follows [Fig. 1(b)]:

$$\frac{d^2\boldsymbol{x}_c}{d\tau^2} = k(\ell - \ell_0)\,\hat{e} - \mu\frac{d\boldsymbol{x}_c}{d\tau}, \tag{1}$$

where k and μ are spring constant and friction coefficient; ℓ is the distance between the focal point $\boldsymbol{x}_f$ and the camera agent, or $\ell = |\boldsymbol{x}_f - \boldsymbol{x}_c|$; and $\hat{e}$ is unit vector $\hat{e} = (\boldsymbol{x}_f - \boldsymbol{x}_c)/\ell$. We numerically integrate Eq. (1) for τ, assuming that the focal point $\boldsymbol{x}_f$ is fixed during the integration. In other words, time t stops during the τ's integration. On the other hand, the focal point moves according to the environmental change, or the development of the simulation in t, while the motion of the agent by Eq. (1) is suspended. We alternately apply the dual time integrations. We set ℓ_0, a free parameter in this method, as $\ell_0 = 30$, with the unit length being the cell size.

The camera agent assumes a part (or sometimes all) of the simulation region called local box, which is defined around the focal point [Fig. 1(c)]. As the environment changes (or as the simulation progresses in time t), the local box range, and accordingly its central focal point, moves. According to the above procedure, the camera agent smoothly tracks the motion of VOI, almost always keeping the appropriate distance ℓ_0 [Fig. 1(c)]. The VOI tracked by the camera agent depends

on the initial position of the agent. This uncertainty of VOI will be resolved if we introduce multiple agents in future.

4 Application Tests

Agent-based in situ visualization is a general idea that can be applied to various kinds of complex simulations, such as fluid turbulence simulations. Here we choose 3D cellular automata (CA) as test target simulations because they potentially exhibit unpredictable behavior.

4.1 3-D Cellular Automata

We consider 3D cartesian lattice cells with discrete (integer) states and a simple ruleset to change the states in the next time step. The rules are local, i.e., the next state of a cell is determined by its state and that of its neighbors. CA is known to mimic complex phenomena observed in nature [24]. The complex time evolution of 3D CA described below makes them suitable applications of the proposed agent-based in situ visualization method.

In the following, we call a cell empty, when its state = 0, and alive when its state = 1. The total number of possible states is n: The state of a cell is one of $\{0, 1, 2, \ldots, n-1\}$. The total number of alive cells in neighbors is m.

We adopt the $Rule\,[\alpha/\beta/n/\gamma]$ notation to specify a CA rule set, where α is an integer or a set of integers for m to make an empty cell alive (born); β is an integer or a set of integers for m to keep an alive cell being alive; and γ is either N (Neumann neighbor) or M (Moore neighbor). When m does not match α (when the cell is empty) or β (when the cell is alive), or the cell is neither alive nor empty (state > 1), 1 is added to the state integer modulo n. We will present the situ visualizations of $Rule\,[4/4/5/M]$ and $Rule\,[5/4, 5/2/M]$ below.

We developed a 3D CA code in C++ and incorporated the in situ visualization function using a single camera agent in the code. Our simulation code executes any CA model described by the $Rule[\alpha/\beta/n/\gamma]$ with periodic boundary conditions in all three (x, y, and z) directions. The program is assumed to be executed on a supercomputer system as a batch job. Although the code is not parallelized, it will be done soon.

We place spheres at non-empty cells and the sphere color depends on the state integer of the cell. Kyoto Visualization System (KVS) [22], which is a visualization development framework for C++, was employed for the in-situ rendering of the spheres on π-computer system of Kobe University, comprising 16 nodes of AMD APYC CPU (512 cores in total). Results of the in situ visualization or the output of KVS are stored as a sequence of image files on the hard disk drive system. These images are then combined into a video file playable on PCs.

4.2 CA of $Rule\,[4/4/5/M]$

First, we demonstrate the results of in situ visualization of CA with $Rule\,[4/4/5/M]$, which leads to highly complicated dynamics of cells. We could not find

literatures describing this CA. We recommend a YouTube video [23] to comprehend the brilliant and impressive developments of cells. The *Rule* [4/4/5/*M*] appears at the beginning of the video.

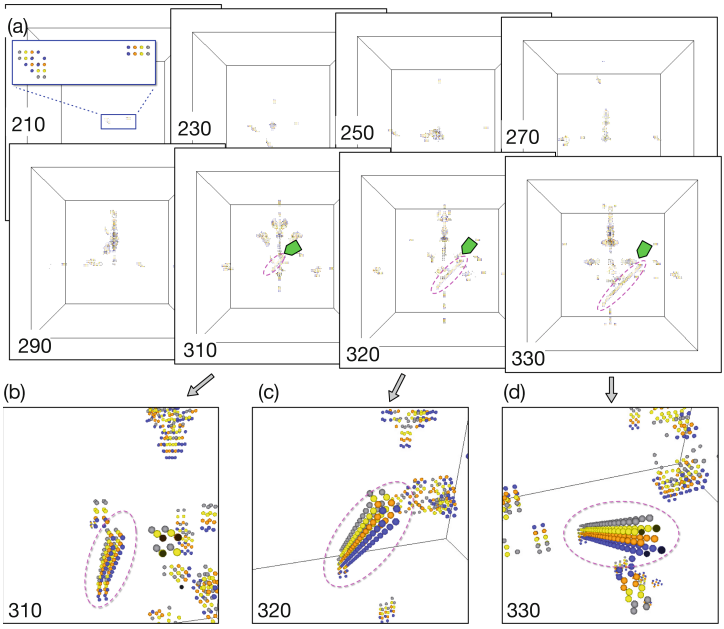

Fig. 2. (a) Snapshots of a 3-D CA. Two clusters collide after 210 time steps and highly complicated structures are then observed. Observe that a camera agent (schematically shown by the blue glyph) tracks a bar-like object (enclosed by a magenta curve). (b), (c), and (d) are images of in the situ visualization by the camera agent. (Color figure online)

Figure 2(a) shows a sequence of snapshots of the CA from 210–330 steps. The cell lattice size is $100 \times 100 \times 100$. At 210 steps, two clusters of non-empty cells are observed (magnified view in the blue box). They collide later and break in multiple fragments at 230 steps. Then, the scattered fragments undergo additional mutual collisions from 250 steps and above. At the 310th step, we observe an emergence of rod-like structures (enclosed by the magenta-dashed line).

Here we define VOI as the center of gravity of alive cells in the local box. When there is no alive cell in the local box, the camera agent does not move, waiting for a change. When a cluster of cells goes into the local box, the camera agent notices its entrance and starts tracking (green glyph in Fig. 2(a)). The viewing direction is toward the center of gravity. In spite of its simplicity, the rule enables the camera agent to follow a bar-like object, as shown in Figs. 2(b), (c), and (d).

4.3 CA of *Rule* [5/4, 5/2/*M*]

The second example of CA to which the agent-based in situ visualization is applied is *Rule* [5/4, 5/2/*M*]. This CA is one of the extensions of Conway's game of life in 3-D [5,6], which enables a "glider," an oscillating structure of a relatively few cells, to translate in the space. In this CA calculation, we intentionally set an initial condition, such that a single glider exists, to confirm the agent's trackability in the event of a sudden change of VOI. The glider goes through a boundary plane and re-appears from the opposite plane because of the periodic boundary conditions. These kinds of abrupt appearances and disappearances should be tracked by a camera agent in complex simulations.

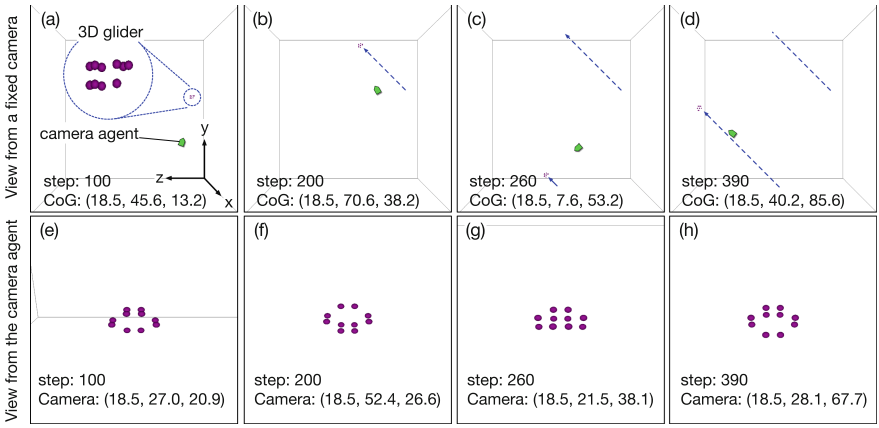

Fig. 3. Agent-based in situ visualization of 3D game of life. (a)–(d): A glider moves in the simulation region under the periodic boundary condition and a camera agent (green glyph) tracks the glider. (e)–(h): Images taken by the camera agent. The cell size is $70 \times 80 \times 90$. (Color figure online)

Figures 3(a)–(d) show the glider's translation (a group of purple cells). The blue arrow denotes the glider trajectory. The green glyph shows the position of the camera agent. (The blue arrow and green glyph are shown for the explanation, they do not appear in the CA computation.) The agent notices the disappearance and appearance of the glider after (b) and before (c), respectively. The successful tracking of the glider's "teleportations" subsequently continues after (d).

Figures 3(e)–(h) show images obtained by the camera agent's in situ visualization at designated time steps corresponding (a)–(d). The glider is recorded at the center of the images, as shown in these figures.

5 Summary

We propose agent-based in situ visualization for effective in situ visualization of HPC. Toward the full capacity of agent-based modeling of visualization cameras, we developed a single camera agent in this paper. We have shown that the camera

agent autonomously tracks VOI in 3D CA, applying in situ visualizations of the VOI during a batch job simulation on an HPC system. Based on the single agent proposed in this paper, we will study multiple agents in the future, expecting the emergence of collective order as observed in general ABMs.

The agent-based in situ visualization is complementary to the omnidirectional stationary cameras in the 4D street view. For the effective analysis of HPC data, we will combine autonomous camera agents and omnidirectional stationary cameras in the future studies.

Acknowledgments. This work was supported by Grant-in-Aid for Scientific Research (KAKENHI) 17H02998. We thank Dr. Naohisa Sakamoto for valuable technical advice and for fruitful discussions.

References

1. Ahrens, J., et al.: In Situ MPAS-ocean image-based visualization. In: SC14 International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 3–6 (2014)
2. Anguelov, D., et al.: Google street view: capturing the world at street level. Comput. (Long. Beach. Calif). **43**(6), 32–38 (2010)
3. Ayachit, U., et al.: ParaView catalyst. In: Proceedings of First Work. Situ Infrastructures Enabling Extreme-Scale Analysis and Visualization - ISAV2015, pp. 25–29. ACM Press (2015)
4. Ayachit, U., et al.: The SENSEI generic in situ interface. In: Proceedings of ISAV 2016 2nd Working Situ Infrastructures Enabling Extreme - Held Conjunction with SC 2016 International Conference High Performance Computing Networking, Storage Analysis, pp. 40–44 (2017)
5. Bays, C.: Candidates for the game of life in three dimensions. Complex Syst. **1**, 373–400 (1987)
6. Bays, C.: A note about the discovery of many new rules for the game of three-dimensional life. Complex Syst. **16**, 381–386 (2006). http://www.cse.sc.edu/bays/CAhomePage
7. Bennett, J.C., Childs, H., Garth, C., Hentschel, B.: In Situ Visualization for Computational Science. vol. 18271, pp. 1–43. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-81627-8
8. Chen, Z., Zhou, J., Sun, R., Kang, L.: A new evolving mechanism of genetic algorithm for multi-constraint intelligent camera path planning. Soft Comput. **25**(7), 5073–5092 (2021). https://doi.org/10.1007/s00500-020-05510-6
9. Demarle, D.E., Bauer, A.: In situ visualization with temporal caching. Comput. Sci. Eng. **23**, 25–33 (2021)
10. Drucker, S.M., Zeltzer, D.: Intelligent camera control in a virtual environment. In: Graph. Interface 1994, pp. 190–199. Banff, Canada (1994)
11. Grignard, A., Drogoul, A.: Agent-based visualization: a real-time visualization tool applied both to data and simulation outputs. In: AAAI-17 Workshops Human-Machine Collaborative Learning, pp. 670–675 (2017)
12. Kageyama, A., Sakamoto, N.: 4D street view: a video-based visualization method. Peer J. Comput. Sci. **6**, e305 (2020)
13. Kageyama, A., Sakamoto, N., Miura, H., Ohno, N.: Interactive exploration of the in-situ visualization of a magnetohydrodynamic simulation. Plasma Fusion Res. **15**, 1401065 (2020)

14. Kageyama, A., Yamada, T.: An approach to exascale visualization: interactive viewing of in-situ visualization. Comput. Phys. Commun. **185**, 79–85 (2014)
15. Kawamura, T., Noda, T., Idomura, Y.: In-situ visual exploration of multivariate volume data based on particle based volume rendering. In: 2nd Workshops of Situ Infrastructures Enabling Extreming Analysis and Visualization, pp. 18–22 (2016)
16. Lofstead, J., Klasky, S., Schwan, K., Podhorszki, N., Jin, C.: Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS). In: Proceedings of 6th International Workshops on Challenges Large Applications Distribution Environments, pp. 15–24 (2008)
17. Matthes, A., Huebl, A., Widera, R., Grottel, S., Gumhold, S., Bussmann, M.: In situ, steerable, hardware-independent and data-structure agnostic visualization with ISAAC. Supercomput. Front. Innov. **3**(4), 30–48 (2016)
18. Ohno, N., Kageyama, A.: In-situ visualization library for Yin-Yang grid simulations. Earth, Planet Space. **73**, 158 (2021)
19. Ohno, N., Ohtani, H.: Development of in-situ visualization tool for PIC simulation. Plasma Fusion Res. **9**, 341071 (2014)
20. O'Leary, P., Ahrens, J., Jourdain, S., Wittenburg, S., Rogers, D.H., Petersen, M.: Cinema image-based in situ analysis and visualization of MPAS-ocean simulations. Parallel Comput. **55**, 43–48 (2016)
21. Sakamoto, N., Nonaka, J., Koyamada, K., Tanaka, S.: Particle-based volume rendering. In: 6th International Asia-Pacific Symposium and Visualization, pp. 129–132. IEEE, February 2007
22. Sakamoto, N., Koyamada, K.: KVS: a simple and effective framework for scientific visualization. J. Adv. Simul. Sci. Eng. **2**, 76–95 (2015)
23. Softology: 3D Cellular Automata. http://www.youtube.com/watch?v=dQJ5aEsP6Fs
24. Wolfram, S.: A New Kind of Science. Wolfram Media, Champaign (2002)
25. Tharwat, A., Elhoseny, M., Hassanien, A.E., Gabel, T., Kumar, A.: Intelligent Bézier curve-based path planning model using chaotic particle swarm optimization algorithm. Cluster Comput. **22**(2), 4745–4766 (2018). https://doi.org/10.1007/s10586-018-2360-3
26. Tikhonova, A., Correa, C.D., Kwan-Liu, M.: Explorable images for visualizing volume data. In: Proceedings of IEEE Pacific Visualization Symposium 2010, PacificVis 2010, pp. 177–184 (2010)
27. Tikhonova, A., Correa, C.D., Ma, K.L.: Visualization by proxy: a novel framework for deferred interaction with volume data. IEEE Trans. Vis. Comput. Graph. **16**(6), 1551–1559 (2010)
28. Wald, I., et al.: OSPRay - a CPU ray tracing framework for scientific visualization. IEEE Trans. Vis. Comput. Graph. **23**(1), 931–940 (2017)
29. Wald, I., Woop, S., Benthin, C., Johnson, G.S., Ernst, M.: Embree: a kernel framework for efficient CPU ray tracing. ACM Trans. Graph. **33**(4), 8 (2014)
30. Whitlock, B., Favre, M.J., Meredith, S.J.: Parallel in situ coupling of simulation with a fully featured visualization system. In: Eurographics Symposium on Parallel Graphics and Visualization, pp. 101–109 (2011)
31. Wilensky, U., Rand, W.: An Introduction to Agent-based Modeling : Modeling Natural, Social, and Engineered Complex Systems with NetLogo. MIT Press, Cambridge (2015)
32. Ye, Y., Miller, R., Ma, K.L.: In situ pathtube visualization with explorable images. In: 13th Eurographics Symposium on Parallel Graphics and Visualization, pp. 9–16. Eurographics Association (2013)