Interactive Exploration of the In-Situ Visualization of a Magnetohydrodynamic Simulation

Akira KAGEYAMA, Naohisa SAKAMOTO, Hideaki MIURA[1] and Nobuaki OHNO[2]

Kobe University, Kobe 657-8501, Japan
[1]*National Institute for Fusion Science, Toki 509-5292, Japan*
[2]*University of Hyogo, Kobe 650-0047, Japan*

(Received 30 March 2020 / Accepted 26 April 2020)

We propose a new visualization method for large-scale computer simulations called "4D Street View". This method uses omnidirectional in-situ visualization cameras to record a simulation from multiple viewpoints. The omnidirectional video dataset produced by the simulation is then analyzed interactively. In this paper, we apply the method to a magnetohydrodynamics turbulence simulation with 64 viewpoints to prove that it is possible to achieve interactive in-situ visualization for supercomputer simulations. Three types of in-situ visualization by omnidirectional cameras (with 4π steradian solid angle) from each viewpoint are applied. At the end of the simulation, a video dataset with 192 omnidirectional video files is produced. We demonstrate that we can change the viewing position, angle, and applied visualization method interactively using a specially designed application program.

Keywords: computer simulation, scientific visualization, in-situ visualization, MHD simulation

DOI: 10.1585/pfr.15.1401065

1. Introduction

In large-scale simulation studies, it is common to apply visualization after a simulation. This style of visualization, called "post-hoc visualization", requires the transfer of large amounts of numerical data from a supercomputer system to a personal computer (PC) or a local workstation. Post-hoc visualization is starting to create a bottleneck in simulation research because of the cost of data transfer and storage.

To resolve this problem, another style of visualization, called "in-situ visualization", is attracting the attention of high performance computing (HPC) researchers [1–4]. In-situ visualizations are applied on supercomputer system while the simulation is running. Images, rather than numerical data, are produced as a result of the simulation, and researchers can analyze those images immediately. Various in-situ visualization libraries are available such as Catalyst [5] and libsim [6]. Generic in-situ interfaces, including ADIOS [7] and SENSEI [8], are also available for in-situ visualization.

The problem with in-situ visualization is that research loses its analytical interactivity. One cannot, for example, move or rotate a visualized object in the window to observe simulated phenomena from different angles. As interactivity is the key to effective visualization, the advent of a fully interactive in-situ visualization method is eagerly awaited.

There are several proposals to incorporate interactivity into the in-situ visualization process, such as the layered image method [9], and particle-based rendering [10, 11]. We have proposed a video-based method for HPC by which we can perform interactive viewing of the in-situ visualizations [12].

The idea for our video-based visualization method arose from an observation that current HPC systems have a large number of computing cores, and it is difficult to use them all effectively by simulation alone. The surplus computing cores can be used for visualization.

The emergence of highly advanced compression algorithms for image and video data is another factor that has influenced our proposed new method. The size of an image file, which is two-dimensional, is much smaller than a three-dimensional raw numerical data file. In this work, we further reduce the size of every still image produced during in-situ visualization by saving in the `PNG` format. The still images are then combined to create a video file in the MP4 codec [13]. We have been able to demonstrate that there is an excellent data-reduction effect while maintaining image quality.

In our video-based visualization method, we separate the visualization process into two stages; the recording stage and the exploration stage. The recording stage is the in-situ visualization on the supercomputer systems, and the exploration stage is conducted as a post-process on a PC. In contrast to regular in-situ visualizations, we apply multiple in-situ visualizations simultaneously from many viewpoints, using various visualization methods. Ahrens *et al.* [14] developed a similar image-based in-situ visualization method using multiple viewpoints implemented as

author's e-mail: kage@port.kobe-u.ac.jp

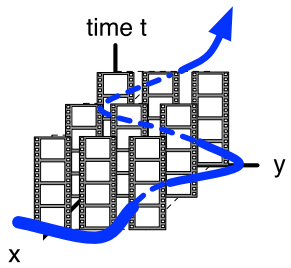

Fig. 1 Schematic view of the video dataset, which can be regarded as a field of videos that are distributed in 3D (x, y, z) space. It could also be described as a field of still images in 4D (x, y, z, t) space. Researchers specify a path (blue line) interactively in 4D space and an image sequence on the path is extracted by a dedicated application program called "4D Street Viewer".

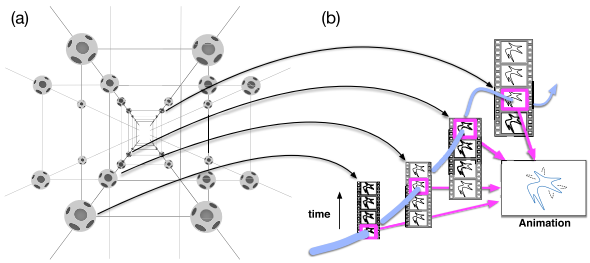

Fig. 2 Concept of 4D Street View. (a) Viewpoints for in-situ visualization are scattered in the simulation region. Visualization videos are generated for all positions. (b) A sequence of images is interactively extracted from the video files. The images are presented as an animation.

a Cinema viewer, which has been applied to various simulations [15, 16].

In our method, we apply the in-situ visualization to every viewpoint in small time intervals. After the simulation, an image sequence from each viewpoint is then compiled as a video file. In the exploration stage, we extract the sequence of images from the multiple video files and observe them as an animation.

The video files are organized as a dataset in which each video is associated with the position of the visualization camera. This dataset is identified as a distribution of video files scattered in the simulation's 3D space. As each video contains still images in a moment in time, the dataset can be described as a four-dimensional (4D) field of images, or as a set of still images distributed in 4D spacetime. We then analyze the image field interactively using a specially designed PC application program.

We have recently developed such an application, "4D Street Viewer", in the KVS framework [17]. One can effectively fly with 4D Street Viewer in the virtual 4D space as shown schematically by the blue path in Fig. 1. 4D Street Viewer extracts a sequence of images along the path and presents them on the window as a smooth animation.

In our previous paper [12], the rendering projection from each viewpoint was assumed to be standard one (i.e., the perspective projection with specified field-of-view). When a sufficient number of viewpoints are set in the recording stage, the viewpoints in the exploration stage can be changed interactively to almost any point in the simulation space. However, we cannot change the viewing direction interactively because the front vectors are fixed.

To overcome this problem, we have recently proposed an extension of our video-based visualization method [18]. In this extension, we place virtual omnidirectional cameras at every viewpoint, as shown schematically in Fig. 2 (a). With these cameras, we record the panoramic (full solid angle of 4π steradian) view and store them as omnidirectional video files [see Fig. 2 (b)]. The omnidirectional video dataset thus constructed enables us to make interactive changes to the viewing direction, as well as to the viewing position. We call this extended video-based visualization method "4D Street View" because it is reminiscent of Google Street View [19]. In computer graphics, the continuous function of pixel color as a function of the viewing position, angle, and time is called plenoptic function [20]. The video dataset in our method could be described as a discretization of the plenoptic function.

4D Street View is a simple idea, but its feasibility is not entirely obvious. For example, it may take unreasonable amount of time to implement multiple, omnidirectional, in-situ visualization for a parallel simulation program. It is also possible that it may not be practical to use 4D Street Viewer for interactive analysis of video datasets is with the current performance levels of PCs.

The purpose of this paper is to provide proof-of-concept for this new visualization method. We applied this new method to a magnetohydrodynamics (MHD) simulation with the Hall term. The simulation, which is described in the next section, is a full-fledged simulation that has already produced scientific results. We have applied the 4D Street View method to this simulation to prove its feasibility.

We demonstrate that a large-scale parallel simulation program with multiple, omnidirectional, in-situ visualization can be performed on a supercomputer, and that the output video files can be analyzed by 4D Street Viewer on a PC. Quantitative confirmation is important and will be the subject of future research; however, this paper focuses on proof-of-concept experiments.

A specific type of in-situ visualization library is required to allow a supercomputer system to run for 4D Street View. In contrast to standard in-situ libraries, it should be possible to apply in-situ visualization from tens, hundreds, or possibly thousands of viewpoints simultaneously. It should also be possible to set an omnidirectional (4π steradian) solid view angle from each point, and it should be compatible with massively parallel simulations. In this study, we demonstrate that we can make use of an

existing in-situ library to achieve specific in-situ visualizations during the recording stage of 4D Street View.

2. Recording Stage of 4D Street View

2.1 MHD simulation

The target simulation we have chosen to demonstrate the proof-of-concept of the 4D Street View is an MHD simulation with the Hall term. To investigate the Hall MHD turbulence, 3D simulation was performed by one of the authors (H. M.) [21–23]. In the simulation, the Hall MHD equations were discretized in space using the pseudo-spectral method with a 2/3-truncation de-aliasing technique under triple-periodic boundary condition. The Runge-Kutta-Gill method was adopted for temporal integration. The initial velocity and magnetic fields are given by the energy spectrum proportional to $k^2 \exp(-k/k_0)^2$ ($k_0 = 2$ in this study) and random phases of the Fourier coefficients. The fluid experiences instabilities and nonlinear energy transfer between numerous scales until it reaches a fully developed turbulence. For our experiments, we used the spatial resolution of 256 points in each direction.

We performed several simulations with different numbers of omnidirectional cameras. In this paper, we demonstrate a case in which we set the cameras as a $4\times4\times4 = 64$ rectilinear configuration, as shown in Fig. 3.

2.2 Multiple in-situ visualizations using VISMO

The requirements for a particular in-situ visualization library for 4D Street View are summarized below: (1) Parallel visualization: It should be possible to be called from parallel simulation programs. (2) Multi-camera visualization: It should be possible to set multiple (at least tens of) viewpoints. (3) Omnidirectional visualization: It should be possible to project the full 4π steradian view from each point.

In this study, we take an existing in-situ visualization library called VISMO (short for Visualization Module), and use it as the library for 4D Street View. We have found that it is relatively easy to satisfy the above requirements.

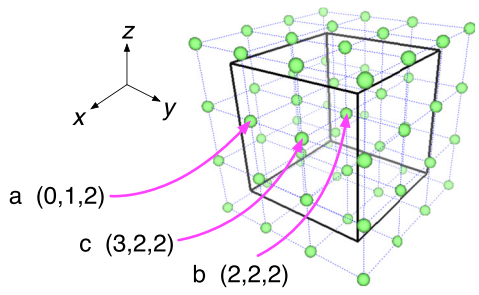

Fig. 3 Configuration of the omnidirectional cameras (light green balls) in this study. Sixty-four cameras were scattered in the simulation space with $4 \times 4 \times 4$ rectilinear configuration. The black thick lines denote the simulation's boundary. The three balls labeled as a, b, and c with grid positions denote the viewpoints in the fly-through demonstration shown in Figs. 10 (a), (b), and (c), respectively.

VISMO is a library for in-situ visualizations developed by one of the authors (N. O.) [24]. It is written in Fortran 2003 and is provided as a Fortran module. It is intended to be used with simulation programs written in Fortran 2003 or subsequent versions. Fundamental visualization methods, such as isosurface, volume rendering, color contour in slice planes, arrow glyphs, and streamlines, are available in VISMO, which is parallelized by MPI and OpenMP.

There are two possible approaches to in-situ visualization. One is a synchronous approach, in which the simulation and visualization computations are applied alternately in the same computational nodes, and the other is an asynchronous approach, in which the two computations are applied simultaneously in different computational nodes. Here we have adopted the former approach. We call VISMO functions from the MHD simulation code and the functions run on the same computer nodes as the simulation.

VISMO is a parallel rendering library. One MPI process is responsible for one spatial domain in a domain-decomposed simulation. When a VISMO visualization function is called in a particular MPI process, partial visualization is applied to the simulation domain. The resulting partial image contains the depth data for each pixel. The partial images are then integrated automatically into a complete image by VISMO.

As the visualization methods are all implemented by ray-casting-based algorithms in VISMO, no additional library such as OSMesa, is required. The only library VISMO calls, other than Fortran compiler, is `libpng` when the user attempts to save the images in `PNG` format. VISMO also allows users to save images in `BMP` or `PPM` formats.

Following the cube-mapping procedure [25, 26], we placed a virtual cube around a viewpoint in such a way that the cube's center was located on the point, with the edges parallel to the x, y, and z axes [see Fig. 4 (a)]. We then applied the standard perspective projections from the point to the surfaces of the cube (i.e., six regular squares with a 90 degree of field-of-view) using VISMO.

Figure 4 (b) shows an example of the six images viewed at a specific time from a viewpoint in the MHD simulation. The images show an isosurface visualization of the squared electric current density $j^2 = |\nabla \times \boldsymbol{B}/\mu_0|^2$, where $\boldsymbol{B}$ and $\mu_0 = 4\pi \times 10^{-7}$ are the magnetic field and the permeability of vacuum, respectively. We call this visualization `vis1`. In this study, we set the resolution for the perspective projection for the six regular squares as 512×512 pixels. The sizes of the six images (`PNG` format) in Fig. 4 (b) range from 251×10^3 B (for the right image) to 353×10^3 B (for the bottom image), with an average of 300×10^3 B.

We combined the six images into a single file as an omnidirectional image by applying equirectangular mapping. The image was also saved in the PNG format. Fig. 4 (c) shows the omnidirectional image that was combined from the six images in Fig. 4 (b). The pixel size of Fig. 4 (c) was $2,048 \times 1,536$. We used this resolution for all other viewpoints. The file size of Fig. 4 (c) was 2.62 MB.

In the test simulation, we applied the omnidirectional in-situ visualizations for 50 times. As a result, we obtained 50 frames per viewpoint. After the simulation, we combined the 50 frames to produce a single video file using MP4 codec [13]. The size of the video file for the viewpoint shown in Fig. 4 (b) and (c) was 20.3 MB. It should be noted that this value (20.3 MB) is approximately 15% of the theoretical size of one still image (2.62 MB) multiplied by the number of frames (50). The size reduction is due to compression with the MP4 codec.

As there were 64 viewpoints in this test, we obtained 64 videos for each visualization. The total size of all the videos for `viz1` was 519 MB.

The simulation and in-situ visualization were performed on a supercomputer system Fujitsu PRIMEHPC FX100 (SPARC64XIfx, 1CPU/node). Using 25 nodes, it took 300 minutes to complete one job. The multiple in-situ visualizations with 64 viewpoints for 50 frames took approximately 100 minutes, with 33% of CPU time is devoted to the multiple visualizations.

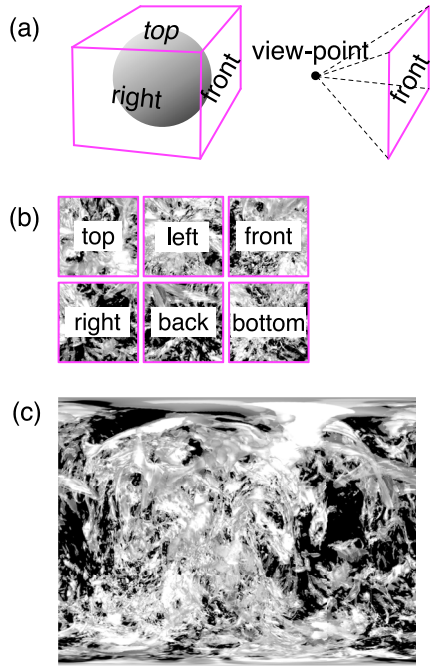

Fig. 4 Procedure to obtain an omnidirectional image from a viewpoint. (a) We apply six perspective projections from the point for the six surfaces of a cube whose center is on the point. (b) The six images are saved separately in the PNG format. (c) The images are combined using the equirectangular mapping to create a single omnidirectional image.

In 4D Street View, it is possible to switch visualization methods as in the post-hoc visualization programs, as long as the video dataset has been created beforehand with different visualization methods from the same viewpoints. In addition to `viz1` (i.e., the isosurfaces of the squared electric current density j^2, as shown in Fig. 4), we also applied two additional visualizations: `viz2` (isosurfaces for enstrophy density $e = |\nabla \times \mathbf{v}|^2$, with $\mathbf{v}$ being the flow velocity) and `viz3` (isosurfaces of both j^2 and e).

3. Exploration Stage of 4D Street View

Figure 5 shows an overview of 4D Street Viewer, which we have recently developed for the exploration stage of 4D Street View. As we will describe the design and implementation of this application software in another paper, we only briefly explain its functions.

Figure 6 shows a snapshot of the three visualizations: (a), (b), and (c) for `viz1`, `viz2`, and `viz3`, respectively. With 4D Street Viewer, clicking the mouse on a button in the window will cycle through the visualizations.

The sizes of the 64 video files in `viz2` and `viz3` are 744 MB and 737 MB, respectively. Therefore, the total size of the video datasets for the test simulation is $(519 + 744 + 737)\,\text{MB} = 2.00\,\text{GB}$, which is small relative to today's standard of drive storage capacity. We use a portable Solid State Drive (SSD) of 2 TB capacity to save video datasets. The 4D Street Viewer reads the video data directly from the SSD when the application is working.

Figure 7 shows snapshots of 4D Street Viewer's screen when the camera position and direction are fixed. First, the browser reads the default video file in the video dataset. Then, extracting a partial view from the omnidirectional image in the first frame (frame number 0) of the video file,

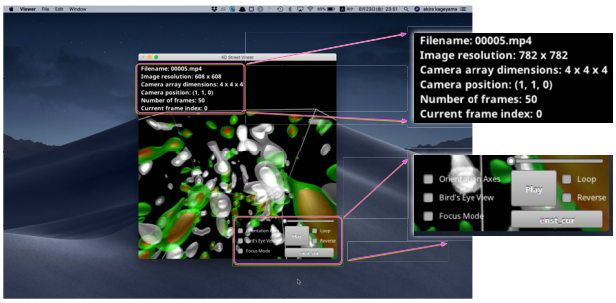

Fig. 5 A snapshot of 4D Street Viewer window. Part of an omnidirectional still image extracted from a video file is shown in the window. The text in the upper-left area indicates the file information. The buttons in the lower-right area are for the user interface. 4D Street Viewer has a standard function as a panoramic video player. When "Play/Stop" button is clicked, it toggles the playing mode of the video. The mouse-drag motion changes the viewing direction. There is also a reverse-play mode.

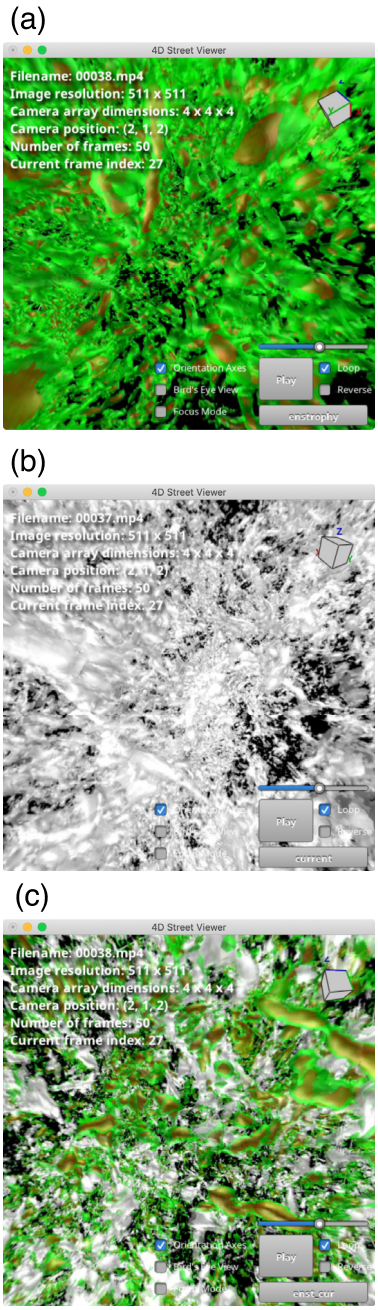

Fig. 6 Snapshots of 4D Street Viewer when visualization methods are changed consecutively by the user. (a) Isosurface visualization of enstrophy density e. (b) Isosurface visualization of squared electric current density j^2. (c) Isosurfaces of both e and j^2.

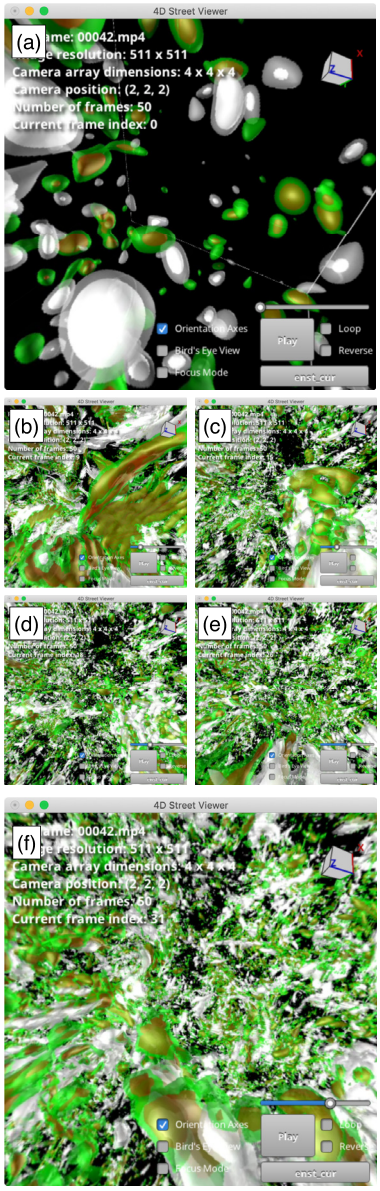

Fig. 7 Snapshots of 4D Street Viewer in the simple "Play" state. When the user click on the "Play" button in the browser window, an animation is displayed in the window. A smooth animation, extracted from the different cameras, are observed.

the browser presents it on the screen, as shown in Fig. 7 (a). When the user pushes the "Play" button in the lower-right corner of the window, it starts playing the animation (i.e., extracting the frame sequences, $1, 2, \ldots$, from the video file). This is similar to conventional video player programs. The development of turbulence in the MHD fluid from the initial laminar condition can seen in Figs. 7 (b) to (f).

In many (post-hoc) visualization software packages, the mouse-drag motion is used to rotate objects. To observe a visualized object from different view angles, the drag motion makes a rotation transformation on the object in the world coordinates, and new renderings are applied each time for a new angle. The view direction can also be changed using a similar mouse-drag motion in 4D Street Viewer. Figure 8 shows snapshots of a rotation process. The direction of the simulation space in the window is depicted by a small box in the upper-right corner in the window.

Contrary to post-hoc visualization software, the view rotation in 4D Street View is achieved by resampling from the omnidirectional image texture to the screen image. Fig-

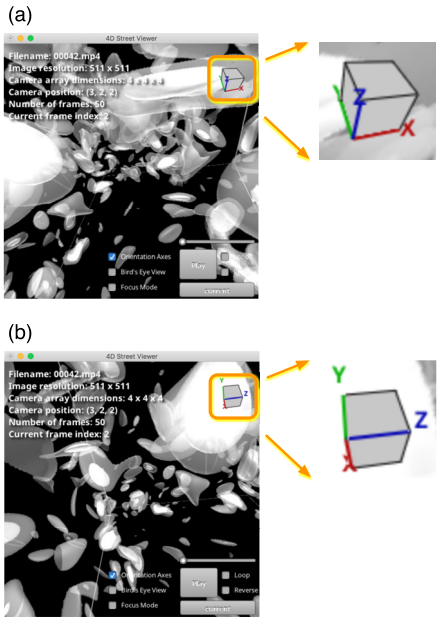

Fig. 8 Snapshots of 4D Street Viewer when the user applied the rotation transform. As other panoramic video player applications, the viewing direction can be altered using the mouse-drag motion. A small box with coordinate labels in the upper-right corner of the window indicates the direction of the view in the simulation coordinates.

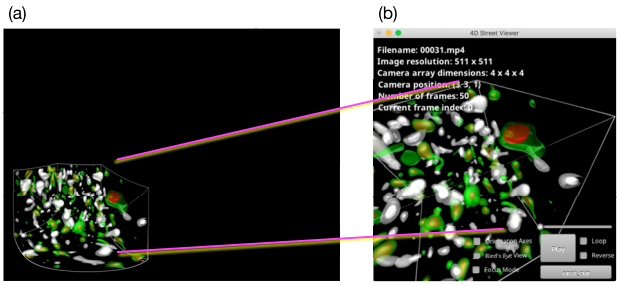

Fig. 9 A sample of the local view, shown in (b), from the corresponding omnidirectional image in (a).

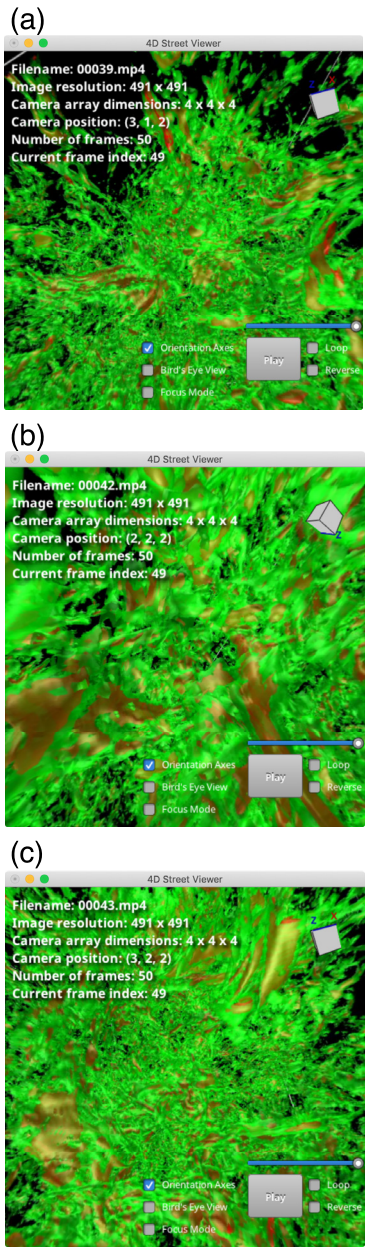

Fig. 10 Snapshots of 4D Street Viewer while the user was performing a fly-through in the simulation region. Isosurfaces of enstrophy density e are shown. It is possible to change the viewpoint using the mouse or keyboard. If an intriguing phenomenon is observed far from the current point in the presented video, it is possible to not only fly there and look around but also go backward/forward in time to establish the cause of the dynamics.

ure 9 (a) shows an example of an omnidirectional image taken by a camera, with `viz3` in its initial condition. As the camera is located outside the simulation region, the omnidirectional image looks highly distorted. Figure 9 (b) is a snapshot of the local image of the omnidirectional image shown in Fig. 9 (a), which has been sampled automatically by 4D Street Viewer.

Ensuring an interactive exploration is the most critical function of 4D Street Viewer. A sequence of images to be displayed as a movie on the PC screen is extracted from the video dataset. Modern PCs read files from a hard disk drive or SSD so fast that this part of the process does cause the bottleneck.

Initially, we were concerned about the processing time for cropping the omnidirectional image, followed by coordinate transformation (distortion correction) in accordance with the viewing direction. We solved this problem by utilizing the high-speed parallel processing of GPU, or GLSL shader, via KVS. We have succeeded in creating a smooth movie in real time, i.e., an interactive exploration, in response to changes in the viewpoint and the viewing direction specified by the user.

An interactive fly-through in the virtual 4D space is one of the key features of 4D Street View. The real-time

change in viewpoint is theoretically simple as it requires only a change in the input video file. All we have to do is to change the input video file of 4D Street Viewer. However, showing image sequences extracted from different video files, without delay, is technically challenging. 4D Street Viewer performs this function smoothly. Figure 10 shows snapshots while the user is flying through the simulation space. Panels (a), (b), and (c) are views from different omnidirectional cameras. Their locations are indicated in Fig. 3. It is possible to "fly" while the video is playing.

4. Summary

Today's simulation researchers typically store 3D numerical data at long intervals during a simulation to suppress the size of the data being stored and transferred. Therefore, it is practically impossible to visually analyze while maintaining a high temporal resolution.

With an ideal HPC visualization, it would be possible to analyze visualization videos of 3D data at a high frame rate. 4D Street View is our proposed approach to achieving such an ideal visualization. During the recording stage, we perform multiple in-situ visualizations with omnidirectional cameras in a supercomputer simulation. We then construct a dataset from the visualization video files, which could be highly compressed using advanced compression algorithms. During the exploration stage, we analyze the dataset interactively using the dedicated application program, 4D Street Viewer.

The recording stage for 4D Street View requires a specific type of in-situ visualization tool that can perform multiple in-situ visualizations with omnidirectional cameras. In this study, we demonstrated that an existing in-situ visualization library can be easily converted into a special in-situ visualization tool. We used VISMO [24] as the base visualization library. As VISMO is a parallel visualization library that accepts two or more viewpoints, it was straightforward to apply multiple in-situ visualizations from different points. It can be assumed that many other in-situ libraries have the same functionality.

On the other hand, there are few in-situ libraries with the omnidirectional visualization function. (It has recently been announced that ParaView will support it.) We have shown that by applying the regular perspective projections six times from each viewpoint, it is possible to obtain an omnidirectional visualization image that can be used in the exploration stage.

As a test, we applied the omnidirectional in-situ visualizations with 64 viewpoints in an MHD turbulence simulation. In this demonstration, we performed three different visualizations from all omnidirectional cameras. As a result, we obtained a video dataset containing three sets of video files associated with their viewpoint information. In this experiment, 67% of total CPU time was spent on the MHD simulation and 33% on the in-situ visualization.

We saved still the images in the `PNG` format and then combined them to create video files in the MP4 format. Each video contained 50 frames of the omnidirectional image in the equirectangular projection. The resolution of each frame in the video was $2,048 \times 1,536$ pixels. The total size of the video dataset (three MP4 video sets) is 2.00 GB, which is sufficiently small.

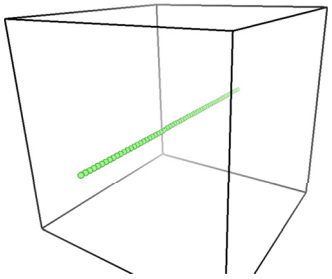

Fig. 11 Another configuration of the omnidirectional cameras used for this study. Sixty-four cameras were placed along the x-axis ($y = z = 0$). 4D Street View with this one-dimensional camera configuration creates a very smooth change of viewpoints while playing the video.

We used the dataset as an input for our recently developed 4D Street Viewer software. It can extract a sequence of images from the video dataset and present it as a smooth animation. The images are taken from a path in the virtual 4D space that is specified interactively by the user through 4D Street Viewer. We have confirmed that an interactive exploration of in-situ visualization is feasible.

Finally, we note that the configuration of omnidirectional cameras in the simulation space is not necessarily a box-shape, as shown in Fig. 3. We also set the same number of cameras (64) along the x-axis ($y = z = 0$), as shown in Fig. 11. We performed the omnidirectional in-situ visualization with this camera configuration for the same simulation. The video dataset obtained with this camera configuration provides very smooth change of viewpoints.

Acknowledgments

We would like to thank Prof. Chandrajit Bajaj for his technical pieces of advice, especially for suggesting using omnidirectional cameras. We thank Ms. Keiko Otsuji for her technical assistance. This work was supported by Grant-in-Aid for Scientific Research (KAKENHI) 16K12434, 16K00173, 17H02998, and 17K05734, "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" and "High Performance Computing Infrastructure" in Japan (Project ID: jh190006-NAJ), SCAT (Support Center for Advanced Telecommunications Technology Research) Foundation, I-O Data Foundation, The Okawa Foundation for Information and Telecommunications, and Tateishi Science and Technology Foundation. The numerical simulations and in-situ visualizations were performed on FUJITSU FX100 supercomputer "Plasma Simulator" of NIFS with the support

and under the auspices of the NIFS Collaboration Research Program (Grants No.NIFS15KNSS062).

[1] K.L. Ma, C. Wang, H. Yu and A. Tikhonova, "In-situ processing and visualization for ultrascale simulations," J. Phys.: Conference Series **78**, no. 1, 1 (2007).

[2] R.B. Ross, T. Peterka, H.-W. Shen, Y. Hong, K.-L. Ma, H. Yu and K. Moreland, "Visualization and parallel I/O at extreme scale," J. Phys.: Conference Series **125**, 012099 (jul 2008).

[3] K.-L. Ma, "In Situ Visualization at Extreme Scale: Challenges and Opportunities," IEEE Comput. Graph. Appl. **29**, Issue 6, 14 (2009).

[4] H. Childs, K.-L. Ma, H. Yu, B. Whitlock, J. Meredith, J. Favre, S. Klasky and N. Podhorszki, "In Situ Processing," in *High Performance Visualization: Enabling Extreme Scale Scientific Insight* (E.W. Betherl, H. Childs, and C. Hansen, eds.), ch. 9, pp. 171-198 (Chapman and Hall, 2012).

[5] U. Ayachit, A. Bauer, B. Geveci, P. O'Leary, K. Moreland, N. Fabian and J. Mauldin, "ParaView Catalyst: Enabling In Situ Data Analysis and Visualization," Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization - ISAV2015, pp. 25-29 (2015).

[6] B. Whitlock, J.M. Favre and J.S. Meredith, "Parallel in situ coupling of simulation with a fully featured visualization system," Eurographics Symposium on Parallel Graphics and Visualization, pp. 101-109 (2011).

[7] Q. Liu, J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, J.Y. Choi, S. Klasky, R. Tchoua, J. Lofstead, R. Oldfield, M. Parashar, N. Samatova, K. Schwan, A. Shoshani, M. Wolf, K. Wu and W. Yu, "Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks," Concurrency and Computation: Practice and Experience, **26**, 1453 (may 2014).

[8] U. Ayachit, B. Whitlock, M. Wolf, B. Loring, B. Geveci, D. Lonie and E.W. Bethel, "The SENSEI generic in situ interface," Proceedings of ISAV 2016: 2nd Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization - Held in conjunction with SC 2016: The International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 40-44 (2017).

[9] A. Tikhonova, C.D. Correa and M. Kwan-Liu, "Explorable images for visualizing volume data," IEEE Pacific Visualization Symposium 2010, PacificVis 2010 - Proceedings, vol. D, no. VIDi, pp. 177-184 (2010).

[10] N. Sakamoto, J. Nonaka, K. Koyamada and S. Tanaka, "Particle-based volume rendering," in 6th International Asia-Pacific Symposium on Visualization, pp. 129-132, IEEE (feb 2007).

[11] T. Kawamura, T. Noda and Y. Idomura, "In-situ visual exploration of multivariate volume data based on particle based volume rendering," Proceedings of ISAV 2016: 2nd Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization - Held in conjunction with SC 2016: The International Conference for High Performance Computing, Networking, Storage and Analysis, vol. D, pp. 18-22 (2017).

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

[13] A. Puri, X. Chen and A. Luthra, "Video coding using the H.264/MPEG-4 AVC compression standard," Signal Processing: Image Communication, vol. 19, pp. 793-849 (oct 2004).

[14] 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," International Conference for High Performance Computing, Networking, Storage and Analysis, SC, vol. 2015-Janua, no. January, pp. 424-434 (2014).

[15] 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 Computing **55**, 43 (2016).

[16] D. Banesh, J.A. Schoonover, J.P. Ahrens and B. Hamann, "Extracting, Visualizing and Tracking Mesoscale Ocean Eddies in Two-dimensional Image Sequences Using Contours and Moments," in Workshop on Visualization in Environmental Sciences (EnvirVis) (2017).

[17] N. Sakamoto and K. Koyamada, "KVS: A simple and effective framework for scientific visualization," J. Adv. Simulation. Sci. Eng. **2**, no. 1, 76 (2015).

[18] A. Kageyama and N. Sakamoto, "4D Street View: A Video-based Visualization Method," submitted to PeerJ Computer Science.

[19] 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**, no. 6, 32 (2010).

[20] E.H. Adelson and J.R. Bergen, "The Plenoptic Function and the Elements of Early Vision," in Computational Models of Visual Processing (M. Landy and J.A. Movshon, eds.), pp. 3-20, Cambridge, Mass: The MIT Press (1991).

[21] H. Miura and D. Hori, "Hall effects on local structures in decaying MHD turbulence," J. Plasma Fusion Res. **8**, 73 (2009).

[22] H. Miura, K. Araki and F. Hamba, "Hall effects and subgrid-scale modeling in magnetohydrodynamic turbulence simulations," J. Comput. Phys. **316**, 385 (2016).

[23] H. Miura, "Extended magnetohydrodynamic simulations of decaying, homogeneous, approximately-isotropic and incompressible turbulence," Fluids **4**, 46 (mar 2019).

[24] N. Ohno and H. Ohtani, "Development of in-situ visualization tool for PIC simulation," Plasma Fusion Res. **9**, 3401071 (2014).

[25] N. Greene, "Environment Mapping and Other Applications of World Projections," IEEE Comput. Graph. Appl. **6**, no. 11, 21 (1986).

[26] G. Sellers, R.S. Write Jr. and N. Haemel, *OpenGL Superbible: Comprehensive Tutorial and Reference* (Addison-Wesley, 2015).