LAYER-BASED 3D GAUSSIAN SPLATTING FOR SPARSE-VIEW CT RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a dynamic framework for 3D sparse-view Gaussian Splatting that learns scene representations through layerwise, iterative refinement of the Gaussian primitives. Conventional methods typically rely on dense, one-time initialization, where the placement of Gaussians is guided by 2D projection supervision and density control. However, such strategies can lead to misalignment with the true 3D structure, particularly in regions with insufficient projection information due to sparse-view acquisition. In contrast, we adopt a coarse-to-fine approach beginning with a base representation and progressively expanding it by adding new layers of smaller Gaussians to accommodate finer-grained details. At each such iteration, the placement of new primitives is guided by a 3D error map, obtained by the back projection of 2D projections' residuals. This process acts as adaptive importance sampling in 3D space, directing model capacity to regions with high error. We evaluate our approach on sparse-view computed tomography reconstruction tasks, demonstrating improved performance over existing methods.

1 INTRODUCTION

Computed Tomography (CT) is a widely used imaging technology enabling non-destructive inspection of internal structures in various domains, *e.g.*, industrial quality control and medical diagnostics (Kak & Slaney, 2001; Herman, 2009). A key challenge in CT imaging is the trade-off between scan settings and image quality: more intensive scans provide detailed reconstructions but increase system usage and radiation exposure. To address this, recent research has focused on sparse-view CT reconstruction (Shen et al., 2022; Li et al., 2025; Zha et al., 2022; Xie et al., 2025; Cai et al., 2024b; Zha et al., 2024), which aims to perform accurate reconstruction from minimal projection data.

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) was recently introduced as an explicit scene representation for novel view synthesis under natural light conditions. Following its success, this approach is now being adapted for the unique CT properties (Cai et al., 2024a; Zha et al., 2024; Wang et al., 2025) to solve novel view synthesis and reconstruction tasks. A fundamental aspect of 3DGS model is how Gaussians are introduced and positioned within the volumetric space. Current approaches typically rely on dense, one-time initialisation and a density control strategy guided by gradients computed from 2D projection errors. Although this approach effectively reduce 2D projection errors, they provide only indirect and incomplete information about the true 3D structure. As a result, the model tends to overfit the observed views and produce an inaccurate volumetric representation with artifacts in unobserved regions between sparse viewpoints.

In this work, we propose, instead, a 3D error-guided reconstruction approach within a hierarchical, layer-based framework (*cf.* Figure 1). In this formulation, a layer refers to a new set of Gaussians added at a specific stage to address remaining volumetric errors. The process begins by initializing a coarse-grained representation of the scene, composed of large-scale Gaussian primitives that capture the object's basic shape. Subsequent layers then perform iterative error refinement, introducing progressively smaller Gaussians that are strategically placed to resolve finer details and correct inaccuracies in the existing model. The core of our method is a 3D error-driven strategy that complements standard optimization by guiding both the placement of new Gaussians and the fusion of existing ones. To generate the necessary 3D guidance, our method aggregates 2D projection residuals from all views and back-projects them with a fast tomographic solver to construct a 3D error map. This map provides explicit guidance for both densification and sparsification: areas with posi-

Figure 1: Overview of the layer-based approach. The volume is progressively reconstructed by adding and jointly optimizing new layers of Gaussians, each focusing on residual errors left by previous layers. The point clouds depict Gaussian centers, with colors indicating different layers.

tive error indicate under-represented regions for adding new Gaussians, while negative error regions highlight over-represented areas for merging existing Gaussians.

This design is conceptually inspired by the coarse-to-fine approach (Tanimoto & Pavlidis, 1975; Burt & Adelson, 1987), often used in 3D reconstruction tasks (Gu et al., 2020; Yi et al., 2020; Wang et al., 2021; Barron et al., 2021; Yu et al., 2021; Kerbl et al., 2024), and the principle of iterative residual fitting. At its core is the idea that learning a corrective update to an existing representation is a more effective optimization strategy than learning the entire transformation from scratch. This principle powers successful methods across different fields, including ensemble techniques such as Gradient Boosting (Friedman, 2001), and deep learning architectures such as Residual Networks (He et al., 2015). In our method, each layer of Gaussians is placed and optimized to correct the volumetric errors highlighted by the 3D error map generated in the previous layers. This step-by-step refinement reaches superior performance in the sparse view reconstruction.

The main contributions of this work can be summarized as follows: (a) we introduce a hierarchical, layer-based framework for 3DGS CT reconstruction. The scene is refined iteratively, with each new layer of Gaussians correcting volumetric errors identified in the previous stage; (b) we propose a 3D error-driven strategy to guide densification and sparsification. A volumetric error map, generated from back-projected 2D residuals, provides direct structural guidance for adding Gaussians in under-represented regions and fusing them in over-represented regions; (c) we demonstrate, with a series of experimental evaluations, that our method achieves state-of-the-art performance and geometric fidelity on the sparse-view CT reconstruction task.

2 Related Work

2.1 Traditional Reconstruction

The foundational task in computed tomography, specifically for cone-beam CT, is the reconstruction of a volume from its 2D projections, a process framed as the inverse Radon transform (Kak & Slaney, 2001). Traditional methods to solve this problem fall into two main categories: analytical and iterative. The filtered backprojection algorithm FDK (Feldkamp et al., 1984) remains the standard analytical method for cone-beam CT reconstruction due to its computational efficiency and simplicity. Iterative methods, including CGLS (Hestenes & Stiefel, 1952), SART, Andersen & Kak (1984), SART TV Biguri et al. (2016), ASD-POCS (Sidky & Pan, 2008), formulate reconstruction as an optimization problem, aiming to recover a volume that best explains the measured projections. However, traditional methods assume dense and high-quality projections. In sparse-view or noisy settings, they often produce artifacts and noise that reduce reconstruction accuracy.

2.2 Continuous Representation

To address these limitations, recent methods have moved beyond traditional solvers by modeling the volume as a continuous field, using implicit neural representation (Sitzmann et al., 2020) and explicit representation with 3D Gaussian primitives (Kerbl et al., 2023). They follow two main strategies for 3D reconstruction: augmenting the input data via novel view synthesis or directly optimizing a continuous volumetric representation.

One line of work follows a two-stage strategy, where novel view synthesis (NVS) generates images from unseen viewpoints to augment the available projections for 3D reconstruction. Representative methods include neural radiance fields (Mildenhall et al., 2020; Zha et al., 2022; Cai et al., 2024b), modeling the scene as an implicit continuous function parameterized by neural network weights. Recently, 3DGS (Kerbl et al., 2023; Cai et al., 2024a) offers an explicit alternative, representing scenes with a collection of learnable Gaussians. These primitives are projected onto image planes through efficient splatting operations, enabling fast and high-fidelity rendering.

A more direct paradigm bypasses the intermediate NVS step, and instead learns a continuous representation of the volume that is optimized end-to-end from the sparse projections. Methods in this category include implicit neural representations (Zha et al., 2022; Xie et al., 2025; Shen et al., 2022) as well as explicit representations (Zha et al., 2024; Li et al., 2025), which have shown state-of-the-art performance. For example, 3DGR-CT (Li et al., 2025) renders projections by first voxelizing the Gaussian field into a 3D grid and then applying a differentiable CT projector. R2-Gaussian (Zha et al., 2024) employs a custom radiative rasterizer, while using a voxelized grid solely to apply a total variation loss for regularization. Our work builds upon this direct, end-to-end paradigm, but introduces a hierarchical Gaussian representation guided by a 3D error map. This design enables iterative refinement of the volume via targeted updates, improving reconstruction accuracy in the sparse-view settings.

2.3 Hierarchical representation

To improve efficiency and final quality, a common strategy is to structure the representation hierarchically. Prior work has employed multi-resolution feature grids to accelerate training (Müller et al., 2022; Zha et al., 2022), as well as adaptive tree structures such as the octree (Rückert et al., 2022; Martel et al., 2021) to dynamically allocate model capacity. Other hierarchical designs include training pyramids of models for scale-aware rendering (Turki et al., 2023) and organizing Gaussian primitives in a coarse-to-fine manner for large datasets (Kerbl et al., 2024). Similarly, S2Gaussian (Wan et al., 2025) employs a hierarchical, two-stage framework to achieve super-resolution from sparse, low-resolution views. In contrast, our work adopts a different hierarchical strategy: the representation is built progressively through layers of Gaussian primitives, where each layer is guided by a 3D error map to correct the residuals of the previous layers.

3 Methodology

A CT scan is the representation of an object (volume) through its *radiodensity field* $\sigma(\mathbf{x}) : \mathbb{R}^3 \to [0, 1]$ [1], associating with each coordinate $\mathbf{x}$ of the scanned volume an X-ray attenuation value representing the internal structure of the object. The primary goal of CT reconstruction is to recover the volume's radiodensity field leveraging a set of spatially localised 2D projections $\{(\mathbf{I}_v, \Gamma_v)\}_{v=1}^{V}$ as the supervised signal. Each projection $\mathbf{I}_v : \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}_{\geq 0}$ is a 2D X-ray image acquired from a specific viewpoint v. The geometry information Γ_v specifies the acquisition parameters of the cone-beam scanner, including the source-to-detector distance, projection angle, detector pixel size, and more. The task consists, then, of devising an opportune representation of the volume whose projections taken from the same viewpoints coincide with the original projection. Our approach represents the volume leveraging a 3D Gaussian Splatting model, where a set of Gaussian primitives models the tomographic data reconstruction. Instead of training all the Gaussians at once, in our approach, we define a hierarchical representation where the Gaussians are divided into layers. New layers are progressively included and fit to the representation to mitigate the error "unmodeled" by previous ones. In the following, we describe our proposed methodology to perform reconstruction.

3.1 3D Gaussian representation for X-ray imaging

A popular approach to CT reconstruction represents the volume through a collection of Gaussian primitives $\{\mathcal{G}_i\}_{i=1}^{N}$. Each primitive $\mathcal{G}_i$ defines a localized distribution in space, which is geometrically described by a center position $\mu_i \in \mathbb{R}^3$ and a covariance matrix $\mathbf{\Sigma}_i \in \mathbb{R}^{3\times3}_{\succeq 0}$, controlling its

[1] The radiodensity may have a different codomain, based on the application.

shape and orientation. In essence,

$$\mathcal{G}_i(\mathbf{x}; \mu_i, \mathbf{\Sigma}_i) \propto \exp\Big\{ -\frac{1}{2}(\mathbf{x}-\mu_i)^\top \mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mu_i)\Big\}. \tag{1}$$

The radiodensity field $\sigma(\mathbf{x})$ is then modeled as a linear combination of N Gaussian primitives, each scaled by a corresponding central density parameter $\alpha_i \in [0,1]$:

$$\sigma(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \, \mathcal{G}_i(\mathbf{x}; \mu_i, \mathbf{\Sigma}_i) \tag{2}$$

The model is therefore parameterised by $(\alpha_i, \mu_i, \mathbf{\Sigma}_i)_{i=1}^N$. To train these parameters, the 3D radiodensity field must be rendered into 2D projections that can be compared with the ground truth. This rendering process simulates the physics of X-ray imaging, which follows the Beer-Lambert law (Kak & Slaney, 2001). Specifically, the value of each pixel in a projection image corresponds to the line integral of the radiodensity field along the ray traced from the X-ray source to that pixel. For a single ray $\mathbf{r}(t)$, the projected value $\mathcal{I}(\mathbf{r})$ is given by:

$$\mathcal{I}(\mathbf{r}) = \int_{\mathbf{r}} \sigma(\mathbf{x}) \, \mathrm{d}t = \int_{\mathbf{r}} \sum_{i=1}^{N} \alpha_i \, \mathcal{G}_i(\mathbf{x}; \mu_i, \mathbf{\Sigma}_i) \, \mathrm{d}t = \sum_{i=1}^{N} \alpha_i \int_{\mathbf{r}} \mathcal{G}_i(\mathbf{x}; \mu_i, \mathbf{\Sigma}_i) \, \mathrm{d}t. \tag{3}$$

While the line integral defines the physical process, its direct computation is inefficient. Therefore, we employ a differentiable rasterization approach based on the principles of splatting (Zwicker et al., 2002; Kerbl et al., 2023). Specifically, we adopt the rasterization logic from the R2-Gaussian framework (Zha et al., 2024), designed for tomographic reconstruction. In this method, each 3D Gaussian is projected onto the 2D detector plane, and the final pixel values are computed by summing the contributions of these projected 2D Gaussians. This process yields a fully differentiable rendered projection, $\hat{\mathbf{I}}_v$, which can be compared to the ground truth image $\mathbf{I}_v$ for optimization.

3.2 Layer-based approach

Our reconstruction strategy adopts a layered architecture, whose pipeline is illustrated in detail in Figure 2. The process begins by initializing a base layer of Gaussians, $G^{(0)}$, by sampling from an initial volume created with a classical reconstruction method. At each subsequent layer $l \geq 1$, a new set of N_ℓ primitives is introduced and added to the cumulative model from all previous layers, $G^{(l-1)}$. The updated model is formed by the union $G^{(l)} = G^{(l-1)} \cup \{\mathcal{G}_j^{(l)}\}_{j=1}^{N_\ell}$, where the newly added Gaussians, $\{\mathcal{G}_j^{(l)}\}_{j=1}^{N_\ell}$, are strategically placed to correct the residual error left uncorrected by the previously optimized layers in $G^{(l-1)}$.

3D error reconstruction Let $\hat{\mathbf{I}}_v^{(l)}$ denote the rendered projection obtained with the first l layers. We can then quantify the error of such representation as $\mathbf{e}_v^{(l)} = \mathbf{I}_v - \hat{\mathbf{I}}_v^{(l)}$. Notably, instead of directly optimizing the Gaussian positions in the 2D projection space, we use the residual maps $\mathbf{e}_v^{(l)}$ from different views to solve an inverse problem, yielding a 3D volumetric error map $\mathbf{E}^{(l)}(\mathbf{x})$ with the same size as the original volume. The 3D error reconstruction is achieved by solving the linear least-squares problem using the Conjugate Gradient for Least Squares (CGLS) (Hestenes & Stiefel, 1952; Biguri et al., 2016).

Sampling procedure To guide the model's refinement, we first decompose the 3D error map into its positive and negative components. We then independently sample locations from each map using the Gumbel-Max trick (Gumbel, 1958), which correspondingly guide densification and sparsification (*cf.* Figure 2). The sampling is error-guided and explained in detail in Appendix B. Based on the sign of the error at the sampled locations, two complementary updates can be performed. Positive error regions indicate underfitting, where the model lacks sufficient representation. We densify these regions by placing new Gaussian primitives, thereby enriching the scene with finer detail and improving reconstruction fidelity (*cf.* Figure 1). Negative error regions reveal overfitting or redundancy, where too many Gaussians contribute additional value. In these areas, we reduce density through local fusion, merging nearby Gaussians into a single, less dense primitive. This acts as a form of regularization by reallocating capacity and promoting the use of larger Gaussian primitives that generalize better. In the following, we describe both operations in detail.

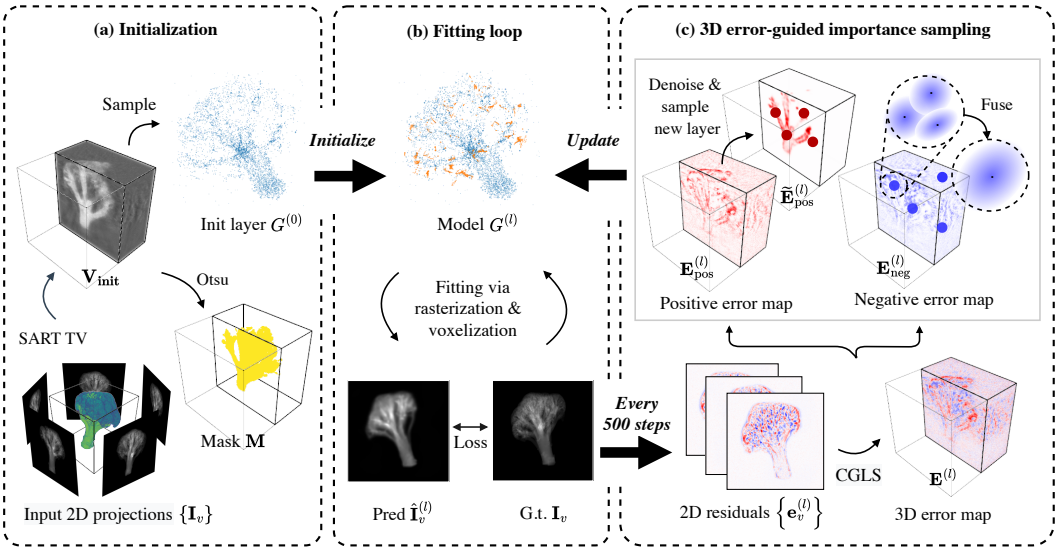

Figure 2: Overview of the layer-based reconstruction pipeline. (a) A classical tomographic reconstruction generates an initial volume to guide the sampling of the first Gaussian layer and to compute an object mask. (b) The iterative loop renders the current set of Gaussians via rasterization and voxelization to produce predicted projection images, which are compared with the ground truth to compute residual maps. (c) These residuals are reconstructed into a 3D volumetric error map, which guides an importance sampling strategy: positive-error regions are sampled for densification (adding a new layer of Gaussians), while negative-error regions are sampled for sparsification (fusing existing Gaussians). The properties of the new Gaussians are adaptively initialized based on the local error magnitude and the current model state.

Layered densification In the case of a location associated with positive error, we place a new Gaussian $\mathcal{G}_i^{(l)}$, whose density $\alpha_i^{(l)}$ is initialized in proportion to the local error $e_i^{(l)} = \mathbf{E}^{(l)}(\mathbf{x}_i)$ at sampled point $\mathbf{x}_i$. However, directly using this error would lead to overestimation in the 2D projection domain, because the contribution of a new primitive accumulates with existing Gaussians across all projection views. To mitigate this, the initial density is normalized by the model's current capacity, approximated by the number of primitives $N^{(l-1)}$. This normalization is motivated by the physical process of the line integral of density, which is simulated by the splatting-based rasterizer. Assuming a quasi-uniform distribution, the number of primitives intersected by a ray scales with $\sqrt[3]{N^{(l-1)}}$. We therefore apply the following scaling, tuned by the hyperparameter C_α, to ensure a stable contribution in the final rendering:

$$\alpha_i^{(l)} = C_\alpha \frac{e_i^{(l)}}{\sqrt[3]{N^{(l-1)}}}, \tag{4}$$

The initial size of new Gaussians is set to a fraction of the average scale of all existing primitives present in the model. This enforces a coarse-to-fine refinement strategy: as the model's average scale naturally decreases with each layer, new primitives are born progressively smaller, ensuring they are dedicated to capturing finer details rather than re-learning the established broad structure.

Layered sparsification In case of a location associated with negative error, we sample a set of points from these regions to serve as fusion centers. At each center, we fuse all Gaussians within a small neighborhood $\mathcal{N}_\varepsilon(i)$ into a single primitive. The density $\alpha_i^{(l)}$ of the fused Gaussian is calculated by summing the densities of the neighbors and then reducing the total by the local error (in absolute value) scaled by the model's capacity, similar to the initialization in Equation 4:

$$\alpha_i^{(l)} = \sum_{j \in \mathcal{N}_\varepsilon(i)} \alpha_j^{(l-1)} - C_\alpha \frac{|e_i^{(l)}|}{\sqrt[3]{N^{(l-1)}}}. \tag{5}$$

Other properties are aggregated based on neighbor density: position and scale are computed via a weighted average, rotation is inherited from the most opaque neighbor. This ensures the fused primitive represents the dominant local structure while avoiding the complexities of rotational averaging.

3.3 TRAINING

Guided by this 3D error map, we add a new layer of Gaussian primitives, $\mathcal{G}^{(l+1)}$, strategically placed within high-error regions. After that, the updated model is jointly optimized to minimize the reconstruction error across all projections using a differentiable projection-domain loss function $\mathcal{L}_{\text{total}}$. Following Zha et al. (2024), we compute the optimization loss and update our model on a per-view basis, progressively selecting them in random order:

Similar to Zha et al. (2024), $\mathcal{L}_{\text{total}}$ comprises a photometric $\mathcal{L}_1$ and a structural fidelity $\mathcal{L}_{\text{SSIM}}$ term. A 3D total variation $\mathcal{L}_{\text{TV}}$ regularization term is also applied to randomly sampled volumetric patches:

$$\mathcal{L}_{\text{total}}(\mathbf{I}_v, \hat{\mathbf{I}}_v^{(l)}) = \mathcal{L}_1(\mathbf{I}_v, \hat{\mathbf{I}}_v^{(l)}) + \lambda_{\text{SSIM}}\, \mathcal{L}_{\text{SSIM}}(\mathbf{I}_v, \hat{\mathbf{I}}_v^{(l)}) + \lambda_{\text{TV}}\, \mathcal{L}_{\text{TV}}(\boldsymbol{X}_p; G^{(l)}), \tag{6}$$

where λ_{SSIM} and λ_{TV} are two hyperparameters, and $\boldsymbol{X}_p$ is a 3D patch randomly sampled from the current model $G^{(l)}$ used to compute the total variation loss.

4 EXPERIMENTS

In this section, we describe the experimental setup used to assess the performance of our method and discuss the results we have obtained. Additionally, we conduct several ablation studies to gain additional insight into the motivation behind the reported performance.

4.1 EXPERIMENTAL SETTINGS

Dataset Following (Cai et al., 2024a; Zha et al., 2024), we conduct experiments on both synthetic and real-world datasets representing diverse object types. The synthetic dataset includes fifteen 3D volumes categorized into three classes: medical, food, and everyday objects. We use the TIGRE (Biguri et al., 2016) toolbox to generate cone-beam X-ray projections, simulating realistic imaging conditions by incorporating Compton scatter and electronic noise. The real-world dataset includes scans of walnut, seashell, and pine. Similarly, we treat the fully reconstructed high-resolution volumes of these objects as ground truth and simulate a sparse-view acquisition using the same projection pipeline as the synthetic set.

Implementation details We implement our method in PyTorch with CUDA acceleration [2]. All experiments are conducted on an NVIDIA H200 GPU. As initialization step, we reconstruct a coarse volume using the SART-TV algorithm (Biguri et al., 2016). A binary object soft mask is then calculated using Otsu thresholding (Otsu, 1979). Next, we construct the Gaussian field using a 20-layer hierarchical strategy, where a new layer of 2500 Gaussians is introduced every 500 iterations, guided by the reconstructed 3D error map. The positive error map is first denoised using the object mask and a 3D Gaussian blur with a standard deviation of $\sigma = 2$ to yield an importance map. From this map, we sample locations for new Gaussians using the Gumbel-max trick with temperature $\tau = 5 \times 10^{-3}$. To progressively capture finer details, these new Gaussians are initialized with a reduced scale (half the current average) and a density scaled by $C_\alpha = 0.5$ (Equations (4–5)). Concurrently, the negative error map guides our layered sparsification. We sample 30K fusion centers and aggregate nearby Gaussians within a radius of $\epsilon = 0.05$ into a single, less dense primitive. This layer-building phase continues until all layers are placed, with all existing Gaussians being jointly optimized. Afterwards, vanilla density control (Kerbl et al., 2023) is enabled for a final fine-tuning stage. Across all experiments, the total optimization runs for 30K iterations. To quantitatively assess reconstruction quality, we use the 3D PSNR and SSIM metrics, following the implementation provided in R2-Gaussian (Zha et al., 2024). More details can be found in Appendices A and B.

[2] Our code is available at the anonymous GitHub repository

Table 1: Quantitative comparison on the 3D reconstruction task across 5, 10, 15, and 25 views settings. The reported metrics are computed over the full volumes and averaged across all scans. We apply colors to the first, second, and third ranked numbers.

Methods	5 views			10 views			15 views			25 views		
	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓
Real dataset												
FDK	14.71	0.066	–	17.77	0.106	–	19.34	0.138	–	23.30	0.335	–
CGLS	24.57	0.546	0.7s	26.21	0.585	0.7s	27.18	0.611	0.8s	28.26	0.673	0.9s
SART	25.84	0.648	6.1s	28.21	0.696	11.0s	29.61	0.722	16.0s	31.52	0.790	26.0s
SART TV	26.65	0.720	33.0s	29.68	0.795	54.7s	31.24	0.829	1.3m	32.89	0.836	2.1m
ASD-POCS	26.65	0.711	43.6s	29.84	0.788	51.6s	31.60	0.827	52.2s	32.38	0.826	2.0m
NAF	27.94	0.802	1.3m	32.47	0.859	2.5m	33.90	0.876	3.7m	32.76	0.783	7.2m
X-Gaussian	20.72	0.639	3.1m	20.73	0.637	3.0m	20.73	0.638	2.9m	20.72	0.636	5.4m
R2-Gaussian	27.24	0.715	4.3m	31.90	0.812	4.7m	34.40	0.854	4.9m	35.52	0.843	7.7m
Ours	28.58	0.825	5.7m	33.04	0.890	6.8m	34.62	0.903	7.3m	36.48	0.851	8.8m
Synthetic dataset												
FDK	12.66	0.045	–	15.26	0.068	–	16.81	0.090	–	22.99	0.317	–
CGLS	22.79	0.482	0.7s	24.64	0.512	0.7s	25.61	0.535	0.8s	27.99	0.664	0.9s
SART	24.10	0.638	5.7s	26.31	0.669	10.7s	27.58	0.683	15.5s	31.14	0.825	25.4s
SART TV	24.88	0.709	31.6s	27.70	0.766	55.6s	29.20	0.795	1.3m	31.48	0.864	2.3m
ASD-POCS	24.98	0.725	42.4s	27.91	0.779	45.9s	29.52	0.806	47.7s	33.92	0.907	1.6m
NAF	25.11	0.724	1.2m	28.29	0.781	2.4m	29.82	0.804	3.6m	33.48	0.893	6.1m
X-Gaussian	17.45	0.620	3.5m	17.46	0.620	3.3m	17.46	0.620	3.1m	17.46	0.620	3.7m
R2-Gaussian	23.48	0.670	9.8m	27.00	0.759	8.4m	29.60	0.813	7.6m	35.39	0.926	5.7m
Ours	25.75	0.795	6.0m	29.45	0.858	6.8m	31.16	0.882	7.3m	33.45	0.912	7.9m

Baselines We benchmark our method against traditional reconstruction techniques, as well as recent implicit and explicit representation techniques. This comparison is limited to self-supervised approaches, as our work focuses on single-scene reconstruction, where supervised methods requiring external datasets are not applicable. As classical baselines, we use the analytical FDK algorithm (Feldkamp et al., 1984) and several iterative methods. While many variations of iterative reconstruction algorithms exist, we focus on a few representative ones: CGLS (Hestenes & Stiefel, 1952), SART (Andersen & Kak, 1984), SART-TV (Biguri et al., 2016), and ASD-POCS (Sidky & Pan, 2008). We tune the number of iterations for each classical algorithm to achieve optimal results across all datasets. Alongside traditional approaches, we evaluate recent implicit and explicit reconstruction methods. For implicit reconstruction, we use NAF (Zha et al., 2022), which models voxel-wise attenuation coefficients as continuous neural fields. For explicit reconstruction, we consider X-Gaussians (Cai et al., 2024a) and R2-Gaussian (Zha et al., 2024) models. Since X-Gaussians is designed for novel view synthesis, we first generate intermediate projections and then apply the classical reconstruction algorithm ASD-POCS to obtain the volume. R-Gaussian, in contrast, reconstructs the volume directly using voxelization.

4.2 Baseline comparison

The performance of our layer-based method is evaluated both quantitatively and qualitatively. Table 1 reports the volumetric 3D PSNR and SSIM metrics for various sparse-view configurations, where our method consistently demonstrates improved performance. More detailed results are provided in Appendix C. These numerical gains are visually supported in Figure 3. The qualitative comparison shows that our layer-based Gaussian model achieves higher fidelity, producing cleaner transitions between neighboring views and preserving finer details.

Analysis of reconstruction accuracy Our method demonstrates its main strengths in sparse-view settings (*e.g.*, 5-15 views), while comparable in more densely-sampled scenarios. This advantage stems from our progressive, coarse-to-fine strategy, which acts as a regularizer against overfitting.

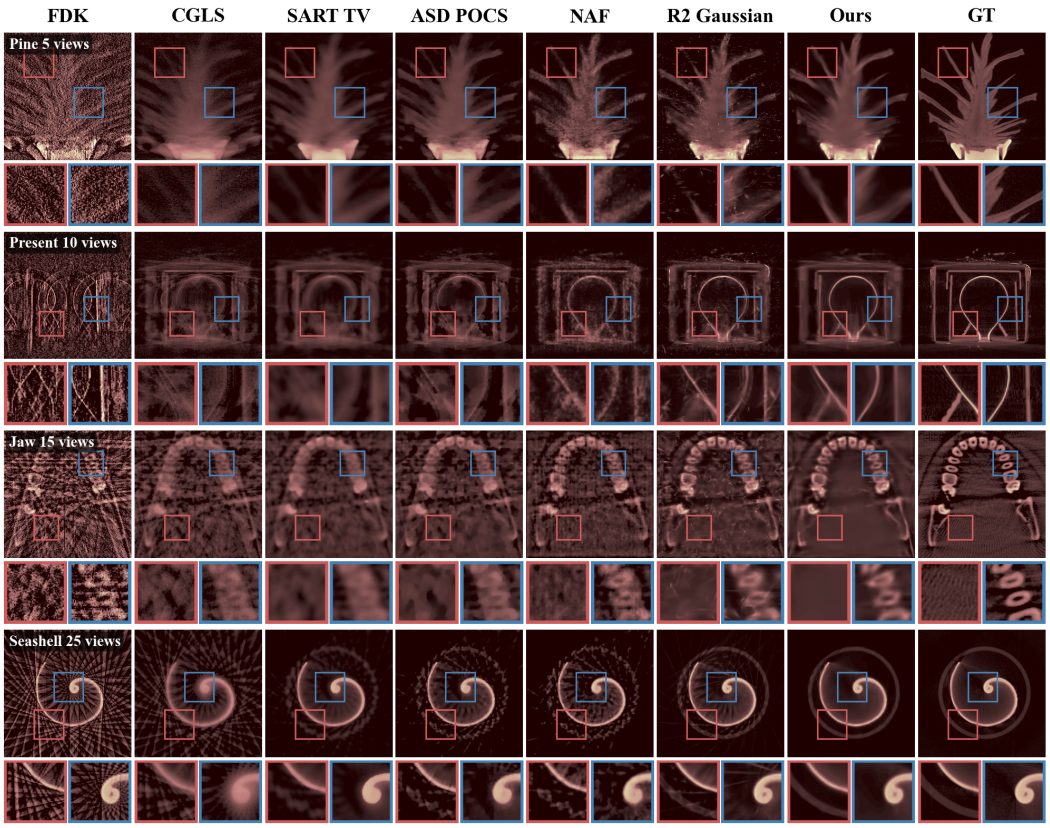

Figure 3: Qualitative evaluation of reconstruction algorithms under varying degrees of data sparsity. Our coarse-to-fine approach uses a 3D volumetric error map to strategically allocate model capacity, focusing refinement only on regions with high structural error. This targeted strategy avoids overfitting to sparse views, resulting in a cleaner and more geometrically accurate reconstruction. Zoomed-in patches are provided for a clearer inspection of reconstruction quality.

The initial layers establish a robust, low-frequency structure using large Gaussians, capturing the object's general form. Subsequent refinements are guided by the 3D volumetric error map, which strategically places smaller, lower-density Gaussians to correct true volumetric inaccuracies. This ensures better geometric fidelity and generalization to in-between views. Conversely, in more data-rich settings (*e.g.*, 25 views), the large number of projections provides more reliable signals from all directions. This is often sufficient to effectively guide a standard optimization of all Gaussians at once. In these cases, the implicit regularization from our layered approach is less critical.

Analysis of runtime Our method's overall timing is comparable with state-of-the-art approaches. Our process introduces extra computations from the initial SART-TV reconstruction and the periodic layered densification and sparsification. However, these costs are effectively balanced by the efficiency gains from our layered fitting strategy. Instead of optimizing all N Gaussians from the start, our method incrementally builds the scene in L stages. For a significant portion of the training, we operate on a much smaller subset of primitives, each time including N/L Gaussians. Furthermore, our layered sparsification step prunes redundant Gaussians. The combined effect produces a more compact representation and comparable fitting time, as empirically validated in Table 3.

4.3 Ablation Study

Layered densification We investigate how reconstruction quality is affected by the choice in the placement of new Gaussian layers. Specifically, we focus on the hierarchical depth, defined by the number of layers, and (2) the spatial density, defined by the initial number of Gaussians per layer.

We set a total of 50K primitives to be introduced during the layer-building phase. In a multi-layer setup with L layers, primitives are added incrementally in batches of $50\text{K}/L$ per layer. This is compared against a single-layer baseline where all 50K primitives are present from the start. Table 3 presents results for different architectures of layered Gaussian model. We report 3D metrics for reconstruction quality, the total number of Gaussians after 30K optimization steps, and training time. Results show that generally multi-layered approaches outperform the single-layer baseline. Moreover, multi-layer design achieves this performance with less number of primitives and, therefore, less training time, with the 20 layer configuration achieving the best results across all views. Figure 4 provides visual comparison of multi-layer and one-layer approaches.

Layered sparsification We analyze the key hyperparameters of our sparsification mechanism, the number of sampled fusion centers and the fusion radius, which together control the degree of structural regularization. This volumetric, error-guided fusion is distinct from standard density-based pruning. Its goal is to manage model complexity by correcting for over-represented regions identified in the 3D error map. An ablation study was performed to identify the optimal configuration that maximizes reconstruction accuracy, as detailed in Table 4, revealing the need to balance between overfitting from insufficient fusion and over-smoothing from an overly aggressive approach.

Masking We analyze the impact of applying the soft Otsu mask during our error-guided densification. As shown in Table 5, the mask provides a spatial prior that yields improvements in both reconstruction quality and model compactness. Without this prior, the model tends to place Gaussians in empty space to minimize 2D projection errors, a form of overfitting that degrades the 3D structure and inflates model size. By constraining densification to the object's volume, the mask ensures model capacity is used to refine true geometric details. This results in superior 3D metrics and a more regularized model with substantially fewer Gaussians.

Layer selection We explore training strategies to mimic a boosting-like approach: train only the newest layer, train the last few layers, and probabilistically select layers, either as a contiguous chain or independently. Although these methods can reduce computation, optimizing all layers consistently produces the best results as shown in Table 6.

5 Discussion

Our framework demonstrates a robust approach to sparse-view reconstruction, yet it leaves room for future exploration. First, the accuracy of the guiding 3D error map is tied to the quality of the back-projection solver. In highly sparse scenarios, this map can become noisy, especially after many layers, potentially leading to the placement of new primitives that capture noise artifacts instead of true structural errors. We mitigate this by denoising the error map with soft object mask and 3D Gaussian blur, but more advanced techniques could be explored in the future. Second, a promising direction is to move beyond a fixed number of primitives per layer towards an adaptive strategy where the number and properties of new Gaussians are determined theoretically by the local error distribution. Third, beyond CT data, this concept may generalize to other tomographic modalities, highlighting the broader relevance of explicit error-guided reconstruction. Finally, the core principle of our method is fundamentally representation-agnostic. For example, the 3D error map can guide importance sampling of training coordinates for an implicit neural representation, directing the network's capacity toward high-error regions and mirroring the coarse-to-fine refinement strategy.

6 Conclusion

In this work, we introduced a hierarchical, layer-based coarse-to-fine framework for sparse-view CT reconstruction leveraging a 3D error map to guide the iterative refinement of a 3D Gaussian representation. Our densification and sparsification strategy allocates model capacity more effectively by directly addressing volumetric inaccuracies. This mitigates a key problem in baseline methods, often overfitting to the training projections when initialized with a dense set of primitives. As shown in our experiments, our approach yields reconstructions with superior geometric fidelity, particularly in highly sparse settings. Therefore, the principle of explicit 3D error correction offers a promising path towards more robust and reliable CT reconstruction in data-limited scenarios.

REFERENCES

Anders H. Andersen and Avinash C. Kak. Simultaneous algebraic reconstruction technique (sart): a superior implementation of the art algorithm. *Ultrasonic Imaging*, 6(1):81–94, 1984.

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pp. 5835–5844. IEEE, 2021.

Ander Biguri, Manjit Dosanjh, Steven Hancock, and Manuchehr Soleimani. Tigre: a matlab-gpu toolbox for cbct image reconstruction. *Biomedical Physics & Engineering Express*, 2(5):055010, 2016.

Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pp. 671–679. Elsevier, 1987.

Yuanhao Cai, Yixun Liang, Jiahao Wang, Angtian Wang, Yulun Zhang, Xiaokang Yang, Zongwei Zhou, and Alan Yuille. Radiative gaussian splatting for efficient x-ray novel view synthesis. In *ECCV*, 2024a.

Yuanhao Cai, Jiahao Wang, Alan Yuille, Zongwei Zhou, and Angtian Wang. Structure-aware sparse-view x-ray 3d reconstruction. In *CVPR*, 2024b.

Lee A. Feldkamp, Lloyd C. Davis, and James W. Kress. Practical cone-beam algorithm. *Journal of the Optical Society of America*, 1(6):612–619, 1984.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001.

Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, pp. 2492–2501. Computer Vision Foundation / IEEE, 2020.

E. J. Gumbel. *Statistics of Extremes*. Columbia University Press, 1958.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Gabor T Herman. *Fundamentals of computerized tomography: image reconstruction from projections*. Springer Science & Business Media, 2009.

Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952.

Avinash C Kak and Malcolm Slaney. *Principles of computerized tomographic imaging*. Society for Industrial and Applied Mathematics, 2001.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics*, 43(4), 2024.

Yingtai Li, Xueming Fu, Han Li, Shang Zhao, Ruiyang Jin, and S. Kevin Zhou. 3dgr-ct: Sparse-view ct reconstruction with a 3d gaussian representation. *Medical Image Analysis*, 103:103585, 2025.

Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: adaptive coordinate networks for neural scene representation. *ACM Transactions on Graphics*, 40(4), 2021.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4), 2022.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

Darius Rückert, Yuanhao Wang, Rui Li, Ramzi Idoughi, and Wolfgang Heidrich. Neat: neural adaptive tomography. *ACM Transactions on Graphics*, 41(4), 2022.

Liyue Shen, John Pauly, and Lei Xing. Nerp: implicit neural representation learning with prior embedding for sparsely sampled image reconstruction. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Emil Y Sidky and Xiaochuan Pan. Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization. *Physics in Medicine & Biology*, 53(17):4777, 2008.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 33:7462–7473, 2020.

Steven Tanimoto and Theo Pavlidis. A hierarchical data structure for picture processing. *Computer graphics and image processing*, 4(2):104–119, 1975.

Haithem Turki, Michael Zollhöfer, Christian Richardt, and Deva Ramanan. Pynerf: Pyramidal neural radiance fields. In *NeurIPS*, 2023.

Yecong Wan, Yuanshuo Cheng, Mingwen Shao, and Wangmeng Zuo. S2gaussian: Sparse-view super-resolution 3d gaussian splatting. *CVPR*, 2025.

Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *CVPR*, pp. 14194–14203. Computer Vision Foundation / IEEE, 2021.

Feiran Wang, Jiachen Tao, Junyi Wu, Haoxuan Wang, Bin Duan, Kai Wang, Zongxin Yang, and Yan Yan. X-field: A physically grounded representation for 3d x-ray reconstruction. *arXiv preprint arXiv:2503.08596*, 2025.

Wangduo Xie, Richard Schoonhoven, Tristan Van Leeuwen, and Matthew B. Blaschko. Ac-ind: Sparse ct reconstruction based on attenuation coefficient estimation and implicit neural distribution. In *WACV*, 2025.

Hongwei Yi, Zizhuang Wei, Mingyu Ding, Runze Zhang, Yisong Chen, Guoping Wang, and Yu-Wing Tai. Pyramid multi-view stereo net with self-adaptive view aggregation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *ECCV*, volume 12354 of *Lecture Notes in Computer Science*, pp. 766–782. Springer, 2020.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, pp. 5732–5741. IEEE, 2021.

Ruyi Zha, Yanhao Zhang, and Hongdong Li. Naf: Neural attenuation fields for sparse-view cbct reconstruction. In *MICCAI*, 2022.

Ruyi Zha, Tao Jun Lin, Yuanhao Cai, Jiwen Cao, Yanhao Zhang, and Hongdong Li. R^2-gaussian: Rectifying radiative gaussian splatting for tomographic reconstruction. In *NeurIPS*, 2024.

Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. EWA Splatting . *IEEE Transactions on Visualization & Computer Graphics*, 8(03), 2002.

A MODEL INITIALIZATION

A.1 INITIAL RECONSTRUCTION

For the initial approximation of the object volume, we employ the Simultaneous Algebraic Reconstruction Technique with Total Variation regularization (SART-TV) using the TIGRE toolkit (Biguri et al., 2016). Other approaches include uniform distribution initialization (Cai et al., 2024a), reconstruction with FDK (Zha et al., 2024), and mixed methods (Wang et al., 2025). We select SART-TV for its superior ability to produce a high-fidelity volume with well-defined edges and reduced artifacts for sampling the first layer and calculating the object mask.

A.2 FIRST LAYER INITIALIZATION

To maintain a consistent sampling methodology throughout our layer-based framework, the initialization of the first layer of Gaussians, $G^{(0)}$, also employs an importance sampling strategy based on the Gumbel-Max trick. Unlike subsequent layers, which are guided by a 3D error map, the first layer is guided by the initial volumetric reconstruction produced by the SART-TV solver. This initial volume serves as a coarse density map, where voxel intensities represent the probability of belonging to the object. Although our method is more robust to initialization due to its iterative, error-correcting design, this starting estimate is still important for generating the object mask via Otsu thresholding used in subsequent denoising steps.

A.3 OTSU SOFT MASKING

The mask is computed once at the beginning of the pipeline. To separate the object from the background, we apply Otsu's thresholding (Otsu, 1979) to determine the optimal binary threshold t^*:

$$t^* = \arg\max_t \text{Var}_b(t), \tag{7}$$

where $\text{Var}_b(t)$ denotes the between-class variance for a given threshold t.

In a sparse-view setting, the initial volume is prone to artifacts, which makes a traditional binary mask created by hard thresholding unreliable. Such a mask would incorrectly classify uncertain regions, potentially removing parts of the object or leaving background noise unfiltered. To mitigate these issues, we replace the binary mask with a probabilistic *soft* mask. First, for each voxel $\mathbf{x}$, we calculate a normalized and scaled distance $d(\mathbf{x})$ from the Otsu threshold t^*:

$$d(\mathbf{x}) = \beta \frac{\mathbf{V}_{\text{init}}(\mathbf{x}) - t^*}{\sigma_V}, \tag{8}$$

where $\beta > 0$ is a steepness parameter and σ_V is the standard deviation of the volume's intensities. This step quantifies how far each voxel is from the decision boundary.

Second, we apply the logistic sigmoid function to map this distance into a probabilistic value:

$$\mathbf{M}(\mathbf{x}) = \frac{1}{1 + \exp(-d(\mathbf{x}))}. \tag{9}$$

This formulation produces a smooth mask with values in $[0, 1]$, representing the probability of a voxel belonging to the object. This method preserves uncertain boundary regions and provides a more reliable guide for subsequent processing steps.

B SAMPLING PROCEDURE

To guide the placement of new Gaussians (densification) and the fusion of existing ones (sparsification), we require a robust method for sampling locations from the 3D error map, $\mathbf{E}^{(l)}$. Our procedure begins by decomposing this map into its positive and negative components, $\mathbf{E}^{(l)}_{\text{pos}}$ and $\mathbf{E}^{(l)}_{\text{neg}}$, which are sampled independently. The positive map, which guides densification, undergoes a denoising step prior to sampling to ensure new primitives are placed in regions of coherent error rather than noise. The negative map is sampled directly to identify candidates for fusion. For both maps, we employ an error-guided importance sampling strategy.

B.1 Error-guided Gumbel sampling

To sample positions from either the positive or negative error maps, we use the Gumbel-Max trick (Gumbel, 1958). This technique allows for efficient importance sampling from a discrete distribution defined by unnormalized scores. This is particularly beneficial in large 3D volumes where computing a partition function would be computationally expensive.

Let $e_i^{(l)} = \mathbf{E}^{(l)}(\mathbf{x}_i)$ denote the absolute error value at position $\mathbf{x}_i$, and let τ be a temperature parameter controlling the stochasticity of the process. We generate Gumbel noise g_i from the standard Gumbel distribution $g_i = -\log(-\log(u_i))$, where $u_i \sim \text{Uniform}(0, 1)$. The final score for position $\mathbf{x}_i$ is computed as $s_i^{(l)} = |e_i^{(l)}|/\tau + g_i$. We then select the top-k highest-scoring indices $i_1, \ldots, i_k$ to form the set of sampled locations for layer l.

B.2 Denoising of the error map

The error volume reconstructed from sparse-view data, $\mathbf{E}^{(l)}$, often contains artifacts such as streaks and noise. To ensure that new Gaussians are placed in regions of meaningful error rather than artifacts, we apply a two-stage denoising process to the positive component, $\mathbf{E}_{\text{pos}}^{(l)}$, before sampling. First, we apply a probabilistic soft mask, $\mathbf{M}(\mathbf{x})$, via a Hadamard product: $\widetilde{\mathbf{E}}_{\text{pos}}^{(l)} = \mathbf{M} \odot \mathbf{E}_{\text{pos}}^{(l)}$. The generation of this mask is detailed in Appendix A.3. Second, the masked volume is smoothed with a Gaussian blur. This step suppresses high-frequency noise and enhances spatial coherence, yielding a robust importance map that guides the subsequent Gumbel sampling for densification.

C Per-scene comparison

We compare our layer-based approach with the R2-Gaussian model in Table 2. To ensure a fair comparison on the sparse-view datasets, we address overfitting issue in standard R2-Gaussian model (Zha et al., 2024). R2-Gaussian model trained with its original parameters tends to overfit to sparse views, resulting in severe needle-like artifacts, especially in settings with very few input images. To mitigate this effect and establish a stronger baseline, we adjusted parameters in favor to the sparse-view setting. Specifically, we (1) increased Total Variation regularization with a weight of $\lambda_{\text{TV}} = 0.5$ to encourage smoother geometry; (2) increasing the minimum allowed Gaussian scale to 0.005 to prevent overly thin structures; and (3) increased the densification gradient threshold to 0.001 to reduce excessive splitting and cloning. Collectively, these changes regularize the model and improve robustness under sparse-view conditions. However, they introduce a trade-off: a less-regularized model achieves higher fidelity in higher-view settings (e.g., 25 views), while the more-regularized model tends to oversmooth results and lowers the metrics. Despite this, our layer-based strategy achieves better results.

D Ablations

D.1 Layered densification

In Table 3, we present an ablation on the number of layers across different sparse-view settings. We report 3D PSNR, 3D SSIM, the number of Gaussians, and training time. Multi-layer architectures generally outperform the single-layer baseline while using fewer primitives and less training time. A configuration with 20 layers results in the best trade-off across all view settings.

D.2 Layered sparsification

In Table 4, we present an ablation on sparsification hyperparameters, fusion radius, and the number of sampled fusion centers, comparing performance on real and synthetic datasets across different sparse-view settings. We report 3D PSNR, 3D SSIM, the number of Gaussians, and training time. The best parameters are highlighted. In Figure 4, we additionally include a visual comparison between the 1-layer and 20-layer models.

Table 2: 3D PSNR comparison between R2-Gaussian and our method across different numbers of sparse views on synthetic and real datasets. Gray-colored numbers indicate R2-Gaussian metrics obtained with a set of parameters optimized for the sparse-view setting.

Scene	5 views		10 views		15 views		25 views	
	R2-Gaussian	Ours	R2-Gaussian	Ours	R2-Gaussian	Ours	R2-Gaussian	Ours
Real dataset								
Pine	29.93 / 31.39	31.97	33.94 / 35.02	35.65	36.52 / 36.42	37.09	38.10 / 37.67	38.25
Seashell	29.00 / 28.80	30.63	34.99 / 33.96	36.24	37.81 / 35.46	37.84	39.53 / 38.34	40.86
Walnut	22.79 / 23.04	23.15	26.76 / 26.68	27.24	28.87 / 28.15	28.93	28.94 / 30.04	30.34
Average	27.24 / 27.74	28.58	31.90 / 31.89	33.04	34.40 / 33.34	34.62	35.52 / 35.35	36.48
Synthetics dataset								
Chest	19.39 / 22.51	23.01	22.58 / 26.30	26.50	26.53 / 27.81	28.01	32.18 / 29.88	30.60
Foot	22.63 / 24.64	24.67	25.91 / 27.13	27.52	27.78 / 28.71	29.08	30.38 / 29.91	30.20
Head	24.04 / 26.57	26.99	28.79 / 30.69	31.14	30.77 / 31.80	32.27	36.86 / 34.72	35.16
Jaw	24.45 / 24.53	25.14	27.20 / 27.31	28.56	29.31 / 29.10	30.48	33.35 / 31.85	32.63
Pancreas	22.01 / 25.30	25.47	25.61 / 27.11	27.36	28.43 / 28.80	29.01	33.08 / 30.51	31.43
Beetle	32.52 / 32.93	33.01	34.98 / 33.98	34.42	37.39 / 35.05	35.76	40.09 / 35.57	36.37
Bonsai	21.68 / 24.98	25.10	23.78 / 28.34	28.64	26.18 / 29.77	30.03	33.06 / 31.67	32.16
Broccoli	18.37 / 19.94	20.01	20.95 / 22.40	22.53	23.13 / 24.75	24.89	29.25 / 27.41	27.85
Kingsnake	31.80 / 33.73	34.09	35.33 / 36.08	36.55	36.19 / 36.31	36.67	39.03 / 36.87	37.35
Pepper	16.25 / 17.98	18.36	20.16 / 26.12	26.14	24.21 / 28.42	28.74	35.08 / 31.58	33.11
Backpack	26.60 / 27.84	28.25	29.13 / 29.13	29.75	31.07 / 29.91	30.53	34.97 / 30.68	31.68
Engine	17.65 / 20.22	20.38	22.27 / 24.37	24.82	27.10 / 28.57	29.44	35.23 / 32.08	33.32
Mount	19.99 / 24.88	24.95	21.58 / 30.44	30.85	25.00 / 32.27	33.00	37.39 / 34.99	36.01
Present	26.10 / 26.35	26.94	28.73 / 28.10	28.94	30.26 / 28.84	29.79	35.04 / 30.64	31.99
Teapot	28.77 / 29.17	29.80	37.91 / 37.46	38.09	40.70 / 39.18	39.70	45.81 / 41.83	41.94
Average	23.48 / 25.44	25.75	27.00 / 29.00	29.45	29.60 / 30.62	31.16	35.39 / 32.68	33.45

Table 3: Ablation on the number of layers (L) across different sparse-view settings. Multi-layer architectures generally outperform the single-layer baseline while using fewer primitives (N) and less training time. $L = 20$ results in the best trade-off across all view settings.

		Real Dataset				Synthetic Dataset			
	L	PSNR↑	SSIM↑	N↓	Time↓	PSNR↑	SSIM↑	N↓	Time↓
5 views	1	27.68	0.773	50K	8.1m	25.46	0.765	66K	8.7m
	5	28.07	0.794	23K	6.1m	25.58	0.778	54K	7.7m
	10	28.18	0.801	19K	5.5m	25.61	0.781	39K	7.0m
	20	28.34	0.806	17K	5.1m	25.68	0.786	31K	6.3m
	30	28.37	0.809	15K	4.9m	25.73	0.788	28K	5.8m
10 views	1	31.79	0.849	50K	8.5m	28.97	0.839	56K	9.2m
	5	32.33	0.866	32K	7.0m	29.16	0.849	45K	8.1m
	10	32.47	0.872	26K	6.4m	29.20	0.850	40K	7.5m
	20	32.68	0.880	22K	6.1m	29.30	0.852	37K	7.0m
	30	32.40	0.875	19K	6.0m	29.24	0.848	33K	6.7m
15 views	1	33.62	0.880	50K	8.6m	30.76	0.870	54K	9.3m
	5	34.10	0.894	37K	7.5m	30.91	0.877	47K	8.3m
	10	34.21	0.898	31K	7.1m	30.93	0.878	43K	7.8m
	20	34.29	0.900	27K	6.6m	30.99	0.878	41K	7.4m
	30	33.84	0.891	24K	6.5m	30.87	0.874	37K	7.0m
25 views	1	36.23	0.862	55K	9.6m	33.34	0.909	54K	7.3m
	5	36.46	0.856	52K	8.1m	33.42	0.912	49K	6.5m
	10	36.49	0.855	48K	7.5m	33.45	0.913	45K	6.1m
	20	36.38	0.853	47K	6.8m	33.46	0.913	42K	5.6m
	30	33.95	0.812	32K	6.0m	32.95	0.904	35K	5.3m

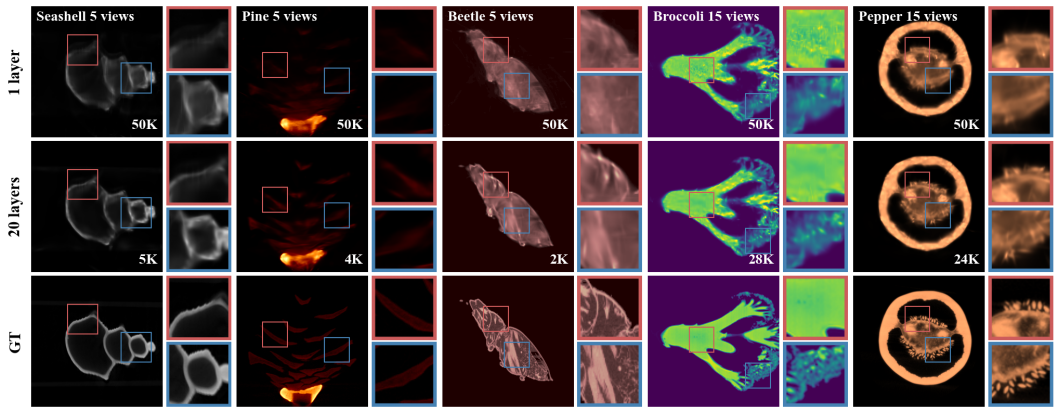

Figure 4: Comparison between 1- & 20-layer models. The total number of Gaussians is shown in the bottom-right corner.

Table 4: Ablation study on sparsification hyperparameters across different sparse-view settings. We vary the fusion radius (r) and the number of sampled fusion centers (k). A fusion radius of $r = 0.05$ and the number of sampled fusion centers $k = 30$K provide the best trade-off between model compactness and fidelity.

			Real Dataset				Synthetic Dataset			
	r	k	PSNR↑	SSIM↑	N↓	Time↓	PSNR↑	SSIM↑	N↓	Time↓
5 views	0.03	1K	28.39	0.807	17K	5.5m	25.68	0.787	31K	6.8m
	0.03	10K	28.42	0.808	13K	5.6m	25.67	0.788	26K	6.6m
	0.03	30K	28.52	0.813	9K	5.6m	25.74	0.791	19K	6.5m
	0.05	10K	28.52	0.815	9K	5.4m	25.70	0.791	18K	6.1m
	0.05	30K	28.58	0.825	6K	5.7m	25.75	0.795	11K	6.0m
	0.10	10K	28.49	0.820	6K	5.1m	25.71	0.794	11K	5.6m
	0.10	30K	28.48	0.811	6K	5.4m	25.71	0.795	8K	6.2m
10 views	0.03	1K	32.68	0.880	22K	6.1m	29.30	0.852	37K	7.0m
	0.03	10K	32.75	0.881	17K	6.2m	29.32	0.853	29K	6.9m
	0.03	30K	32.85	0.882	12K	6.5m	29.37	0.855	21K	7.0m
	0.05	10K	32.95	0.885	11K	6.2m	29.34	0.855	20K	6.7m
	0.05	30K	33.04	0.890	9K	6.8m	29.45	0.858	14K	6.8m
	0.10	10K	33.00	0.888	9K	6.2m	29.40	0.856	13K	6.5m
	0.10	30K	32.93	0.883	8K	6.2m	29.43	0.857	11K	7.1m
15 views	0.03	1K	34.30	0.900	27K	6.7m	31.00	0.878	41K	7.4m
	0.03	10K	34.38	0.900	21K	6.8m	31.06	0.879	32K	7.3m
	0.03	30K	34.50	0.901	16K	7.1m	31.10	0.880	23K	7.4m
	0.05	10K	34.52	0.902	15K	6.8m	31.10	0.881	22K	7.1m
	0.05	30K	34.62	0.903	10K	7.3m	31.16	0.882	15K	7.3m
	0.10	10K	34.52	0.902	10K	6.8m	31.12	0.881	14K	6.9m
	0.10	30K	34.50	0.902	9K	6.9m	31.14	0.881	13K	7.6m
25 views	0.03	1K	36.38	0.853	47K	9.0m	33.45	0.913	42K	7.9m
	0.03	10K	36.40	0.852	43K	9.0m	33.45	0.913	36K	8.0m
	0.03	30K	36.43	0.852	38K	9.1m	33.46	0.912	31K	8.1m
	0.05	10K	36.47	0.851	32K	8.6m	33.44	0.912	27K	7.7m
	0.05	30K	36.48	0.851	26K	8.8m	33.45	0.912	21K	7.9m
	0.10	10K	36.43	0.850	22K	8.3m	33.32	0.911	18K	7.5m
	0.10	30K	36.44	0.849	22K	9.0m	33.31	0.910	17K	8.1m

D.3 MASKING

In Table 5, we present an ablation on the use of the soft Otsu mask during 3D error-guided densification. Particularly in sparse-view configurations, the mask acts as a crucial spatial prior, improving reconstruction quality and model compactness. Without it, the model tends to overfit by placing Gaussians in empty space to minimize 2D projection errors, degrading the 3D geometry and inflating model size. The mask constrains densification to the object's volume, focusing model capacity on refining true geometric details.

Table 5: Ablation on background masking, comparing performance on real and synthetic datasets across different sparse-view settings for a fixed 20-layer model. Mask improves reconstruction quality (PSNR, SSIM) and leads to a more compact model with fewer primitives (N).

		Real Dataset				Synthetic Dataset			
	Masking	PSNR↑	SSIM↑	N↓	Time↓	PSNR↑	SSIM↑	N↓	Time↓
5 views	without	28.11	0.786	15K	5.0m	25.52	0.766	20K	5.8m
	with	28.65	0.827	6K	5.6m	25.73	0.794	11K	6.0m
10 views	without	32.54	0.856	16K	5.4m	29.16	0.840	20K	5.8m
	with	33.05	0.890	9K	6.7m	29.43	0.858	14K	6.8m
15 views	without	34.23	0.885	18K	5.5m	30.90	0.870	20K	5.8m
	with	34.65	0.903	10K	7.0m	31.14	0.882	15K	7.3m
25 views	without	36.57	0.855	33K	6.5m	33.46	0.912	24K	5.7m
	with	36.46	0.851	26K	8.7m	33.39	0.913	21K	8.0m

D.4 LAYER SELECTION

We investigated several layer selection strategies aimed at reducing computational cost by selectively updating subsets of layers. These included boosting-like approaches, such as training only the newest layer or optimizing a sliding window of recent layers. However, as shown in Table 6, joint optimization of all layers consistently yielded superior results. We attribute this to the need for global coherence: freezing earlier layers prevents them from adapting to the details introduced by new layers.

Table 6: Ablation on layer training strategies for a 20-layer model under the 10-view setting. The superior full-training strategy is highlighted.

		Real Dataset				Synthetic Dataset			
	Strategy	PSNR↑	SSIM↑	N↓	Time↓	PSNR↑	SSIM↑	N↓	Time↓
10 views	Train newest layer	32.28	0.873	20K	6.9m	28.98	0.841	24K	7.3m
	Train last 2 layers	32.41	0.876	18K	6.8m	29.05	0.844	23K	7.5m
	Train last 3 layers	32.41	0.878	17K	6.7m	29.04	0.844	23K	7.4m
	Prob. chain	32.64	0.880	13K	6.4m	29.25	0.851	16K	7.4m
	Prob. independent	32.66	0.878	13K	6.3m	29.19	0.850	17K	7.0m
	Train all layers	33.02	0.889	7K	6.6m	29.45	0.857	14K	6.8m