EPJ Web of Conferences **302**, 11002 (2024) https://doi.org/10.1051/epjconf/202430211002
SNA + MC 2024

Extension of particle-based in-situ visualization for multi-point VR visualization

Takuma Kawamura[1,*], *Kazuya* Shimomura[1,**], *Tsukasa* Osaki[1,***], and *Yasuhiro* Idomura[1,****]

[1]Japan Atomic Energy Agency, Center for Computational Science & e-Systems, Kashiwa, Japan

Abstract. In the field of nuclear engineering, complex simulations on exascale supercomputers generate large-scale data. To facilitate efficient analysis of such simulation data, one needs to share them among scientists at remote locations. However, data I/O and data transfer for such large-scale data are quite costly. To resolve these issues, we developed a remote in-situ visualization system IS-PBVR based on the particle-based volume rendering (PBVR), which is suitable for parallel processing on modern supercomputers. In this study, we extend IS-PBVR for VR visualization on multiple client PCs, thus developing a multi-point remote VR visualization. We apply this technique to fluid simulations on GPU-based supercomputers and verify its utility by sharing in-situ VR visualization between multiple client PCs.

1 Introduction

In the field of nuclear engineering, complex simulations on exascale supercomputers generate large-scale data. To facilitate efficient analysis of such simulation data, one needs to share it among scientists at remote locations. To achieve this goal, one needs to address the following challenges. Firstly, data I/O and data transfer for large-scale simulation data are quite costly. Secondly, in analyzing complex three dimensional (3D) data, VR visualization, which allows intuitive understanding of 3D data, is needed. VR visualization is now facilitated by head-mounted displays (HMDs), which are utilized not only for entertainment but also for scientific visualization. Thirdly, multi-point visualization is needed to share visualization results and data exploration in it among scientists at remote locations. To resolve bottlenecks of data I/O and data transfer, in-situ visualization is a promising solution. However, the conventional polygon-based visualization methods often generate large polygon data, which is nearly the same size as the volume data, and thus, their in-situ visualization requires client-server communication of large polygon data or off-screen rendering. These approaches may not be sufficient for rendering at interactive frame rates (>60 frame per second (FPS)) and interactive changes of camera parameters, which are essential for VR visualization with head tracking on HMDs. Therefore, in-situ VR visualization and its extension for collaboration among multiple clients are challenging.

*e-mail: kawamura.takuma@jaea.go.jp
**e-mail: kazuya.shimomura@jaea.go.jp
***e-mail: osaki.tsukasa@jaea.go.jp
****e-mail: idomura.yasuhiro@jaea.go.jp

In this study, we develop a multi-point is-situ VR visualization technique using the particle-based visualization methods. The Particle-based Volume Rendering (PBVR) [1, 2] converts volume data into compressed visualization particle data, which is typically on the order of tens of MB. The smallness of visualization particle data enables rendering at interactive frame rates and interactive changes of camera parameters. In our previous work, we developed an in-situ visualization system IS-PBVR based on PBVR, and demonstrated the capability of remote in-situ visualization for the simulation with hundreds of millions of grid points on GPU supercomputers [3]. In this study, we extend IS-PBVR for VR visualization, and develop an in-situ VR visualization system VR IS-PBVR. Furthermore, we extend VR IS-PBVR for multi-point visualization. By applying VR IS-PBVR to simulations on GPU supercomputers, we demonstrate the utility of multi-point in-situ VR visualization using multiple client PCs.

2 Related Work

Generic visualization systems such as ParaView and VisIt provide libraries and frameworks to support large-scale in-situ visualization. ParaView Catalyst [4] is a visualization library that is tightly coupled to the simulation to enable in-situ visualization using visualization functions on ParaView. It also provides computational steering capability by calling Catalyst routines in the simulation. Libsim is a visualization library for in-situ visualization using visualization functions on VisIt [5]. It is also tightly coupled to the simulation and pauses the simulation while data manipulation is taking place. While these tools are optimized for use on supercomputers, they may not provide sufficient interactive frame rates required for VR visualization. For example, in Ref. [3], we conducted an in-situ visualization benchmark between IS-PBVR and ParaView Catalyst on a GPU-based supercomputer, and it was shown that even with an off-screen rendering mode, the rendering speed of ParaView Catalyst was far below one FPS, while IS-PBVR achieved interactive frame rates. As rendering at interactive frame rates and interactive changes of camera parameters are essential for VR visualization, in-situ VR visualization is challenging.

2.1 PBVR

Volume rendering is used to visualize the distribution of 3D volume data by mapping color and opacity to physical values. This approach is effective in analyzing 3D volume data in computational fluid dynamics (CFD) and structural analysis. Particle-Based Volume Rendering (PBVR) [1, 2] is a projection-based volume rendering technique that converts volume data into particle data. The generation and projection of particles and the calculation of pixel values are based on the optical model of volume rendering (the density emission model) [6, 7]. The resulting particle data is independent of viewpoint, eliminating the need for recomputation of particles upon changes in camera parameters. Moreover, the size of particle data is determined by opacity and screen resolution, typically resulting in sizes of several tens of MB for a resolution of 1024^2, which is significantly smaller than the size of large-scale simulation data. Additionally, PBVR is extended for multivariate visualization capabilities [8]. Users can define compositions of multiple volume datasets and multiple transfer functions using algebraic expressions, allowing them to design transfer functions tailored for multivariate data.

2.2 Client Server PBVR

Client-Server PBVR (CS-PBVR) [9] is a remote visualization system constructed by dividing particle generation and particle projection into client and server components. The configura-

tion of CS-PBVR is shown in the upper part of figure 1. CS-PBVR consists of Particle Sampler for particle generation on the server and Particle Renderer for image production through particle projection on the user PC. Particle Sampler and Particle Renderer are connected via an ssh tunnel over the internet, allowing interactive visualization by transferring compressed particle data at low data transfer costs. The view-independent particle data can be easily rendered at interactive frame rates on the user PC, which also enables interactive changes of camera parameters. This interactive visualization capability of CS-PBVR mekes VR visualization possible using the Oculus HMD [10]. It calculates camera parameters based on positional information obtained from the headset, and generates images for both eyes from the particle data. Additionally, gesture control is implemented to manipulate visualization data in VR space. In gesture control, hand movements are captured via VR controllers, and coordinate transformations (rotation, scaling, translation) of objects are computed in response to user's gestures. The image generation process in CS-PBVR is implemented based on the visualization library KVS and OpenGL. Subsequently, the images are sent to the HMD using the API of the Oculus Software Development Kit (SDK). As a result, the use of VR functionality is limited to Oculus.

2.3 In-Situ PBVR

In-Situ PBVR (IS-PBVR) [11, 12] enables in-situ visualization by generating compressed particle data at runtime within the simulation and transferring it to the user PC. IS-PBVR consists of three components: In-situ Sampler, In-situ Daemon, and Particle Renderer. The configuration of IS-PBVR is shown in the lower part of figure 1. In-situ Sampler is tightly coupled with the simulation, and it generates particle data files at each time step, which are stored in the storage. In-situ Daemon checks updates of the particle data files, and when all of them are updated, it gathers them from the storage and transfers them to the user PC via socket communication. Additionally, it receives interactive changes of visualization parameters such as transfer functions from the user PC and updates the visualization parameter file on the storage, which is checked before particle generation by In-Situ Sample. This file-based control enables asynchronous update of visualization parameters, without interfering the simulation.

3 VR In-Situ PBVR

The GUIs of Particle Renderers in CS-PBVR and IS-PBVR share many common functionalities, thus we integrate their GUIs to construct a common VR-capable Particle Renderer in IS-PBVR. In-situ Sampler and In-situ Daemon on the server are not changed from the conventional IS-PBVR, sending particle data for each time step at runtime. An overview of VR support on IS-PBVR (VR IS-PBVR) is shown in the lower part of figure 1.

The VR IS-PBVR client is developed using OpenXR as the interface of the HMD. OpenXR is an open source, royalty-free standard for VR devices, and by using the OpenXR SDK's API, users can use the VR functions developed on many OpenXR-compatible HMDs. OpenXR detects the type of HMD being used during the initialization process and creates instances of the corresponding APIs and extensions. In the Oculus SDK, textures rendered as stereo images with OpenGL are displayed on the HMD by connecting them to the Oculus API. In OpenXR SDK, however, the concept of the API for displaying images is different, and images are displayed on the HMD by registering them on a multi-buffer called a swap chain. In this study, we developed a function to add textures drawn with OpenGL to the swap chain to absorb the difference between the function for Oculus and the OpenXR specification.

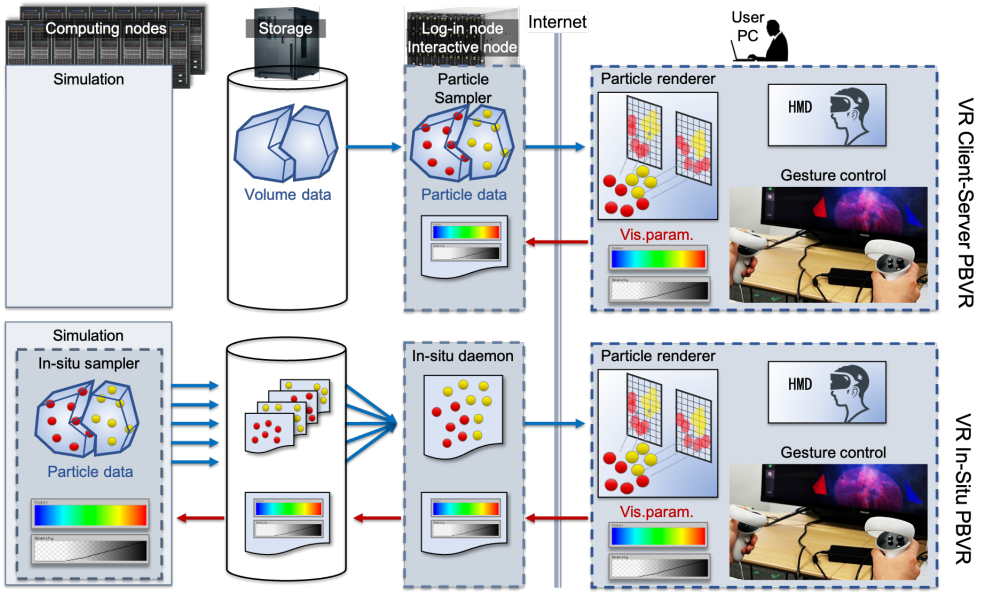

Figure 1. Overview of PBVR. Top row: VR CS-PBVR, bottom row: VR IS-PBVR.

Sending and receiving timestep particle data is implemented as before on a TCP socket, and particle data for each timestep is transferred to the GPU for high-speed rendering. In updating the time step, the client receives the new time step particle data from the particle sampler side and transfers it to the GPU memory to replace the particle data of the previous time step. An overview of the developed VR IS-PBVR process and whether each process is implemented with KVS+OpenGL, OpenXR, or socket communication is shown in figure 2. Particle data for each time step is stored on the user PC, enabling history search in VR space. The GUI for time step control is drawn on the texture of the rendering result and displayed in layover. In general, different types of HMDs have different controller hardware configurations. In this study, we mapped the activation of geometric transformations and the manipulation functions of the GUI in the VR space to joysticks, buttons, and triggers, which are universally available on many HMD controllers.

4 Multi-point In-Situ PBVR

In multi-point visualization, multiple users share visualization results and change visualization parameters for data exploration. To share the visualization results, multiple In-situ Daemons are launched by users, and when the particle data files are generated from a single Particle Sampler, each In-situ Daemon transfers them to each user PC. Here, user establish ssh tunnels between In-Situ Daemon and Particle Render with their respective port numbers, and In-situ Daemons asynchronously send particle data to the corresponding user PCs at runtime. In data exploration, a host, who is responsible for changing visualization parameters, and guests, who are passive users, are designated. In VR IS-PBVR, a host and guests are freely assigned to users based on their port numbers. The host In-situ Daemon outputs interactive changes of visualization parameters to the visualization parameter file, which controls Particle Sampler, while the guest In-situ Daemon reads it and transfers the modified visualization parameters to the user PC, so that they are shown on the GUI of Particle Renderer. An

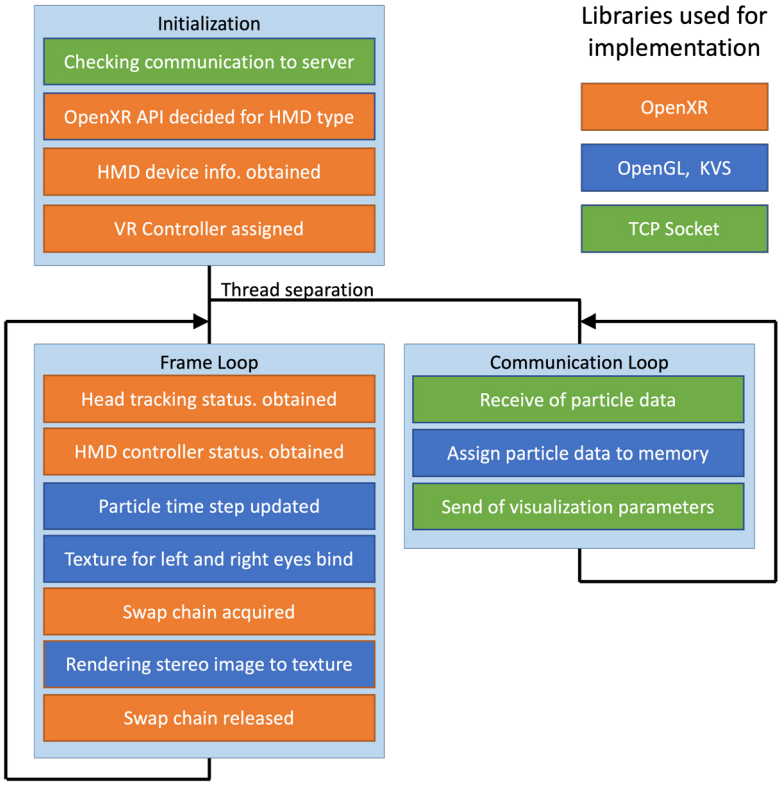

Figure 2. High-level overview of the APIs that make up VR IS-PBVR. The orange, blue, and green boxes mean implementation by OpenXR, OpenGL + KVS, and TCP Socket, respectively.

overview of multi-point VR IS-PBVR is shown in figure 3. In addition to the conventional visualization parameters such as transfer functions for multivariate visualization, the visualization parameter file is extended including the host's focal point, so that it is shared by all users using glyph rendering in the VR space.

5 Experimental Result and Discussion

To validate the effectiveness of multi-point VR IS-PBVR, we applied it to debris air cooling analysis on a GPU-based supercomputer. In the experiment, we utilized the HMD Meta Quest Pro along with its accompanying Touch controllers as VR devices. Our numerical experiments are conducted using two computation nodes with 8 GPUs on the HPE SGI8600, which is connected to two user PCs with the network bandwidth of 3.0MB/s (see table 1). Here, both PCs are equipped with the HMD Meta Quest Pro. The debris air cooling analysis has the block structured AMR grid system consisting of 24,336 leaves for Lv.1 and 499,904 leaves for Lv.2, and each leaf is given by 43 Cartesian grids, leading to 33,551,360 total grids.

Firstly, the numerical experiment is conducted using a single user PC (User PC1, see Table 1). In the numerical experiment, Particle Sampler is processed at every 200 time steps, which are processed in 12 seconds. In-situ Sampler generated approximately 103 MB of particle data at each in-situ visualization step within 10.1 seconds, and In-situ Daemon transferred them to the user PC in 34.3 seconds. The transferred particle data was rendered at 90

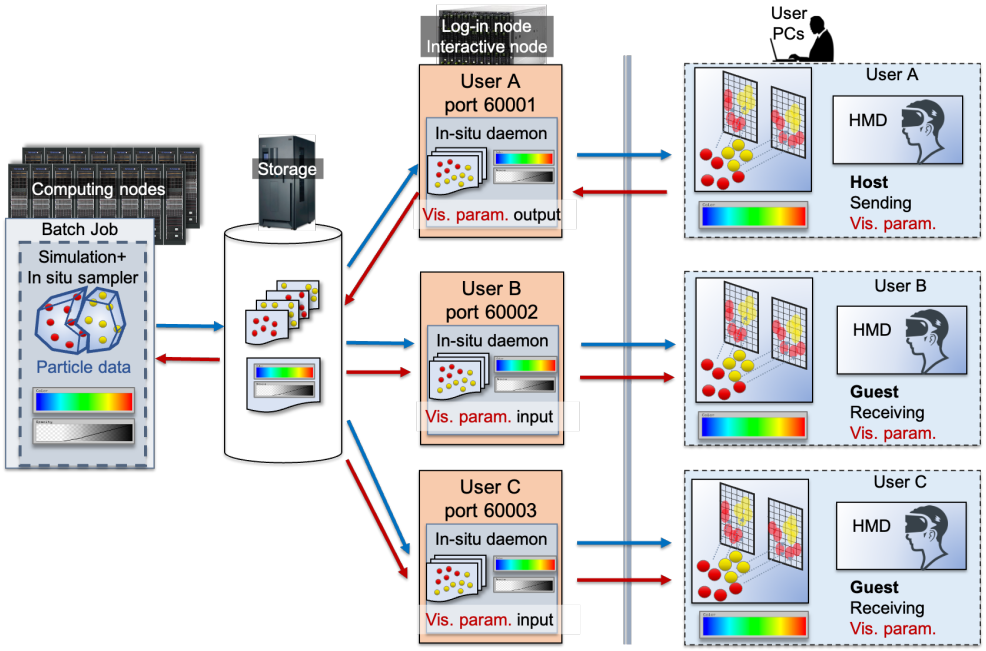

Figure 3. Configuration of multi-point VR IS-PBVR.

Table 1. Specification of a single node of HPE SGI8600, the user PC1 and the user PC2.

	HPE SGI8600	User PC1	User PC2
CPU	2x Intel Xeon Gold 6248R (3.0GHz, 24 cores)	Intel Corei7 (2.6GHz, 8 cores)	Intel Xeon Silver (2.20GHz, 12 cores)
GPU	4x NVIDIA Tesla V100 SXM2 (32GB)	NVIDIA GeForce RTX2080 (8GB)	NVIDIA Quadro RTX 5000 (16GB)
Memory	384GB	32GB	64GB

FPS on the HMD. The visualization results are shown in figure 4. It is noted that Paraview Catalyst required 388 seconds to render the same data [3]. The achievement of interactive in-situ VR visualization is attributed to compressed particle data in PBVR.

We then conducted the numerical experiment of multi-point visualization by adding the second user PC (User PC2). Setting PC1 as the guest and PC2 as the host, we verified that each user PC could launch In-situ Daemon and share the in-situ visualization results. The visualization results are illustrated in figure 5. Since view-independent particle data is transferred to each user PC, each user can independently change viewpoints and perform geometric transformations in the VR space. Additionally, we confirmed that the host could modify the transfer function and the position of the focal point glyph, and the guest could receive these updates accordingly.

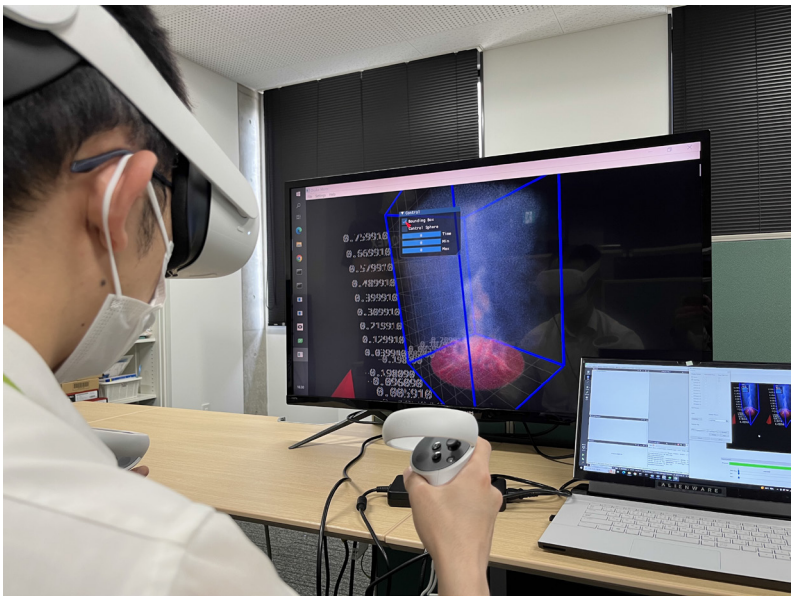

Figure 4. Debris air cooling analysis using VR IS-PBVR with HMD Meta Quest Pro.

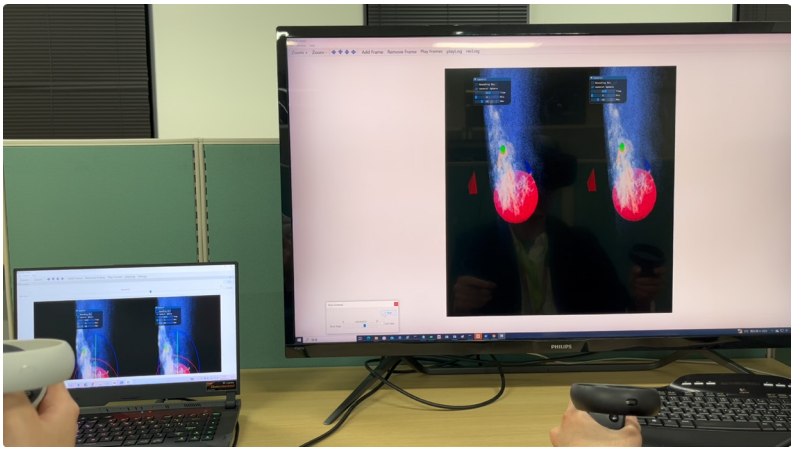

Figure 5. Debris air cooling analysis with multi-point VR IS-PBVR. The guest is on the left, and the host is on the right. The green glyph indicating a point of interest is shared from the host to the guest.

6 Conclusion

In this study, we extended IS-PBVR for VR visualization on HMDs. VR visualization is newly implemented via OpenXR so that it universally works across various HMDs. We applied VR IS-PBVR to simulations on GPU supercomputers, demonstrating interactive and in-situ immersion in the computational space. Achieving in-situ VR visualization was challenging in the conventional in-situ visualization systems, because the interactive frame rates were essential in VR visualization on HMDs. However, VR IS-PBVR accomplished this requirement by rendering compressed particle data on the user PC.

We further extended VR IS-PBVR for multiple clients. The compressed particle data generated from a single Particler Sampler is transferred to the user PCs in an asyncronous manner via multiple In-situ Daemon. By configuring host and guest functions on each client, we demonstrated the ability to share data exploration through changes in visualization parameters and a focal point glyph across multiple clients. This technology enables sharing of VR spaces on HMDs with remote scientists, facilitating efficient analysis of large-scale simulations on supercomputers. In the future, we aim to enhance user usability by developing information visualization features and GUIs for visualization parameters in VR spaces.

Acknowledgement

This research was supported by JSPS KAKENHI Grant Number 23K11130. The research used the GPU supercomputer HPE SGI8600 belonging to Japan Atomic Energy Agency.

References

[1] N. Sakamoto, T. Kawamura, K. Koyamada, K. Nozaki, Comput. Graph. **34**, 34–42 (2010)

[2] T. Kawamura, N. Sakamoto, K. Koyamada, Journal of Computer Science and Technology **25**, 905 (2010)

[3] T. Kawamura, Y. Hasegawa, Y. Idomura, Journal of Visualization **27**, 89 (2024)

[4] N. Fabian, K. Moreland, D. Thompson, A.C. Bauer, P. Marion, B. Gevecik, M. Rasquin, K.E. Jansen, *The ParaView Coprocessing Library: A scalable, general purpose in situ visualization library*, in *2011 IEEE Symposium on Large Data Analysis and Visualization* (2011), pp. 89–96

[5] B. Whitlock, J.M. Favre, J.S. Meredith, *Parallel In Situ Coupling of Simulation with a Fully Featured Visualization System*, in *Eurographics Symposium on Parallel Graphics and Visualization*, edited by T. Kuhlen, R. Pajarola, K. Zhou (The Eurographics Association, 2011), ISBN 978-3-905674-32-3, ISSN 1727-348X

[6] P. Sabella, SIGGRAPH Comput. Graph. **22**, 51 (1988)

[7] P.L. Williams, N. Max, *A Volume Density Optical Model*, in *Proceedings of the 1992 Workshop on Volume Visualization* (Association for Computing Machinery, New York, NY, USA, 1992), VVS '92, p. 61–68, ISBN 0897915275, https://doi.org/10.1145/147130.147151

[8] T. Kawamura, Y. Idomura, H. Miyamura, H. Takemiya, Journal of Visualization **20**, 151 (2017)

[9] T. Kawamura, Y. Idomura, H. Miyamura, H. Takemiya, N. Sakamoto, K. Koyamada, *Remote visualization system based on particle based volume rendering*, in *Visualization and Data Analysis 2015*, edited by D.L. Kao, M.C. Hao, M.A. Livingston, T. Wischgoll, International Society for Optics and Photonics (SPIE, 2015), Vol. 9397, p. 93970S, https://doi.org/10.1117/12.2083501

[10] T. Kawamura, N. Sakamoto, T. Osaki, Journal of Advanced Simulation in Science and Engineering **10**, 31 (2023)

[11] T. Kawamura, T. Noda, Y. Idomura, Supercomputing Frontiers And Innovations: Int. J. **4**, 43 (2017)

[12] T. Kawamura, Y. Idomura, Journal of Visualization **23**, 695 (2020)