GaussianPile: A Unified Sparse Gaussian Splatting Framework for Slice-based Volumetric Reconstruction

Supplementary Material

8. Derivation of Forward rendering process

We will analytically construct the computational pathway from the imaging target composed of 3D Gaussian primitives $g(\mathbf{x})$ with position $\boldsymbol{\mu} \in \mathbb{R}^3$, covariance $\boldsymbol{\Sigma}$, and density a to the final imaging output. Specifically, each primitive can be represented by an anisotropic 3D Gaussian:

$$g(\mathbf{x}) = \alpha \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \quad (13)$$

where $\mathbf{x} = [x, y, z]^\top$ represents the world coordinates. To avoid information redundancy, each Gaussian primitive in the computation contains the following parameters $G = (\boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{q}, \alpha)$, which include: the 3D position $\boldsymbol{\mu}$, the anisotropic scale $\boldsymbol{s} \in \mathbb{R}^3$ controlling the anisotropic scaling, the orientation quaternion $\boldsymbol{q} \in \mathbb{H}$ representing spatial orientation, and the intensity coefficient or opacity α.

During the CUDA rasterization process, to enable differentiable rendering, we need to convert the scale $\boldsymbol{s}$ and orientation $\boldsymbol{q}$ into a 3D covariance matrix $\boldsymbol{\Sigma}$ that describes the Gaussian shape. This conversion follows the classical Gaussian splatting pipeline: first, obtain the diagonal scaling matrix considering the scaling factor through $\boldsymbol{S} = \mathrm{diag}(\mathrm{mod} \cdot \boldsymbol{s})$; second, convert the quaternion to a rotation matrix via $\boldsymbol{R} = \mathrm{quatToMatrix}(\boldsymbol{q})$; finally, synthesize the covariance matrix in the world coordinate system through the transformation $\boldsymbol{\Sigma} = \boldsymbol{RSS}^\top \boldsymbol{R}^\top$.

We conduct our analysis in the camera coordinate system, thus it is first necessary to transform the Gaussian primitives from the world coordinate system to the camera coordinate system. For each virtual slice, we utilize its exported view matrix $[R_c|\mathbf{t}]$ to transform the Gaussian center $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ from the world coordinate system to the camera coordinate system (focused imaging plane). The transformation formulas are as follows:

$$\boldsymbol{\mu}_c = R_c \cdot \boldsymbol{\mu} + \mathbf{t}, \quad (14)$$

$$\boldsymbol{\Sigma}_c = R_c \cdot \boldsymbol{\Sigma} \cdot R_c^\top, \quad (15)$$

where R_c is a 3×3 rotation matrix and $\mathbf{t}$ is a 3×1 translation vector.

We assume the point spread function (PSF) of the imaging system is spatially invariant and can be represented by a 3D Gaussian function:

$$\mathrm{psf}(\mathbf{x}_c) = \frac{1}{(2\pi)^{3/2}\sigma_x\sigma_y\sigma_z} \exp\left(-\frac{1}{2}\left(\frac{x_c^2}{\sigma_x^2} + \frac{y_c^2}{\sigma_y^2} + \frac{z_c^2}{\sigma_z^2}\right)\right), \quad (16)$$

where σ_x, σ_y, and σ_z represent the spatial resolution of the imaging system in the x, y, and z directions, respectively. Notably, σ_z directly reflects the focusing capability of the imaging system in the elevational (axial) direction. Based on this, we define the sensitivity map function $h(z)$ as the reversed system's axial impulse response:

$$h(-z_c) = \exp\left(-\frac{z_c^2}{2\sigma_z^2}\right). \quad (17)$$

Generally, σ_z relates to the numerical aperture NA_{ele} in the elevational direction and the imaging wavelength λ, and can be calculated using the formula:

$$\sigma_z \approx 0.27 \cdot \frac{\lambda}{NA_{ele}}. \quad (18)$$

For specific imaging systems, this formula can be further refined. For instance, for a linear optical imaging system with a numerical aperture NA:

$$\sigma_z \approx \frac{\lambda}{NA^2}. \quad (19)$$

For an ultrasound imaging system:

$$\sigma_z \approx k\frac{\lambda F}{D_{ele}}, \quad (20)$$

where λ is the ultrasonic wavelength, F is the transducer focal length, and D_{ele} is the effective aperture size of the ultrasonic transducer in the elevational direction.

According to the above formulation, for a single Gaussian primitive, the rendered image I can be obtained by the convolution of the Gaussian primitive $g_c(\mathbf{x}_c)$ in the camera coordinates with the reversed sensitivity map function $h(-z)$, evaluated at $z_c = 0$:

$$I(x_c, y_c) = [h(-z) * g_c(\mathbf{x}_c)]\,\big|_{z_c=0}. \quad (21)$$

Expanding the above equation into its integral form yields:

$$I(x_c, y_c, z_c = 0) = \int_{-\infty}^{\infty} h(t) \cdot g_c(x_c, y_c, t)dt. \quad (22)$$

Since the integrand $g_e(\mathbf{x}) = h(\mathbf{x})g_c(\mathbf{x})$ is the product of two Gaussian functions, it remains Gaussian in form. We refer to this effective Gaussian as the focus Gaussian.

When deriving the parameters of the focus Gaussian, we consider the sum of the exponent terms of the two Gaussian

functions. The exponent of the Gaussian $g_c(\mathbf{x})$ in camera coordinates is:

$$-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c). \tag{23}$$

The exponent of the sensitivity map function $h(\mathbf{x})$ is:

$$-\frac{z_c^2}{2\sigma_z^2} = -\frac{1}{2\sigma_z^2}\mathbf{x}^\top(\mathbf{e}_3\mathbf{e}_3^\top)\mathbf{x}, \tag{24}$$

where $\mathbf{e}_3 = [0,0,1]^\top$. Therefore, the total exponent is:

$$-\frac{1}{2}\left[(\mathbf{x}-\boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c) + \frac{1}{\sigma_z^2}\mathbf{x}^\top(\mathbf{e}_3\mathbf{e}_3^\top)\mathbf{x}\right]. \tag{25}$$

Expanding the first term:

$$\begin{aligned}&(\mathbf{x}-\boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c) = \\ &\mathbf{x}^\top \boldsymbol{\Sigma}_c^{-1}\mathbf{x} - 2\mathbf{x}^\top \boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c + \boldsymbol{\mu}_c^\top \boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c.\end{aligned} \tag{26}$$

Thus the complete exponent becomes:

$$\begin{aligned}-\frac{1}{2}\Bigg[&\mathbf{x}^\top\left(\boldsymbol{\Sigma}_c^{-1} + \frac{\mathbf{e}_3\mathbf{e}_3^\top}{\sigma_z^2}\right)\mathbf{x} \\ &-2\mathbf{x}^\top \boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c + \boldsymbol{\mu}_c^\top \boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c\Bigg].\end{aligned} \tag{27}$$

Comparing this with the standard Gaussian form $-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_e)^\top \boldsymbol{\Sigma}_e^{-1}(\mathbf{x}-\boldsymbol{\mu}_e)$, we can identify:

$$\boldsymbol{\Sigma}_e^{-1} = \boldsymbol{\Sigma}_c^{-1} + \frac{\mathbf{e}_3\mathbf{e}_3^\top}{\sigma_z^2}, \tag{28}$$

and

$$\boldsymbol{\Sigma}_e^{-1}\boldsymbol{\mu}_e = \boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c. \tag{29}$$

Solving for $\boldsymbol{\mu}_e$:

$$\boldsymbol{\mu}_e = \boldsymbol{\Sigma}_e\boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c. \tag{30}$$

The remaining constant term $\boldsymbol{\mu}_c^\top \boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c - \boldsymbol{\mu}_e^\top \boldsymbol{\Sigma}_e^{-1}\boldsymbol{\mu}_e$ is absorbed into the opacity modulation term:

$$\text{opacity}_r = \exp\left(-\frac{1}{2}\left(\boldsymbol{\mu}_c^\top \boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c - \boldsymbol{\mu}_e^\top \boldsymbol{\Sigma}_e^{-1}\boldsymbol{\mu}_e\right)\right). \tag{31}$$

Finally, the rendered image $I(x_c, y_c)$ can be expressed as:

$$\begin{aligned}&I(x_c,y_c) = \alpha \cdot \text{opacity}_r \\ &\int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_e)^\top \boldsymbol{\Sigma}_e^{-1}(\mathbf{x}-\boldsymbol{\mu}_e)\right) \quad dz_c.\end{aligned} \tag{32}$$

The integrand is clearly a three-dimensional Gaussian distribution. Integrating over z_c yields the marginal distribution with respect to x_c and y_c, which is a two-dimensional Gaussian distribution. Its mean and covariance matrix are determined by the corresponding components of the original mean vector and covariance matrix. Based on this, we directly express the computed result as:

$$\begin{aligned}&I(x_c,y_c) = \alpha \cdot \frac{\text{opacity}_r}{\sqrt{|\boldsymbol{\Sigma}_{2d}|}} \\ &\cdot \exp\left(-\frac{1}{2}([x_c,y_c]^\top - \boldsymbol{\mu}_{2d})^\top \boldsymbol{\Sigma}_{2d}^{-1}([x_c,y_c]^\top - \boldsymbol{\mu}_{2d})\right),\end{aligned} \tag{33}$$

where $\boldsymbol{\Sigma}_{2d}$ and $\boldsymbol{\mu}_{2d}$ are the components of $\boldsymbol{\Sigma}_e$ and $\boldsymbol{\mu}_e$ along the x and y directions, respectively:

$$\boldsymbol{\Sigma}_{2d} = \begin{bmatrix}\boldsymbol{\Sigma}_e[0,0] & \boldsymbol{\Sigma}_e[1,0] \\ \boldsymbol{\Sigma}_e[1,0] & \boldsymbol{\Sigma}_e[1,1]\end{bmatrix}, \quad \boldsymbol{\mu}_{2d} = [\boldsymbol{\mu}_e[0], \boldsymbol{\mu}_e[1]]^\top. \tag{34}$$

To facilitate the rendering of the 2D image, we transform the above equation into the screen space. For a pixel p, we use $\mathbf{d}$ to denote the 2D vector from the 2D Gaussian center $\boldsymbol{\mu}_{2d}$ to the coordinates of pixel p. The rendered image can then be expressed as:

$$I(p) = \tilde{\alpha} \cdot \exp\left(-\frac{1}{2}\mathbf{d}^\top \boldsymbol{\Sigma}_{2d}^{-1}\mathbf{d}\right), \tag{35}$$

where $\tilde{\alpha}$ represents the modulated final rendering intensity:

$$\tilde{\alpha} = \alpha \cdot \frac{\text{opacity}_r}{\sqrt{|\boldsymbol{\Sigma}_{2d}|}}. \tag{36}$$

Finally, for scenarios involving multiple Gaussian primitives, due to the linearity of the imaging model, we only need to sum over all Gaussian primitives:

$$I(p) = \sum_{i\in\mathcal{N}(p)} \tilde{\alpha}_i \cdot \exp\left(-\frac{1}{2}\mathbf{d}_i^\top \boldsymbol{\Sigma}_{2d,i}^{-1}\mathbf{d}_i\right), \tag{37}$$

where $\mathcal{N}(p)$ denotes the set of Gaussians overlapping with pixel p.

9. Derivation of Differentiable Backward Pass

9.1. Forward Process Review

2D mean:

$$\boldsymbol{\mu}_{2d} = [\boldsymbol{\mu}_e[0], \boldsymbol{\mu}_e[1]]^\top \tag{38}$$

2D inverse covariance matrix:

$$\boldsymbol{\Sigma}_{2d}^{-1} = \frac{1}{\det(\boldsymbol{\Sigma}_{2d})}\begin{bmatrix}\boldsymbol{\Sigma}_e[1,1] & -\boldsymbol{\Sigma}_e[1,0] \\ -\boldsymbol{\Sigma}_e[1,0] & \boldsymbol{\Sigma}_e[0,0]\end{bmatrix} \tag{39}$$

where

$$\boldsymbol{\Sigma}_{2d} = \begin{bmatrix}\boldsymbol{\Sigma}_e[0,0] & \boldsymbol{\Sigma}_e[1,0] \\ \boldsymbol{\Sigma}_e[1,0] & \boldsymbol{\Sigma}_e[1,1]\end{bmatrix}. \tag{40}$$

Rendered intensity:

$$\tilde{\alpha} = \alpha \cdot \text{opacity}_r \cdot \frac{1}{\sqrt{\det(\boldsymbol{\Sigma}_{2d})}}. \tag{41}$$

These parameters form the basis for backward propagation, linking 2D rendering to 3D Gaussian primitives.

9.2. Backward Propagation

Gradient from 2D mean:

$$\frac{dL}{d\boldsymbol{\mu}_e} = \begin{bmatrix} \frac{dL}{d\boldsymbol{\mu}_{2d}[0]} \\ \frac{dL}{d\boldsymbol{\mu}_{2d}[1]} \\ 0 \end{bmatrix}. \tag{42}$$

Gradients from rendered intensity:

$$\frac{dL}{d\alpha} = \frac{dL}{d\tilde{\alpha}} \cdot \frac{\text{opacity}_r}{\sqrt{\det(\boldsymbol{\Sigma}_{2d})}}, \tag{43}$$

$$\frac{dL}{d\text{opacity}_r} = \frac{dL}{d\tilde{\alpha}} \cdot \frac{\alpha}{\sqrt{\det(\boldsymbol{\Sigma}_{2d})}}, \tag{44}$$

$$\frac{dL}{d\det(\boldsymbol{\Sigma}_{2d})} = \frac{dL}{d\tilde{\alpha}} \cdot \left(-\frac{1}{2} \cdot \frac{\alpha \times \text{opacity}_r}{[\det(\boldsymbol{\Sigma}_{2d})]^{3/2}} \right). \tag{45}$$

These compute contributions from rendered intensity, essential for focus-aware rendering.
Gradient from $\boldsymbol{\Sigma}_{2d}^{-1}$ to $\boldsymbol{\Sigma}_{2d}$ (using $\boldsymbol{\Sigma}_{2d} \cdot \boldsymbol{\Sigma}_{2d}^{-1} = \boldsymbol{I}$):

$$\frac{dL}{d\boldsymbol{\Sigma}_{2d}} = -\boldsymbol{\Sigma}_{2d} \cdot \frac{dL}{d\boldsymbol{\Sigma}_{2d}^{-1}} \cdot \boldsymbol{\Sigma}_{2d} + \frac{dL}{d\det(\boldsymbol{\Sigma}_{2d})} \cdot \det(\boldsymbol{\Sigma}_{2d}) \cdot \boldsymbol{\Sigma}_{2d}^{-1}. \tag{46}$$

This combines direct inverse propagation with determinant contribution, ensuring covariance consistency.
Map to $\boldsymbol{\Sigma}_e$:

$$\frac{dL}{d\boldsymbol{\Sigma}_e} = \begin{bmatrix} \frac{dL}{d\boldsymbol{\Sigma}_{2d}[0,0]} & \frac{dL}{d\boldsymbol{\Sigma}_{2d}[1,0]} & 0 \\ \frac{dL}{d\boldsymbol{\Sigma}_{2d}[1,0]} & \frac{dL}{d\boldsymbol{\Sigma}_{2d}[1,1]} & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{47}$$

From $\boldsymbol{\mu}_e$ and $\boldsymbol{\Sigma}_e$ to $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ (with $q = \boldsymbol{\mu}_c^\top \boldsymbol{\Sigma}_c^{-1} \boldsymbol{\mu}_c - \boldsymbol{\mu}_e^\top \boldsymbol{\Sigma}_e^{-1} \boldsymbol{\mu}_e$):

$$\frac{dL}{dq} = \frac{dL}{d\text{opacity}_r} \cdot \left(-\frac{1}{2} \text{opacity}_r \right), \tag{48}$$

$$\frac{dL}{d\boldsymbol{\mu}_c} = \left(\boldsymbol{\Sigma}_e \cdot \boldsymbol{\Sigma}_c^{-1} \right)^\top \cdot \frac{dL}{d\boldsymbol{\mu}_e} + \frac{dL}{dq} \cdot 2\boldsymbol{\Sigma}_c^{-1} \boldsymbol{\mu}_c, \tag{49}$$

$$\frac{dL}{d\boldsymbol{\Sigma}_c^{-1}} = \frac{dL}{dq} \cdot \boldsymbol{\mu}_c \boldsymbol{\mu}_c^\top + \frac{dL}{d\boldsymbol{\Sigma}_e^{-1}}, \tag{50}$$

$$\frac{dL}{d\boldsymbol{\Sigma}_e^{-1}} = \frac{dL}{dq} \cdot (-\boldsymbol{\mu}_e \boldsymbol{\mu}_e^\top) - \boldsymbol{\Sigma}_e \cdot \frac{dL}{d\boldsymbol{\Sigma}_e} \cdot \boldsymbol{\Sigma}_e. \tag{51}$$

From $\boldsymbol{\Sigma}_c^{-1}$ to $\boldsymbol{\Sigma}_c$:

$$\frac{dL}{d\boldsymbol{\Sigma}_c} = -\boldsymbol{\Sigma}_c^{-1} \cdot \frac{dL}{d\boldsymbol{\Sigma}_c^{-1}} \cdot \boldsymbol{\Sigma}_c^{-1}. \tag{52}$$

This propagates gradients back to camera coordinates, maintaining differentiability.
Finally back to world coordinates:

$$\frac{dL}{d\tilde{\boldsymbol{\mu}}} = R_c^\top \cdot \frac{dL}{d\boldsymbol{\mu}_c}, \tag{53}$$

$$\frac{dL}{d\tilde{\boldsymbol{\Sigma}}} = R_c^\top \cdot \frac{dL}{d\boldsymbol{\Sigma}_c} \cdot R_c. \tag{54}$$

Above final steps transform gradients to world space, enabling end-to-end optimization of 3D Gaussians.

10. Implementation details of baseline methods

Comparisons are conducted with various compression and 3D reconstruction methods. The traditional compression algorithm HEVC [34] is employed with a CRF value of 16. For the SOTA NeRF-based method represented by INIF [8], a SIREN-like neural network with frequency-adjusted activations is trained for 40000 iterations using the Adam optimizer and a compression ratio of 16, with automatic batch size tuning and JAX for GPU acceleration. The implementation of INIF uses its official code with default hyperparameters. Inspired by FruitNinja [40], we also develop a slice-adapted version of original 3DGS [14] for evaluation. Two modifications are made: (i) **Internal initialization**: we employ grid-based ray-casting to fill interior voxels with Gaussian primitives, ensuring volumetric coverage beyond surface points; (ii) **Slice rendering**: unlike FruitNinja's infinitesimally thin cutting planes, we render slices with finite thickness by first masking out Gaussians whose distance from the imaging plane exceeds their spatial extent, then projecting the retained primitives to 2D Gaussian ellipses with depth-dependent attenuation. Contributions are additively accumulated to match intensity integration in slice imaging. All methods are run on NVIDIA A800 GPUs.

11. More qualitative results

Main results. We visualize more reconstruction results in Fig. 6. HEVC and INIF introduce strong blocky or oversmoothed artifacts, failing to recover fine cellular structures faithfully. Original 3DGS preserves some details in high signal-to-noise conditions, but produces floating artifacts in low-signal conditions. In contrast, our method successfully recovers sharper morphology with cleaner backgrounds and better local contrast across all conditions, from high-quality TNNI1 to very-low-signal Tribolium dataset, closely matching the ground truth.
Volume visualization. In the main text, the 3D reconstruction results are presented as maximum-intensity projections (MIP), which appear in grayscale. To provide a clearer visualization of the volumetric geometry, we additionally show depth-colored renderings of the reconstructed volumes in Fig. 7. The depth of the dominant signal along the z-axis is encoded using a perceptually uniform colormap, allowing fine-scale structural variations to be more easily perceived. This visualization is intended to improve perceptual understanding of the reconstructed 3D geometry.
Convergence analysis. We conduct a visual convergence analysis of the INIF [8], original 3DGS [14], and Gaussian-

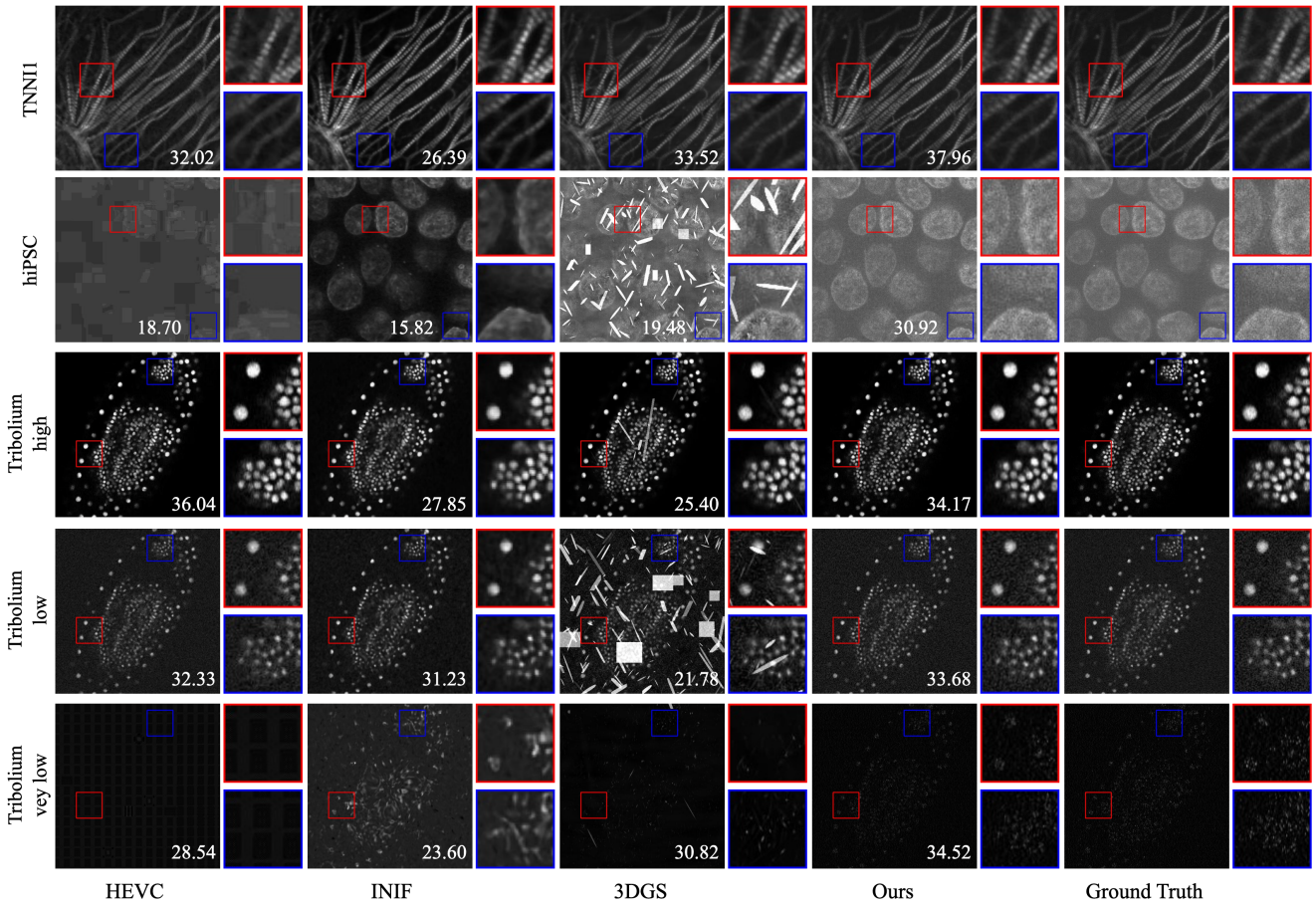

Figure 6. Qualitative comparison of reconstruction results on three cellular datasets. Slice examples of different methods with PSNR (dB) shown at the bottom of each image.

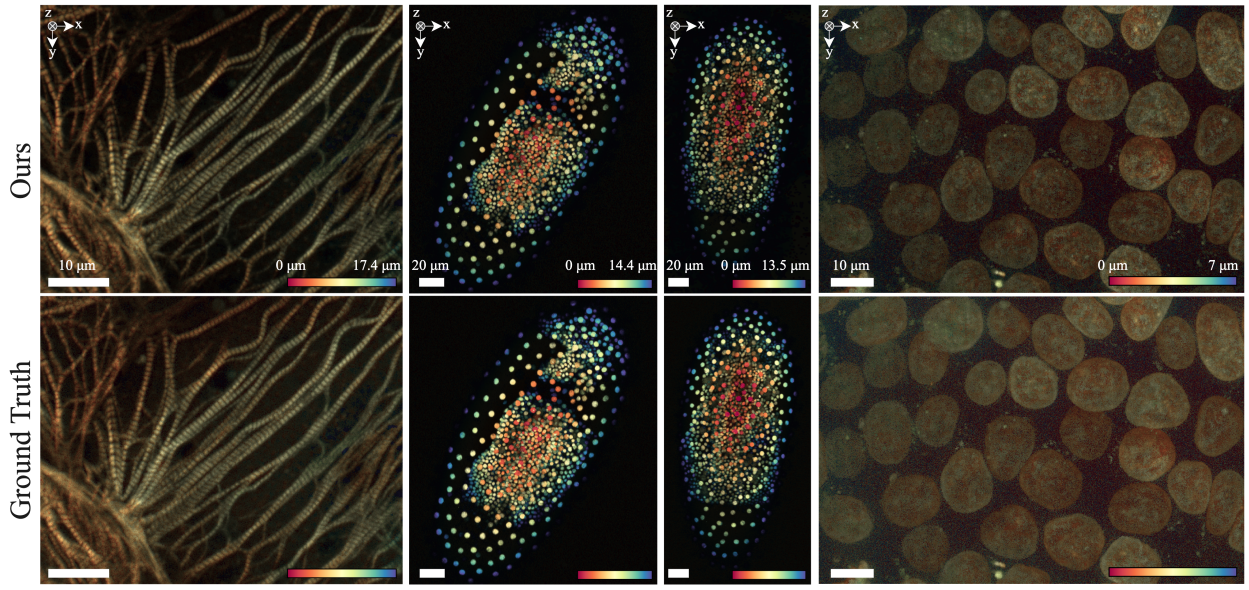

Figure 7. 3D reconstruction results on cellular datasets, presented by depth-encoded color rendering of volumes.

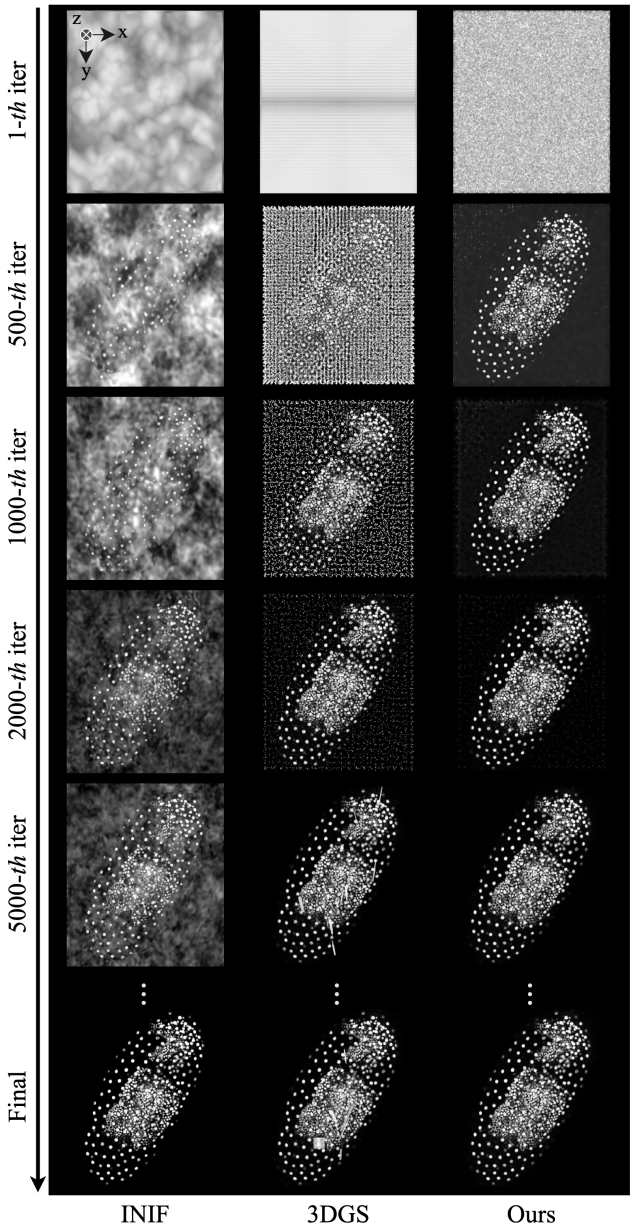

Figure 8. Convergence analysis of baseline methods and GaussianPile. We visualize the intermediate snapshots at different iterations. GaussianPile shows faster and better convergence.

Pile in Fig. 8. INIF gradually forms the structure but converges slowly. Original 3DGS converges faster than INIF, but struggles to cull the dense grid-pattern Gaussians in early iterations and subsequently exhibits persistent floating artifacts. In contrast, GaussianPile rapidly forms a compact and well-localized representation within $\sim$ 2k iterations. These observations demonstrate that GaussianPile achieves faster and more stable convergence while producing cleaner reconstructions.

Application on Industrial Non-Destructive Testing. To further assess the generality of GaussianPile, we evaluate it on the CeraMIRScan Mid-infrared Optical Coherence Tomography (MIR-OCT) dataset, a challenging industrial non-destructive testing (NDT) benchmark comprising 29 MIR-OCT volumes of 3D-printed ceramic components. Unlike biomedical ultrasound and microscopy data used in our main text, MIR-OCT exhibits distinct noise characteristics, signal attenuation patterns, and structural textures, making it an ideal testbed for robustness. As shown in Fig. 9, GaussianPile successfully reconstructs the layered ceramic structures and internal patterns across samples ("hole", "pilot"), producing results that closely match the ground-truth MIR-OCT slices. These results demonstrate that GaussianPile is not limited to biomedical 3D tomography and can be effectively applied to industrial inspection scenarios, highlighting its versatility for broader NDT applications.

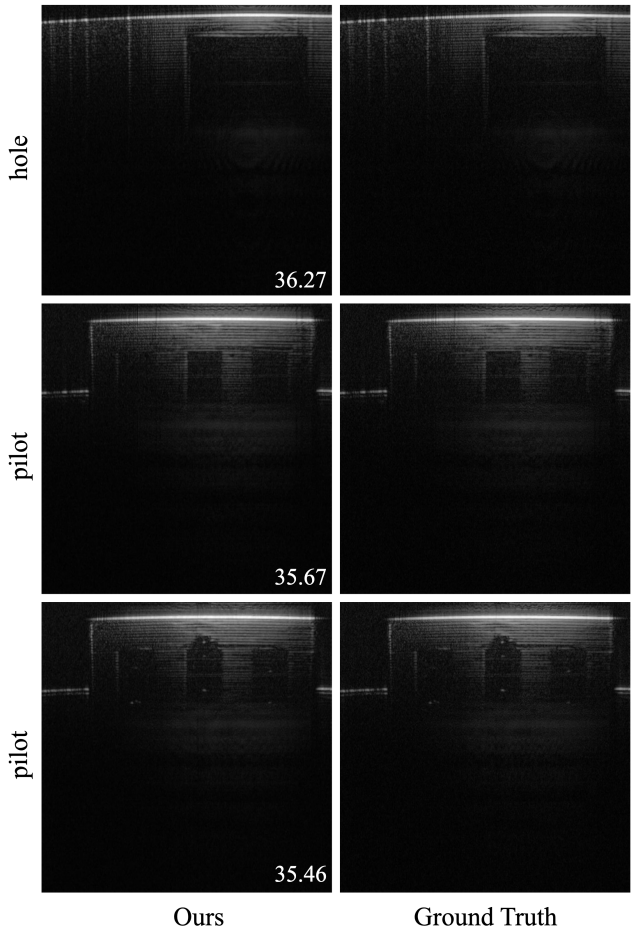

Figure 9. Qualitative results on industrial CeraMIRScan MIR-OCT data, validating GaussianPile's applicability to diverse slice-based imaging modalities.

Generalization to highly anisotropic microscopy and task-driven evaluation. We further evaluate GaussianPile on highly anisotropic serial-section electron microscopy

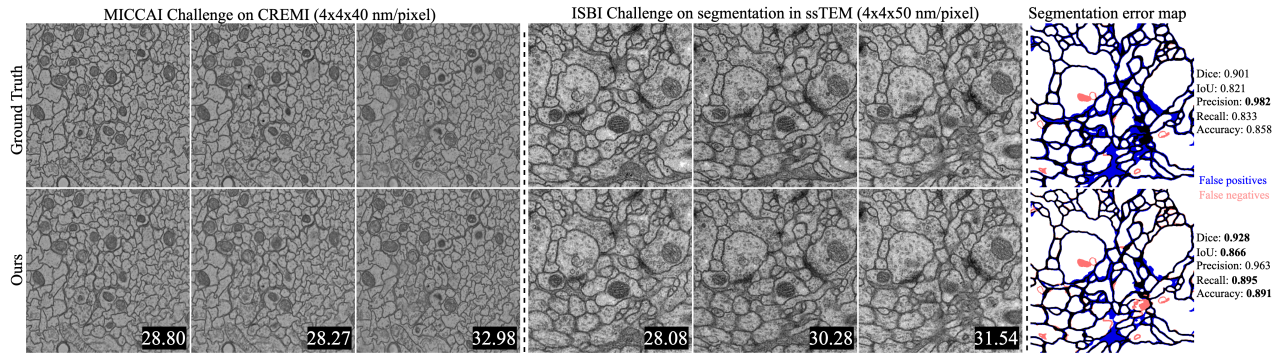

Figure 10. Generalization to highly anisotropic ssEM (CREMI[1], ISBI12 [1], $10\times$ /$12.5\times$ anisotropy respectively). We visualize three consecutive slices for each dataset.

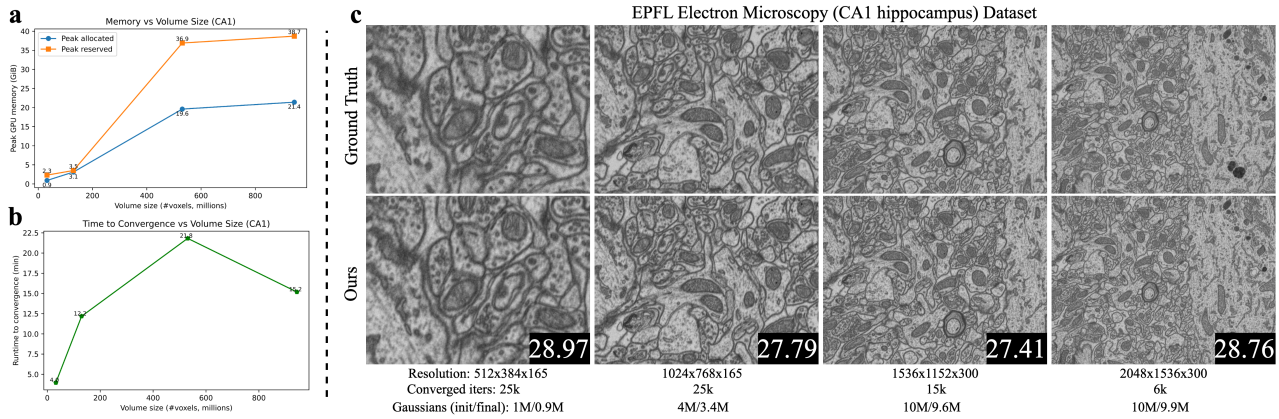

Figure 11. Scalability analysis on the large-scale EPFL CA1 EM dataset [23].

Table 5. Quantitative comparison on highly anisotropic ssEM datasets. We report 2D slice fidelity, 3D volumetric fidelity, and compression/runtime metrics on ISBI12 [1] and CREMI[1].

Method	ISBI12						CREMI					
	2D PSNR↑	2D SSIM↑	3D PSNR↑	3D SSIM↑	CR↑	Time↓	2D PSNR↑	2D SSIM↑	3D PSNR↑	3D SSIM↑	CR↑	Time↓
CoordNet [12]	25.19	0.708	24.98	0.715	3×	51m	22.66	0.620	20.80	0.589	6×	2h34m
NeurComp [22]	24.45	0.695	25.19	0.728	**8**×	**5m**	19.81	0.428	17.62	0.333	**10**×	11h
Ours	**28.79**	**0.815**	**28.98**	**0.881**	**8**×	6m	**29.50**	**0.831**	**28.27**	**0.752**	**10**×	**25m**

(ssEM) datasets, including CREMI[1] and ISBI12 [1], where the axial resolution is much coarser than the in-plane resolution and the imaging process departs substantially from the Gaussian, spatially invariant PSF assumption. As shown in Fig. 10, GaussianPile still preserves coherent slice structures and fine boundaries across consecutive sections, indicating robust generalization under substantial PSF mismatch and severe anisotropy. To complement PSNR/SSIM-style reconstruction metrics, we additionally report segmentation results on ISBI12 using a fixed 2D U-Net in the rightmost column. Segmentations obtained from our reconstructions exhibit improved Dice/IoU and fewer boundary errors, suggesting that GaussianPile better preserves task-relevant structural fidelity beyond voxel-level similarity.

[1] https://cremi.org/

Scalability on large-scale volumes. To further examine scalability, we evaluate GaussianPile on the large-scale EPFL CA1 EM volume, with resolutions up to 0.94B voxels. As shown in Fig. 11, GaussianPile scales to near-billion-voxel data on a single A800 40GB GPU while maintaining stable reconstruction quality. Notably, the memory and runtime costs are governed primarily by the number of optimized Gaussian primitives rather than growing linearly with voxel resolution, highlighting the practicality of the representation for large volumes.

12. More quantitative results

To complement the qualitative generalization results on highly anisotropic ssEM data, we further report quantitative comparisons on ISBI12 [1] and CREMI[1] against represen-

tative INR baselines, including CoordNet [12] and NeurComp [22]. We summarize 2D slice fidelity (PSNR/SSIM), 3D volumetric fidelity (PSNR/SSIM), and compression efficiency/runtime (CR/Time) in Tab. 5. These results further confirm that GaussianPile consistently outperforms the INR baselines on both reconstruction quality and practical efficiency for challenging anisotropic volumes. In particular, the gains on ISBI12 and CREMI support the qualitative observations and the segmentation results shown in Fig. 10.