

# A Proposed Framework for Interactive Virtual Reality *In Situ* Visualization of Parallel Numerical Simulations

Aryaman Gupta\*

Technische Universität Dresden & Center for Systems Biology Dresden (CSBD) & MPI-CBG, Dresden

Ulrik Günther

Pietro Incardona

Ata Deniz Aydin

Raimund Dachsel

Technische Universität Dresden  
Interactive Media Lab Dresden  
Cluster of Excellence Physics of Life

Stefan Gumhold

Technische Universität Dresden  
Cluster of Excellence Physics of Life

Ivo F. Sbalzarini†

Technische Universität Dresden  
CSBD & MPI-CBG, Dresden  
Cluster of Excellence Physics of Life

## ABSTRACT

As computer simulations progress to increasingly complex, non-linear, and three-dimensional systems and phenomena, intuitive and immediate visualization of their results is becoming crucial. While Virtual Reality (VR) and Natural User Interfaces (NUIs) have been shown to improve understanding of complex 3D data, their application to live *in situ* visualization and computational steering is hampered by performance requirements. Here, we present the design of a software framework for interactive VR *in situ* visualization of parallel numerical simulations, as well as a working prototype implementation. Our design is targeted towards meeting the performance requirements for VR, and our work is packaged in a framework that allows for easy instrumentation of simulations. Our preliminary results inform about the technical feasibility of the architecture, as well as the challenges that remain.

**Index Terms:** Computing methodologies—Massively parallel and high-performance simulations; Human-centered computing—Virtual reality;

## 1 INTRODUCTION

The growing disparity between computational throughput and I/O bandwidth to disk increasingly prohibits the classical approach of writing simulation output to disk and analyzing it in post-processing steps. *In situ* techniques, which involve the visualization or analysis of a simulation as it runs, have thus received much research interest.

In addition to helping bypass the file I/O bottleneck, *in situ* visualization also enables computational steering. Images rendered *in situ* may be streamed to a remote client, where a user can visualize the simulation live and also interact with it. This has, e.g., been applied in computational fluid dynamics to interactively vary simulation parameters [8], which can help gain an intuition of the simulated physics.

Forming an intuition of the simulated dynamics and its parameter dependence benefits from Virtual Reality (VR) visualization, which has been shown to improve perception of space and geometry [9]. We therefore believe that *in situ* VR visualization has the potential to aid scientific discovery by providing better understanding of simulation data, and an environment for intuitive interaction with it.

A VR environment lends itself to the use of Natural User Interfaces (NUI) for intuitive interaction. NUIs are based on how humans interact with their physical environment and include the use of body pose, hand gestures, hand-held tools, and eye gaze. It has been shown that scientists better understand data when they are able to interact with it via intuitive interfaces [2].

\*e-mail: argupta@mpi-cbg.de

†e-mail: sbalzarini@mpi-cbg.de

Using VR for *in situ* visualization of simulations, however, is complicated by the high frame rates and low latency required. Simulation data can be large, often tens or hundreds of gigabytes per simulation time step, which typically occurs once per second. *In situ* visualization in VR is therefore difficult to achieve.

Here, we propose a framework for *in situ* VR visualization and NUI steering of simulations that optimizes rendering frame rate and latency. We describe the architecture and present preliminary results, which leads us to identifying challenges that need to be overcome in future work.

## 2 RENDERING AND SIMULATION LIBRARIES USED

The rendering system used in our framework is developed using *scenery* [4], an open-source VR library for both volumetric and geometric data. In simulations, volumetric data typically stem from discretized continuous fields and geometric data from surface meshes or postprocessing results, such as isosurfaces or streamlines. *scenery* supports rendering to both commodity VR headsets and distributed setups, such as VR CAVes and PowerWalls. It uses the high-performance Vulkan API for rendering, and therefore has the potential to harness the power of modern GPUs better than traditional OpenGL-based solutions, as still predominantly used by *in situ* solutions like ParaView Catalyst [1] and VisIt Libsim [6].

The simulation engine used in our framework is OpenFPM [5], an open-source library that enables rapid development of scalable and efficient particle- and mesh-based numerical simulations. OpenFPM largely automates parallelization on high-performance computers, including distributed-memory systems, many-core systems, and GPGPUs. It has been used to implement simulations ranging from Smoothed-Particle Hydrodynamics to Molecular Dynamics to Discrete Element Methods on thousands of CPU cores [5].

Integrating our work into OpenFPM helps us reach a wide audience of computational scientists who benefit from interactive *in situ* visualization and steering, but may lack the time or expertise to write scalable solutions from scratch. We implement *in situ* functionality into OpenFPM in a manner that allows users to enable it in a fraction of the time and effort it normally takes to instrument a simulation with an *in situ* solution.

## 3 IN SITU ARCHITECTURE

We design our *in situ* architecture targeting a rendering frame rate of at least 60 Hz, rendering latency on viewpoint changes of at most 20 ms, and low interaction latency for steering commands.

Key to achieving these goals is to minimize data movement. We thus choose a tightly-coupled structure with simulation and rendering sharing resources, running on the same nodes within a cluster. Figure 1 shows a schematic. Each compute node runs the distributed OpenFPM environment for the simulation, along with a *scenery*-based application that performs local rendering. All simulation data generated on a node are rendered on the node itself. The simula-

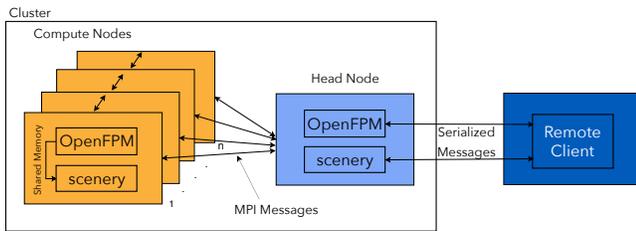


Figure 1: Schematic of our framework architecture.

tion data are shared between OpenFPM and *scenery* via a common pointer to shared memory. This enables zero-copy rendering.

Several *in situ* solutions, including ParaView Catalyst [1] and VisIt Libsim [6], adopt a synchronous execution strategy, where simulation and visualization run sequentially. We choose an asynchronous strategy, aiming to reduce interaction and rendering latency. In our asynchronous strategy, neither simulation nor visualization need to wait for the other, which means that steering commands can be incorporated at the earliest, and rendering can begin while the simulation is in progress. Visualization artifacts that may result from this lack of synchronization are expected to be small, since numerical stability requires the simulation to only change slightly between time steps. For our application, we emphasize low latency over these minor visual artifacts.

The rendering results produced on the compute nodes are then composited using a divide-and-conquer strategy similar to the binary swap algorithm [7]. The *scenery* process on the Head Node is responsible for streaming the composited result to a remote client, as well as for receiving requests for changes in visualization parameters. The remote client also communicates steering commands to the OpenFPM process on the Head Node, which conveys them to all OpenFPM processes via MPI messages.

In contrast to other approaches, we intend to render an explorable representation, such as a Volumetric Depth Image (VDI) [3], instead of a simple image. Such representations of volumetric data can, once generated, be reprojected to a new viewpoint without redoing the ray casting. We intend to render such a representation on the cluster and stream it to the remote client. Small viewpoint changes can then be handled locally by the remote client without involving network latency. This is crucial for meeting the latency requirements of VR, where viewpoint changes occur frequently.

## 4 PRELIMINARY RESULTS

We have implemented a prototype of our framework, performing particle-as-sphere rendering of a distributed OpenFPM simulation. The simulation runs parallel on multiple CPUs, while visualization is done on the GPU. The shared-memory communication between the processes on a single node is extended by a protocol to handle dynamic memory requests and reallocations. The *scenery* application reads the simulation data from shared memory and understands the data structures used by OpenFPM. Our implementation is open source. The *scenery* application can be found at [github.com/scenerygraphics/scenery-insitu](https://github.com/scenerygraphics/scenery-insitu) and the OpenFPM project at [git.mpi-cbg.de/openfpm/openfpm\\_pdata](https://git.mpi-cbg.de/openfpm/openfpm_pdata) (in the “*insitu\_visualization*” branch).

Figure 2 shows our prototype applied to a Molecular Dynamics simulation. The simulation code written using OpenFPM is only 161 lines of C++, and the inclusion of *in situ* visualization requires changes in only two lines.

## 5 FUTURE WORK AND CONCLUSIONS

We have presented a framework architecture for interactive VR *in situ* visualization of parallel numerical simulations. We used

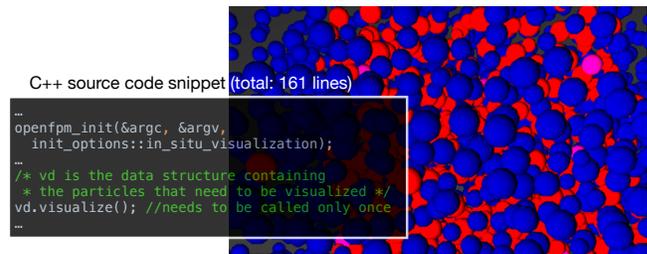


Figure 2: An OpenFPM-based distributed Molecular Dynamics simulation visualized *in situ* using our prototype, and the lines of code that needed to be changed to enable *in situ* visualization. Particles are colored by velocity magnitude.

*scenery* and OpenFPM to base the rendering and simulation parts of the framework on. We have implemented a first prototype of the system, demonstrating the feasibility of achieving our design goals. In our prototype, enabling live *in situ* visualization only required minimal changes in the existing simulation code.

Our prototype does not yet meet the targets we set for immersive *in situ* visualization in Section 3, but it enables us to identify a number of challenges that should be addressed in order to meet them: First, volumetric rendering results need to be in an explorable representation, such as a VDI, as explained in Section 3. Second, we need a solution to efficiently generate and composite such a representation on a cluster. Third, support needs to be added for simulations running on GPGPUs. In addition, we are going to extend our framework to NUI-based steering by instrumenting OpenFPM to receive steering commands on the fly and by expanding *scenery*’s support for NUI interaction.

Taken together, we are confident that many applications in scientific computing will benefit from a more immersive and more explorative mode of interaction.

## REFERENCES

- [1] 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. In *Proc. 1st Workshop on In Situ Infrastructures for Enabling Extreme-Scale Anal. Vis. (ISAV)*, pp. 25–29. ACM, 2015.
- [2] F. P. Brooks Jr, M. Ouh-Young, J. J. Batter, and P. Jerome Kilpatrick. Project GROPEHaptic displays for scientific visualization. In *ACM SIGGraph computer graphics*, vol. 24, pp. 177–185. ACM, 1990.
- [3] S. Frey, F. Sadlo, and T. Ertl. Explorable Volumetric Depth Images from Raycasting. In *Proc. Conference on Graphics, Patterns and Images*, pp. 123–130. IEEE, 2013.
- [4] U. Günther, T. Pietzsch, A. Gupta, K. I. Harrington, P. Tomancak, S. Gumhold, and I. F. Sbalzarini. scenery: Flexible Virtual Reality Visualization on the Java VM. *Proc. IEEE VIS*, in print, 2019. <https://arxiv.org/abs/1906.06726>.
- [5] P. Incardona, A. Leo, Y. Zaluzhnyi, R. Ramaswamy, and I. F. Sbalzarini. OpenFPM: A scalable open framework for particle and particle-mesh codes on parallel computers. *Computer Physics Communications*, 241:155–177, 2019.
- [6] T. Kuhlen, R. Pajarola, and K. Zhou. Parallel In Situ Coupling of Simulation with a Fully Featured Visualization System. In *Proc. 11th EGPGV*, 2011.
- [7] K.-L. Ma, J. S. Painter, C. D. Hansen, and M. F. Krogh. Parallel Volume Rendering Using Binary-Swap Compositing. *IEEE Computer Graphics and Applications*, 14(4):59–68, 1994.
- [8] A. Sanderson, A. Humphrey, J. Schmidt, and R. Sisneros. Coupling the Uintah Framework and the VisIt Toolkit for Parallel In Situ Data Analysis and Visualization and Computational Steering. In *International Conference on High Performance Computing*, pp. 201–214. Springer, 2018.
- [9] M. Slater and M. V. Sanchez-Vives. Enhancing our lives with immersive virtual reality. *Frontiers in Robotics and AI*, 3:74, 2016.