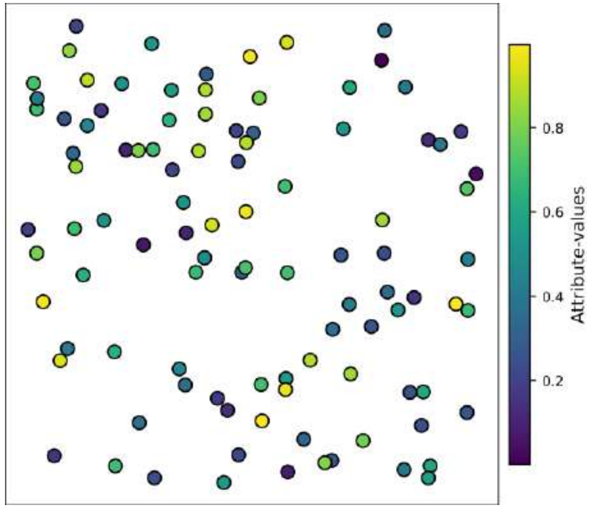
GPU-Accelerated Scalar Field Reconstruction and Volume Rendering

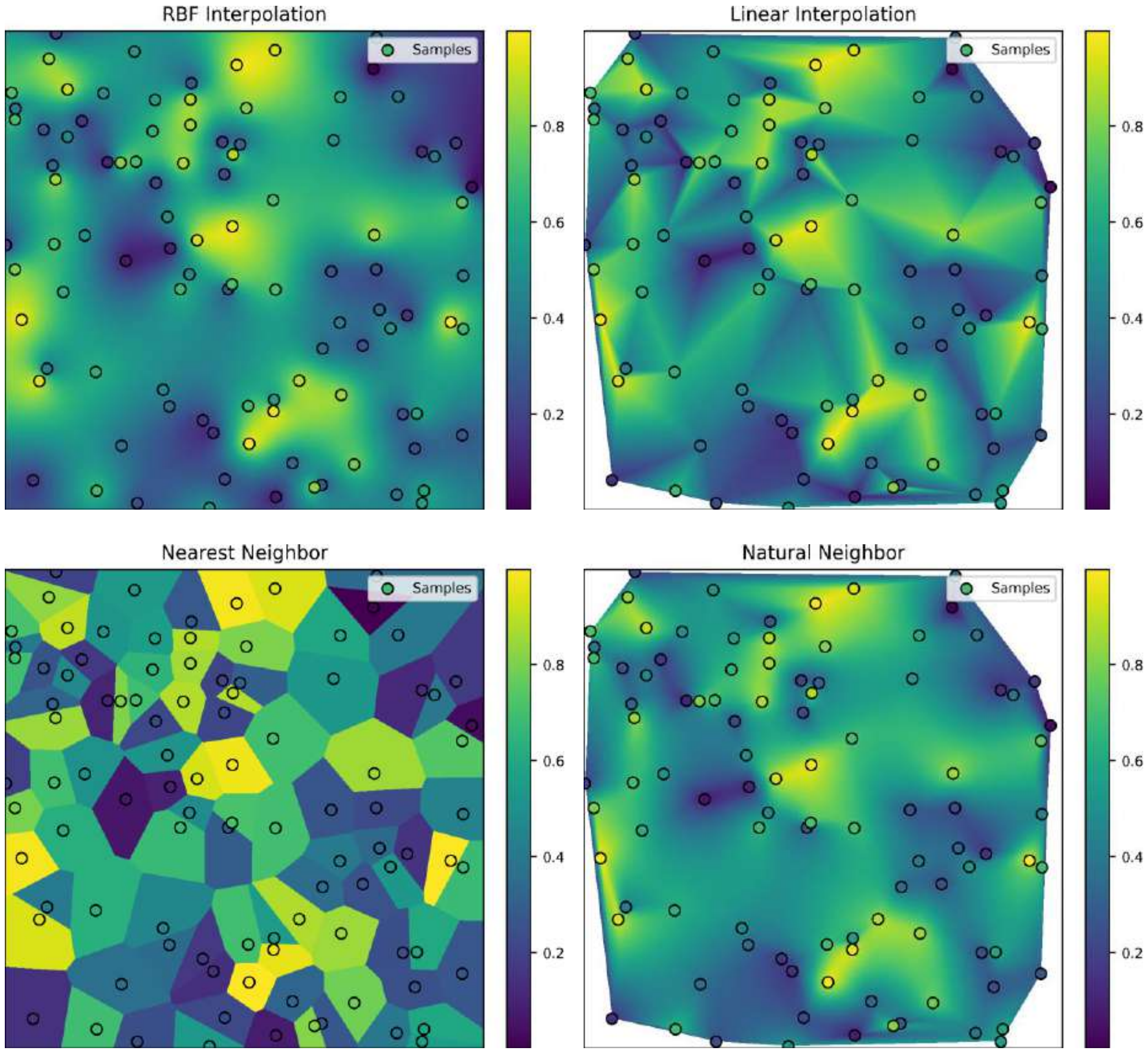
Motivation
Assumption
Interpolations
Results and Metrics

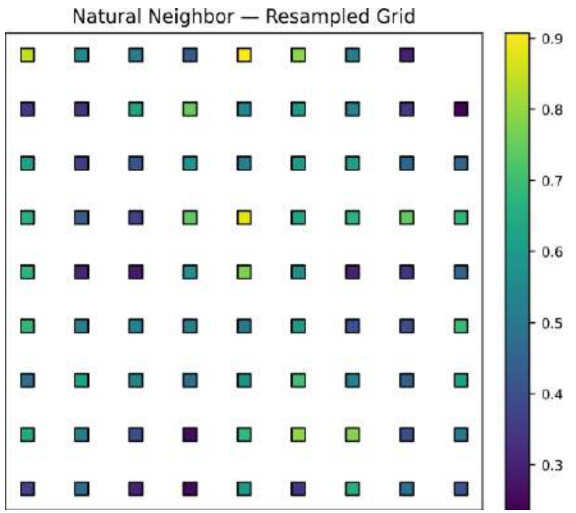
Presented by:
Utkarsh Sharma
Akash Maji

Motivation: Data Format and Challenges

Given an unstructured data from simulations, how can we do Volume Rendering ?







Unstructured Data

$N \times 3$ co-ordinates
$N \times K$ corresponding scalars
(for data with K attributes)

Resampled Grid

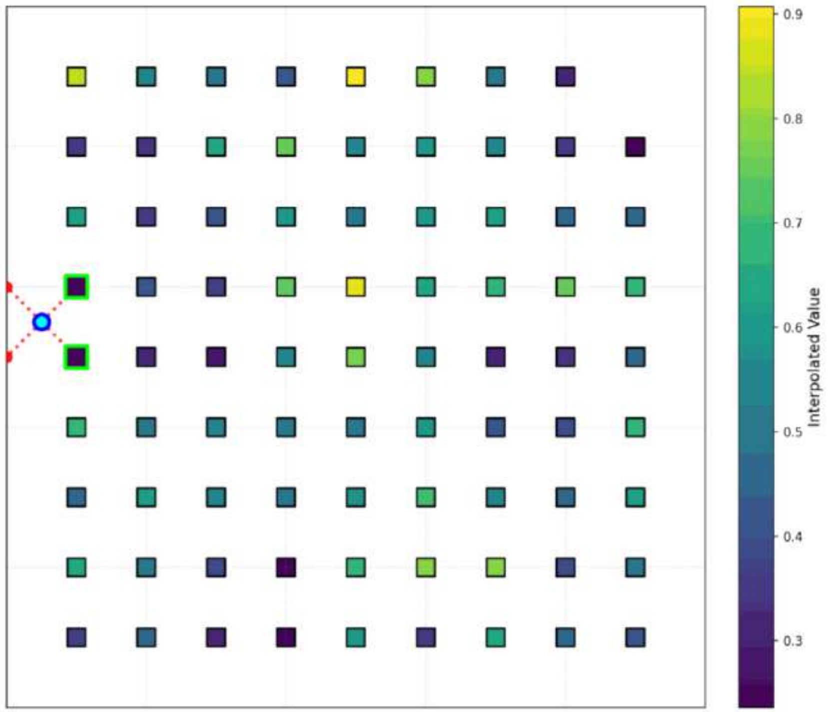
$H \times W \times D \times K$ scalar values and spacing is specified

Motivation: Data Format and Challenges

The Ray Marching using the Unstructured data can use empty space skipping

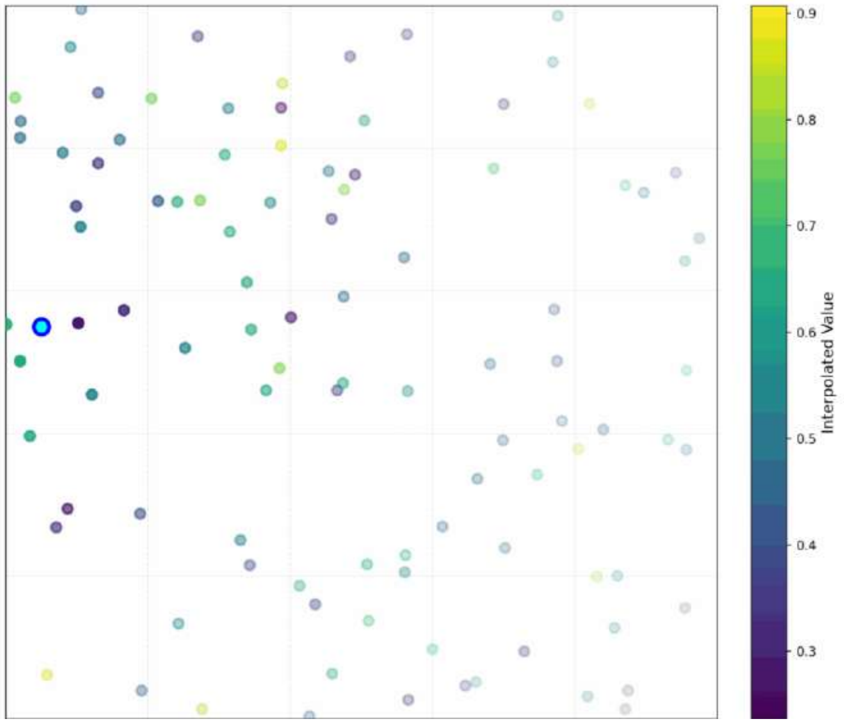
Resample to a grid:

- Well defined algorithm
- Trilinear interpolation is fast
- Slow Pre-processing and high memory usage
- Faster methods have less accuracy



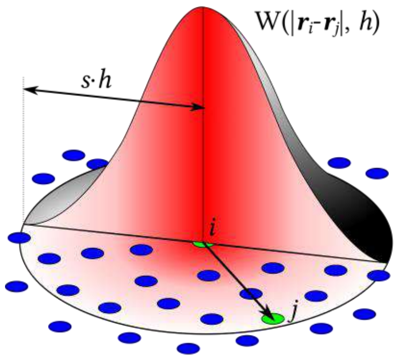
Directly use unstructured data

- Can skip Empty Space
- Can do more accurate interpolation
- Slow interpolation on the fly



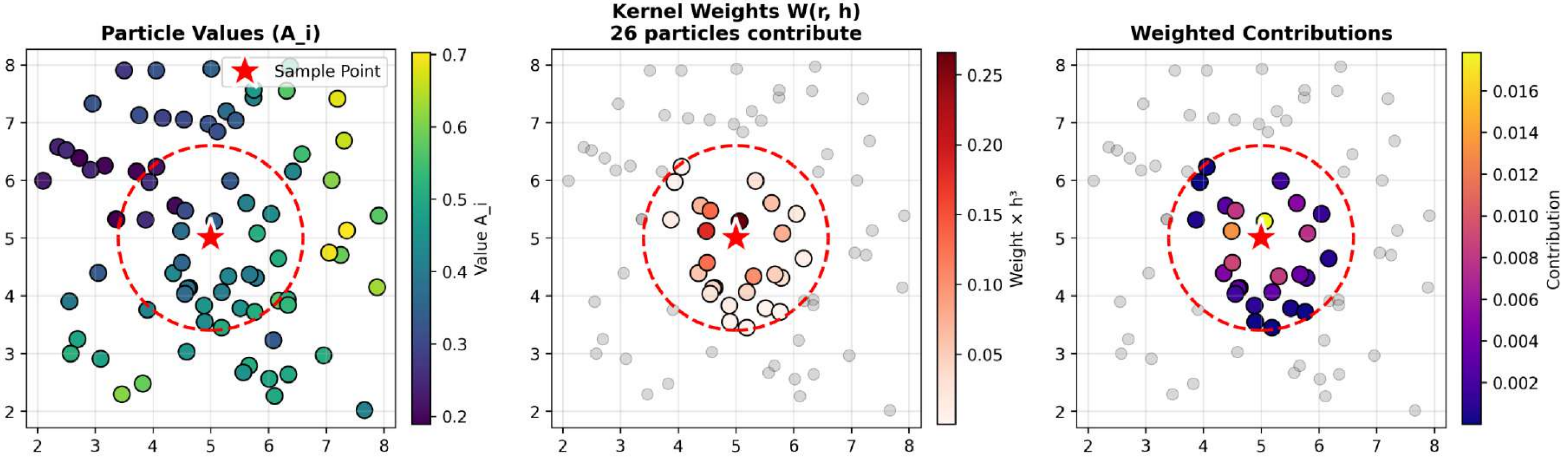
Assumption: SPH Data

Smoothed-particle hydrodynamics (**SPH**) is a computational method used for simulating the mechanics of continuum media (solid mechanics/ fluid flows) which treats a fluid as particles with physical attributes that contributes to a continuous field via a **kernel function** (a smooth weighting function):



$$\rho(x) = \sum_i m_i W(\|x_i - x\|, h)$$

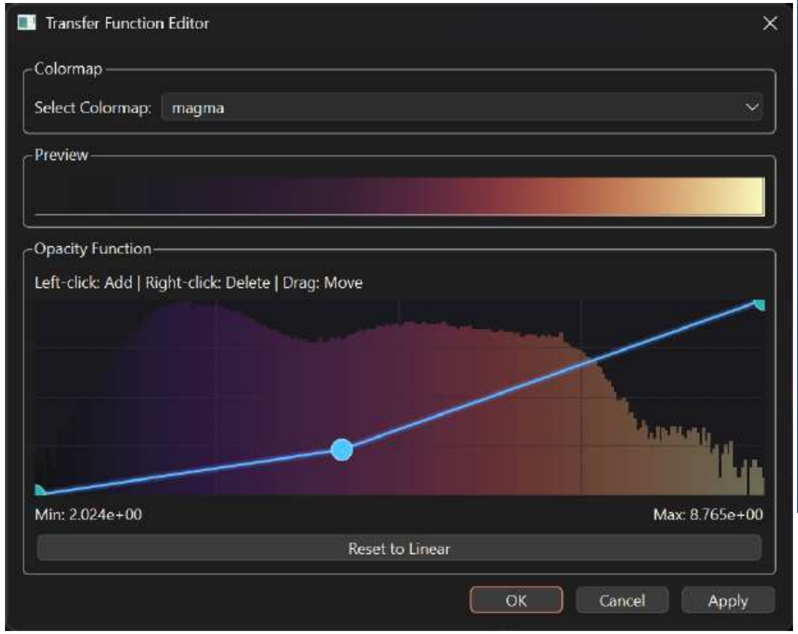
where m_i is particle mass, W is smoothing kernel and h is it's radius.



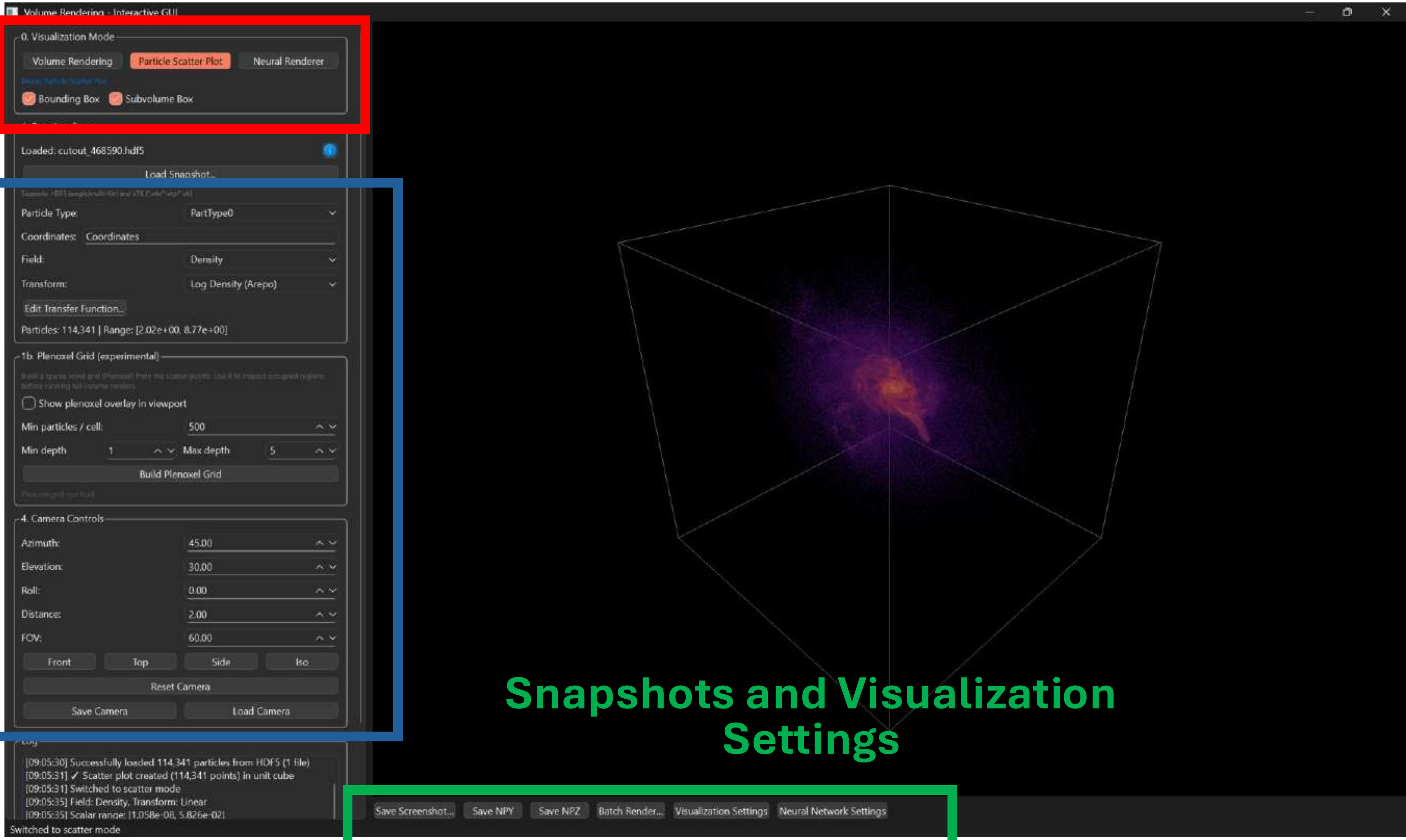
GUI

Visualization Modes

Data and Construction Controls



Transfer Function

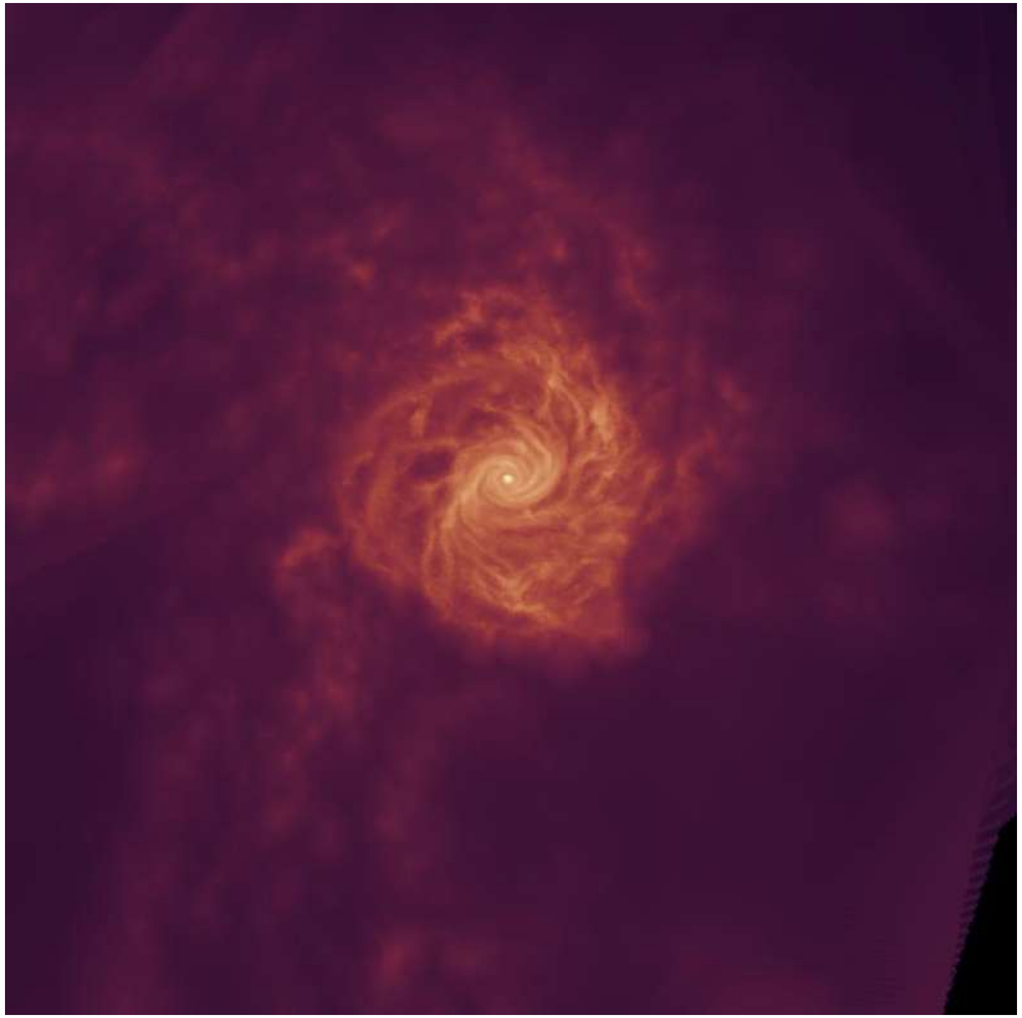


Snapshots and Visualization Settings

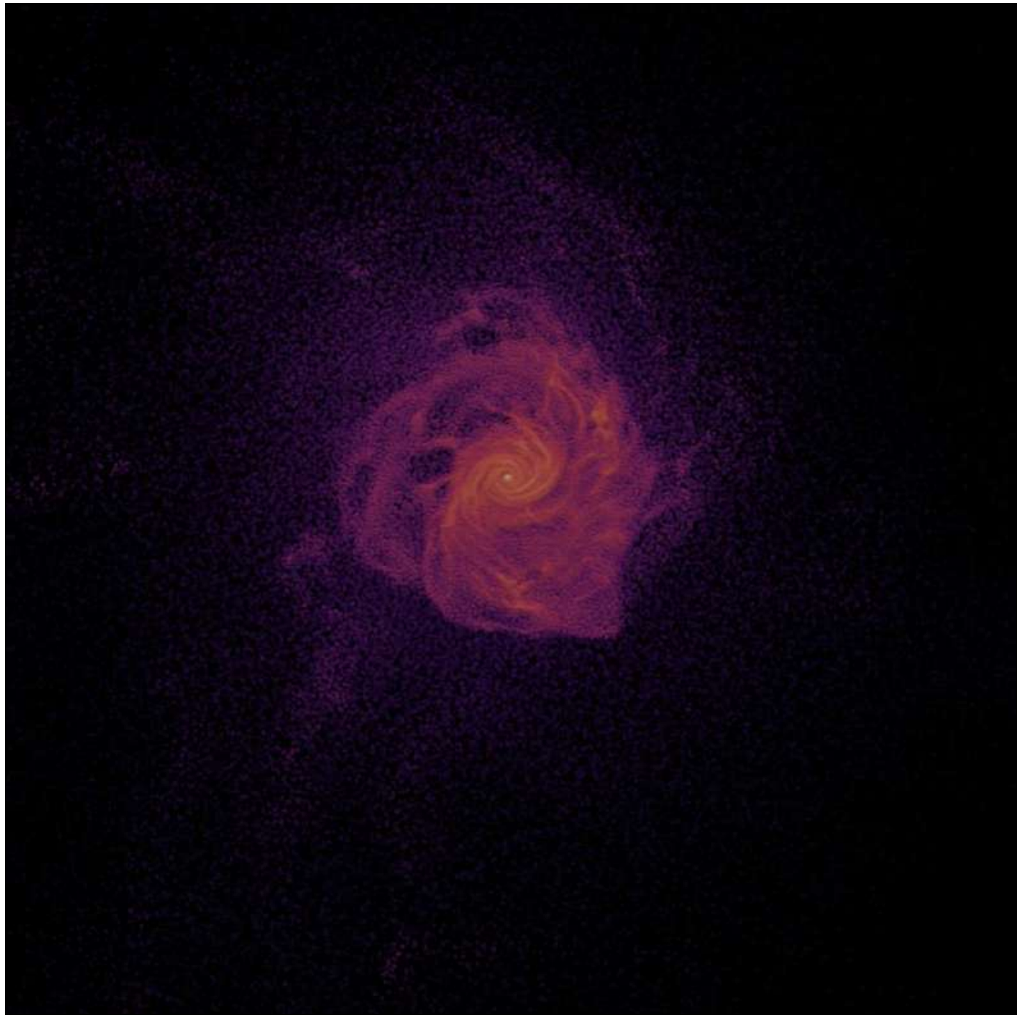
Main Viewer Window

Subhalo 468590 Visualization

The rendered images were saved at a resolution of 800×800. The data set used is: **Subhalo 468590**



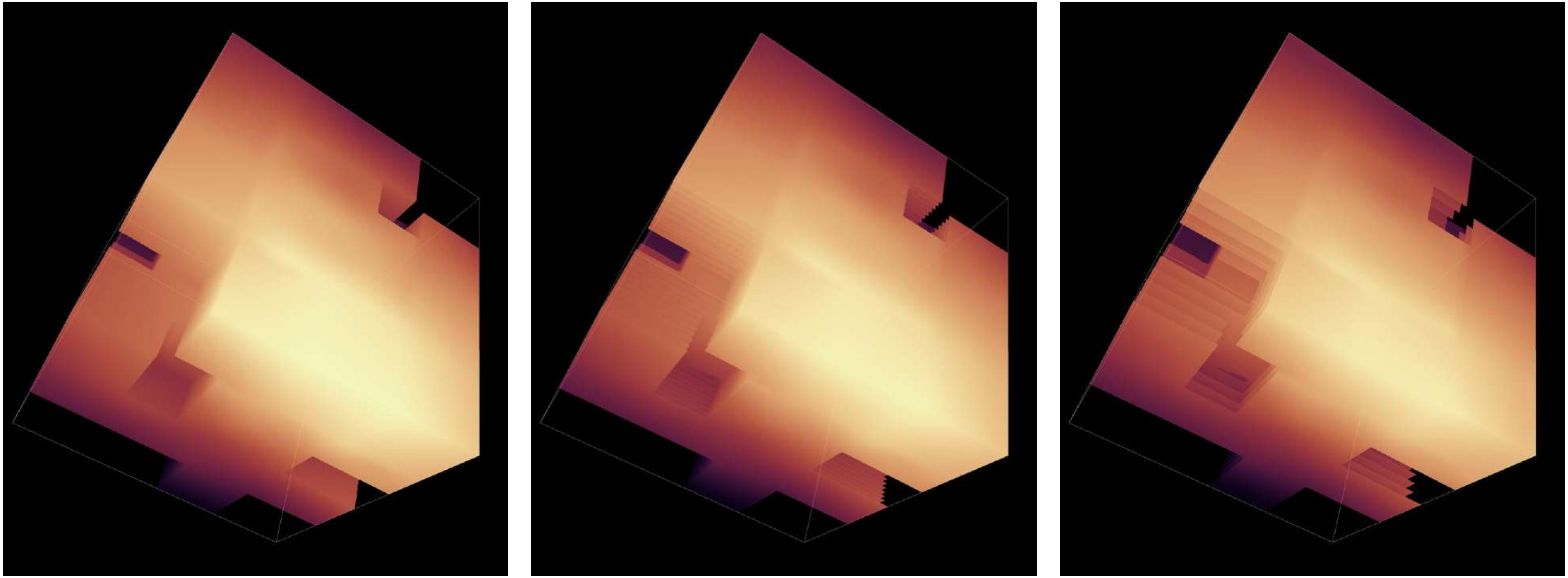
Natural Neighbor



Scatter Plot

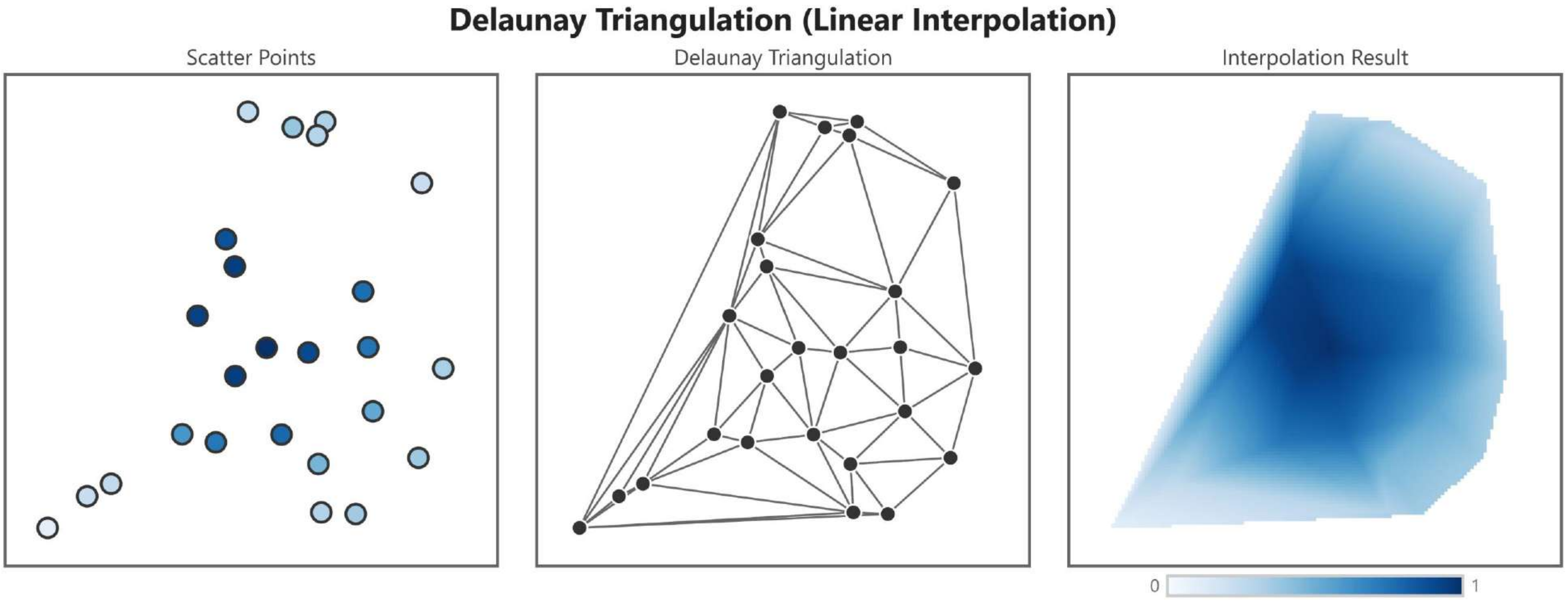
Atrifacts : Step Size

Step size can create the artifacts around the curved area or high frequency detailed area



Method : Linear Interpolation

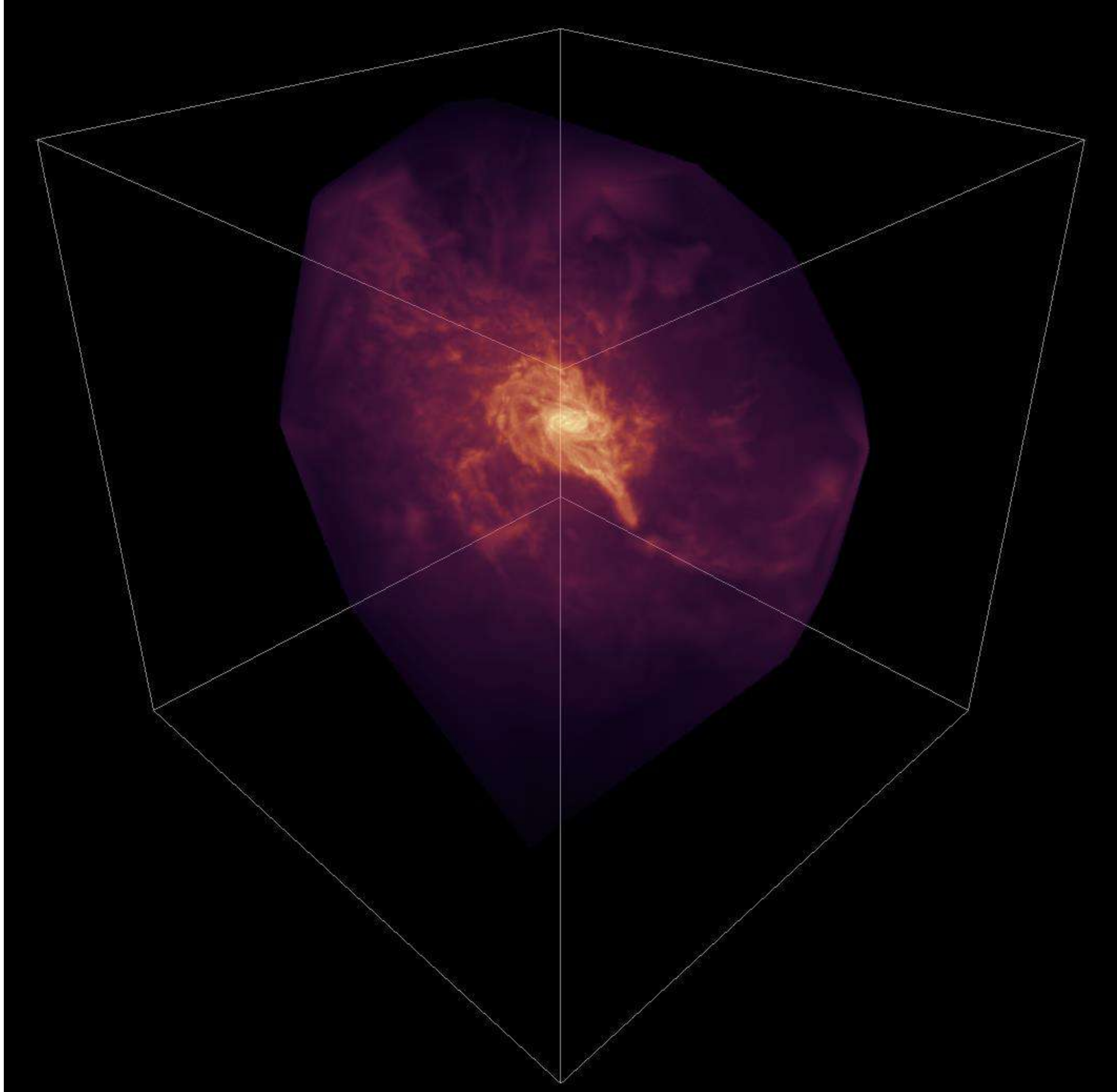
Delaunay tetrahedralization provides a natural geometric structure for particle data. Each tetrahedron enables smooth barycentric interpolation, ensuring C 0 continuity across cell boundaries.



Linear Interpolation

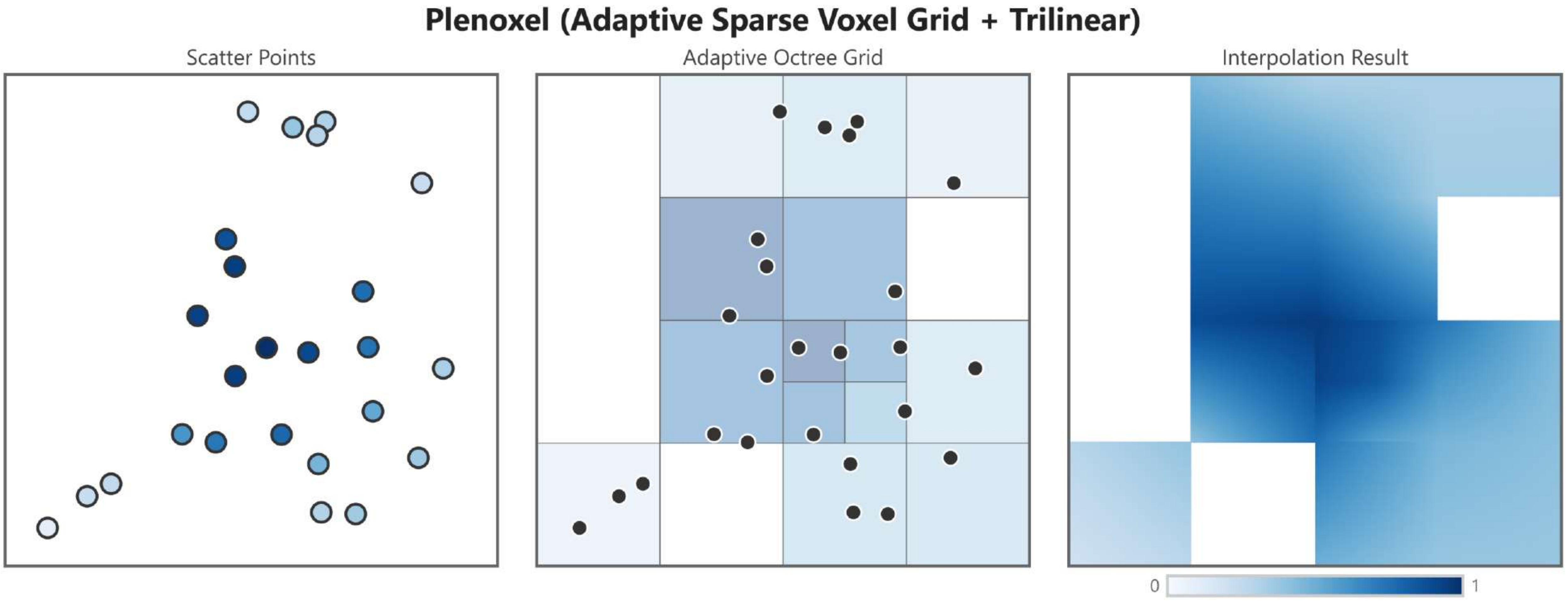
Linear Interpolation uses Delaunay Tetrahedral to linearly interpolate between the points.

The Convex Hull is visible in the image.



Method : Octree

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.



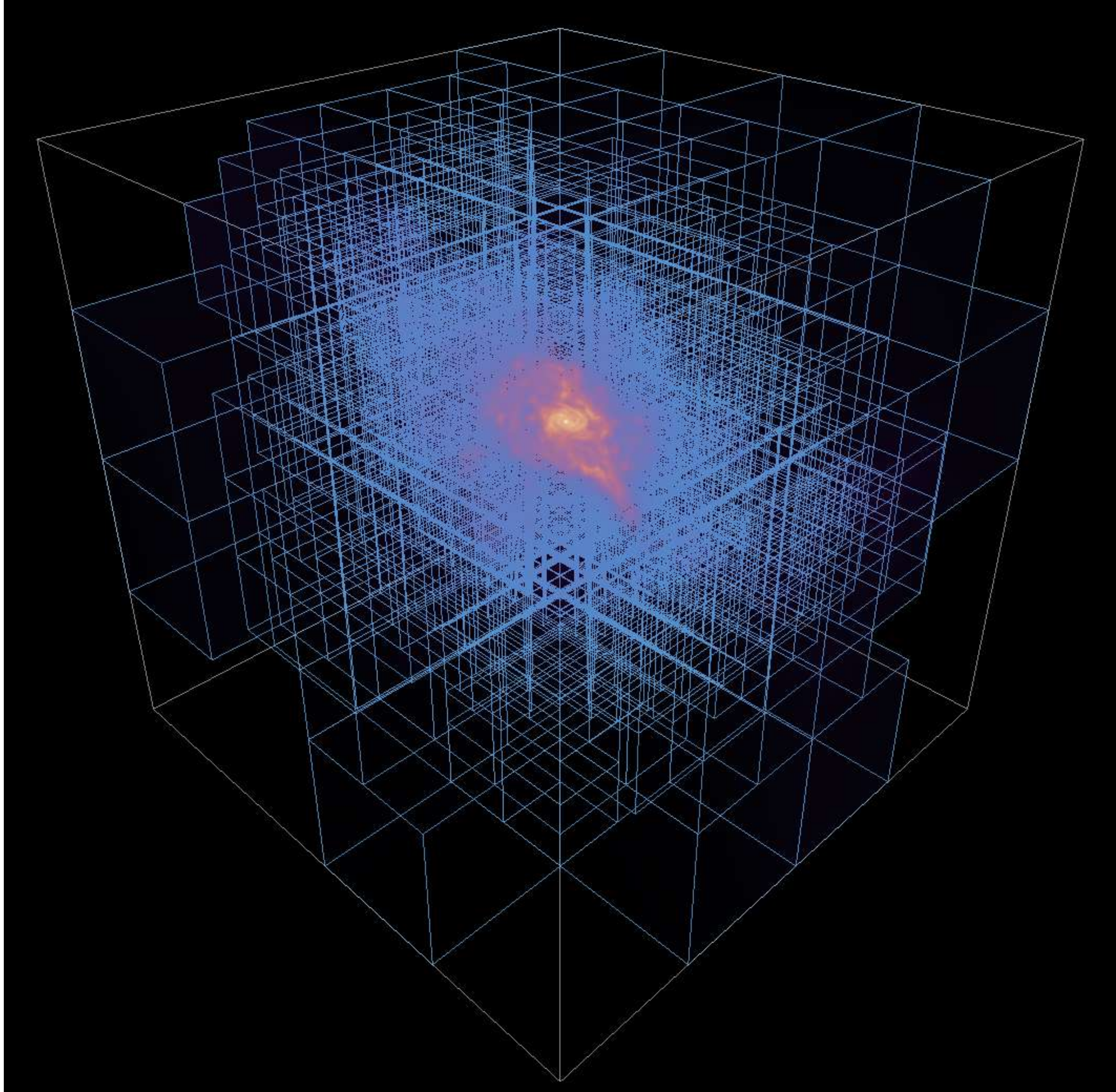
Octree

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.



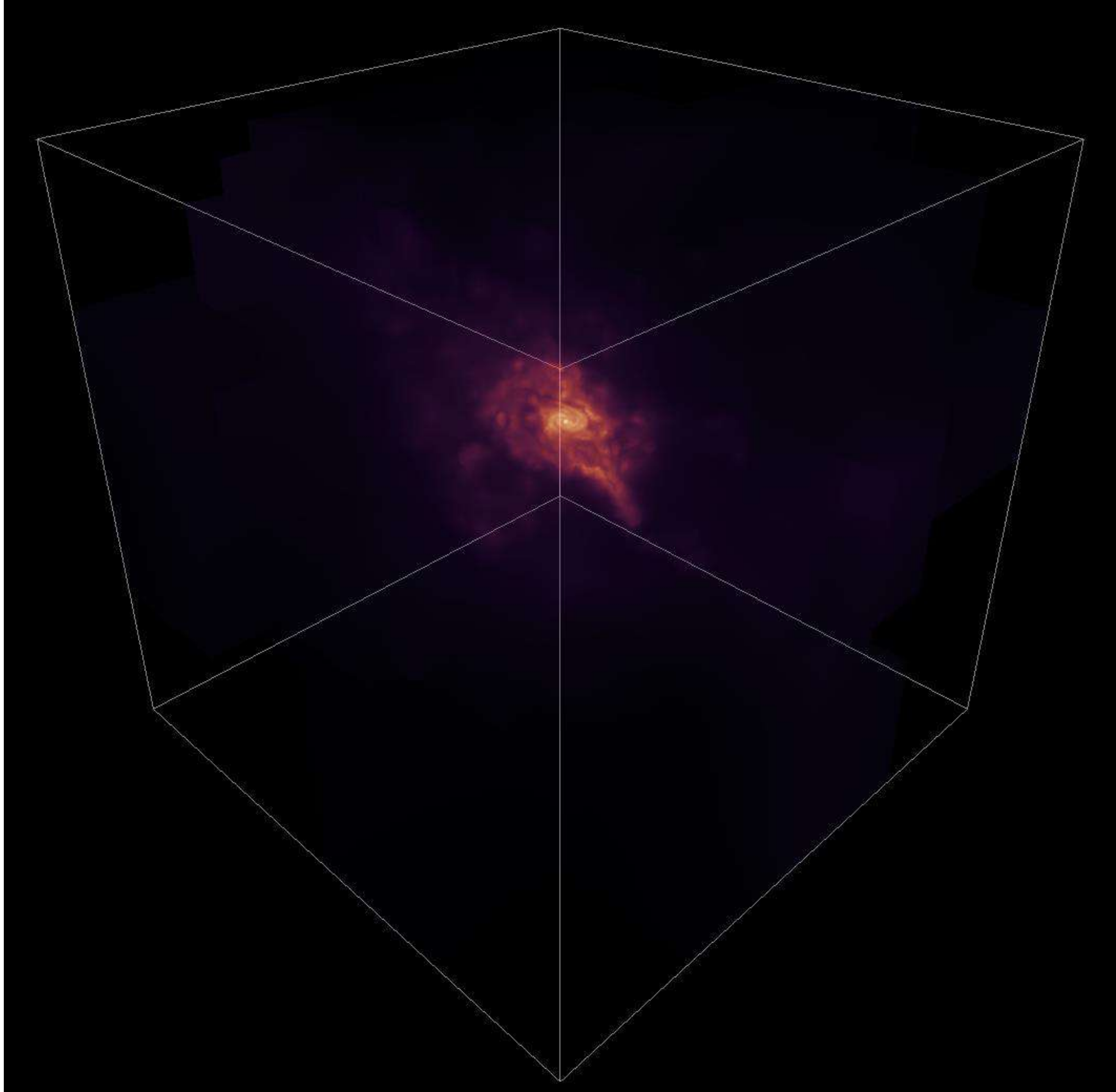
Octree

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:

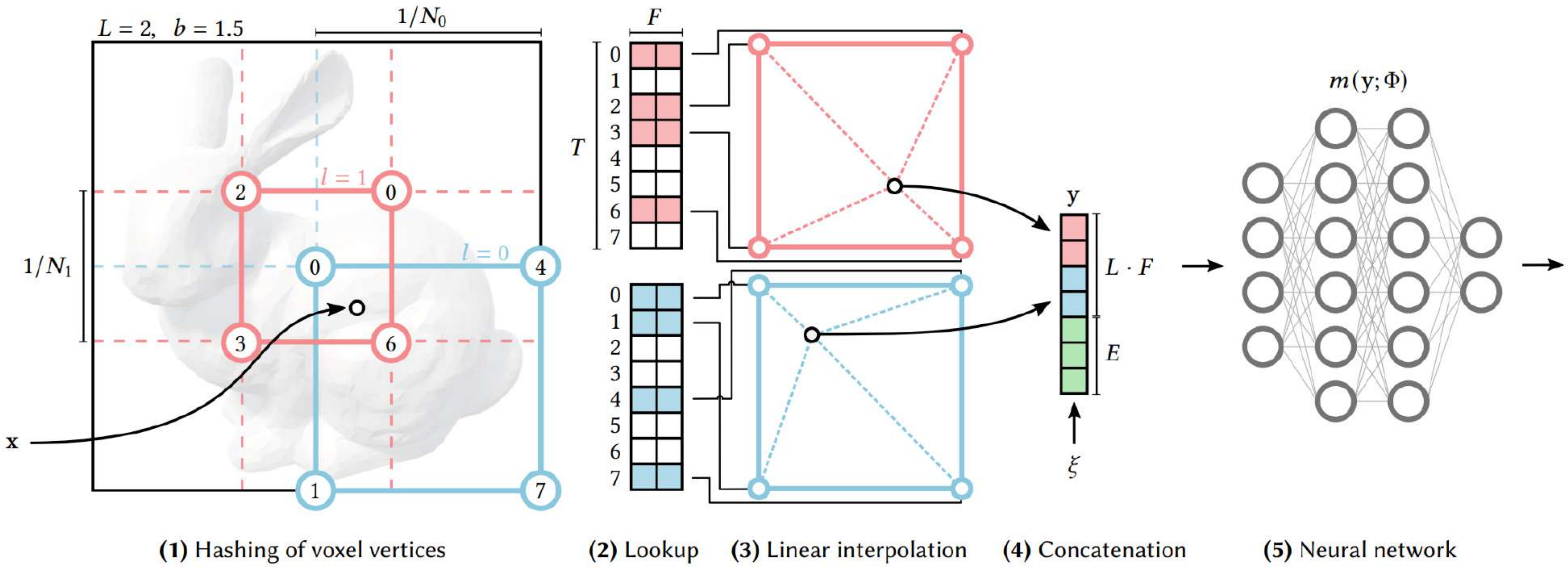
$$\rho(\mathbf{x}) = \sum_{k=0}^{7} \phi_k(u, v, w)\rho_k$$

where $(u, v, w) \in [0, 1]^3$ are normalized coordinates and ϕ_k are trilinear basis functions.



Method : Neural Representation

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

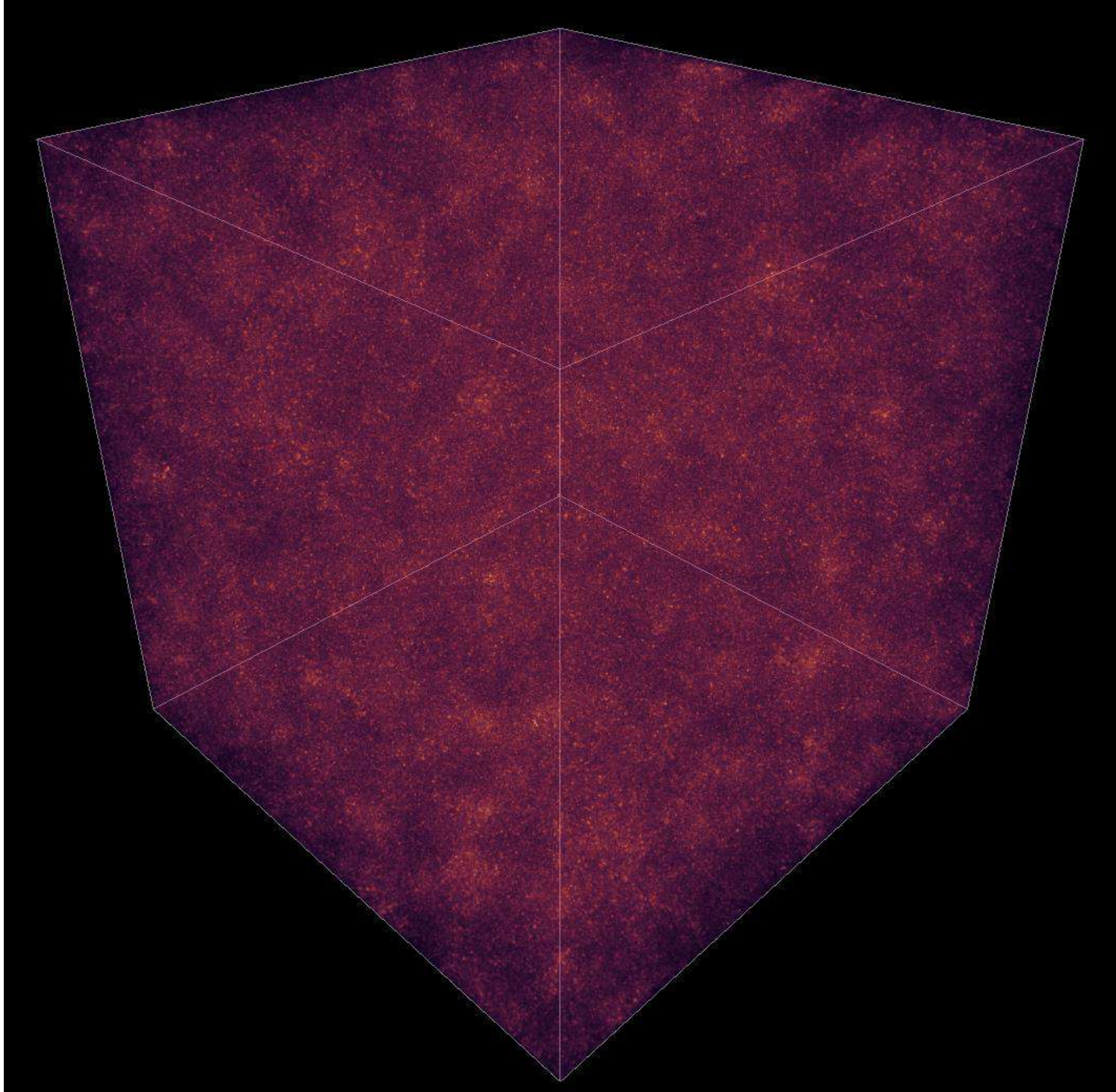


Neural Representation

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

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

Initialized Neural Network

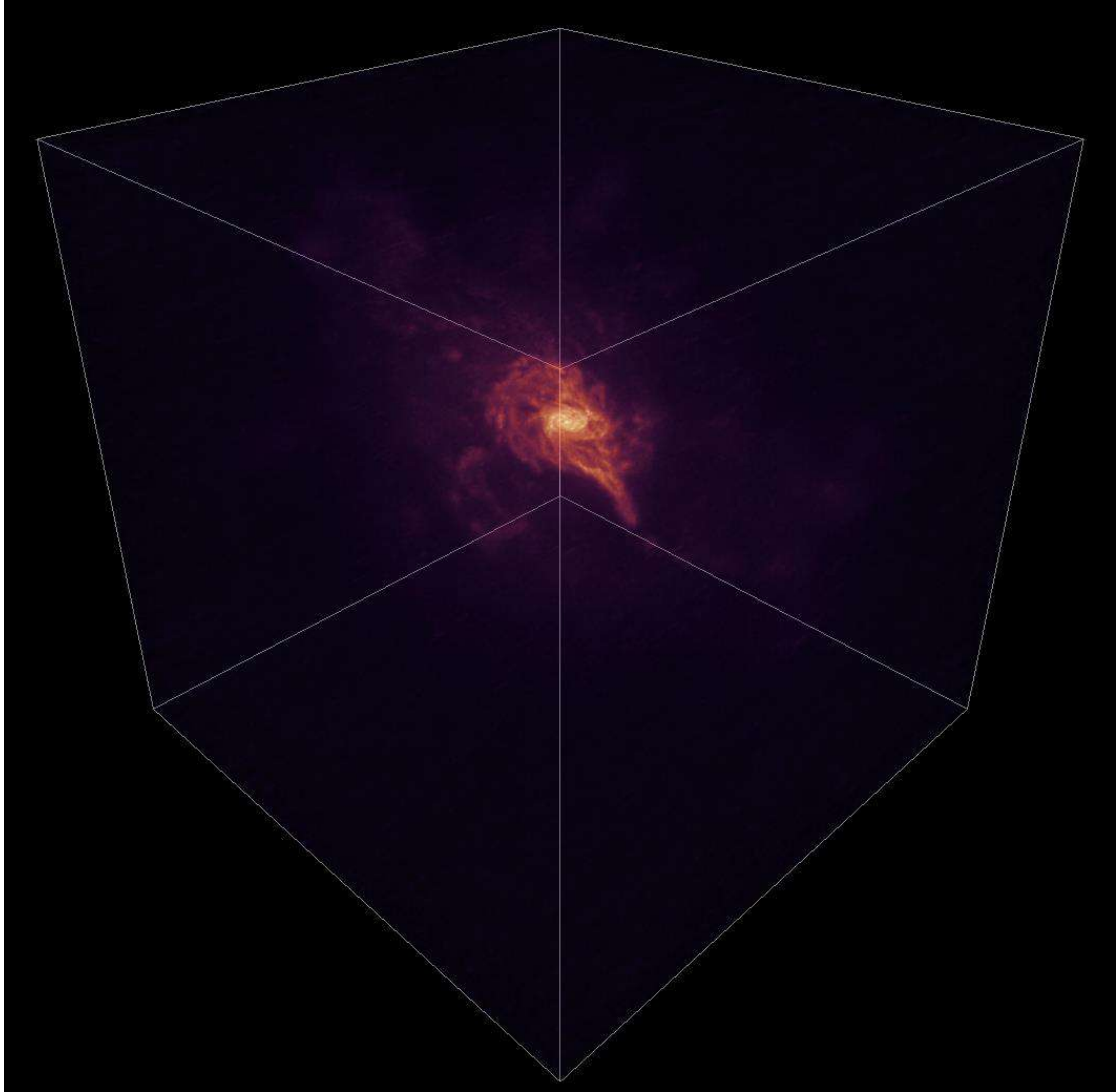


Neural Representation

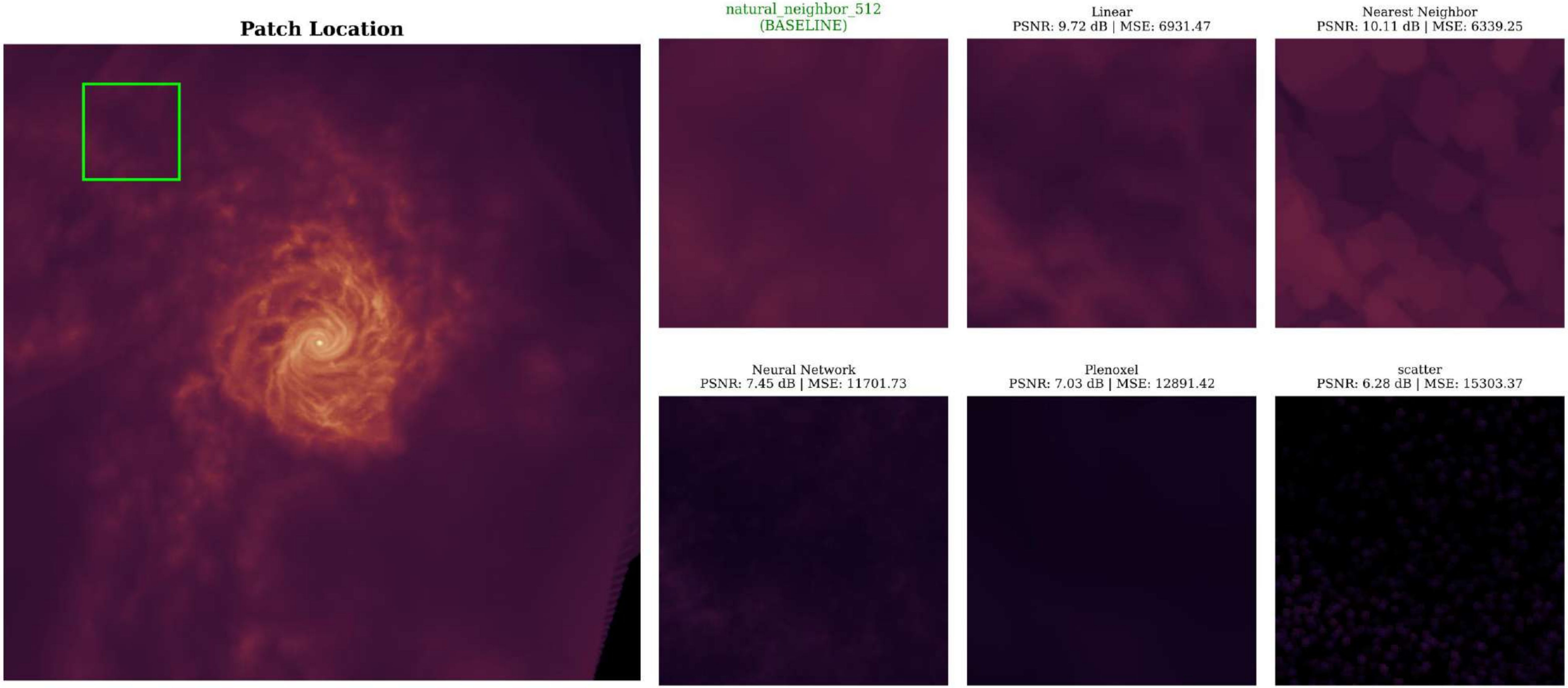
Learning-based continuous representation using multi-resolution hash encoding. Position $\boldsymbol{x}$ mapped to **L** resolution levels, hashed into compact feature tables:

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

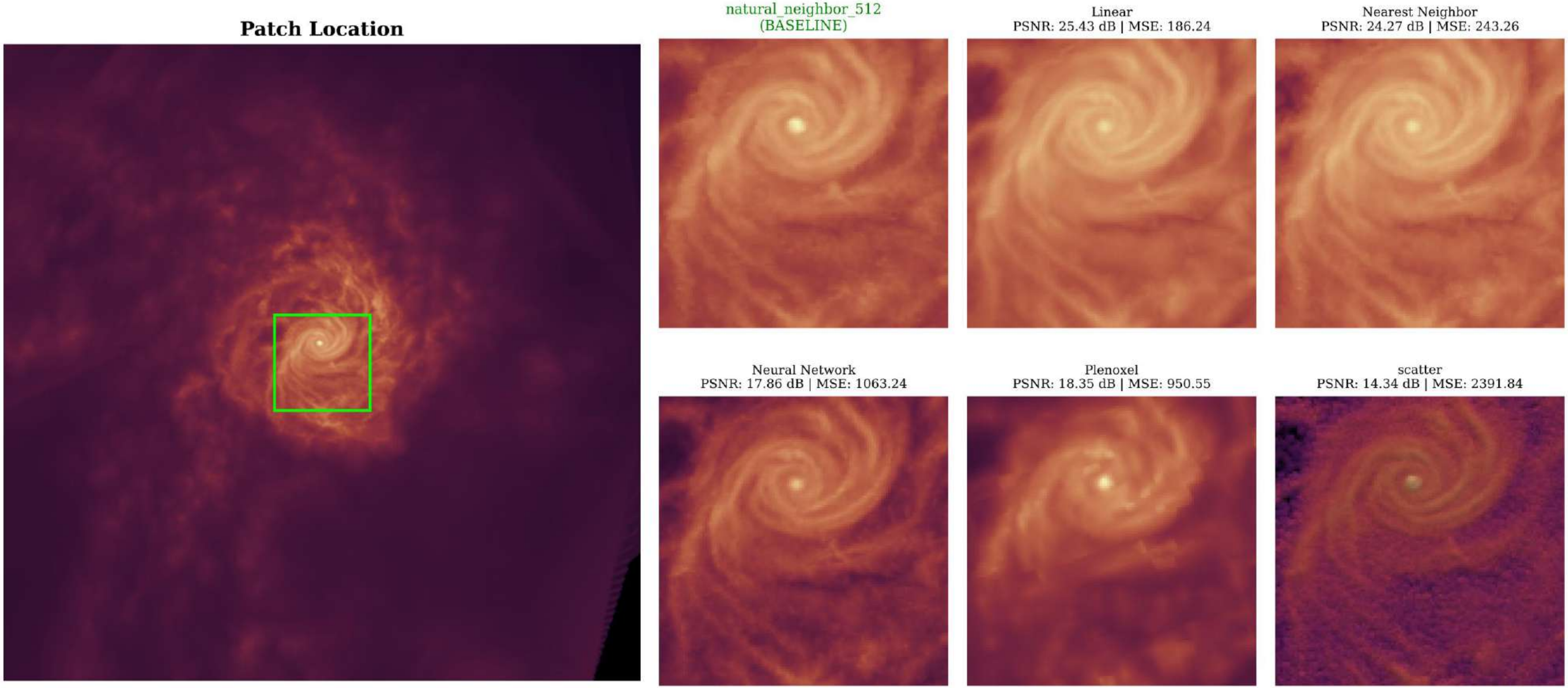
Trained Neural Network



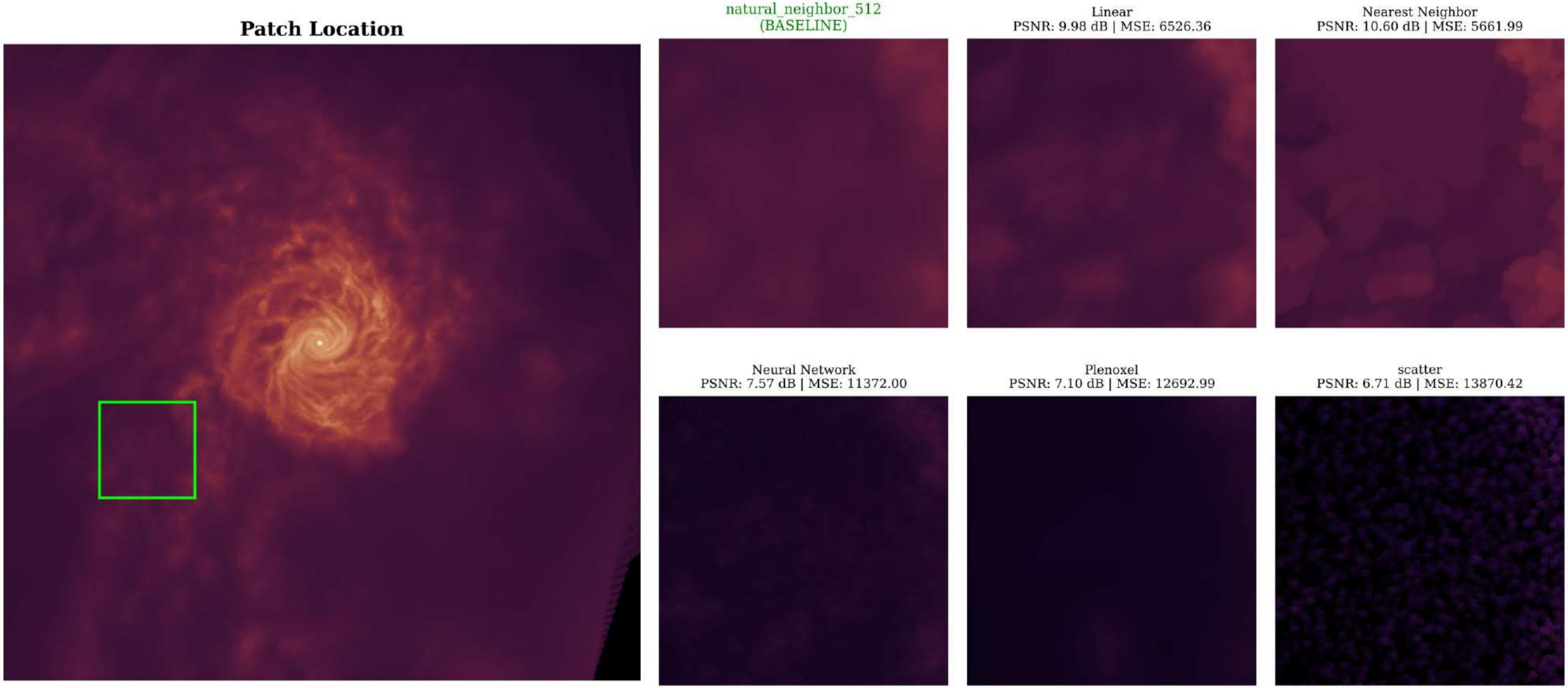
Results: Patch Comparison



Results: Patch Comparison

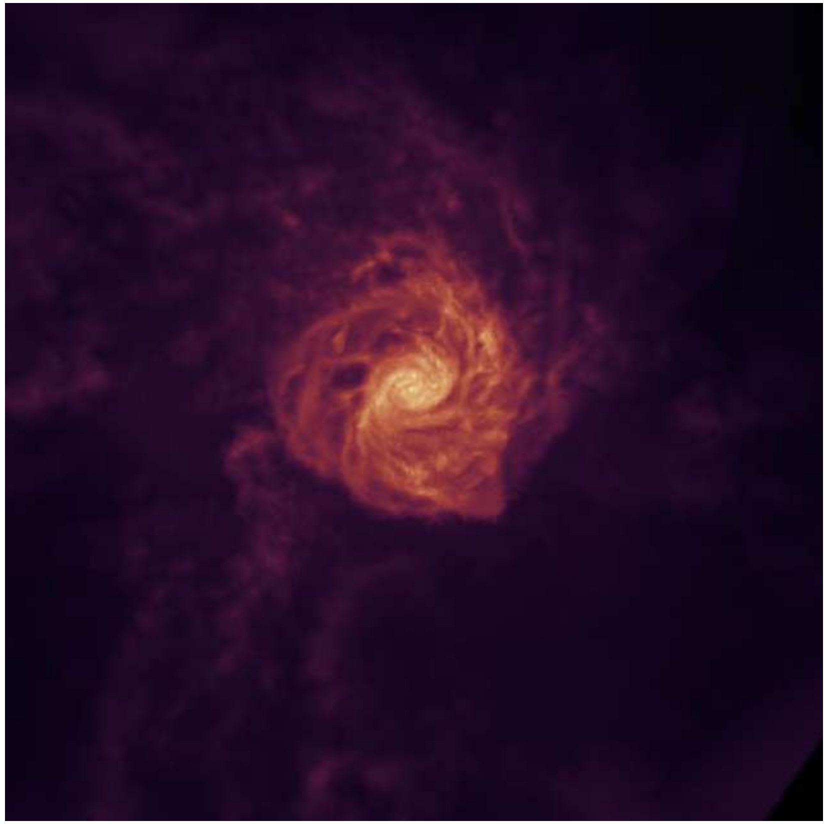


Results: Patch Comparison

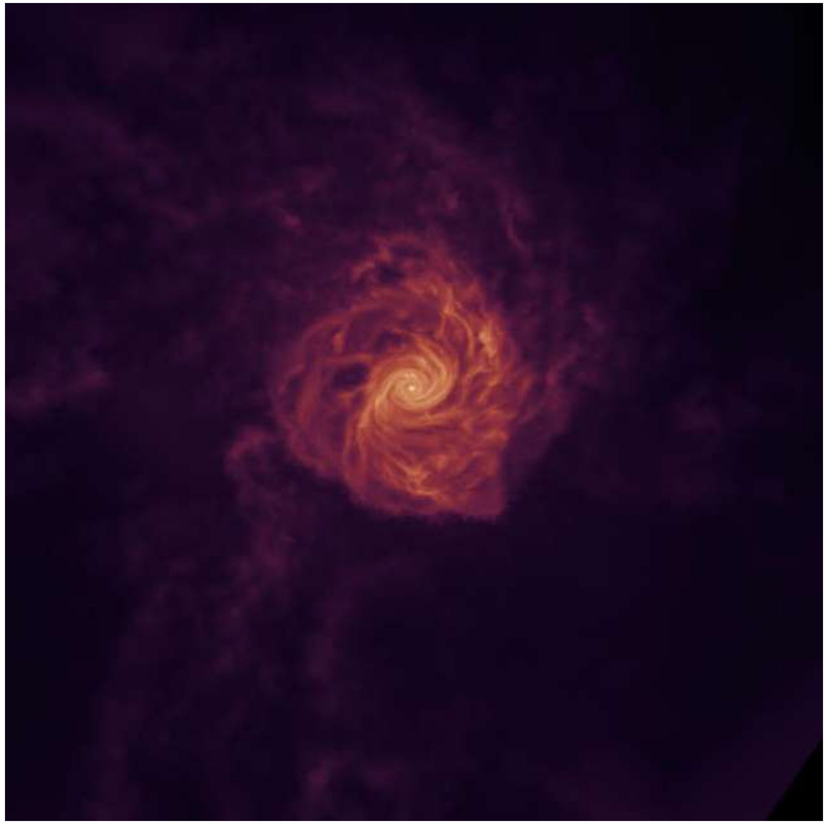


Results: Effects of Resolution

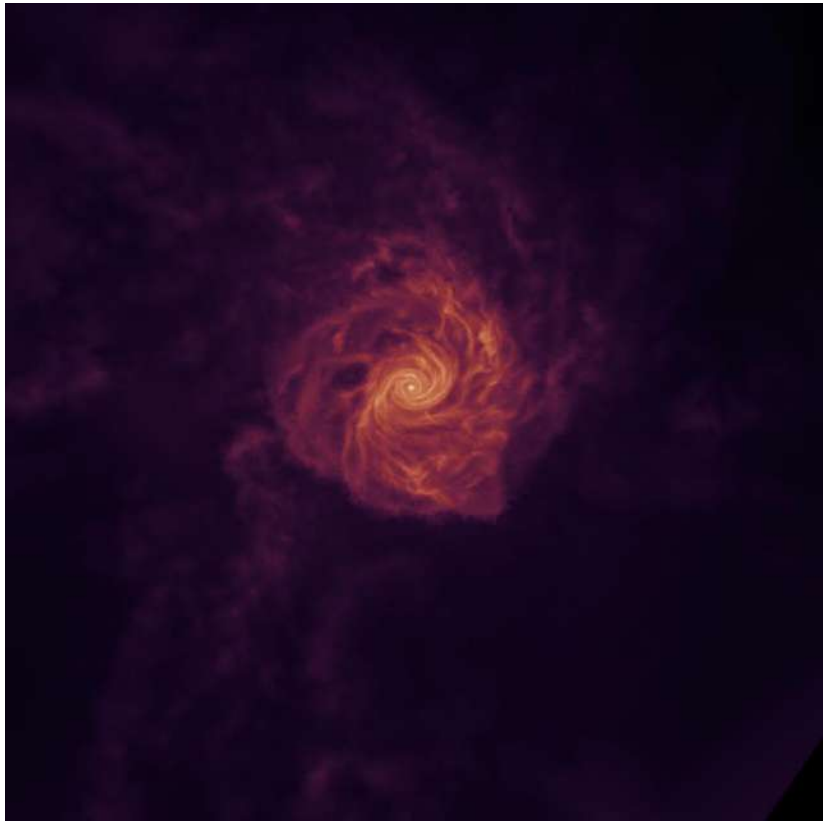
The following show the Linear Interpolation Results at different Resolutions



$\mathbf{256^3}$



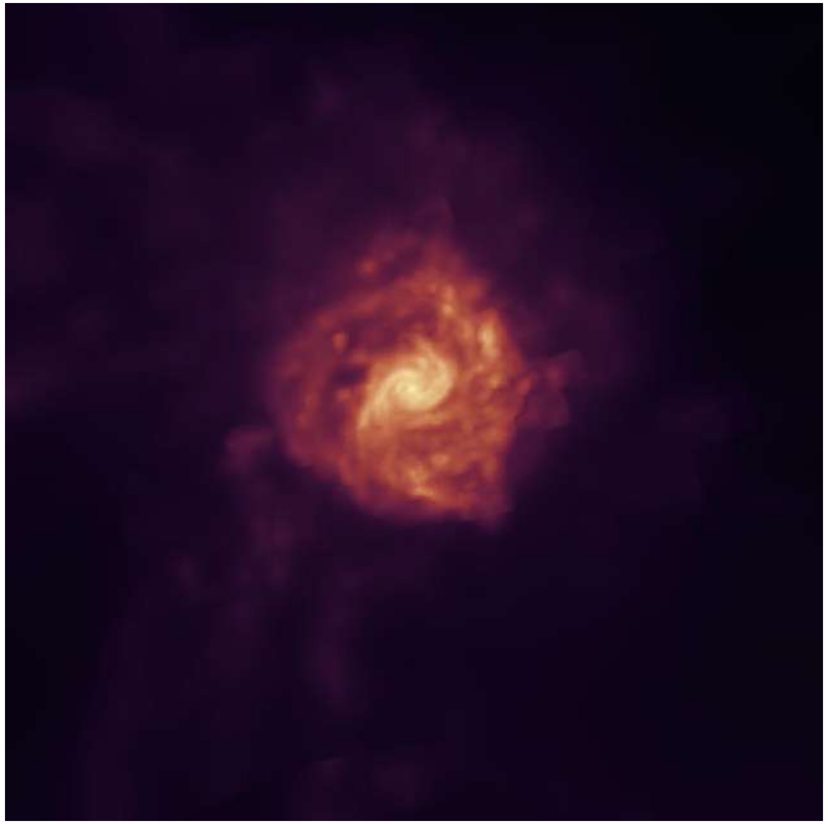
$\mathbf{512^3}$



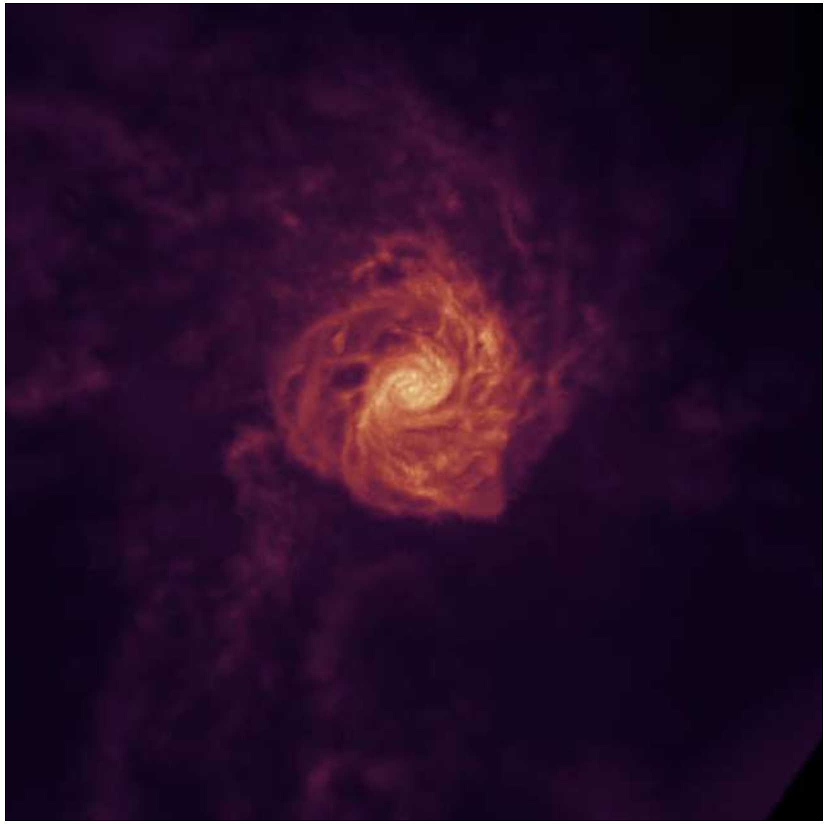
$\mathbf{1024^3}$

Results: Effects of Octree Depth

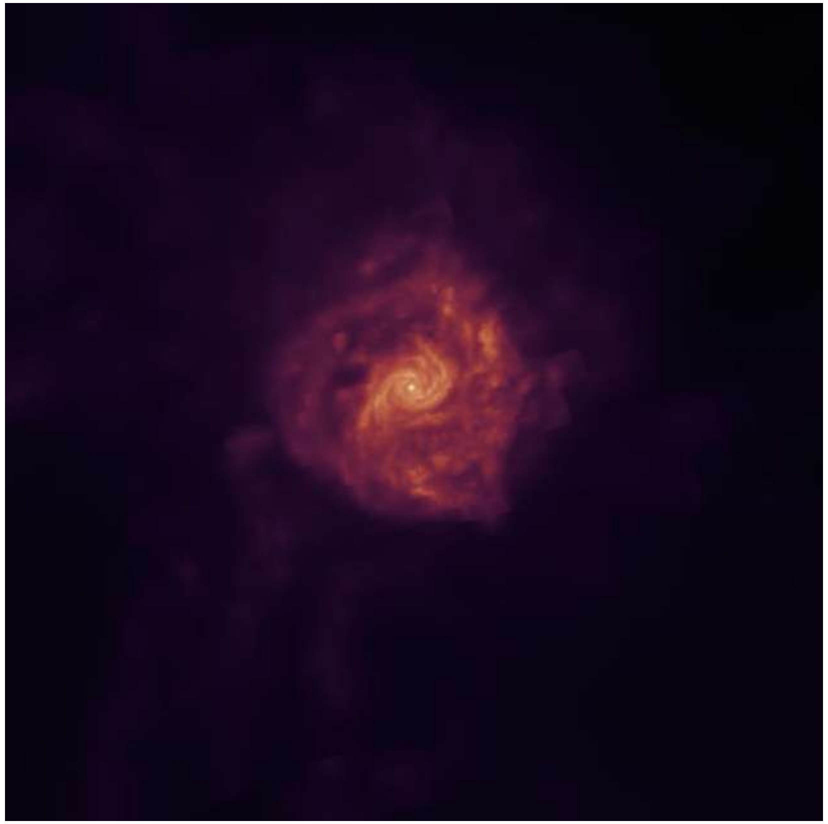
Ran the Resampling on a grid using Octree



$depth = 7$



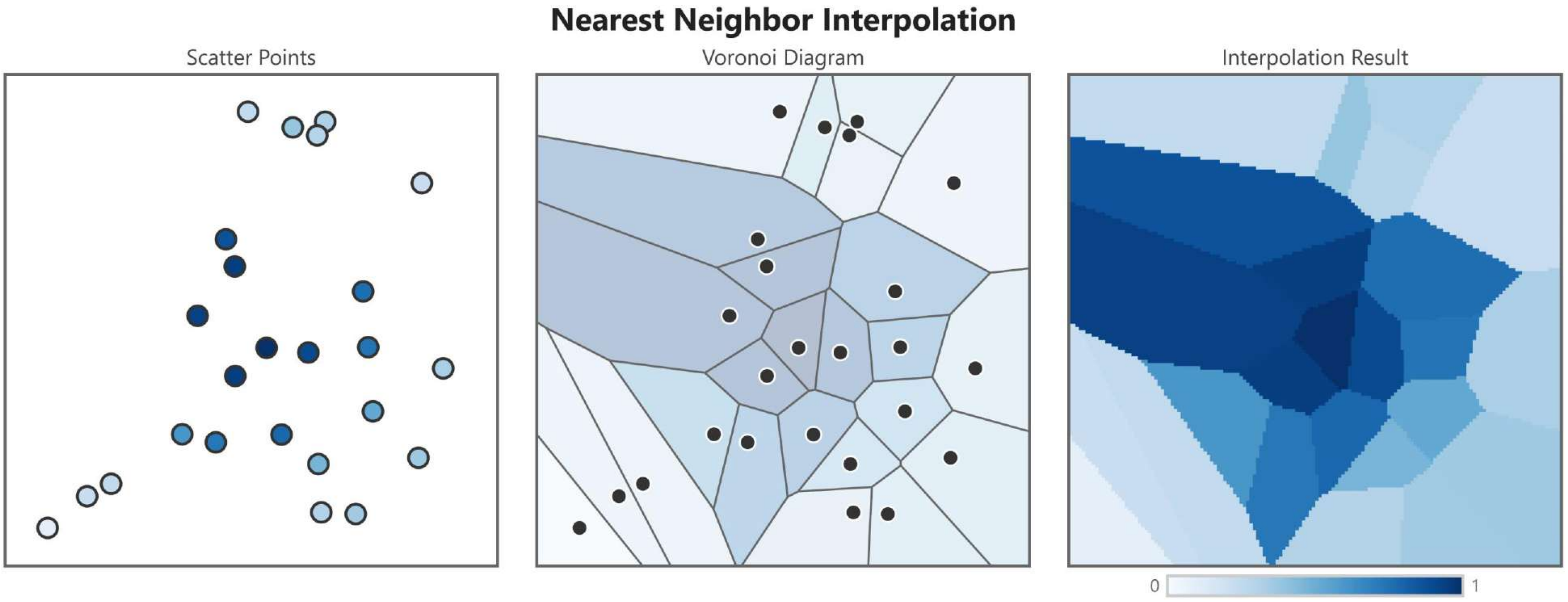
$depth = 8$



$depth = 9$

Method : K-Nearest Neighbors

Evaluates the scalar field using Inverse Distance Weighting (IDW) over K nearest particle

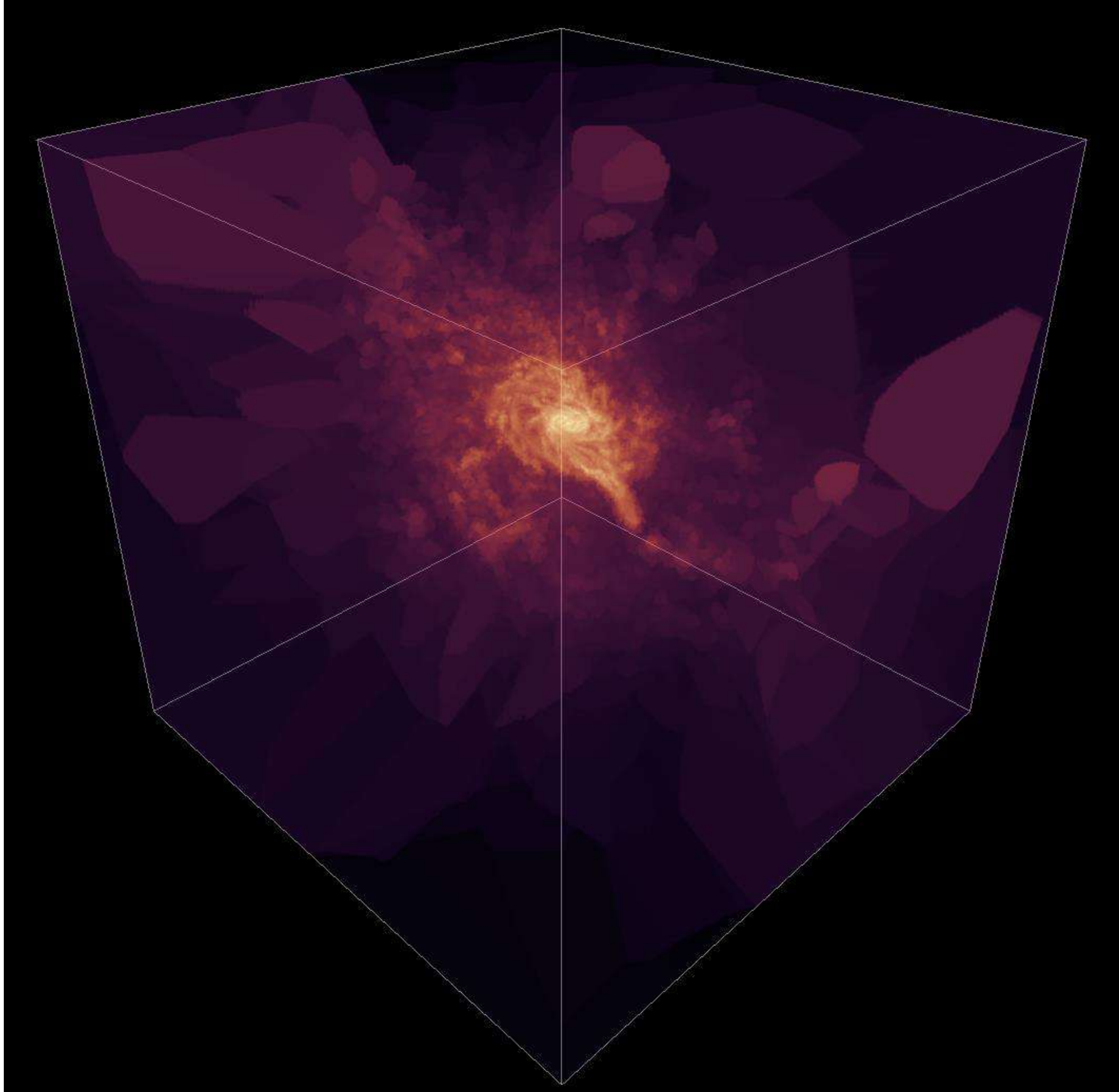


K-Nearest Neighbors

Evaluates the scalar field using Inverse Distance Weighting (IDW) over **K** nearest particle:

$$\rho(\mathbf{x}) = \frac{\sum_{i=1}^{K} w_i(\mathbf{x})\rho_i}{\sum_{i=1}^{K} w_i(\mathbf{x})},$$

$$w_i(\mathbf{x}) = \frac{1}{\|\mathbf{x} - \mathbf{p}_i\|^p}$$



K-Nearest Neighbors

Parameters:
K = no. of neighbours
m = sampling depth (m * m voxels)
GRID_SIZE = 512 ^ 3

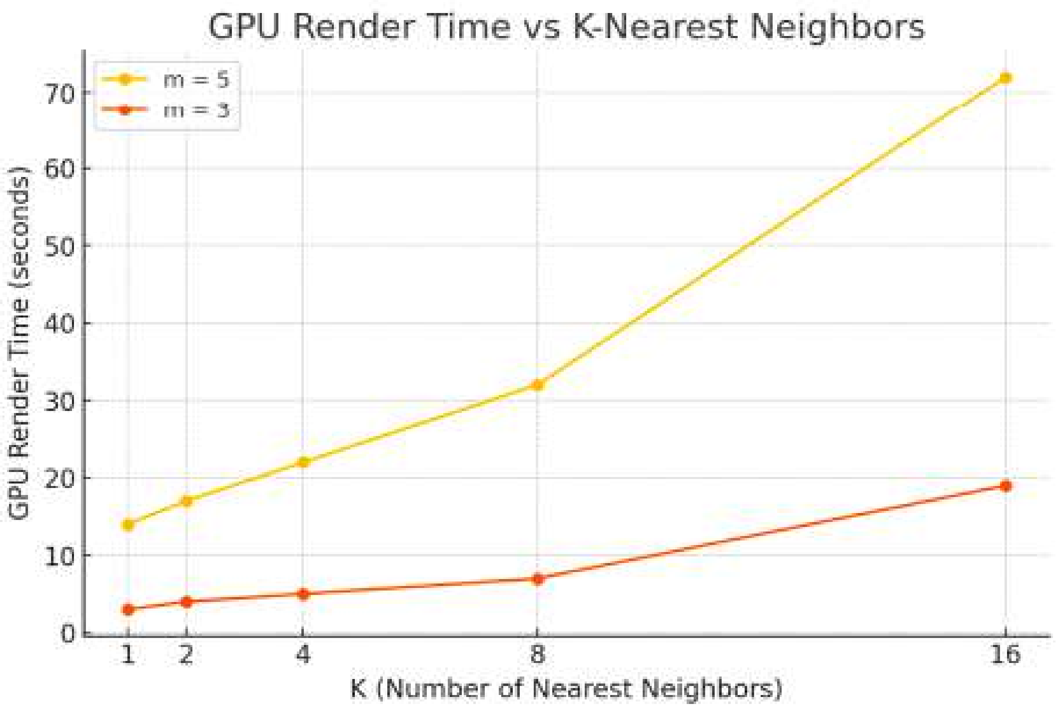
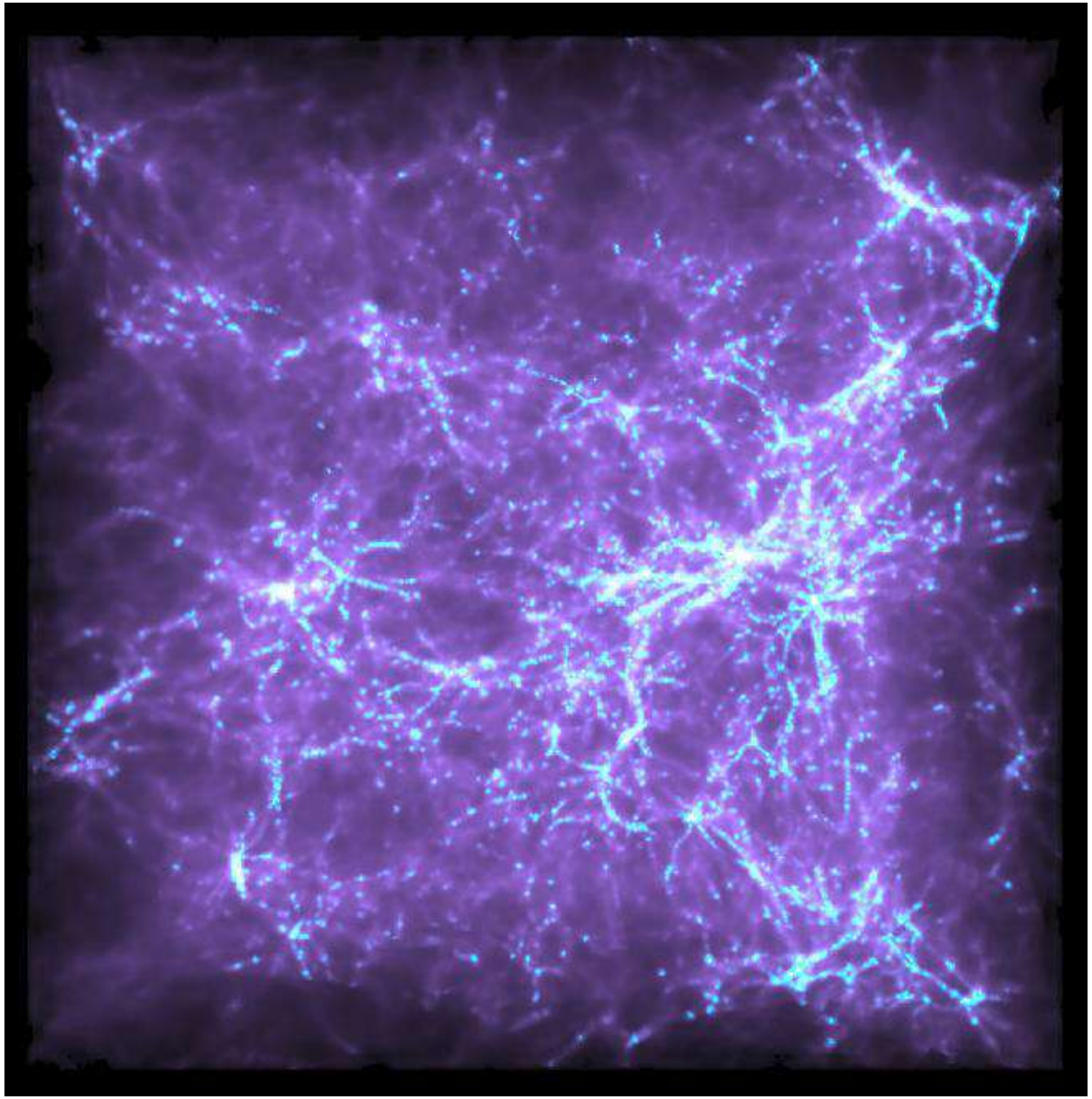
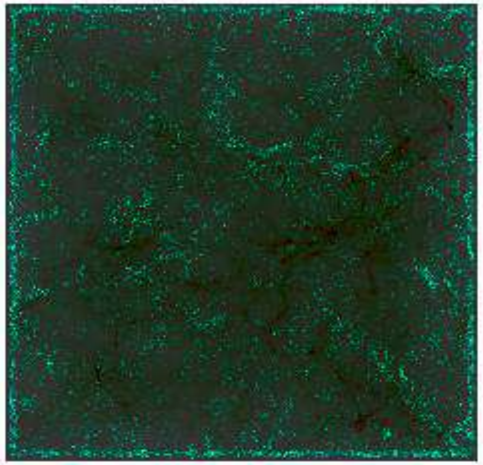


Fig. 9. GPU Render Time vs. K for KNN IDW

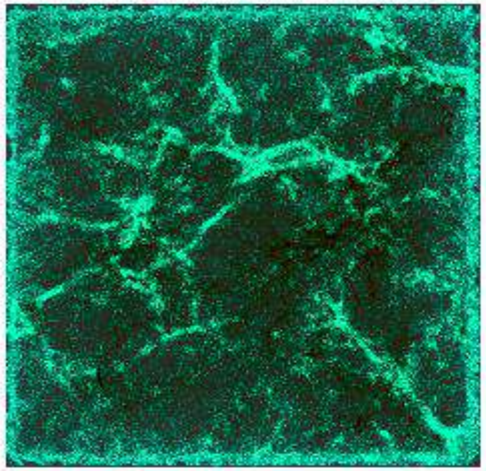


Illustris-1(k=16, m=5)

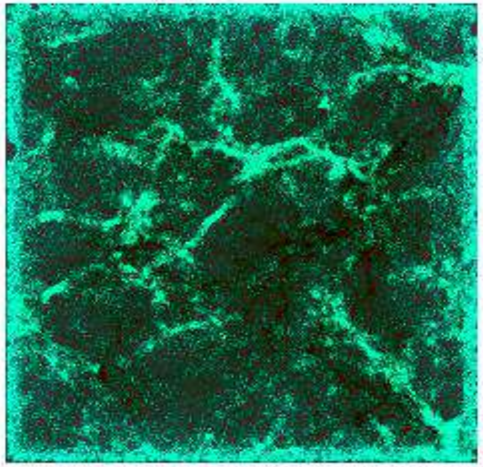
Illustris-1 Metrics



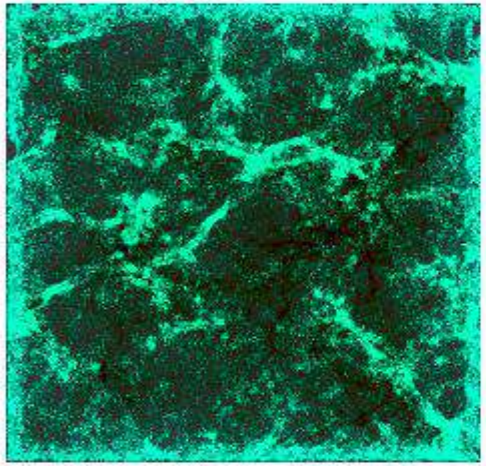
(a) $K = 2$



(b) $K = 4$

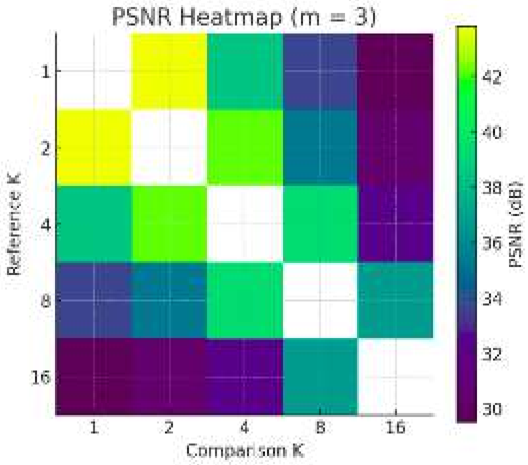


(c) $K = 8$

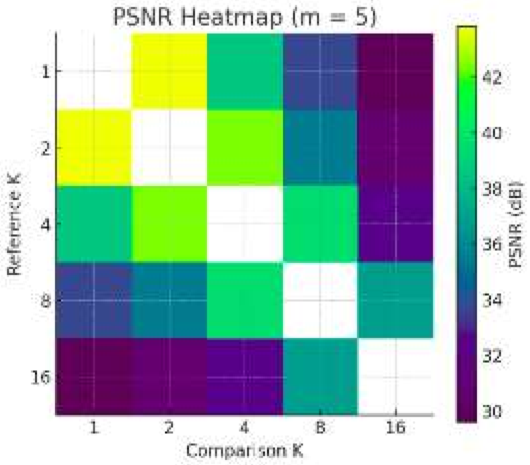


(d) $K = 16$

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

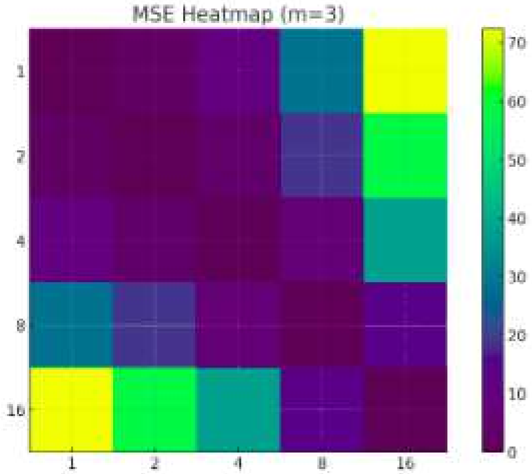


(a) $m = 3$

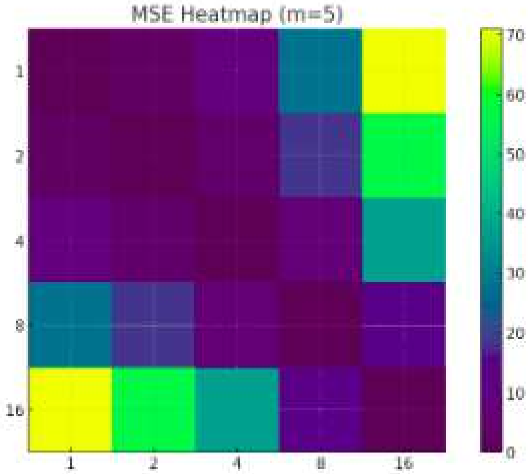


(b) $m = 5$

Fig. 10. Pairwise PSNR comparison between KNN configurations.



(a) $m = 3$

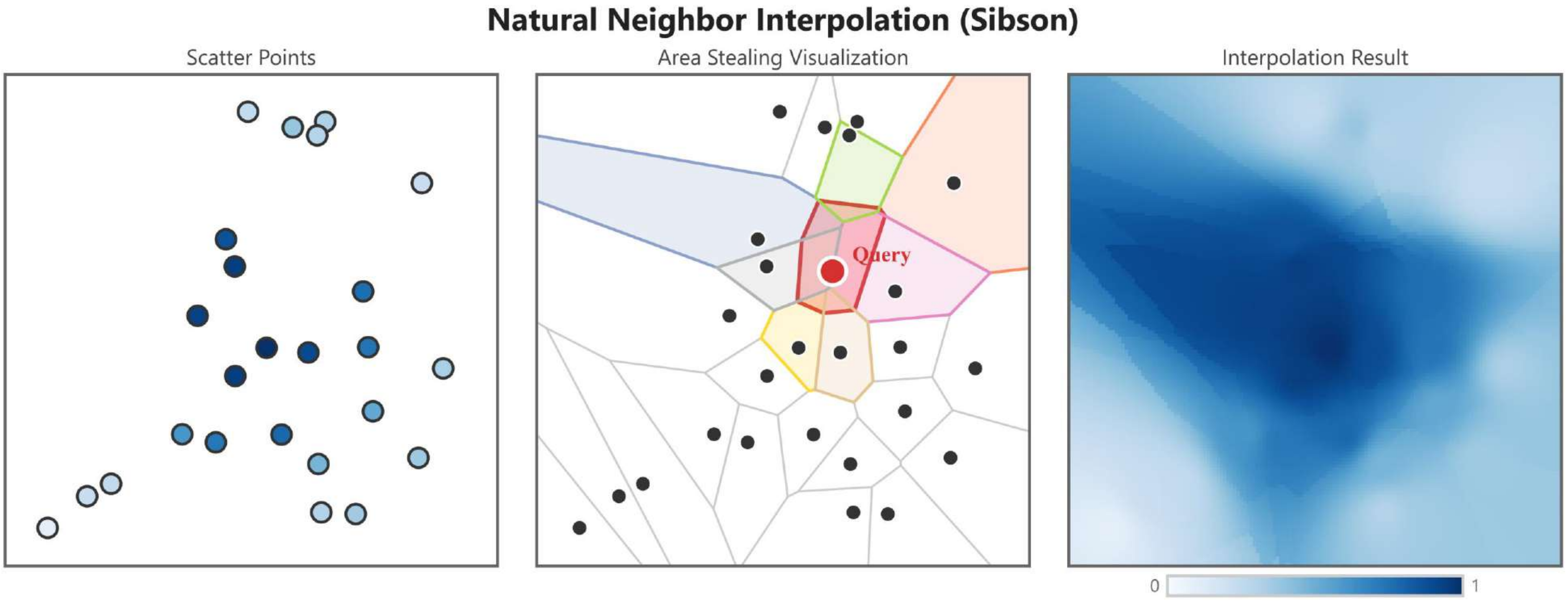


(b) $m = 5$

Fig. 11. Pairwise MSE comparison between KNN configurations.

Method : Natural Neighbors

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



Natural Neighbors

Sibson coordinates (weights)

When you insert x into the Voronoi diagram, each neighbor p_i loses a part of its Voronoi cell. Let:

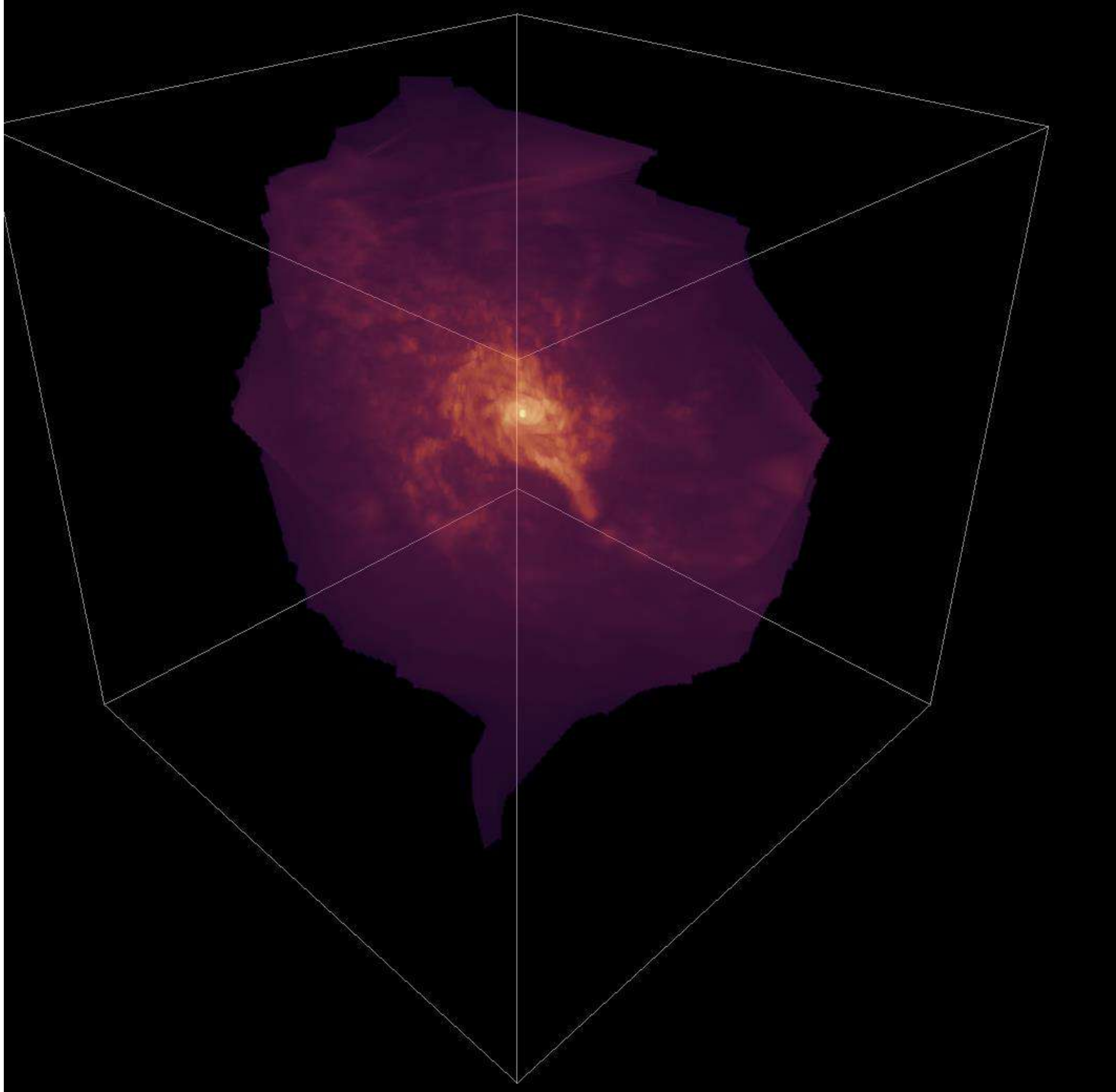
- $A_i(x) =$ area (2D) or volume (3D) stolen from p_i's original Voronoi cell
- $A(x) = \sum_i A_i(x) =$ total inserted Voronoi area/volume

Then the **natural neighbor weight** is:

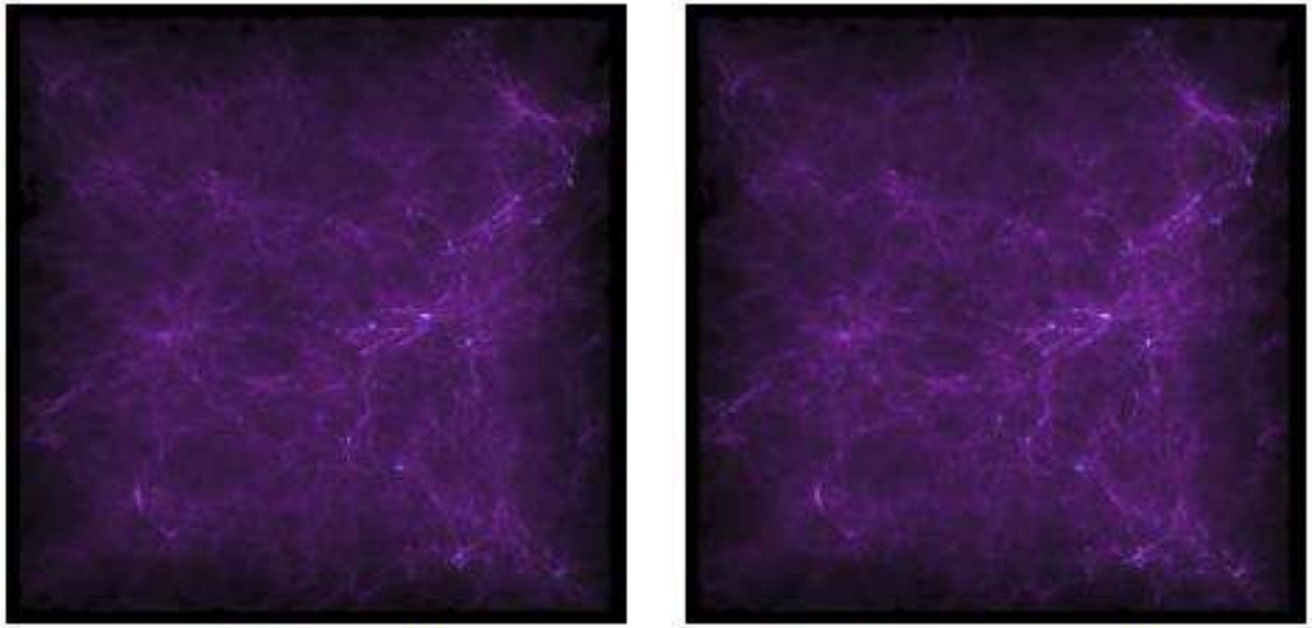
$$w_i(x) = \frac{A_i(x)}{A(x)}$$

The interpolated function value is:

$$f(x) = \sum_i w_i(x)\, f_i$$



Illustris-1 Metrics



(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)

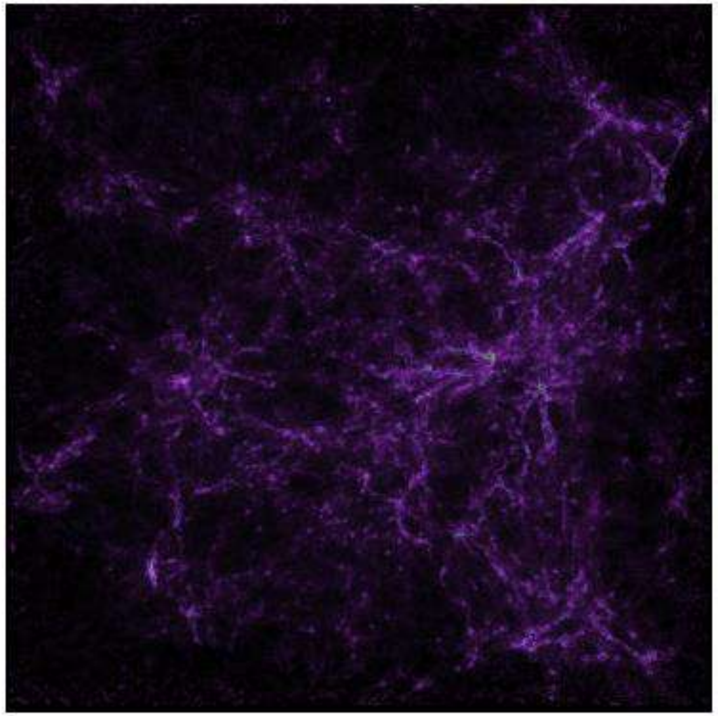
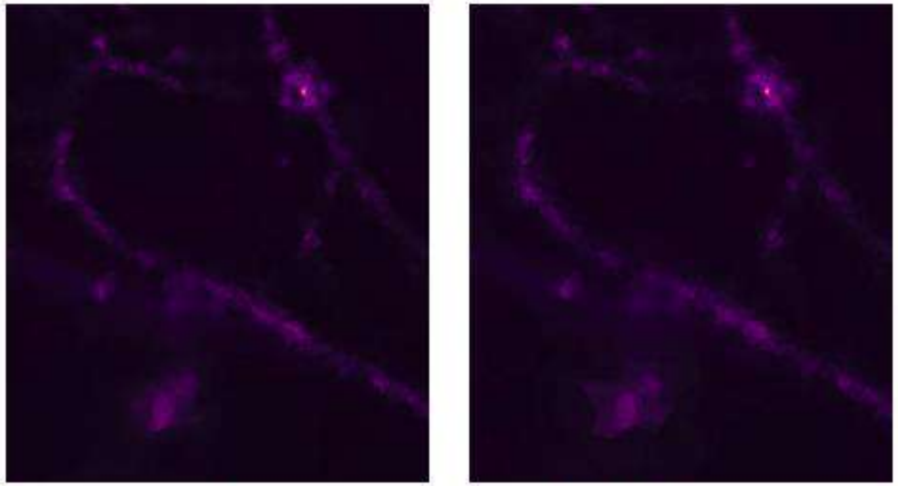


Fig. 7. Voronoi: Difference in pixels (scaled)

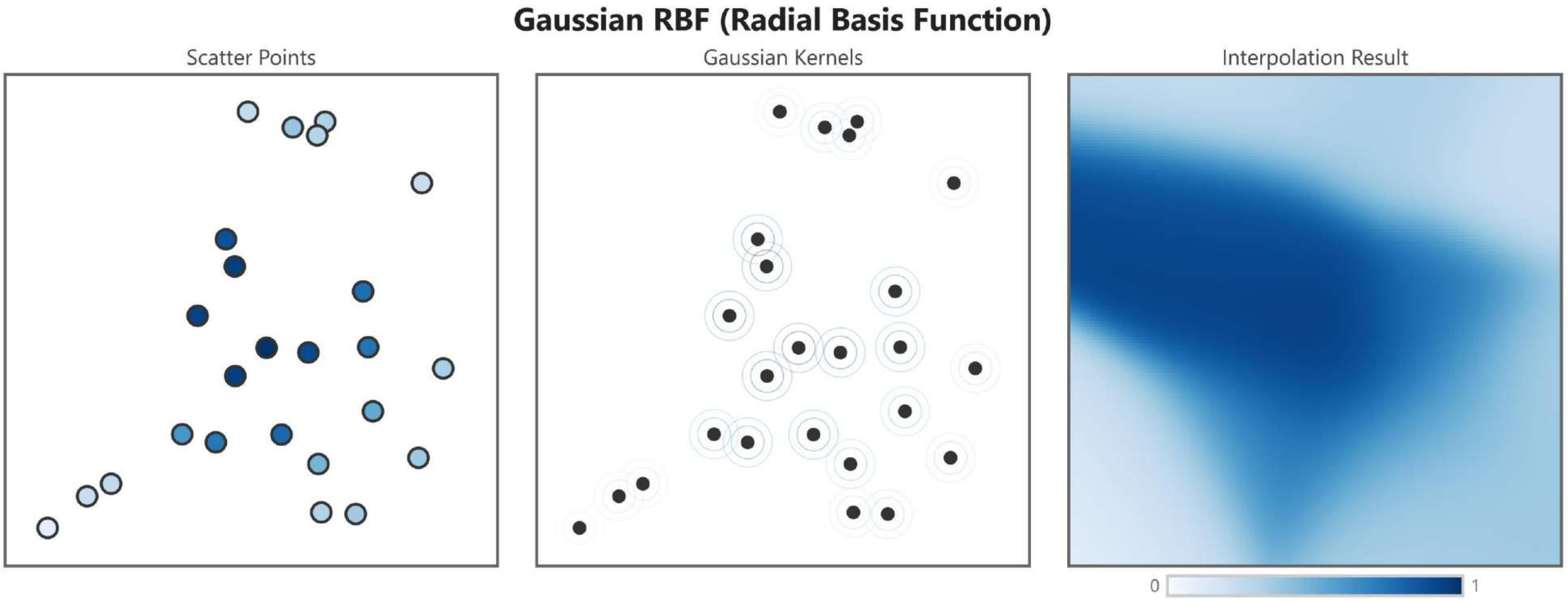


(a) **smooth (cutout)** (b) **blocky (cutout)**

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

Method : Gaussian RBF

Each particle contributes density to surrounding voxels within support radius

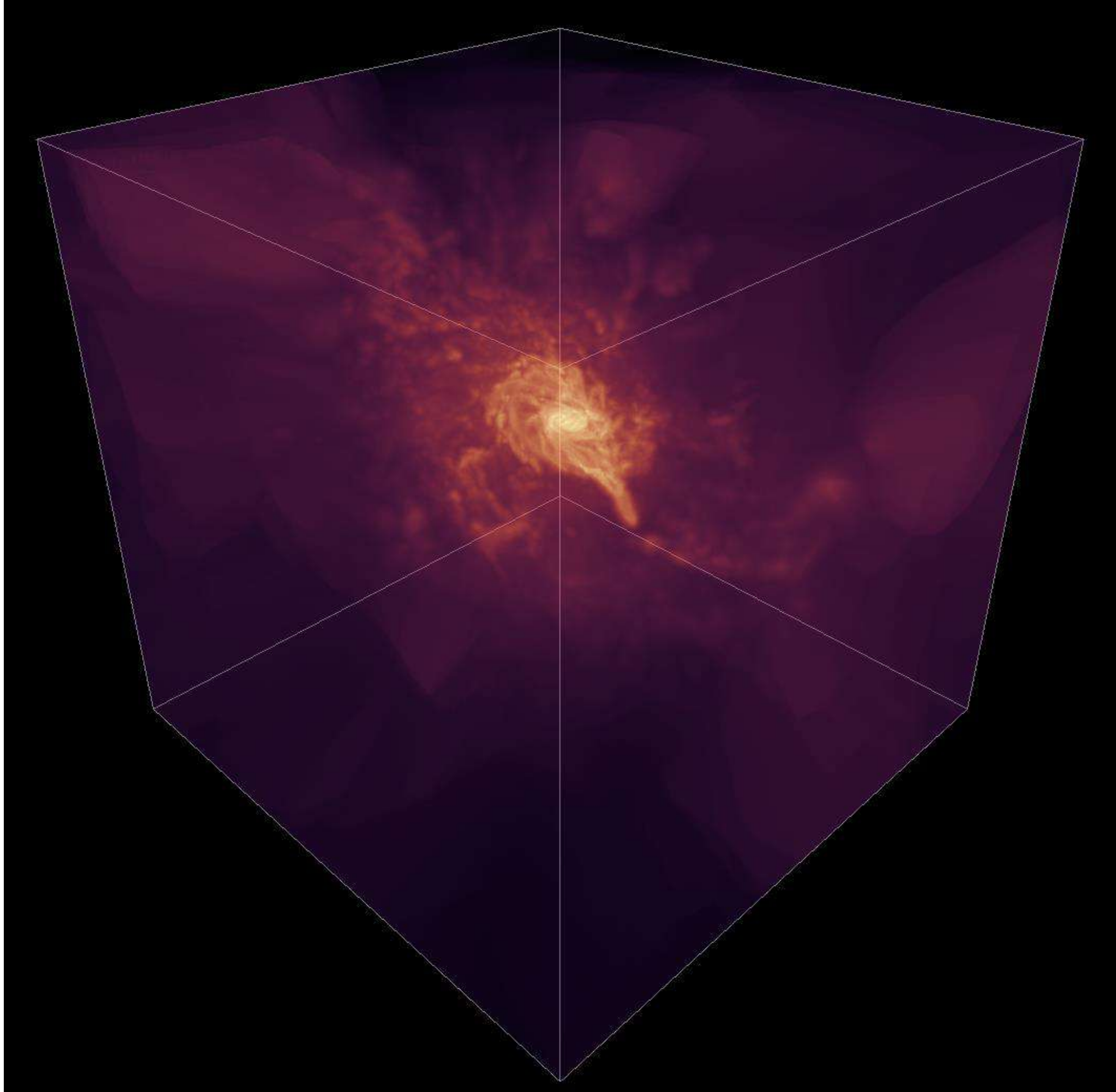


Gaussian RBF

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

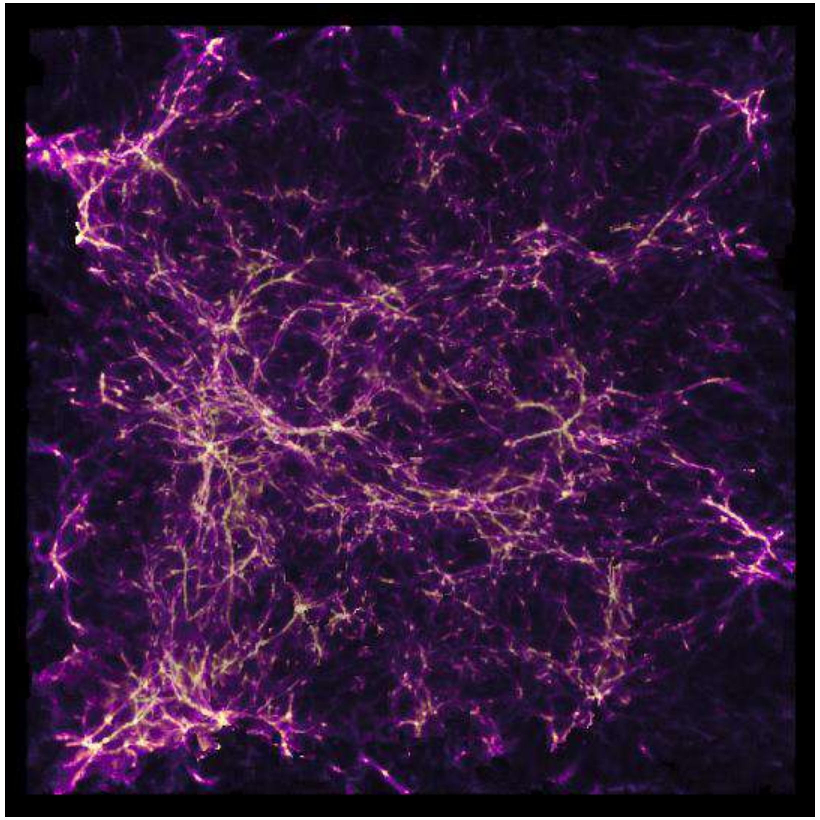
$$\boldsymbol{\rho}(\boldsymbol{x}) = \sum_{\boldsymbol{i}} \boldsymbol{m_i} \boldsymbol{W}(\|\boldsymbol{x_i} - \boldsymbol{x}\|, \boldsymbol{h})$$

where $\boldsymbol{m_i}$ is particle mass, $\boldsymbol{W}$ is smoothing kernel and $\boldsymbol{h}$ is it’s radius.

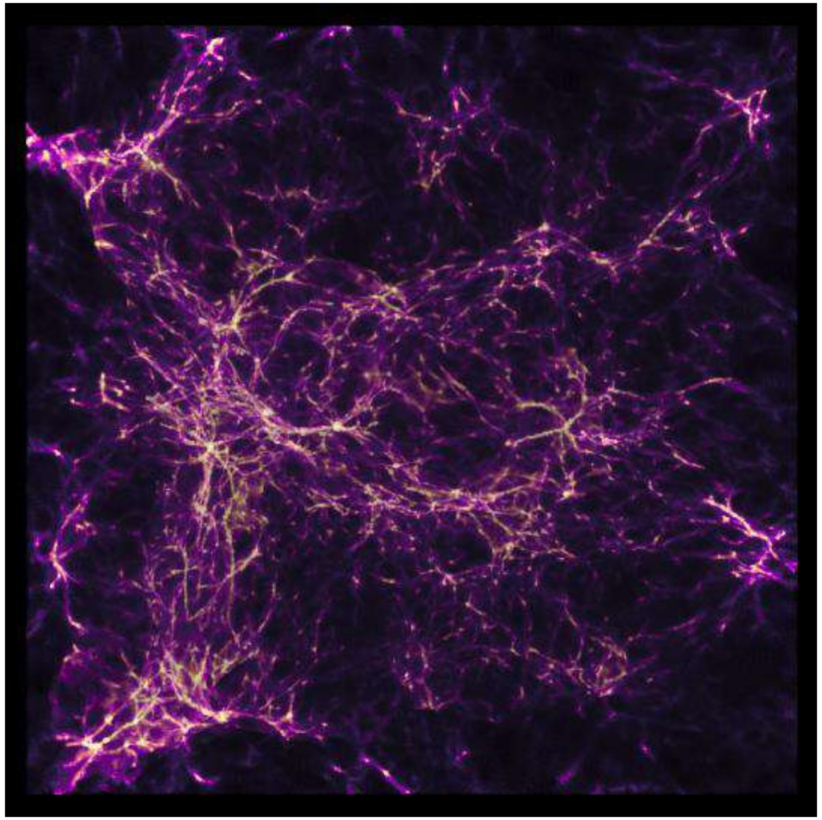


Illustris-1 Visualization

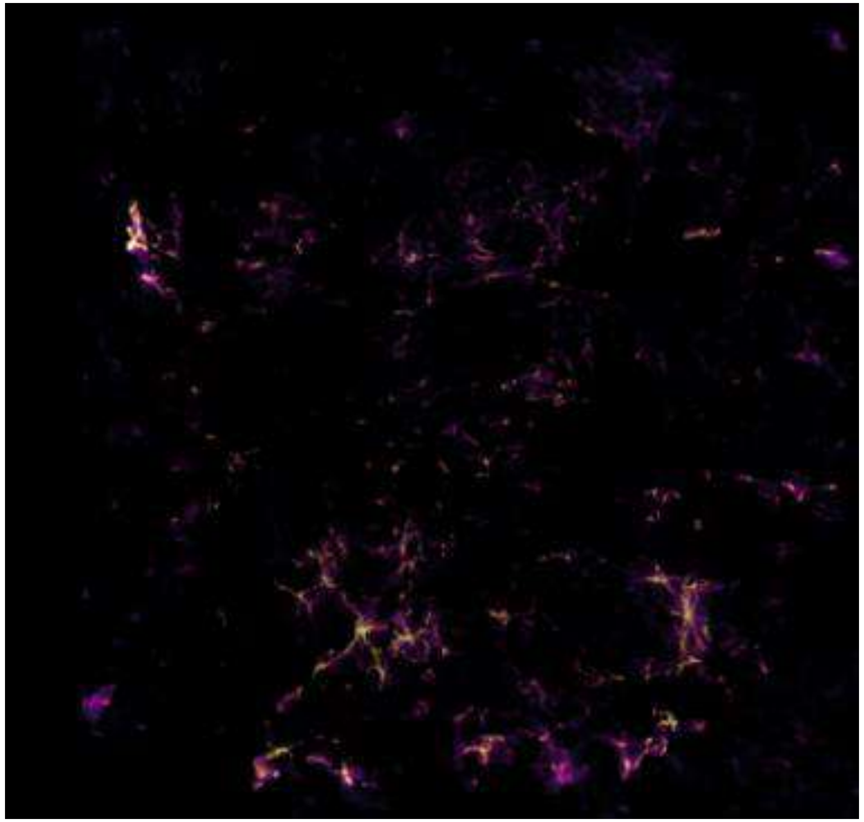
The rendered image was saved at a resolution of 3840×2160. The data set used is: **Illustris-1**



Gaussian Splatting
with adaptive marching.

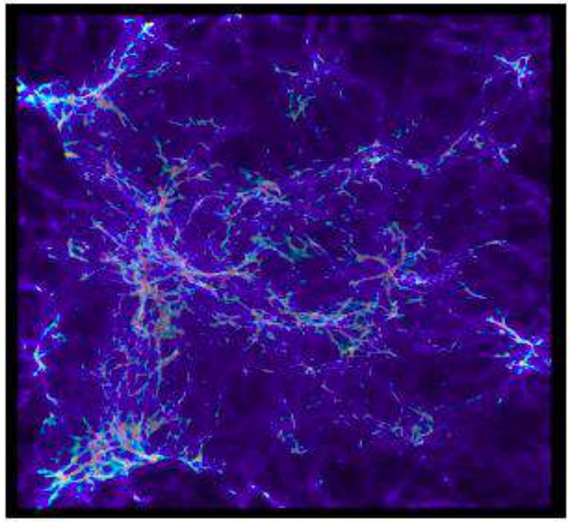


Gaussian Splatting
without adaptive marching.

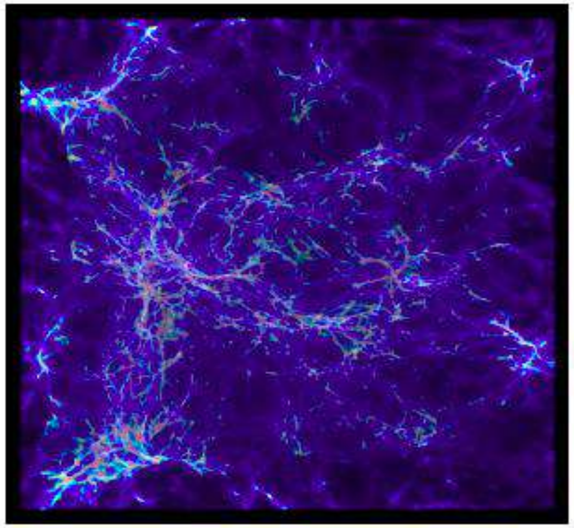


Pixel Difference

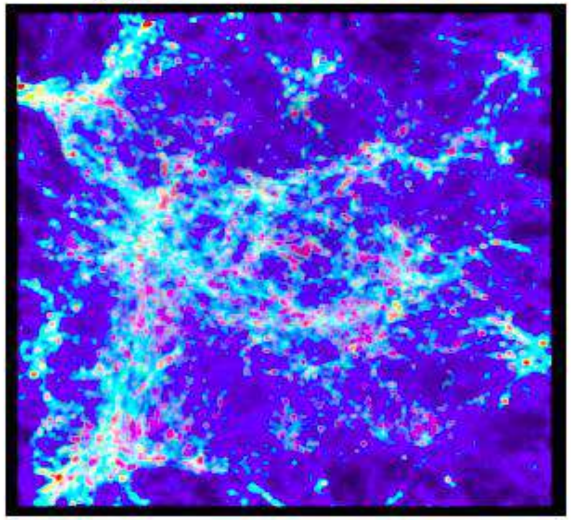
Illustris-1 Visualization



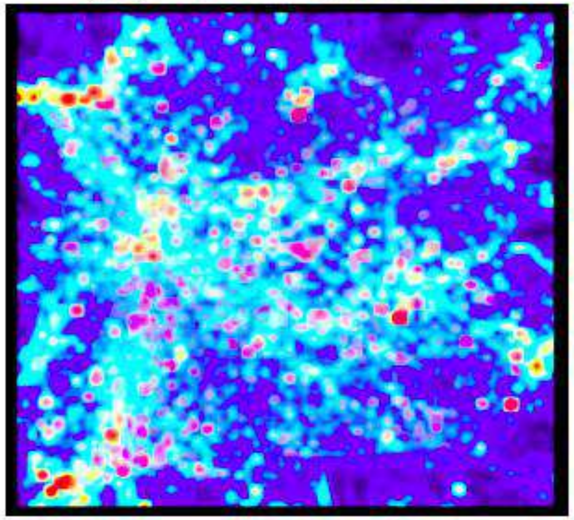
(a) 512^3 **Resolution**



(b) 256^3 **Resolution**



(c) 128^3 **Resolution**



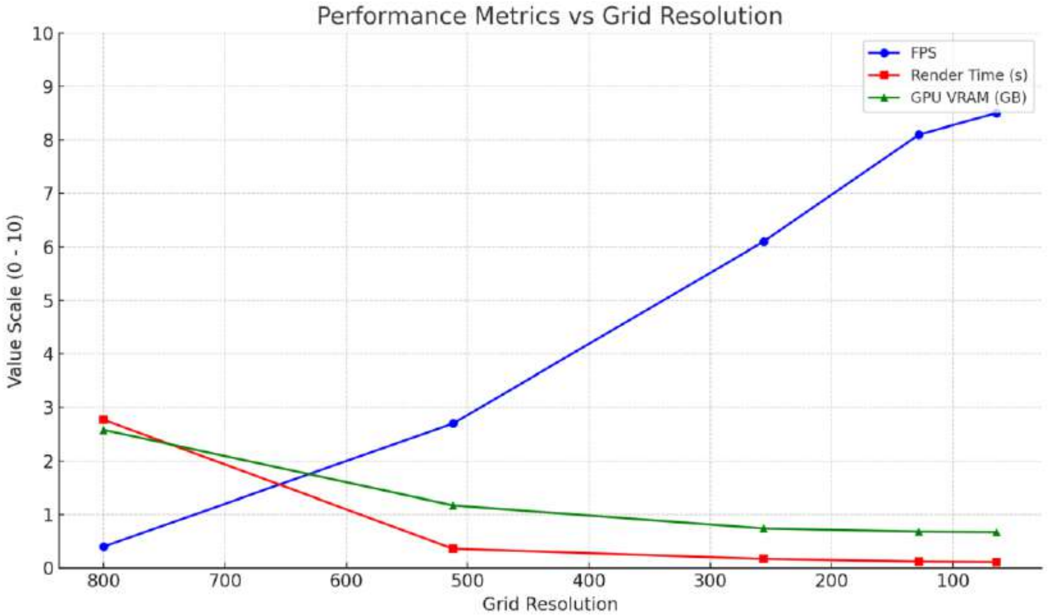
(d) 64^3 **Resolution**

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

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



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

Thank You

Code Link: https://github.com/USharma002/VolPath

Presented by:
Utkarsh Sharma
Akash Maji