3D Gaussian Splatting for Scientific Particle Data Compression and Rendering

ParticleGS

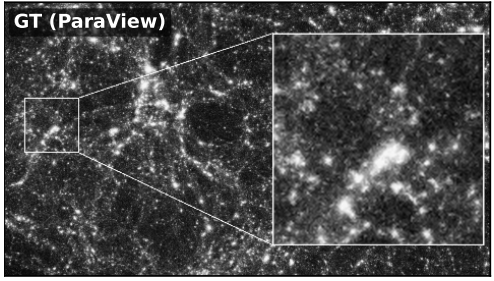
Bo Jiang[1], Youyuan Liu[1], Taolue Yang[1], Sheng Di[2], Sian Jin[1]

1. Temple University 2. Argonne National Laboratory

SC26 · Chicago

Particle simulations produce massive snapshots

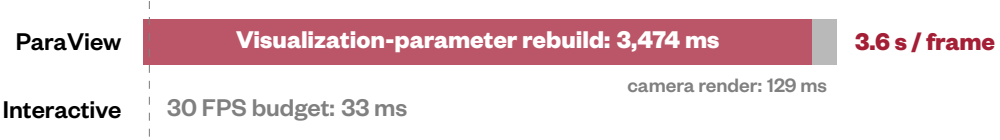
- Cosmology (**HACC**, FIRE-2), molecular dynamics (LAMMPS): millions to billions of particles per snapshot.
- Each particle is a 3D position, an **unstructured** point cloud, not a grid.
- Our test snapshot: **281 M particles, 3.4 GB**.
- The largest HACC runs: **trillions** of particles, tens to hundreds of TB per snapshot, PB per campaign.



HACC snapshot rendered in ParaView

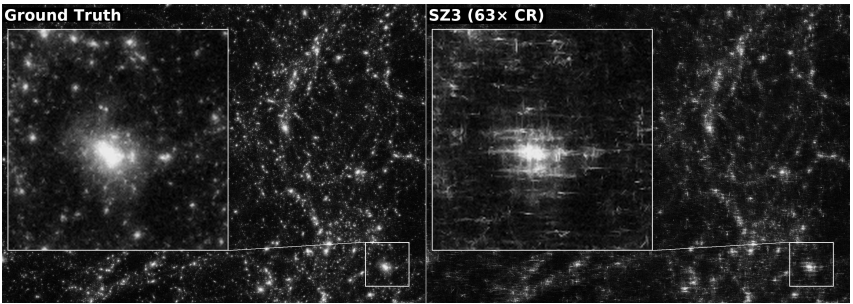
Exploring raw particles is already slow

- ParaView draws every particle as a screen-space **Gaussian splat** with radius *r* and opacity α.
- Scientists keep changing *r* and α: large and faint shows the filament network, small and bright shows local density.
- Every change **rebuilds the pipeline over all 281 M particles**.



Per-frame latency at 1080p, 281 M HACC particles, ParaView 6.0.1, RTX PRO 6000

Data-space compression breaks the rendered image



Close-up of HACC in ParaView: ground truth (left) vs. SZ3 at 63× compression (right)

- Lossy compressors bound the error of each **coordinate value**, not the **image**.
- SZ3 compresses *x*, *y*, *z* separately: errors line up as **axis-aligned streaks**.

Compressors bound data error, not image quality

General-purpose (grid-oriented)

- **SZ3**: predict each value from neighbors, quantize the residual, absolute error bound ε.
- ZFP, MGARD, TTHRESH, HPEZ; GPU versions (cuSZ) reach tens of GB/s.
- On particles: three separate 1D arrays, low spatial coherence.

Particle-specific

- **LCP**: reorder particles along a space-filling curve, then SZ-style prediction.
- Errors become coherent along the curve: **structured** artifacts.

Quality-aware

- Target PSNR/SSIM or quantities of interest, but still measured in data space.

Takeaway No existing compressor optimizes the image that scientists actually look at.

3D Gaussian Splatting: a fast, differentiable renderer

What 3DGS is

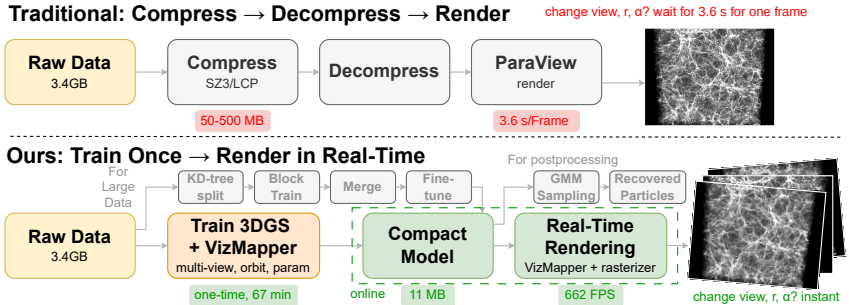
- A scene is a set of 3D Gaussians: center, covariance (scale + rotation), opacity, color.
- Project each to a 2D ellipse, alpha-blend front to back.
- Tile-based CUDA rasterizer: **hundreds of FPS**.
- Fully differentiable: fit all Gaussians to **images** by gradient descent.

Three gaps for particle data

1. **No photos, no cameras**: we only have particle coordinates.
2. **Fixed appearance**: photos never change, but users keep changing r and α.
3. **Scale**: one model has limited capacity for 281 M particles.

Takeaway ParaView already draws Gaussian splats: 3DGS fits, if we close these three gaps.

ParticleGS: compress what scientists see



1. Rendered supervision

No photos: render training views of all particles in ParaView.

2. VizMapper

Fixed appearance: a tiny MLP follows the user's radius and opacity.

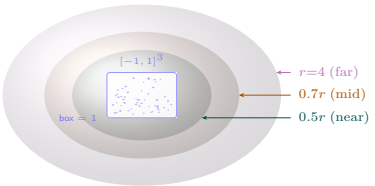
3. KD-tree block training

Scale: train blocks in parallel, merge, fine-tune globally.

Rendered supervision: train on views of all particles

Training views

- Ground truth: ParaView renders **all 281 M particles**.
- Each image has its own $r, \alpha \sim \text{Beta}(3,3)$ around the defaults.
- Init from a 200 K subsample; grows to 171 K Gaussians.



Orbits: **far** 1.0×, **mid** 0.7×, **near** 0.5×

Schedule: 39 K iterations, 67 min

- Stage 1: 1920×1080, 12 K iterations, coarse layout.
- Stage 2: 5760×3240, 27 K iterations, fine detail; +20% cameras inside the volume.

Loss

- L1 on **content pixels** only, not the black background.
- No DSSIM (+68% size, no gain); SH degree 0 (−59% size).

Why a static model is not enough

Option 1: one per setting

Train again for every $(\boldsymbol{r}, \alpha)$. Too expensive, and nothing for settings never trained.

Option 2: one static model

Train on random $(\boldsymbol{r}, \alpha)$ without adaptation. The Gaussians learn an **average** look: blurry.

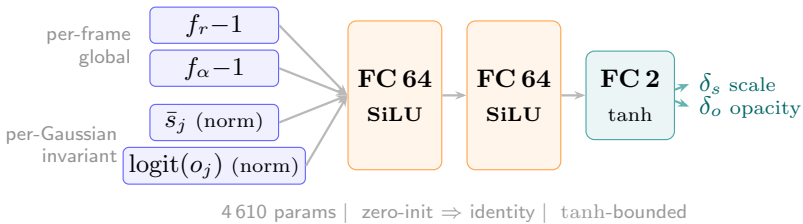
−5.51 dB, even with 30% more Gaussians.

Our choice: VizMapper

One shared model plus a small per-Gaussian correction of **scale** and **opacity**, conditioned on $(\boldsymbol{r}, \alpha)$.

Changing *r* and α only changes particle size and transparency, so VizMapper never touches position or color.

VizMapper: a 4.6 K-parameter MLP per Gaussian



Inputs: the change from the defaults, $f_r - 1$ and $f_\alpha - 1$ with $f_r = r/r_0$, $f_\alpha = \alpha/\alpha_0$, plus the Gaussian's own log-scale and opacity logit.

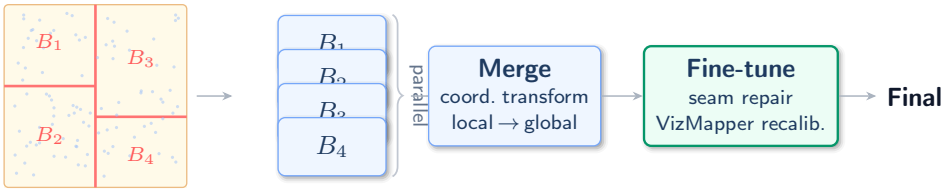
Outputs: bounded residuals δ_s, δ_o (tanh).

$$\mathbf{s}_j' = \mathbf{s}_j \cdot f_r \cdot (1 + \delta_s)$$

$$o_j' = \sigma\big(\mathrm{logit}(o_j) + \log \max(f_\alpha(1 + \delta_o), \epsilon)\big)$$

Zero-initialized output: starts as the identity. Trained jointly, end to end.

Spatial block training with global fine-tuning



1. Partition

KD-tree bisects the longest axis: compact blocks with equal particle counts, so per-block cost is balanced.

2. Train in parallel

Each block has its own normalization and orbits; one GPU per block. Merge back to global frame in 0.11 s.

3. Global fine-tune

60 K iterations at 1080p (15 min) to repair seams and recalibrate VizMapper, with wider correction ranges.

Experimental setup

Datasets

- **HACC**: 281 M particles, $[0, 256]^3$, 3.4 GB (primary).
- HACC: 4 disjoint regions, $\sim$268 M each.
- **FIRE-2**: 268 M dark-matter particles, 3.2 GB, a different simulation code.

Ground truth

- ParaView 6.0.1 Point Gaussian, 1920$\times$1080.
- 3 orbits $\times$ 80 frames = 240 test views, each with its own sampled r, α.

Hardware: 2$\times$ NVIDIA RTX PRO 6000.

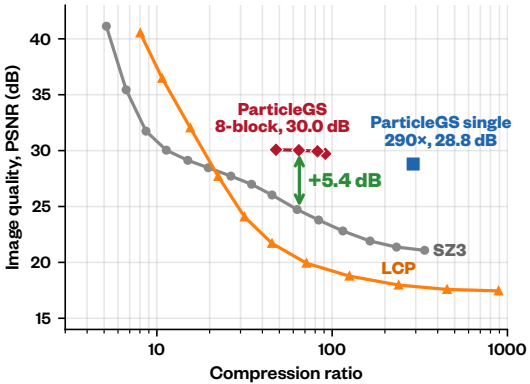
Metrics

- Rendered-image **PSNR** and model size (compression ratio vs. 3.4 GB of raw positions).
- Also SSIM and masked PSNR (foreground pixels only).

Baselines

- **SZ3** (absolute error bound, per axis) and **LCP** (particle-specific).
- Decompressed particles go through the **same ParaView pipeline**; we sweep error bounds.

Rate-distortion: 5 to 11 dB better at matched ratios



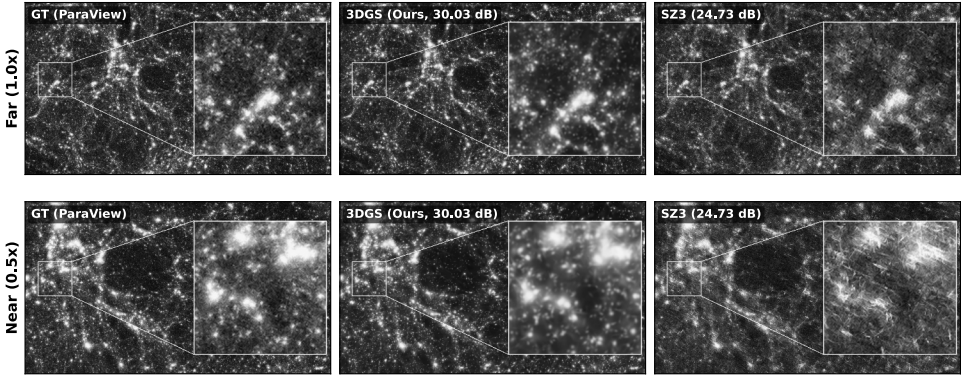
Method	CR	Size	PSNR
SZ3	290×	11 MB	21.2
LCP	290×	11 MB	17.9
Ours, single	**290×**	11.1 MB	**28.80**
SZ3	65×	50 MB	24.7
LCP	65×	50 MB	20.3
Ours, 8-block	**65×**	49.8 MB	**30.03**

290×: **+7.6 dB** vs. SZ3, +10.9 vs. LCP

65×: **+5.4 dB** vs. SZ3, +9.7 vs. LCP

HACC, 281 M particles · PSNR in dB, averaged over far/mid/near test orbits

Visual comparison at 65× compression



Top: far orbit (1.0×) · bottom: near orbit (0.5×) · center 30% crop, inset zoom

Performance: 2,386× faster per frame

Per frame, 1080p	ParaView	Ours	Speedup
Viz-param adapt.	3,474 ms	1.21 ms	2,871×
Camera render	129 ms	0.38 ms	339×
Combined	3,603 ms	1.51 ms	**2,386×**
Combined, 4K	3,648 ms	1.93 ms	1,890×
Data loading	39.4 s	41 ms	968×

662 FPS at 1080p (518 at 4K)
vs. 0.28 FPS in ParaView

462 MB GPU memory at inference

One-time cost
Train 67 min, incl. ground truth
8-block: +15 min fine-tune

ParaView renders 281 M particles; ours renders 769 K Gaussians (8-block model) · RTX PRO 6000

Ablation: every component matters

Core training (single block)

Variant	PSNR	Δ	Gaussians
Full pipeline	28.80	—	171 K
w/o VizMapper	23.30	−5.51	223 K
w/o progressive res.	26.42	−2.39	169 K
w/o multi-orbit	28.04	−0.77	191 K

Fine-tune recipe (8-block)

Recipe	PSNR	Size
L1 + densify	30.03	49.8 MB
w/o densify	28.61	57.9 MB
L1 + 0.1 DSSIM	28.26	54.2 MB
Pure DSSIM	26.60	298.2 MB

Takeaway VizMapper matters most; extra Gaussians cannot replace it.

Block training: fine-tuning is essential

Blocks	Merged	Fine-tuned	Δ	Size
1	—	28.80	—	11.1 MB
2	28.99	29.72	+0.92	35.2 MB
4	28.85	29.94	+1.14	39.0 MB
8	28.32	**30.03**	+1.23	49.8 MB
16	26.80	30.08	+1.28	67.2 MB

- Merging alone **degrades** with more blocks (26.80 dB at 16).
- Fine-tuning recovers and **beats** the single model every time.
- Gains saturate after 4 blocks; more blocks still add training parallelism.
- Largest gain on near views: **+1.49 dB** masked PSNR at 8 blocks.

HACC, 281 M particles · PSNR in dB · fine-tune: 60 K iterations

Generalization across data, views, and parameters

New data, same hyperparameters

Dataset	PSNR	Size	CR
HACC, 4 regions	27.36 ± 0.04	11.3 MB	273×
FIRE-2	29.27	5.3 MB	577×

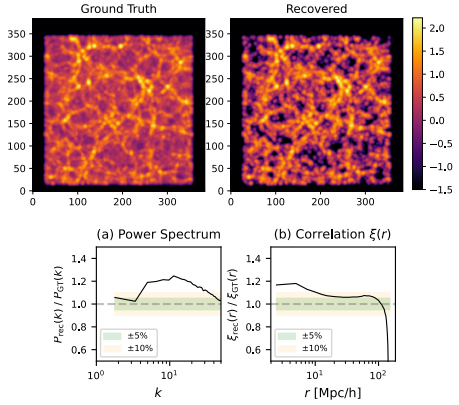
- Model size grows **sub-linearly**: 94 K Gaussians for 17.6 M particles, 171 K for 281 M.
- **Unseen** r, α: quality falls smoothly outside the trained range (21.85 dB at 2× radius).

Unseen camera orbits (single model, masked PSNR in dB)

Orbit radius	1.3× extrap.	**1.0×** trained	0.85× interp.	**0.7×** trained	0.6× interp.	**0.5×** trained	0.35× extrap.
Masked PSNR	26.20	26.32	26.33	26.43	26.37	26.07	**25.10**

Within 0.2 dB of the trained orbits, except −1.3 dB at the very close 0.35×.

Byproduct: approximate particle recovery



Gaussians as a GMM, $w_j \propto o_j$: sample 281 M particles in 74.5 s.

- **Density correlation 0.928.**
- **$P(k)$: $\pm$10% up to $k = 2.63$ h/Mpc.**
- **$\xi(r)$: $\pm$10% for $r > 10$ Mpc/h.**
- **Not per-particle: NN distance 0.250 vs. 0.665.**

8-block model · top: density slice · bottom: ratio to ground truth, $\pm$5% and $\pm$10% bands

Conclusion

ParticleGS

- Learns what scientists **see**: a visualization-aware 3DGS model.
- **VizMapper**: one model, any radius and opacity.
- **Block training**: parallel blocks, merge, global fine-tune.
- **+5.4 dB** over SZ3 at 65×; **290×** with 11 MB; **662 FPS**.

Limits and scope

- Best inside the trained orbits and (r, α) range.
- Shown up to 281 M particles; trillions are future work.
- Recovery is density-level only.
- An **interactive visual surrogate**, not ground-truth measurement.

Next: time-varying data, other rendering modes, other domains.

Takeaway Code and artifact: github.com/BoJiang03/ParticleGS

Thank you!

Questions are welcome!

Bo Jiang · bo.jiang@temple.edu

Code: `github.com/BoJiang03/ParticleGS`



