https://doi.org/10.2352/EI.2026.38.1.VDA-350

Distributed 3D Gaussian Splatting for High-Resolution Isosurface Visualization

Mengjiao Han[1], Andres Sewell[2], Joseph Insley[1], Janet Knowles[1], Victor A. Mateevitsi[1,3], Michael E. Papka[1,3], Steve Petruzza[2], and Silvio Rizzi[1]

[1]Argonne National Laboratory, Lemont, IL, USA
[2]Utah State University, Logan, UT, USA
[3]University of Illinois Chicago, Chicago, IL, USA

Abstract

3D Gaussian Splatting (3D-GS) has recently emerged as a powerful technique for real-time, photorealistic rendering by optimizing anisotropic Gaussian primitives from view-dependent images. Unlike neural implicit methods such as NeRF, 3D-GS avoids the need for a neural network forward pass at inference, making it significantly faster while maintaining high visual fidelity. While 3D-GS has been extended to scientific visualization, prior work remains limited to single-GPU settings, restricting scalability for large scientific datasets on high-performance computing (HPC) systems. In this study, we present a distributed 3D-GS pipeline tailored for scientific data on HPC. Our approach partitions data across nodes, trains Gaussian splats in parallel using multi-nodes and multi-GPUs, and merges splats for global rendering. To eliminate artifacts, we add ghost cells at partition boundaries and apply background masks to remove irrelevant pixels. Benchmarks on the Richtmyer–Meshkov datasets (about 106.7M Gaussians) show up to 3X speedup across 8 nodes on Polaris while preserving image quality. These results demonstrate that distributed 3D-GS enables scalable visualization of large-scale scientific data and provides a foundation for future in situ applications.

Introduction

3D Gaussian Splatting (3D-GS) [Kerbl et al.(2023)] is a recent technique that enables photorealistic, real-time rendering of complex 3D scenes by optimizing anisotropic Gaussian primitives from view-dependent images. Unlike neural implicit methods such as NeRF [Mildenhall et al.(2021)], 3D-GS avoids the need for a neural network forward pass at inference, making it significantly faster while maintaining high visual fidelity.

Recent work has begun to apply 3D-GS to scientific visualization [Ai et al.(2025), Tang et al.(2025), Sewell et al.(2024), Yao and Wang(2025)]. These approaches have shown promise, but existing pipelines are typically limited to single-GPU execution. This limitation hinders the scalability of large complex datasets produced by high-performance computing (HPC) simulations, which are often distributed between multiple compute nodes in subsets that cannot be centralized in a single node's memory.

To address this limitation, we propose a distributed 3D-GS pipeline tailored for HPC environments. Our approach is based on the isosurface-based method of Sewell et al. [Sewell et al.(2024)] and adapts the distributed training concept of Grendel-GS [Zhao et al.(2024)], but with critical modifications to handle distributed scientific datasets on HPC. Instead of loading the full dataset onto a master node, we allow each node to process only its own subset of data, parallel train Gaussian splats independently on each node, and then combine splats for global rendering. Additionally, we incorporate ghost cells and background masking to address rendering artifacts such as gaps and white streaks. Our contributions are as follows.

- **Distributed 3D-GS for HPC**: A multi-node, multi-GPU pipeline for large-scale scientific visualization, designed for datasets partitioned across HPC nodes.
- **Scalability Benchmarks**: Performance evaluation with different datasets and across different image resolutions and node counts.
- **Foundation for In Situ Visualization**: A first step toward scalable in situ workflows using 3D-GS for scientific data visualization.

A preliminary version of this work was presented as posters at the IEEE eScience 2025 conference (covering the multi-GPU experiments) and at Super Computing 2025 (covering the multi-node experiments). In this paper, we extend those results with a unified presentation and introduce new contributions, including a load-balancing strategy along with comprehensive analysis and discussion. The source code for our distributed 3D-GS pipeline is publicly available at `https://github.com/MengjiaoH/Grendel-GS-SciVIS`.

Background and Related Work

3D Gaussian Splatting

3D-GS [Kerbl et al.(2023)] is a recently introduced technique for high-fidelity, real-time rendering of complex 3D scenes. Instead of relying on volumetric grids and implicit neural representations such as NeRF [Mildenhall et al.(2021)], 3D-GS represents a scene as a collection of anisotropic Gaussian primitives and each primitive is parameterized by its position, scale, orientation, color, and opacity. During the training process, a set of 3D Gaussians is first initialized to represent the scene. These Gaussians are then projected into 2D to reconstruct images, and their parameters are optimized through differentiable rasterization against ground truth views. The process also includes Gaussian removal and densification to adaptively refine Gaussians to get closer to the ground truth views.

A key advantage of 3D-GS is its efficiency at both training and inference. Kerbl et al. [Kerbl et al.(2023)] have shown that 3D-GS can be at least 10 times faster than NeRF on the same scene. In addition, since rendering requires only rasterizing

Gaussians rather than evaluating a deep neural network, 3D-GS achieves interactive to real-time performance while maintaining visual quality comparable to NeRF. However, the original study only supports single GPU training, which limits the usage of 3D-GS for large-scale scenes.

To extend 3D-GS training beyond a single GPU, Zhao et al. [Zhao et al.(2024)] introduced Grendel-GS, which demonstrated that Gaussian Splatting can be scaled using data-parallel training across multiple GPUs. Grendel-GS distributes training images across GPUs and synchronizes Gaussian parameters during optimization, allowing datasets too large for a single GPU to be trained effectively. This work established the feasibility of scaling Gaussian Splatting to multi-GPU systems and highlighted the potential for broader adoption in HPC contexts.

Nevertheless, Grendel-GS remains requiring loading all initial Gaussians and images on the master GPU and then distributed to the workers. In addition, synchronization of Gaussian parameters during optimization relies on images of the global domain, which is not feasible for scientific data visualization on HPC systems. In practice, HPC simulation outputs are already spatially decomposed across nodes, could reaching terabyte scale, and cannot be trivially centralized. Instead, visualizations are typically performed independently on each node and later composed into a final result [Moreland(2011)]. This limitation motivates our distributed 3D-GS pipeline, which applies to scientific visualization and introduces new strategies, including ghost cells, background masking, and load balancing to ensure scalability and quality on HPC systems.

3D-GS for Scientific Visualization

Owing to the efficiency and rendering speed of 3D-GS, recent research has begun to explore the application of 3D-GS in scientific visualization. Sewell et al. [Sewell et al.(2024)] pioneered this direction by adapting 3D-GS to reconstruct isosurface visualizations of scientific data. Tang et al. [Tang et al.(2025)] introduced an inverse volume rendering approach using 3D-GS, enabling interactive editing such as dynamic transfer function adjustments. Ai et al. [Ai et al.(2025)] further extended this concept by integrating Large Language Models (LLMs) to facilitate user-driven editing of visualizations through natural language input. In another line of work, Yao et al. [Yao and Wang(2025)] proposed a segmentation and tracking framework for dynamic volumetric scenes using 3D deformable Gaussians.

Despite these advances, none of the existing studies have addressed the use of distributed HPC environments for 3D-GS-based scientific visualization. Given the scale and complexity of scientific datasets, which are often generated on HPC systems using multiple nodes, there is a pressing need for scalable visualization solutions. In this study, we present an initial investigation into distributed 3D-GS training and rendering of isosurfaces across multiple compute nodes, aiming to facilitate efficient 3D reconstruction of large-scale scientific data.

Method

Our distributed workflow (Figure 1) is designed to enable scalable training of Gaussian Splatting models on large-scale scientific datasets without requiring central data aggregation. The pipeline consists of six major stages:

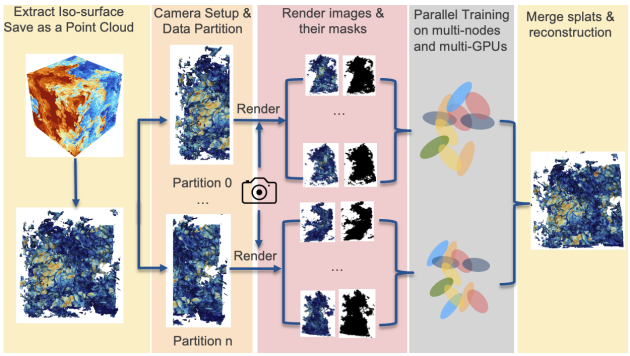


Figure 1: Workflow of our distributed 3D-GS pipeline for large-scale isosurface visualization on HPC systems.

- **Isosurface Extraction**: We first extract isosurface point clouds from volumetric simulation data using ParaView[1]. These point clouds are used as the initial Gaussian primitives.
- **Camera Setup**: To maintain rendering consistency across nodes, we generate a structured orbital set of synthetic camera views. Each node uses an identical set of camera configurations, including position, orientation, field-of-view and image resolution. This ensures that Gaussian splats are trained under uniform conditions for correct final merging. The orbital setup also provides comprehensive angular coverage, which is critical for avoiding reconstruction bias.
- **Data Partitioning**: The dataset is divided into n spatial partitions, one for each compute node. To mitigate boundary artifacts, we extend each partition with ghost cells (refer to Section *Rendering Improvements*), which include overlapping data along shared boundaries. These ghost regions guarantee continuity of geometry, reducing visual seams in the final rendering.
- **Image Rendering and Masking**: On each node, visualizations are rendered for its assigned partition using the predefined cameras. Alongside RGB images, we generate binary background masks that delineate the occupied geometry (refer to Section *Rendering Improvements*). These masks prevent the creation of splats in empty regions and ensure that computational resources are focused only on relevant structures. This significantly improves visual fidelity, as artifacts from splats from background are avoided.
- **Parallel Training**: Each node independently trains a Gaussian Splatting model using a multi-GPU data-parallel strategy [Zhao et al.(2024)]. The independence of training minimizes communication overhead and allows nodes to scale nearly linearly with available GPU resources. Since each partition corresponds to a physically local subset of the data, this design aligns naturally with HPC domain decomposition strategies commonly used in simulation codes.
- **Global Reconstruction**: Finally, trained splats from all nodes are collected and merged into a single global model. The merged splat set can then be used for high-quality rendering or further analysis.

This design removes the need to centralize large-scale datasets, instead relying on in-place training across distributed

[1] `https://www.paraview.org`

partitions. By aligning with the spatial decomposition commonly available in HPC simulation outputs, the workflow makes distributed Gaussian Splatting tractable and scalable for large scientific visualization tasks on HPC.

Load Balancing

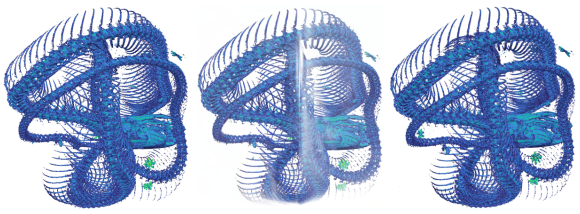
The size of the data assigned to each node directly impacts training performance, making balanced partitioning essential for scalability. We evaluated two strategies for dividing the domain (refer to Section *Load Balancing* in *Results*). The first strategy partitions the domain uniformly along spatial axes by specifying the number of divisions per axis. While straightforward, this approach does not guarantee balanced workloads, since different partitions may contain significantly different numbers of points. The second strategy explicitly aims to achieve load balance. Given a desired number of partitions, the algorithm recursively splits along the longest axis so that each partition contains approximately the same number of points. This results in a more even distribution of computation across nodes and reduces idle time caused by workload imbalance.

Rendering Improvements

A central challenge in distributed Gaussian Splatting on HPC is the presence of visual seams and artifacts when merging splats from independently trained partitions. If left unaddressed, these artifacts manifest as seams at partition boundaries and white streaks from background pixels (refer to Figure 4b). We mitigate these problems using two strategies:

1. **Ghost cells:** By extending each partition one cell into neighboring domains, we preserve overlapping geometry across boundaries. This ensures that splats trained on each node naturally blend at the interfaces, eliminating visible seams and producing continuous surfaces in the global model.
2. **Background masks:** During training, background pixels are excluded using binary masks generated alongside synthetic images. This prevents unnecessary splat generation in empty space and removes the white streak artifacts for the final rendering.

Together, these improvements yield reconstructions that are visually indistinguishable from centralized training, while maintaining the scalability of distributed workflows on HPC systems (Figure 2c).



(a) Ground Truth (b) W/O GC and Masks (c) Our Method W/ GC and Masks

Figure 2: Visualization comparison using the Kingsnake scan dataset. (a) Ground truth, (b) rendering without ghost cells (GC) or background masks, and (c) our method with GC and background masks.

Results

We evaluated our distributed Gaussian Splatting workflow on three representative datasets from small to large:

- **Kingsnake** (110.3 MB, ∼4M points) from DigiMorph[2], which serves as a compact biological dataset with complex geometry but moderate scale.
- **Rayleigh–Taylor instability** (491 MB, ∼18.2M points) [Cook et al.(2004)], a canonical fluid dynamics benchmark with highly turbulent structures and strong interface instabilities.
- **Richtmyer–Meshkov instability** (5.3 GB, ∼106.7M points) [Cohen et al.(2002)], a large-scale hydrodynamics dataset exhibiting shock-driven mixing, representing the largest and most challenging test in our study.

All experiments were conducted on the Polaris supercomputer at Argonne National Laboratory[3], with 4 NVIDIA A100 GPUs per node. For each dataset, we generated 448 training images with orbital camera configurations, providing consistent angular coverage across all spatial partitions.

Scaling Efficiency

In this section, we benchmark the scaling efficiency across multi-GPU and multi-node configurations. All benchmarks used the first partitioning strategy, which divides the domain uniformly along spatial axes by specifying the number of divisions per axis. For each run, the reported training time corresponds to the slowest node.

Single Node Scaling

We first benchmarked the Kingsnake and Rayleigh–Taylor datasets on a single node while varying the number of GPUs. Training times across different image resolutions and GPU counts are reported in Tables 1. The results show that increasing the number of GPUs substantially reduces training time, especially for high-resolution inputs such as 2048×2048. For example, on Kingsnake at 2048×2048, using 4 GPUs achieved a 5.6× speedup compared to a single GPU. As noted by Zhao et al. [Zhao et al.(2024)], a single A100 GPU can support up to approximately 11.2M Gaussians. Larger datasets, such as Rayleigh–Taylor with ∼18M Gaussians, exceed this memory limit and therefore require at least multi-GPU training. This demonstrates the necessity of distributed 3D-GS to enable large-scale scientific visualization.

Table 1: Training time (minutes) for Kingsnake and Rayleigh–Taylor at different resolutions and GPU counts. 'X' indicates runs that exceeded a single A100 GPU's memory.

	Kingsnake (∼4M)		Rayleigh–Taylor (∼18M)	
#GPUs	1024	2048	1024	2048
1	18.60	48.00	X	X
2	10.48	15.46	21.88	50.10
4	5.97	8.50	12.20	16.84

In addition to scaling efficiency, 3D-GS achieves high reconstruction quality, as visualizations illustrated in Figure 3. Quan-

[2] https://www.digimorph.org/index.phtml
[3] https://www.alcf.anl.gov/polaris

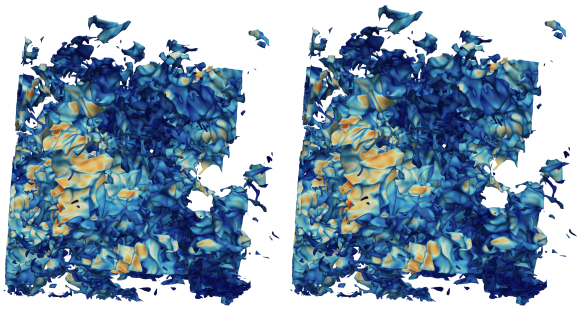
titative results are summarized in Tables 2 and 3, with average metrics of PSNR 29.32, SSIM 0.97, and LPIPS 0.03 for Kingsnake, and PSNR 36.37, SSIM 0.9905, and LPIPS 0.011 for Rayleigh–Taylor.

Table 2: PSNR (↑), SSIM (↑), and LPIPS (↓) for the Kingsnake dataset across different image resolutions and GPU counts.

	512			1024			2048		
#GPUs	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
1	25.52	0.95	0.056	26.90	0.96	0.056	25.12	0.93	0.089
2	25.87	0.96	0.046	28.63	0.97	0.035	29.33	0.97	0.030
4	25.87	0.96	0.046	25.03	0.93	0.067	29.32	0.97	0.030

Table 3: PSNR (↑), SSIM (↑), and LPIPS (↓) for the Rayleigh–Taylor dataset across different image resolutions and GPU counts.

	512			1024			2048		
#GPUs	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
2	31.62	0.99	0.014	34.21	0.99	0.010	36.30	0.99	0.011
4	31.63	0.99	0.014	34.22	0.99	0.010	36.37	0.99	0.011



(a) Ground Truth (b) Distributed 3D-GS

Figure 3: Visualization on the Rayleigh–Taylor dataset: (a) ground truth and (b) reconstruction with our distributed 3D-GS method, achieving high fidelity (PSNR = 36.37, SSIM = 0.9905, LPIPS = 0.011).

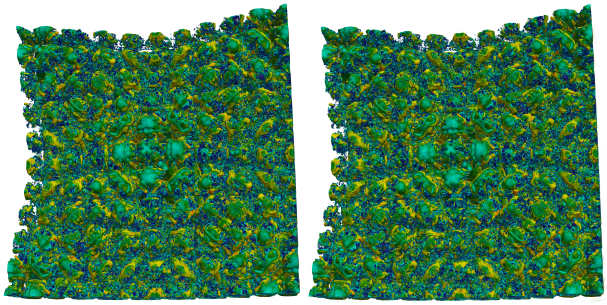
Multi-Node Scaling

To evaluate scalability, we benchmarked the distributed 3D-GS pipeline on the Rayleigh–Taylor and Richtmyer–Meshkov datasets using multi-node runs with 4 GPUs per node.

For Rayleigh–Taylor, training scaled efficiently with a 1.4× speedup when increasing from 2 to 4 nodes (Table 4), while maintaining high reconstruction quality (Table 5). However, because the dataset size is relatively modest, further scaling from 4 to 8 nodes yielded only limited reductions in training time. In contrast, the much larger Richtmyer–Meshkov dataset required at least 4 nodes to fit in memory. Scaling to 8 nodes achieved a 3.1× speedup at 2048^2 resolution (Table 4), while preserving stable image quality (Table 6). Visualizations closely matched ground truth, with no noticeable degradation (Figure 4).

We did observe some decline in reconstruction quality when scaling to higher node counts. For example, for Rayleigh–Taylor at 2048×2048 resolution, PSNR decreased from 37.15 (4 nodes) to 35.42 (8 nodes). This degradation primarily comes from larger boundary errors introduced as the domain is split into more partitions. Addressing these boundary effects remains an important direction for future work.

Overall, these results demonstrate that distributed 3D-GS scales effectively across multiple nodes: delivering substantial speedups for both moderate and large-scale datasets while consistently preserving visualization fidelity.



(a) Ground Truth (b) Distributed 3D-GS Result

Figure 4: Visualization of the Richtmyer–Meshkov instability dataset [Cohen et al.(2002)] containing 106.7M Gaussians. (a) Ground truth image rendered directly from the point cloud. (b) Reconstruction from our distributed 3D-GS method at 2048×2048 resolution. Training was performed on 8 Polaris nodes at Argonne, each with 4 NVIDIA A100 GPUs, completing in 8 minutes. The reconstruction achieves high quality: PSNR=30, SSIM=0.97, LPIPS=0.019.

Table 4: Training time (minutes) for Rayleigh–Taylor and Richtmyer–Meshkov at different resolutions and node counts. 'X' indicates run failed due to memory limits.

	Rayleigh–Taylor (∼18M)		Richtmyer-Meshkov (∼106.7M)	
#Nodes	1024	2048	1024	2048
2	7.22	11.97	X	X
4	5.08	8.33	10.07	32.03
8	5.30	7.45	7.87	10.18

Impact of Load Balancing on Performance

From previous experiments, we observed that Rayleigh–Taylor runs at 2048×2048 resolution exhibited significant load imbalance when using 8 nodes. As shown in Figures 5a and 5b, the first partitioning approach produced uneven data distributions, resulting in training times ranging from 251 s on the fastest node to 447 s on the slowest—a disparity of more than three minutes. To address this, we used the same configuration to benchmark our load-balancing strategy. With the second approach, partitions were generated such that each node contained a comparable number of points (Figures 5c and 5d), leading to a more balanced workload across nodes. After applying the load-balancing approach, training times ranged from 392 s to 420 s, reducing the variation across nodes to about 30 s compared to the previous three-minute disparity. In addition, the load-balancing strategy preserved reconstruction quality, achieving PSNR 35.23, SSIM 0.99, and LPIPS 0.012.

Conclusion and Future Work

We presented a distributed 3D Gaussian Splatting pipeline for large-scale scientific visualization on HPC systems. Our design enables each node to train only on its local data subset, eliminating the need for centralized data loading and supporting scalability across multiple nodes. By incorporating ghost cells and background masks, we ensure artifact-free merging of splats from different nodes. Experiments on three scientific datasets demonstrated the necessity of a distributed 3D-GS pipeline for handling

Table 5: PSNR (↑), SSIM (↑), and LPIPS (↓) for the Rayleigh–Taylor dataset across different image resolutions and compute node counts.

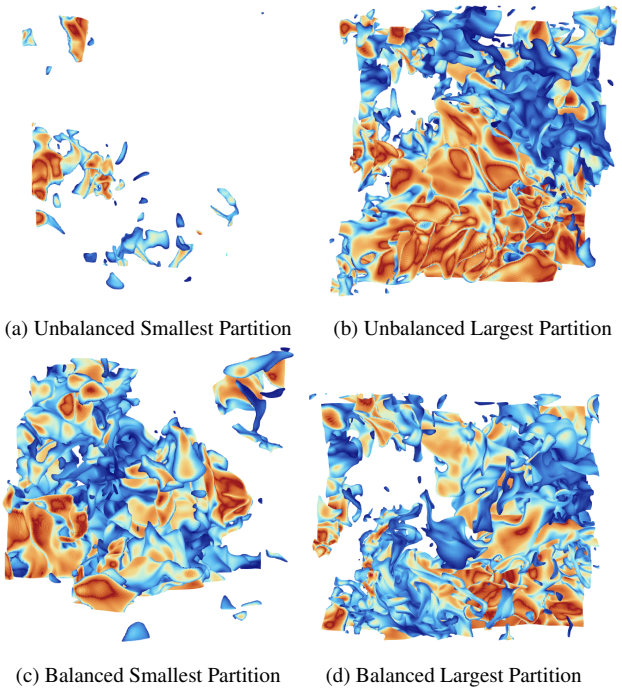
	1024x1024			2048x2048		
#Nodes	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
2	33.64	0.99	0.012	37.22	0.99	0.008
4	33.61	0.99	0.012	37.15	0.99	0.008
8	33.64	0.99	0.012	35.42	0.99	0.012

Table 6: PSNR (↑), SSIM (↑), and LPIPS (↓) for the Richtmyer-Meshkov dataset across different image resolutions and compute node counts.

	1024x1024			2048x2048		
#Nodes	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
4	28.46	0.97	0.018	30.04	0.97	0.019
8	28.20	0.96	0.019	30.04	0.97	0.019

large-scale data. For example, the Rayleigh–Taylor dataset exceeds the memory capacity of a single GPU, while the Richtmyer–Meshkov dataset requires more than two compute nodes (8 GPUs) for training. In addition, we observed substantial performance gains: up to 5.6× speedup with multi-GPU training on a single node and up to 3× speedup across multiple nodes, all while maintaining high visual quality (PSNR over 30, SSIM about 0.99, and low LPIPS).

For future work, we plan to investigate strategies for improving reconstruction quality near partition boundaries, addressing the degradation observed at higher node counts. We also aim to integrate our workflow directly into simulation pipelines to enable real-time, in situ 3D-GS visualization, including the development of a distributed renderer that avoids gathering all splats for final rendering. Finally, we intend to explore methods for estimating reconstruction confidence, providing scientists with richer and more reliable insights from visualized data.



(a) Unbalanced Smallest Partition (b) Unbalanced Largest Partition

(c) Balanced Smallest Partition (d) Balanced Largest Partition

Figure 5: Visualization of the Rayleigh–Taylor instability dataset [Cook et al.(2004)] containing 18M Gaussians. (a) and (b) show the smallest and largest partitions without load balancing, while (c) and (d) show the corresponding partitions with load balancing. The load-balancing strategy produces a more even workload across nodes while preserving reconstruction quality.

Acknowledgments

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357. Finally, this work was partially supported by the National Science Foundation (NSF) through collaborative research award #2221812.

References

[Ai et al.(2025)] Kuangshi Ai, Kaiyuan Tang, and Chaoli Wang. 2025. NLI4VolVis: Natural Language Interaction for Volume Visualization via LLM Multi-Agents and Editable 3D Gaussian Splatting. *arXiv preprint arXiv:2507.12621* (2025).

[Cohen et al.(2002)] Ronald H. Cohen, William P. Dannevik, Andris M. Dimits, Donald E. Eliason, Arthur A. Mirin, Ye Zhou, David H. Porter, and Paul R. Woodward. 2002. Three-dimensional simulation of a Richtmyer–Meshkov instability with a two-scale initial perturbation. *Physics of Fluids* 14, 10 (oct 2002), 3692–3709. `https://doi.org/10.1063/1.1504452`

[Cook et al.(2004)] Andrew W. Cook, William Cabot, and Paul L. Miller. 2004. The Mixing Transition in Rayleigh-Taylor Instability. *Journal of Fluid Mechanics* 511 (2004), 333–362. `https://doi.org/10.1017/S0022112004009681`

[Kerbl et al.(2023)] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/`

[Mildenhall et al.(2021)] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

[Moreland(2011)] Kenneth D Moreland. 2011. *IceT Users' Guide and Reference.* Technical Report. Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA.

[Sewell et al.(2024)] Andres Sewell, Landon Dyken, Victor A Mateevitsi, Will Usher, Jefferson Amstutz, Thomas Marrinan, Khairi Reda, Silvio Rizzi, Michael E Papka, Sidharth Kumar, et al. 2024. High-quality Approximation of Scientific Data using 3D Gaussian Splatting. In *2024 IEEE 14th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 73–74.

[Tang et al.(2025)] Kaiyuan Tang, Siyuan Yao, and Chaoli Wang. 2025. iVR-GS: Inverse Volume Rendering for Explorable

Visualization via Editable 3D Gaussian Splatting. *IEEE Transactions on Visualization and Computer Graphics* (2025).

[Yao and Wang(2025)] Siyuan Yao and Chaoli Wang. 2025. VolSegGS: Segmentation and Tracking in Dynamic Volumetric Scenes via Deformable 3D Gaussians. *arXiv preprint arXiv:2507.12667* (2025).

[Zhao et al.(2024)] Hexu Zhao, Haoyang Weng, Daohan Lu, Ang Li, Jinyang Li, Aurojit Panda, and Saining Xie. 2024. On Scaling Up 3D Gaussian Splatting Training. In *European Conference on Computer Vision*. Springer, 14–36.

Author Biography

Mengjiao Han is a Postdoctoral Researcher at the Argonne Leadership Computing Facility. Her work focuses on scientific visualization, particularly flow field and uncertainty visualization using deep learning, as well as 3D Gaussian splatting for scientific data. She received her MS in Computer Science from the University of Delaware and a Ph.D. in Computing from the University of Utah.

Andres Sewell is a PhD candidate at Utah State University, specializing in high-performance computing and scientific visualization data. His current research focuses on in situ 3D Gaussian Splatting for large-scale scientific data on HPC.

Joseph A. Insley is Team Lead for Visualization and Data Analysis at Argonne National Laboratory's Leadership Computing Facility and Associate Research Professor at Northern Illinois University. His research focuses on scalable visualization methods, immersive technologies, and collaborative data exploration. Insley has an MFA and an MS in Computer Science, both from the University of Illinois at Chicago.

Janet Knowles is a Principal Software Engineering Specialist at the Argonne Leadership Computing Facility. Her interests include immersive environments, 3D modeling, computer animation, and scientific visualization. Janet holds an BSE in Electrical Engineering from the University of Pennsylvania and an MS in Computer Science from the University of Illinois-Chicago. In her free time she loves to run and do jigsaw puzzles.

Victor A. Mateevitsi is a Computer Scientist at the Argonne Leadership Computing Facility. His research focuses on large-scale visualization, augmented and virtual reality, and novel interaction techniques. He holds a PhD in Computer Science from the University of Illinois at Chicago.

Michael E. Papka is a Senior Scientist and Distinguished Fellow at Argonne National Laboratory, where he serves as Deputy Associate Laboratory Director for Computing, Environment, and Life Sciences and Director of the Argonne Leadership Computing Facility. He is also Professor of Computer Science at the University of Illinois Chicago, with research spanning high-performance computing, visualization, and interdisciplinary collaboration.

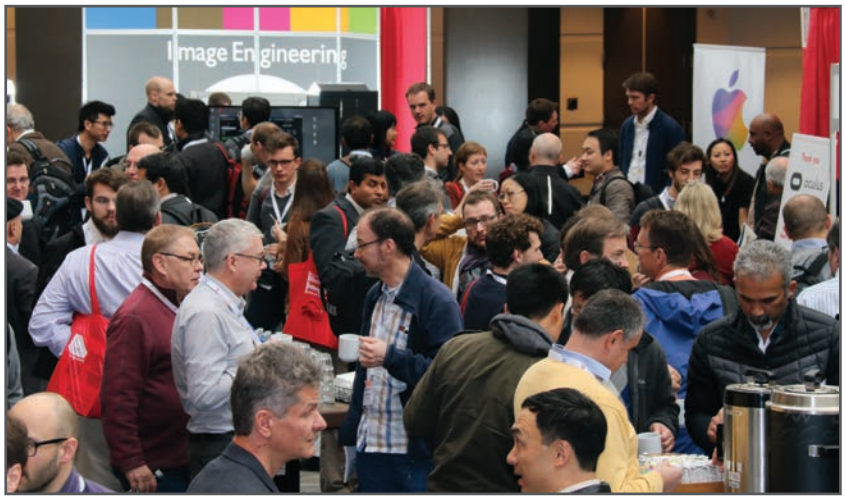
Steve Petruzza is an Assistant Professor at Utah State University. He received his M.S. and B.E. in Computer Engineering and a Ph.D. in Computer Science from the University of Rome "Tor Vergata". His research spans high-performance computing, large-scale data analytics, and GPU-based computing, with prior experience in the defense industry and as a Postdoctoral Fellow at the Scientific Computing and Imaging Institute, University of Utah.

Silvio Rizzi is a Computer Scientist at the Argonne Leadership Computing Facility. His research interests are large-scale and in-situ scientific visualization and analysis, display technologies, and immersive visualization environments. He has an MS in Electrical and Computer Engineering and a PhD in Industrial Engineering and Operations Research, both from the University of Illinois-Chicago.

JOIN US AT THE NEXT EI!

electronic IMAGING

Imaging across applications . . . Where industry and academia meet!



• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

ADVANCING IMAGING SCIENCE 75 1947 IS&T 2022 imaging.org