GPU-Accelerated Scalar Field Reconstruction and Volume Rendering

Utkarsh Sharma [24116]
utkarsh2024@iisc.ac.in

Akash Maji [24212]
akashmaji@iisc.ac.in

***Abstract*—We present a GPU-accelerated volume renderer for astrophysical data from large-scale simulations such as IllustrisTNG. Raw particle attributes (e.g., positions, densities) are converted into volumetric fields and visualized via CUDA/OpenGL ray casting. Emphasis is placed on visual fidelity through advanced shading, adaptive sampling, and transfer functions to reveal structures like cosmic filaments, halos, and shock fronts.**

***Index Terms*—Volume Rendering, GPU Ray Casting, CUDA, OpenGL, Astronomy, IllustrisTNG**

I. Objective

The goal of this project is to build a GPU-based direct volume rendering (DVR) system that preserves high visual quality when visualizing astrophysical scalar fields. We focus on evaluating how different particle-to-volume representations - traditional grid resampling, SPH kernels, and learned neural field models affect reconstruction accuracy and final render fidelity.

II. Work Done

We first explore three core methods:

- **Gaussian SPH Kernel Splatting**: Smoothly reconstructs particle data into a voxel density field using Gaussian kernel contributions.
- **Nearest Neighbor Interpolation**: Assigns each voxel the value of its closest particle, providing a fast but coarse reconstruction of the field.
- **Voronoi Tessellation-Based Linear Reconstruction**: Uses Voronoi cells with per-particle gradients for smooth, continuous field interpolation without voxelization.

We then explored following reconstruction techniques:

- **Sibson Natural Neighbor**: Voronoi-based interpolation using area-weighted contributions from natural neighbors for smooth reconstruction.
- **PlenOctree Corner Interpolation**: Octree-based spatial hierarchy with trilinear interpolation at corner nodes for adaptive resolution rendering.
- **Neural Hash Grid (Instant-NGP)**: Multi-resolution hash encoding with MLP decoder for learning continuous density representations.

III. Implementation Details

This work investigates GPU-accelerated particle-based field reconstruction and volume rendering approaches. Each method offers different trade-offs in visual smoothness, physical accuracy, and computational complexity. CUDA-OpenGL Interop allows direct writing into a PBO for interactive performance at high resolutions.

A. Gaussian SPH Kernel Splatting

Converts discrete particles into a smooth voxel-based density field using scatter-based GPU voxelization. Each particle contributes density to surrounding voxels within support radius $2.5h$ via Gaussian kernel:

$$W(r,h) \propto \exp\left(-\frac{r^2}{h^2}\right)$$

Contributions accumulate via `atomicAdd`, followed by normalization. A hierarchical macro-volume structure (8^3 blocks) stores maximum densities for empty-space skipping during ray traversal. Local gradients approximate normals via central differences: $\mathbf{N}(\mathbf{x}) \approx -\nabla\rho(\mathbf{x})$.

B. Nearest Neighbor Interpolation

Meshless direct rendering evaluates the scalar field on-the-fly using Inverse Distance Weighting (IDW) over K nearest particles:

$$\rho(\mathbf{x}) = \frac{\sum_{i=1}^{K} w_i(\mathbf{x})\rho_i}{\sum_{i=1}^{K} w_i(\mathbf{x})}, \quad w_i(\mathbf{x}) = \frac{1}{\|\mathbf{x}-\mathbf{p}_i\|^p}$$

Particles are hashed into a uniform grid with sorted lookup tables enabling $O(1)$ neighbor access within 3^3 search regions. Ray marching directly evaluates $\rho(\mathbf{x})$ at each sample point.

C. Voronoi Tessellation-Based Linear Reconstruction

Hybrid CPU-GPU approach enabling first-order field interpolation. Voronoi cells computed via `voro++` define natural neighbor relationships for weighted least-squares gradient estimation:

$$\nabla\rho_i = \arg\min_{\mathbf{g}} \sum_j w_{ij} \left(\mathbf{g}\cdot(\mathbf{p}_j-\mathbf{p}_i) - (\rho_j-\rho_i)\right)^2$$

Field reconstructed piecewise-linearly: $\rho(\mathbf{x}) \approx \rho_i + \nabla\rho_i \cdot (\mathbf{x}-\mathbf{p}_i)$ within each Voronoi cell, eliminating blocky artifacts. Spatial hash grid accelerates GPU-based nearest-particle queries.

D. Sibson Natural Neighbor Interpolation

Extends Voronoi approach using area-based weighting for C-continuous interpolation. For sample point $\mathbf{x}$, temporarily insert into tessellation to identify natural neighbors. Sibson coordinates computed from "stolen" areas:

$$\lambda_i(\mathbf{x}) = \frac{A_i(\mathbf{x})}{\sum_j A_j(\mathbf{x})}, \quad \rho(\mathbf{x}) = \sum_{i \in \mathcal{N}(\mathbf{x})} \lambda_i(\mathbf{x}) \rho_i$$

Satisfies partition of unity and reproduces linear functions exactly, but computationally intensive due to geometric area calculations.

E. PlenOctree Corner Interpolation

Hierarchical spatial structure with adaptive resolution. Particles binned into octree (max depth D) with density values cached at 8 corners of leaf nodes. Trilinear interpolation within cells:

$$\rho(\mathbf{x}) = \sum_{k=0}^{7} \phi_k(u, v, w) \rho_k$$

where $(u, v, w) \in [0, 1]^3$ are normalized coordinates and ϕ_k are trilinear basis functions. Ray traversal uses octree hierarchy for empty-space skipping with adaptive step sizes and early termination when $\alpha > 0.95$.

F. Neural Hash Grid (Instant-NGP)

Learning-based continuous representation using multi-resolution hash encoding. Position $\mathbf{x}$ mapped to L resolution levels, hashed into compact feature tables:

$$h(\mathbf{x}, \ell) = \left(\bigoplus_{d=1}^{3} \lfloor x_d \cdot N_\ell \rfloor \cdot \pi_d \right) mod T$$

Features trilinearly interpolated at corners, concatenated across levels, then decoded by compact MLP.

IV. Results

In subsections A-C, the rendered image was saved at a resolution of 3840×2160. The data set used is: Illustris-1.

In subsections D-I, the rendered image was saved at a resolution of 800×800. The data set used is: Subhalo 468590.

A. Gaussian Splatting Results

We evaluate the performance impact of adaptive empty-space skipping on the Gaussian splatting renderer implemented in `CUDA`.

Gaussian: Image Quality Comparison
MSE: 60.8140
PSNR: 30.2908 dB
SSIM: 0.9731 (max 1.0)

Configuration	FPS	GPU vRAM	Render Time
Without adaptive marching	2.7	1.17 GB	0.37 s
With adaptive marching	2.1	1.17 GB	0.46 s

TABLE I
RENDERING PERFORMANCE COMPARISON FOR GAUSSIAN SPLATTING

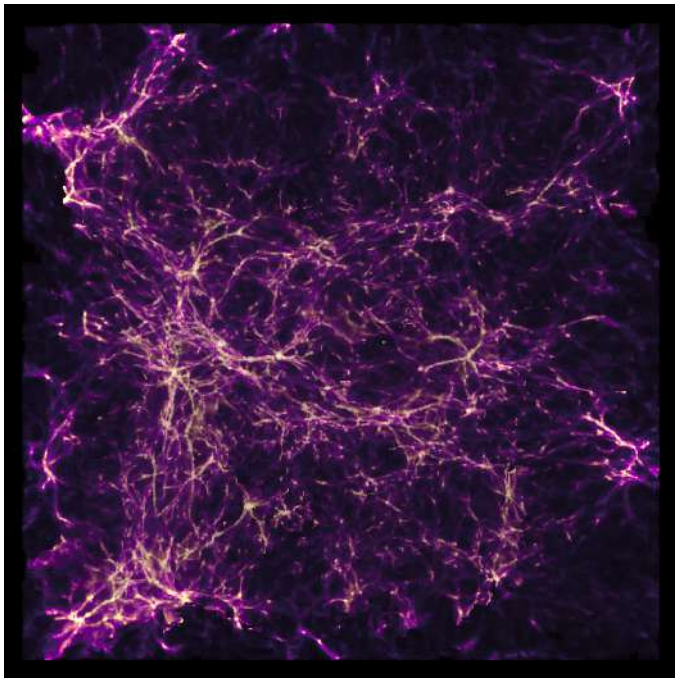


Fig. 1. Gaussian Splatting with adaptive marching.

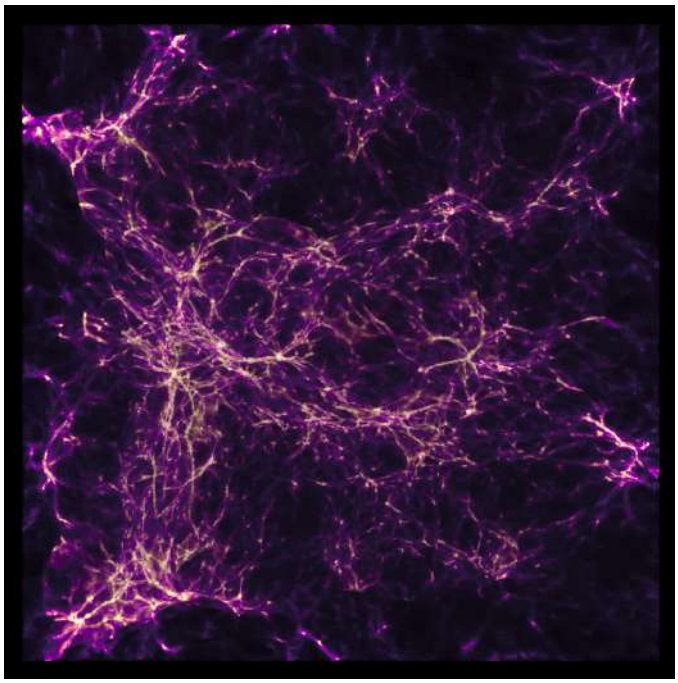


Fig. 2. Gaussian Splatting without adaptive marching.

1) Gaussian SPH at Different Voxel Grid Resolutions: We analyze how voxel grid resolution impacts performance, memory usage, and visual fidelity in the Gaussian splatting pipeline. The highest resolution (800^3) is used as the reference image for other resolutions as shown below:

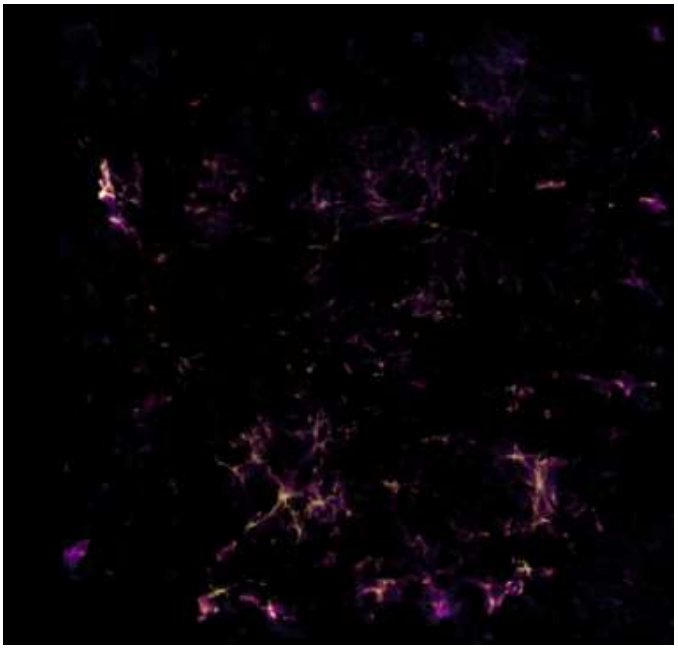


Fig. 3. Gaussian: Pixel Difference

Grid Resolution	GPU vRAM	FPS	Render Time
800^3	2.58 GB	0.4	2.77 s
512^3	1.17 GB	2.7	0.36 s
256^3	0.74 GB	6.1	0.17 s
128^3	0.68 GB	8.1	0.14 s
64^3	0.67 GB	8.5	0.11 s

TABLE II
PERFORMANCE SCALING OF GAUSSIAN SPH

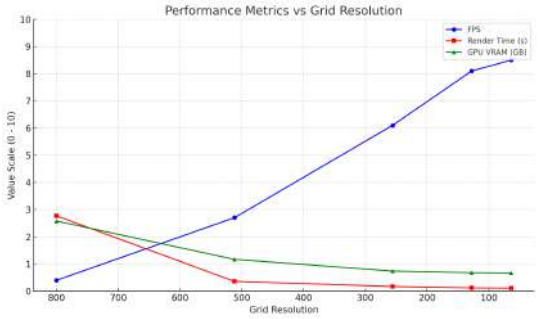
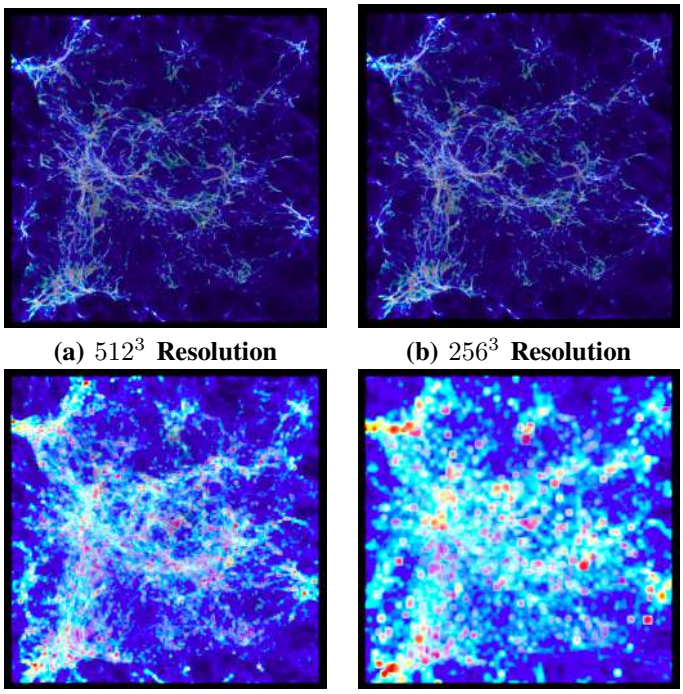


Fig. 4.

Quality Comparison vs 800^3 Reference			
Resolution	**MSE**	**PSNR (dB)**	**SSIM**
512^3	1002.0069	18.1221	0.8261
256^3	1550.0694	16.2273	0.8075
128^3	5626.6464	10.6283	0.7529
64^3	7519.8710	9.3687	0.7477

B. Voronoi Reconstruction Results

To evaluate the effect of gradient-enhanced field reconstruction in the Voronoi-based method (`in CUDA`), we compare two configurations:



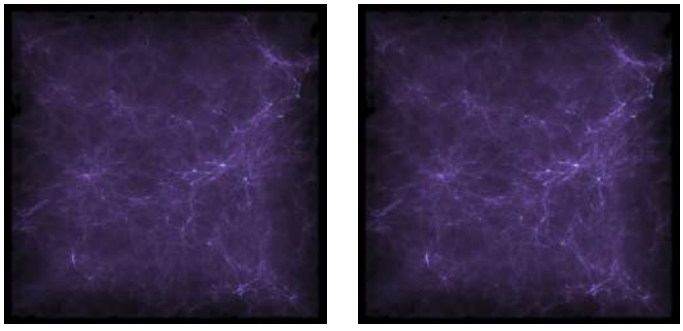
(a) 512^3 **Resolution** **(b)** 256^3 **Resolution**

(c) 128^3 **Resolution** **(d)** 64^3 **Resolution**

Fig. 5. Gaussian SPH rendering outputs at different voxel grid resolutions.

- **With gradient**: Linear reconstruction using $\rho(\mathbf{x}) = \rho_i + \nabla\rho_i \cdot (\mathbf{x} - \mathbf{p}_i)$
- **Without gradient**: Constant reconstruction using only ρ_i of the nearest particle

The gradient-based approach produces smooth transitions within each Voronoi cell and reduces the visibility of cell boundaries. Conversely, the constant reconstruction results in blocky artifacts and noticeable faceting, particularly in regions of high density variation.



(a) With gradient **(b) Without gradient**

Fig. 6. Visual comparison of Voronoi-based rendering.

Voronoi: Image Quality Comparison
MSE: 2.9613
PSNR: 43.4159 dB
SSIM: 0.9957 (max 1.0)

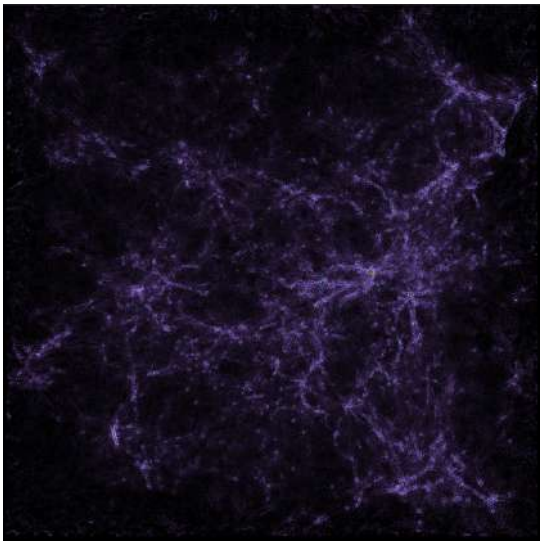
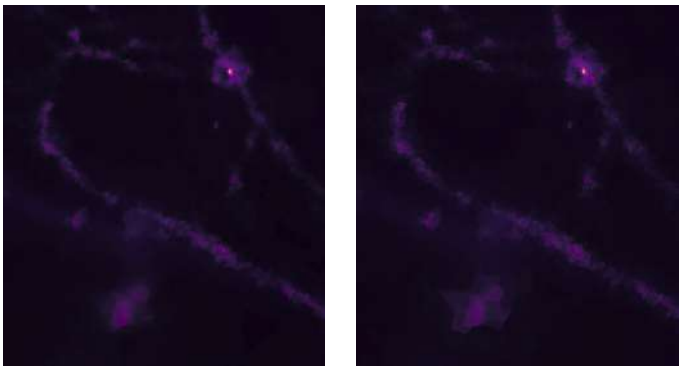


Fig. 7. Voronoi: Difference in pixels (scaled)



(a) smooth (cutout) **(b) blocky (cutout)**

Fig. 8. Visual comparison of Voronoi-based rendering.

Voronoi: Performance Metrics
GPU vRAM: 1.08 GB
CPU Voronoi Time: ∼165 sec
GPU Render Time: 4.84 sec

C. K-Nearest Neighbor Results

In the meshless KNN-IDW rendering approach, the number of neighbors K determines the locality and smoothness of the reconstructed density field. A lower K retains sharper particle-driven detail but may introduce noise, while a higher K reduces variance and produces smoother interpolation, takes more time.

1) Effect of K-Nearest Neighbors on Image Quality: To evaluate this effect, we compare metrics using $m = 3$ and $m = 5$ ray sampling and varying K, where width m means we look for k neighbours among $m*m$ voxels around the voxel the particle is in. The results are summarized in Table VII.

These results demonstrate that the IDW-based K-nearest method is robust to changes in K for this dataset. Moderate values such as $K = 4$ or $K = 8$ offer the best balance between computational cost and reconstruction accuracy, while even minimal settings retain excellent visual fidelity.

MSE (m=3)	**k1**	**k2**	**k4**	**k8**	**k16**
k1	0.0000	2.7089	9.3369	28.1032	72.4126
k2	2.7089	0.0000	3.9308	18.8206	58.3462
k4	9.3369	3.9308	0.0000	7.3100	37.3427
k8	28.1032	18.8206	7.3100	0.0000	13.7412
k16	72.4126	58.3462	37.3427	13.7412	0.0000

TABLE III
PAIRWISE MSE MATRIX FOR $m = 3$.

MSE (m=5)	**k1**	**k2**	**k4**	**k8**	**k16**
k1	0.0000	2.7032	9.1149	27.5294	70.9944
k2	2.7032	0.0000	3.7468	18.3163	57.0339
k4	9.1149	3.7468	0.0000	7.1410	36.6184
k8	27.5294	18.3163	7.1410	0.0000	13.5039
k16	70.9944	57.0339	36.6184	13.5039	0.0000

TABLE IV
PAIRWISE MSE MATRIX FOR $m = 5$.

PSNR (dB, m=3)	**k1**	**k2**	**k4**	**k8**	**k16**
k1	∞	43.8028	38.4288	33.6433	29.5327
k2	43.8028	∞	42.1860	35.3845	30.4707
k4	38.4288	42.1860	∞	39.4916	32.4087
k8	33.6433	35.3845	39.4916	∞	36.7506
k16	29.5327	30.4707	32.4087	36.7506	∞

TABLE V
PAIRWISE PSNR MATRIX FOR $m = 3$.

PSNR (dB, m=5)	**k1**	**k2**	**k4**	**k8**	**k16**
k1	∞	43.8120	38.5333	33.7328	29.6186
k2	43.8120	∞	42.3942	35.5024	30.5695
k4	38.5333	42.3942	∞	39.5932	32.4938
k8	33.7328	35.5024	39.5932	∞	36.8262
k16	29.6186	30.5695	32.4938	36.8262	∞

TABLE VI
PAIRWISE PSNR MATRIX FOR $m = 5$.

K	**MSE**	**PSNR (dB)**	**SSIM**	**Verdict**
$K = 1$	0.0000	∞	1.0000	Perfect Match
$K = 2$	0.0074	69.4314	0.9999	Excellent
$K = 4$	0.0817	59.0087	0.9996	Excellent
$K = 8$	0.1570	56.1724	0.9993	Excellent
$K = 16$	0.2293	54.5264	0.9993	Excellent

TABLE VII
IMAGE SIMILARITY METRICS (M=3 VS M=5)

2) Performance and Similarity Analysis for K-Nearest Neighbor Rendering: To evaluate the influence of the neighborhood size K and sampling depth m on rendering performance and reconstruction fidelity, we analyze both GPU render time and pairwise perceptual similarity across configurations.

Figure 9 shows that GPU runtime increases monotonically with K. This trend is more pronounced for the higher sampling rate ($m = 5$), where each additional neighbor increases the density lookup cost during ray marching.

To further assess the consistency of the reconstruction under different K values, we compute the Peak Signal-to-Noise Ratio (PSNR) and Mean-Squared-Error (MSE) between all pairwise combinations of renderings. The resulting PSNR and MSE matrices for $m = 3$ and $m = 5$ are visualized as heatmaps in Fig. 10 and Fig.11.

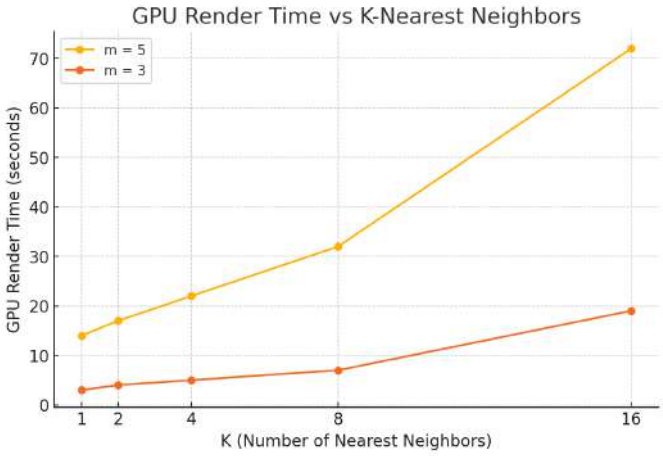
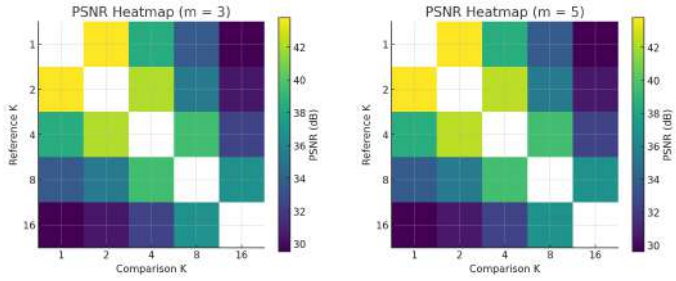
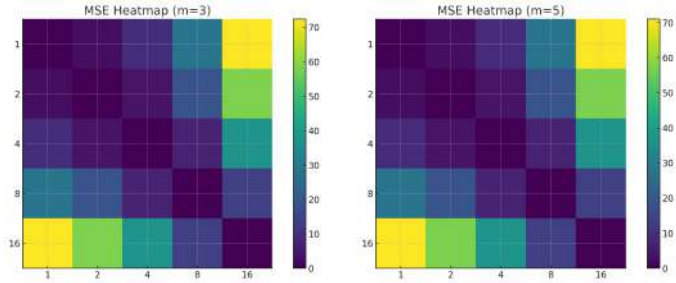


Fig. 9. GPU Render Time vs. K for KNN IDW



(a) $m = 3$ **(b)** $m = 5$

Fig. 10. Pairwise PSNR comparison between KNN configurations.



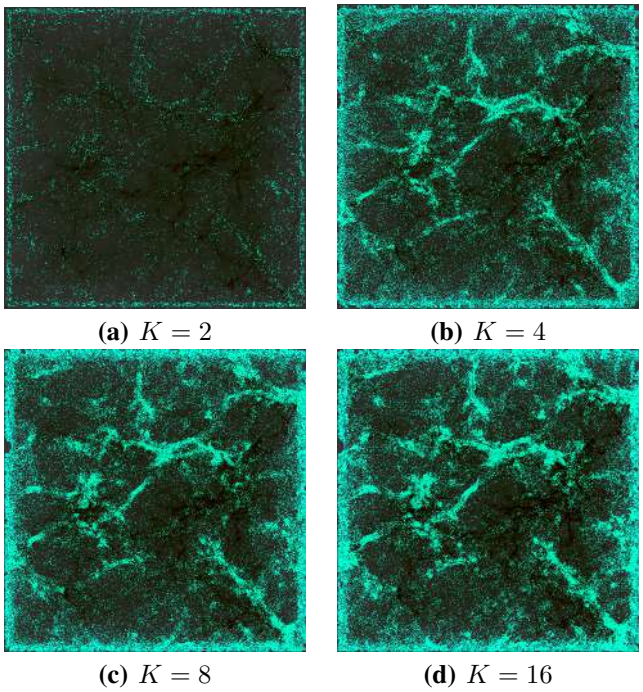
(a) $m = 3$ **(b)** $m = 5$

Fig. 11. Pairwise MSE comparison between KNN configurations.

In Fig.10, the bright diagonal elements indicate perfect self-consistency (∞ dB), while off-diagonal entries show that similarity gradually decreases as the difference between K values increases.

In Fig.11, the MSE grows monotonically with increasing separation in K, confirming that reconstruction stability is strongly dependent on neighborhood size consistency. Meanwhile, the close alignment of the $m = 3$ and $m = 5$ heatmaps suggests that the depth (to look for k neighbors) minimal influence on comparative structural differences between configurations.

3) Pixel-wise Difference Images for K-Nearest Neighbor Rendering: The pixel-wise difference visualizations highlight how reconstruction quality varies as K changes for images with m=3 and m=5. Small K values retain sharper details with minimal changes across sampling depths, while large K values introduce smoother, more noticeable deviations.



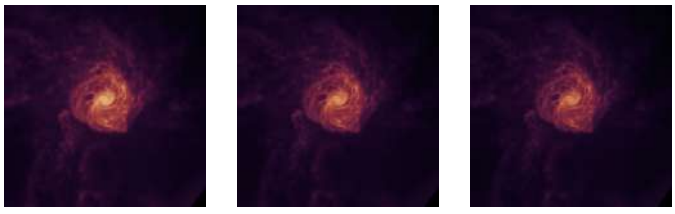
(a) $K = 2$ **(b)** $K = 4$ **(c)** $K = 8$ **(d)** $K = 16$

Fig. 12. Pixel-wise difference across K values.

4) Conclusion: Overall, the results demonstrate that the K-nearest neighbor approach is highly stable: even small K values maintain strong perceptual similarity, whereas large K values incur significant rendering overhead with limited visual improvement. Therefore, moderate settings such as $K = 4$ or $K = 8$ offer an optimal balance between performance and quality.

D. Tetrahedra Linear Interpolation Results

Delaunay tetrahedralization provides a natural geometric structure for particle data. Each tetrahedron enables smooth barycentric interpolation, ensuring C^0 continuity across cell boundaries. This method offers high fidelity reconstruction particularly suited for irregularly distributed particles.



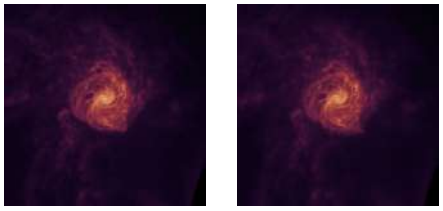
(a) 256^3 **(b)** 512^3 **(c)** 1024^3

Fig. 13. Tetrahedra linear interpolation at varying sampling resolutions.

E. Sibson Natural Neighbor Interpolation Results

Natural neighbor interpolation based on Voronoi tessellation uses area-weighted contributions from neighboring particles.

This Sibson coordinate approach ensures smooth, continuous fields without requiring gradient estimation, offering a middle ground between simple nearest-neighbor and gradient-enhanced methods.



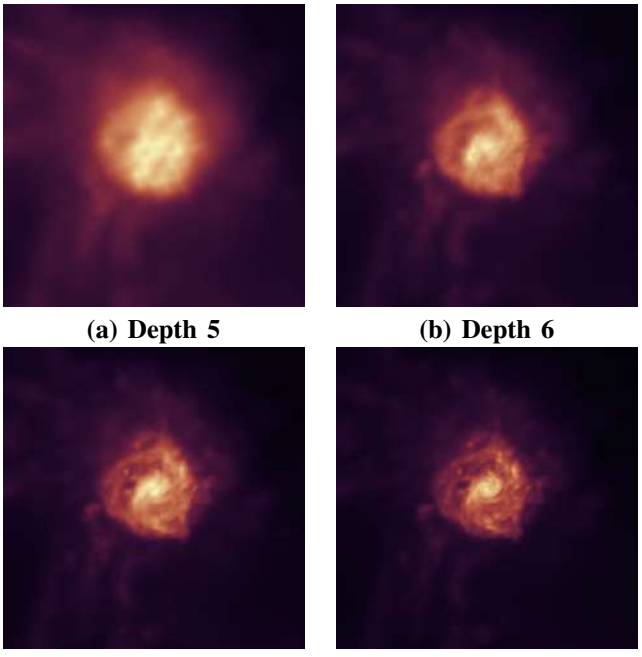
(a) 256^3 **(b)** 512^3

Fig. 14. Natural neighbor reconstruction at different resolutions.

This is taken as baseline for all comparisons below.

F. PlenOctree Corner Interpolation Results

PlenOctree-based rendering uses an octree spatial hierarchy where density values are stored at octree corners. During ray marching, trilinear interpolation between corner values provides smooth field reconstruction. Deeper octree levels offer higher spatial resolution but increased memory and traversal cost.



(a) Depth 5 **(b) Depth 6**

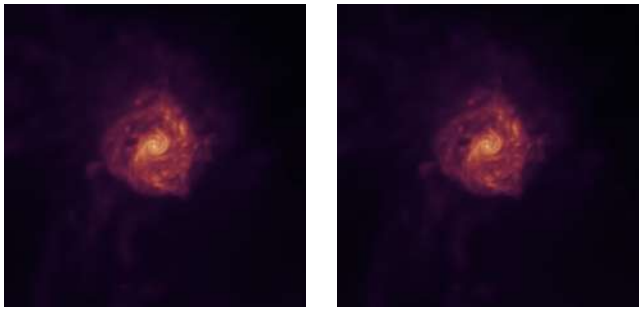
(c) Depth 7 **(d) Depth 8**

Fig. 15. PlenOctree rendering at increasing octree depths.

The octree structure enables adaptive resolution, focusing detail where needed while maintaining efficiency in homogeneous regions.

G. Neural Hash Grid Rendering Results

Neural hash grid encoding (Instant-NGP style) uses multi-resolution hash tables to encode 3D positions into feature vectors, which a small MLP decodes into density. This learned representation can capture complex field structures after training on the particle data.



(a) Depth 9 **(b) Depth 10**

Fig. 16. PlenOctree at maximum evaluated depths showing finest detail.

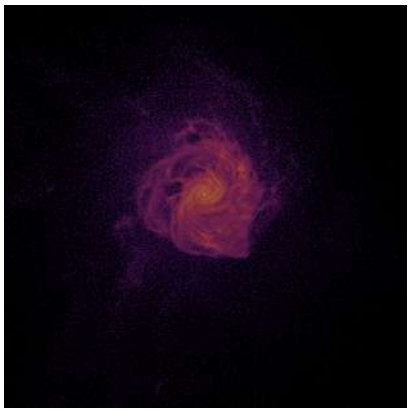
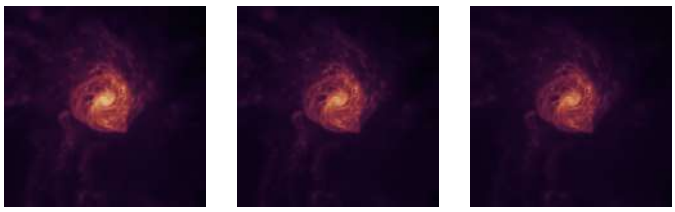


Fig. 17. Input particle distribution used for neural network training (Scatter plot).



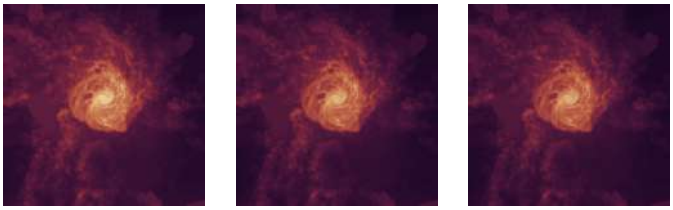
(a) 256^3 **(b)** 512^3 **(b)** 800^3

Fig. 18. Neural hash grid rendered at different sampling resolutions

The neural approach offers excellent smoothness and can learn to represent complex structures. However, it requires training time and careful hyperparameter tuning. Once trained, rendering is fast and produces high-quality smooth fields.

H. Nearest Neighbor Baseline Results

Simple nearest neighbor lookup assigns each ray sample the density of the closest particle. This serves as a performance baseline—extremely fast but produces blocky, discontinuous artifacts.



(a) 256^3 **(b)** 512^3 **(c)** 800^3

Fig. 19. Nearest neighbor baseline showing characteristic Voronoi cell artifacts.

I. Patch Wise Comparison for above methods

Fig. 20. Patch extraction location and aligned comparison.

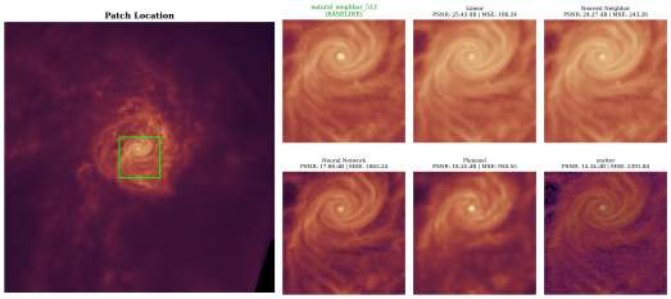


Fig. 21. Detailed center patch comparison showing local reconstruction quality.

V. Issues

- Neural Network produces artifacts in areas that did not have any data points (far empty spaces)
- Nearest Neighbors also produces artifacts in the large empty regions and can give high scalar value to that area
- The resolution of sampling can create aliasing or loss of high frequency features (see lower resolution results above) whereas very high resampling lead to wasted samples in areas which did not have any points or were empty
- The boundaries of Linear Interpolation and Natural Neighbor can create edges (see the bottom right parts of renders above)
- Memory requirements increase significantly as resampling increased the points by orders of magnitude in some cases

VI. Challenges

Key challenges include:

- Managing large simulation datasets within GPU memory.
- Balancing adaptive sampling quality with interactivity.
- Avoiding bias and artifacts from non-uniform sampling
- Choice of parameters affect quality and performance.

VII. Possible Extensions

The renderer can be extended with the following:

- Multi-dimensional transfer functions to jointly visualize features (like density and temperature)
- Support snapshots for visualizing cosmic time evolution.

References

[1] C. Kulla and M. Fajardo, "Importance Sampling Techniques for Path Tracing in Participating Media," *Eurographics Symposium on Rendering (EGSR)*, 2012. [Online]. Available: https://blogs.autodesk.com/media-and-entertainment/wp-content/uploads/sites/162/egsr2012_volume.pdf

[2] C. Ikits, J. Kniss, A. Lefohn, and C. Hansen, "Volume Rendering Techniques," in *GPU Gems*, Ch. 39, NVIDIA, 2004. [Online]. Available: https://developer.nvidia.com/gpugems/gpugems/part-vi-beyond-triangles/chapter-39-volume-rendering-techniques

[3] D. Nelson, F. Pillepich, V. Springel, L. Hernquist, K. Naiman, P. Torrey, A. Marinacci, R. Pakmor, A. Weinberger, and M. Vogelsberger, "The IllustrisTNG Simulations: Public Data Release," *arXiv preprint arXiv:1812.05609*, 2018. [Online]. Available: https://arxiv.org/pdf/1812.05609

[4] D. Nelson, M. Vogelsberger, S. Genel, D. Sijacki, D. Kereš, V. Springel, and L. Hernquist, "Moving Mesh Cosmology: Tracing Cosmological Gas Accretion," *Monthly Notices of the Royal Astronomical Society*, vol. 429, no. 4, pp. 3353–3370, Mar. 2013. doi: 10.1093/mnras/sts595. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2013MNRAS.429.3353N

[5] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "PlenOctrees for Real-time Rendering of Neural Radiance Fields," *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[6] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ACM Transactions on Graphics*, vol. 41, no. 4, 2022.

[7] A. Yu, S. Yeung, M. Tancik, and A. Kanazawa, "Plenoxels: Radiance Fields without Neural Networks," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[8] R. Sibson, "A Brief Description of Natural Neighbor Interpolation," in *Interpreting Multivariate Data*, V. Barnett, Ed., Wiley, 1981, pp. 21–36.