arXiv:2411.04844v4 [eess.IV] 22 Oct 2025

Discretized Gaussian Representation for Tomographic Reconstruction

Shaokai Wu[1], Yuxiang Lu[1], Yapan Guo[4], Wei Ji[2], Suizhi Huang[1], Fengyu Yang[3], Shalayiding Sirejiding[1], Qichen He[1], Jing Tong[1], Yanbiao Ji[1], Yue Ding[1], Hongtao Lu[1*]
[1]Shanghai Jiao Tong University, [2]National University of Singapore,
[3]Yale University, [4]Suzhou Xiangcheng People's Hospital
{shaokai.wu, luyuxiang_2018, huangsuizhi, salaydin}@sjtu.edu.cn
{ayombeach, tj_19_hf, jiyanbiao, dingyue, htlu}@sjtu.edu.cn
guo_yapan@outlook.com, jiwei@nus.edu.sg, fengyu.yang@yale.edu

Abstract

Computed Tomography (CT) enables detailed cross-sectional imaging but continues to face challenges in balancing reconstruction quality and computational efficiency. While deep learning-based methods have significantly improved image quality and noise reduction, they typically require large-scale training data and intensive computation. Recent advances in scene reconstruction, such as Neural Radiance Fields and 3D Gaussian Splatting, offer alternative perspectives but are not well-suited for direct volumetric CT reconstruction. In this work, we propose Discretized Gaussian Representation (DGR), a novel framework that reconstructs the 3D volume directly using a set of discretized Gaussian functions in an end-to-end manner. To further enhance efficiency, we introduce Fast Volume Reconstruction, a highly parallelized technique that aggregates Gaussian contributions into the voxel grid with minimal overhead. Extensive experiments on both real-world and synthetic datasets demonstrate that DGR achieves superior reconstruction quality and runtime performance across various CT reconstruction scenarios. Our code is publicly available at https://github.com/wskingdom/DGR.

1. Introduction

When a patient undergoes a Computed Tomography (CT) scan at a hospital, the initial output is not the familiar image seen on the screen, but rather a series of projection data generated by X-ray detectors [21]. This data is then processed by a CT reconstruction algorithm to transform the raw information into a 3D volume, which radiologists examine to assess the patient's condition. This task is challenging due to three key factors. Firstly, the ionizing radiation exposure from a complete CT scan poses significant health risks to patients, limiting the projections that can be acquired [41]. Secondly, time constraints in emergency situations require fast reconstruction to enable timely diagnosis [3]. Thirdly, the method must be generalizable across various CT configurations, such as different scan geometries, to accommodate heterogeneous imaging protocols and patient populations.

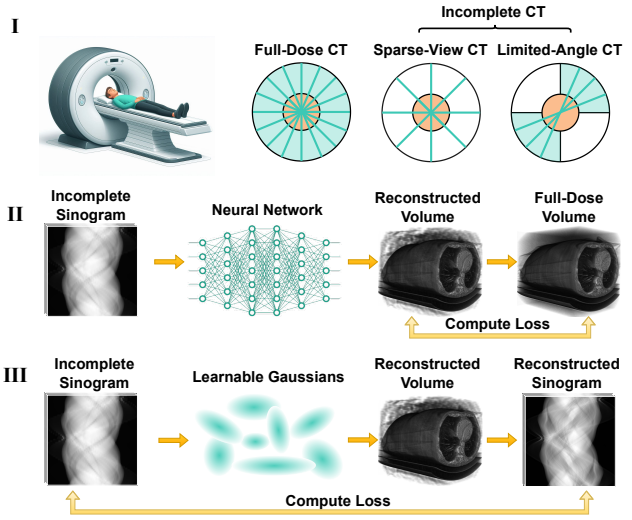


Figure 1. I: CT imaging workflow: X-ray projections are acquired from multiple angles. II: Deep Learning-based Reconstruction (DLR): Networks are trained on paired projection-image datasets to reconstruct volumes, requiring extensive pre-training. III: Instance reconstruction (exemplified by 3DGS): Instance-adaptive optimization of 3D Gaussians via differentiable rendering, tailored for each instance without the need for training datasets.

As illustrated in Figure 1, existing CT reconstruction methods broadly fall into two categories: Deep Learning-based Reconstruction (DLR) and instance reconstruction. Early DLR approaches, such as FBPConvNet [16] and RED-CNN [4], primarily focused on image denoising within the reconstructed domain. Subsequent methods

*Corresponding author.

like HDNet [15] advanced to joint optimization across both projection and image domains. More recently, advanced diffusion-based DLR methods, including DiffusionMBIR [6] and SWORD [37], leverage score-based diffusion models to guide the reconstruction process. Despite these advancements, DLR methods face persistent challenges: high training costs and limited generalization across diverse CT scanner configurations. For instance, models trained on Cone-Beam CT data often perform poorly on Fan-Beam CT systems, and networks optimized for chest scans may show degraded performance on head scans.

In response to these challenges, instance reconstruction methods have emerged, specifically designed to yield patient-specific results. Inspired by the success of scene reconstruction techniques, particularly Neural Radiance Fields (NeRF) [29] and 3D Gaussian Splatting (3DGS) [19], numerous adaptations have been proposed for CT reconstruction. Exemplars of NeRF-based methodologies, such as NAF [40], Intra-Tomo [39], and SAX-NeRF [2], implicitly represent the tomographic scene as a continuous function of 3D spatial coordinates and optimize a neural network within the projection domain. However, these methods typically require hours per reconstruction, which renders them impractical for real-time clinical deployment. On the other hand, 3DGS-based methodologies, such as 3DGR-CAR [8] and X-Gaussian [3], leverage explicit Gaussian functions to represent tomographic scenes. Nevertheless, directly applying 3DGS to CT leads to a strong integration bias [41] due to density inconsistency inherent in 3DGS. While R^2-Gaussian [41] attempts to mitigate these issues with a rectified 3DGS rasterizer and voxelizer, it introduces computational overhead stemming from the fundamental discrepancy between 3DGS's view-oriented rendering and the requirements of discretized tomographic reconstruction.

These limitations highlight a critical gap: existing DLR methods are hindered by dataset dependency and training complexity, while instance reconstruction paradigms are not well-aligned with tomographic objectives. Motivated by these challenges, we rethink the framework's design based on three core principles: discretized representation, efficient reconstruction, and unified global optimization.

From these foundational considerations, we propose our Discretized Gaussian Representation (DGR) method. For **representation**, given that the target volumetric grid is discretized, we directly model the 3D volume as a set of discretized Gaussian functions. To ensure differentiability and accurate preservation of continuous Gaussian contributions within local grid regions, we developed a novel alignment technique. For **reconstruction**, we advance beyond traditional voxelization with our highly parallelized Fast Volume Reconstruction method, which can reconstruct a $256 \times 256 \times 256$ volume from over 150,000 Gaussians in just 0.09 seconds per iteration. In terms of **optimization**, the synergy of our lightweight representation and fast reconstruction allows for joint optimization of all Gaussians, a departure from 3DGS-based methods that rely on selective view or block optimization. We evaluate DGR on three public CT datasets: FIPS [34], AAPM-Mayo LDCT [30], and FUMPE [27], covering both real-world and synthetic data. Comprehensive experiments across multiple reconstruction tasks demonstrate that DGR surpasses state-of-the-art DLR and instance reconstruction methods in both quantitative metrics and qualitative assessments.

Our main contributions can be summarized as follows:

- We propose DGR, an end-to-end CT reconstruction method that directly reconstructs 3D volumes by learning from discretized Gaussian functions.
- We introduce a fast volume reconstruction technique that efficiently aggregates Gaussian contributions into a discretized volume using a highly parallelized approach, significantly reducing reconstruction time.
- We demonstrate DGR's superior performance over both DLR and instance reconstruction methods through extensive experiments on real-world and synthetic datasets.

2. Background

2.1. Deep Learning Reconstruction

Deep Learning Reconstruction (DLR) methods have consistently been a prominent area in CT reconstruction since the successful application of deep neural networks in image processing [32]. Conventional DLR approaches are primarily categorized into direct-learning and indirect-learning approaches [38]. Direct learning methods involve training networks to directly reconstruct images from projection data (sinograms). Prominent instances include AUTOMAP [42], which employs manifold approximation, and iRadonMAP [12], which utilizes learnable filtering and back-projection techniques. Conversely, indirect learning methods [4, 11, 16, 18] frame CT reconstruction as a high-level denoising problem. These methods first transform raw projection data into the image domain, typically via filtered back-projection (FBP) [1] or iterative reconstruction (IR), and subsequently train networks to map these noisy initial images to their clean counterparts. Certain indirect methods, such as HDNet [15], further integrate both projection and image domains' knowledge to guide the reconstruction.

Recent advancements in DLR have led to the development of diffusion models that further guide the reconstruction process. Models like MCG [5], DiffusionMBIR [6], and SWORD [37] learn the score function as a prior, which helps manage noise and artifacts in the reconstructed images. However, diffusion-based methods are computationally more intensive than traditional DLR, limiting their practicality in clinical settings. Despite this, DLR methods can improve generalization and performance by training on large datasets with diverse imaging conditions.

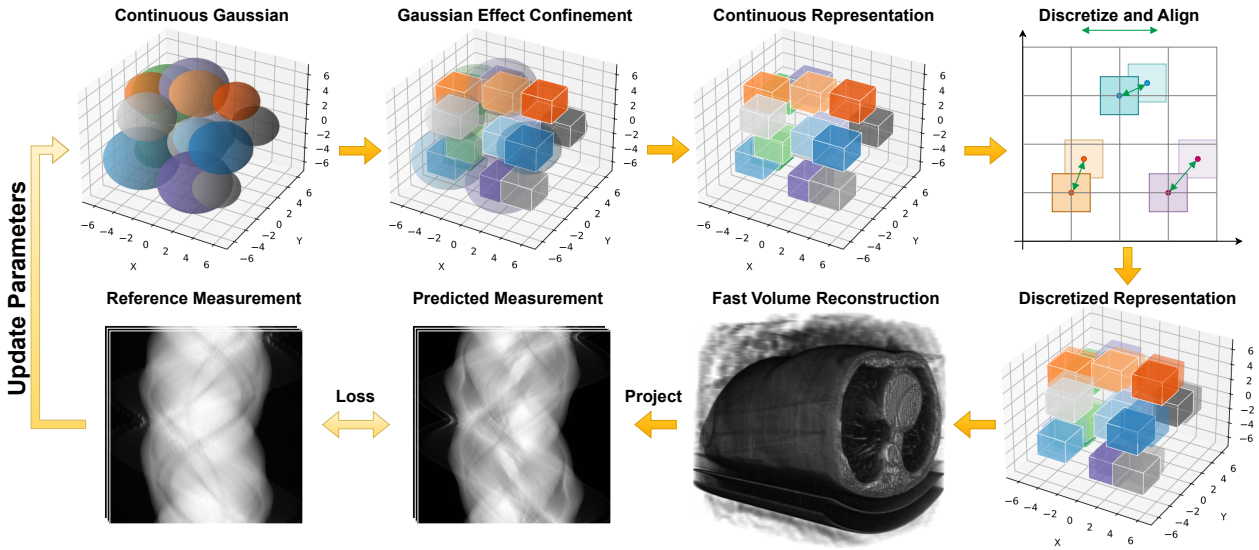


Figure 2. Pipeline of DGR. The 3D volume is initially represented by a set of continuous Gaussians, with each Gaussian's contribution confined to a local region surrounding the voxel. The Gaussians are then discretized onto the 3D grid, where their contributions are aligned to directly reconstruct the entire volume. Fast Volume Reconstruction technique gathers these contributions in a highly parallelized manner. Within each iteration, these Gaussians are reconstructed and then projected into the measurement domain for optimization.

2.2. Instance Reconstruction

Instance Reconstruction-based methods constitute another category of CT reconstruction techniques, focused on reconstructing images through solving an optimization problem. These methods are fundamentally rooted in iterative reconstruction (IR) [9], where an objective function balances data fidelity with regularization terms. These terms impose prior knowledge or constraints, such as sparsity, smoothness, or edge preservation, to guide the reconstruction. Common approaches like algebraic reconstruction techniques (ART) [13], simultaneous iterative reconstruction technique (SIRT) [35], and total variation (TV) [33] regularization are widely adopted to enhance image quality and reduce artifacts compared to direct analytical methods like filtered back projection (FBP) [1].

Recent advancements for CT imaging have seen the adaptation of instance reconstruction methods, such as Neural Radiance Fields (NeRF) [29] and 3D Gaussian Splatting (3DGS) [19]. NeRF-based approaches like NAF [40] and Intra-Tomo [39] optimize by minimizing the difference between synthetic and ground-truth projections, incorporating both local and global geometric priors. SAX-NeRF [2] further enhances Sparse-View X-ray reconstruction through transformer-based structure-aware modeling. These NeRF-based methods typically require hours of inference time per instance [41], making them impractical for clinical use.

Generally, 3DGS-based methods surpass NeRF-based methods in both speed and image quality. Following the original 3DGS, they represent the CT scene as a set of 3D Gaussians. 3DGR-CT [24, 25] integrates 3D Gaussians with a differentiable CT projector for Cone-Beam CT reconstruction. 3DGR-CAR [8] leverages a U-Net [32] to initialize Gaussian centers before employing 3DGS for final reconstruction. DIF-Gaussian [26] uses 3D Gaussians to model feature distributions for estimating attenuation coefficients. X-Gaussian [3] adapts 3DGS for novel view synthesis in X-ray imaging. However, these 3DGS-based methods are not ideal for direct 3D volume reconstruction. R^2-Gaussian [41] identified integration bias in the standard 3DGS formulation and proposed rectified rasterizer and differentiable voxelizer to mitigate this limitation.

3. Preliminary

Problem Formulation Clinical CT imaging involves capturing X-ray projections of the patient from various angles, which are then processed to reconstruct a 3D volume that represents the internal structures. The CT imaging system for a 2D image slice can be described as:

$$y = Ax + \beta, \tag{1}$$

where $x \in \mathbb{R}^{w \times h}$ denotes the 2D image with width w and height h, and $y \in \mathbb{R}^{m \times n}$ represents the projection matrix, where m is the number of projection views and n is the number of detector elements per view. The matrix $A \in \mathbb{R}^{m \times n \times w \times h}$ is the Radon Transform [31] matrix, which maps the geometric relationship between the object and the X-ray source/detector system. The noise term $\beta \in \mathbb{R}^{m \times n}$ represents measurement noise.

Optimization Goal. The goal of CT reconstruction is to accurately estimate the underlying image x from measured projection data y. While the Inverse Radon Transform theoretically offers a perfect mapping, real-world issues like data noise and the underdetermined nature of the inverse problem makes perfect reconstruction unattainable.

The reconstruction process is commonly formulated as an optimization problem:

$$x^* = \arg\min_x \mathcal{E}(Ax, y) + R(x), \tag{2}$$

where $\mathcal{E}$ is the data fidelity term, which measures the discrepancy between the estimated measurement Ax and the actual measurement y, often using the L1 or L2 norm. The regularization term $R(x)$ enforces properties like smoothness or sparsity in the reconstructed image, with Total Variation (TV) [33] regularization being a popular choice [28].

4. Methodology

We illustrate our DGR framework in Figure 2. DGR aims to directly reconstruct the 3D tomographic volume in an end-to-end and efficient manner. We organize our method section into three parts: Discretized Gaussian Representation (Section 4.1), Fast Volume Reconstruction (Section 4.2), and Global Optimization (Section 4.3).

4.1. Discretized Gaussian Representation

Continuous Gaussian Representation We begin by introducing the basic formulation of the continuous Gaussian representation for tomographic volume reconstruction. We define the Gaussian function centered at μ with covariance Σ as:

$$G(p, \mu, \Sigma) = e^{-\frac{1}{2}(p-\mu)^\top \Sigma^{-1}(p-\mu)}, \tag{3}$$

where $p \in \mathbb{R}^d, d = 3$ is the 3D point in the scene (For clarity in the Einstein Summation representation presented in this section, we maintain the notation of a constant $d = 3$). The Gaussian function is bell-shaped and symmetric around the mean μ, with its spread controlled by the standard deviation σ, determining its extent in 3D space. We utilize isotropic Gaussians rather than anisotropic ones to represent the 3D volume. This choice is justified by the isotropic attenuation properties of CT tissues, particularly soft tissues, which exhibit minimal directional dependence. As noted in [17], small distributed source elements can be treated as isotropic, consistent with tissue behavior.

Direct Reconstruction Let the target volume be denoted by $V \in \mathbb{R}^{w \times h \times c}$, where w, h, and c correspond to the width, height, and depth (or length) of the volume, respectively. The intensity of a voxel $V(p)$ at a spatial coordinate p within this volume is expressed as the cumulative contribution from all n constituent Gaussians:

$$V(p) = \sum_{i=1}^{n} G(p, \mu_i, \Sigma_i) \cdot I_i, \tag{4}$$

where $I_i \in \mathcal{R}^+$ represents the scalar intensity value of the i-th Gaussian, functioning as both its individual contribution to the voxel intensity and its weight. The quadratic form $(p-\mu)^\top \Sigma^{-1}(p-\mu)$ is recognized as the squared Mahalanobis distance, which quantifies the statistical distance of point p from the Gaussian center μ, normalized by the distribution's shape. This quantity will be referred to as D^2 in subsequent discussions for conciseness.

The direct computation of every Gaussian's contribution to the entire volume presents a significant challenge, leading to unacceptable computational expense (see Table 5). For a deeper understanding of this computational complexity, please refer to Appendix E.2. To mitigate this issue, we restrict each Gaussian's effect to a localized region around its assigned voxel, as detailed below.

Gaussian Effect Confinement The influence of a Gaussian on a given voxel is known to attenuate with increasing distance from the Gaussian's centroid. Leveraging this principle, restricting the contributions to only those Gaussians within a specified proximity of each voxel can substantially accelerate the reconstruction process.

To formalize this concept, we define the effective proximity for each Gaussian as a cuboid $w_0 \times h_0 \times c_0$, where w_0, h_0, c_0 are all odd integers. This box is centered on the voxel, and we denote the set of all 3D coordinates within this cuboid as $B_0 \subset \mathbb{R}^3$.

Naturally, the specific coordinates within the cuboid B_0 depend on the Gaussian function's center μ. For a Gaussian centered at μ, its influence is evaluated at coordinates $B = B_0 - \mu$, and B is a constant tensor. This is because the coordinates within B_0 are relative to μ, meaning that subtracting μ yields a fixed pattern of offsets independent of its absolute position. Mathematically, this reflects the translation invariance of the Gaussian function's shape: shifting μ only changes the peak's location, not the function's overall form.

Now, we can compute the squared Mahalanobis distance D^2 for a given relative coordinate in B with respect to the Gaussian's mean (which is implicitly at the origin for these relative coordinates) as:

$$D^2 = B^\top C^{-1} B. \tag{5}$$

Here, C denotes the covariance matrix of the Gaussian. The contribution of the i-th Gaussian to the voxel intensity at a relative coordinate described by B is then given by:

$$e^{-\frac{1}{2}D^2} \cdot I_i. \tag{6}$$

While this localized confinement might lead to a loss of information from more distant Gaussians, it is worth noting that all Gaussians are trainable and can compensate for any missing information from more distant Gaussians.

Discretize and Align In reconstructing a 3D volume V, directly using the continuous mean μ for indexing and summing all Gaussian contributions presents a challenge: the gradient with respect to μ cannot propagate through this summation. This occurs because the discretized 3D volume V relies on integer coordinates, while μ is continuous.

Simply discretizing μ by rounding it to the nearest integer for indexing would render the reconstruction process non-differentiable, thereby hindering optimization. To circumvent this, rather than directly discretizing μ, we calculate each Gaussian's contribution at the discretized grid positions aligned with their *floored* integer coordinates. More precisely, we select the floored integer of the box's center as the indexing point, denoting this integer-centered box as $\lfloor B \rfloor$.

For enhanced clarity, we define the residual between the continuous center μ and its floored integer counterpart as:

$$\Delta\mu = \mu - \lfloor \mu \rfloor, \tag{7}$$

where $\lfloor \mu \rfloor$ signifies the floor function. This $\Delta\mu$ quantifies the offset between the continuous and discretized positions, thereby enabling precise alignment between the continuous Gaussian center and the discretized grid.

Subsequently, we establish a relationship between the coordinates of the new box $\lfloor B \rfloor$ and the original box B by accounting for this offset. The transformation from B to $\lfloor B \rfloor$ is expressed as:

$$\lfloor B \rfloor_{n,w_0,h_0,c_0,d} = B_{w_0,h_0,c_0,d} - \Delta\mu_{n,1,1,1,d,}. \tag{8}$$

Here, $\Delta\mu$ is implicitly broadcast across the dimensions of B to enable element-wise subtraction. This adjustment guarantees that each Gaussian's contribution is accurately evaluated at its corresponding position on the discretized grid, relative to its floored mean. This 'Discretize and Align' procedure thus enables seamless optimization by ensuring the unimpeded flow of gradients through the aggregation of Gaussian contributions, thereby preserving the differentiability of the reconstruction pipeline.

4.2. Fast Volume Reconstruction

Parallel Computation To improve computational efficiency, we compute the squared Mahalanobis distance D^2 for each local region in parallel. For each voxel, D^2 is calculated using the discretized box $\lfloor B \rfloor$, leading to:

$$D^2 = \lfloor B \rfloor^\top C^{-1} \lfloor B \rfloor. \tag{9}$$

To represent this computation more clearly and compactly, we utilize the Einstein summation convention, which implicitly sums over repeated indices. The expression for D^2 thus becomes:

$$D^2_{n,w_0,h_0,c_0} = \sum_d \lfloor B \rfloor_{n,w_0,h_0,c_0,d} C^{-1}_{n,d,d} \lfloor B \rfloor_{n,w_0,h_0,c_0,d}.$$

Once D^2 is computed, the contributions of all Gaussians are summed to form the reconstructed volume.

Decomposition To further reduce computation in Equation 10, we decompose the computationally expensive Einstein summation into smaller, more manageable summations. The squared Mahalanobis distance D^2 can be explicitly decomposed into a combination of the following four smaller Einstein sums:

$$B^T C^{-1} B_{n,w_0,h_0,c_0} = \sum_d B^T_{w_0,h_0,c_0,d}\, C^{-1}_{n,d,d}\, B_{w_0,h_0,c_0,d},$$

$$B^T C^{-1} \Delta\mu_{n,w_0,h_0,c_0} = \sum_d B^T_{w_0,h_0,c_0,d}\, C^{-1}_{n,d,d}\, \Delta\mu_{n,1,1,d},$$

$$\Delta\mu^T C^{-1} B_{n,w_0,h_0,c_0} = \sum_d \Delta\mu^T_{n,1,1,d}\, C^{-1}_{n,d,d}\, B_{w_0,h_0,c_0,d},$$

$$\Delta\mu^T C^{-1} \Delta\mu_{n,1,1,1} = \sum_d \Delta\mu^T_{n,1,1,d}\, C^{-1}_{n,d,d}\, \Delta\mu_{n,1,1,d},$$

Subsequently, we compute the squared Mahalanobis distance D^2 as:

$$\begin{aligned} D^2 =& (B^\top C^{-1} B) - (B^\top C^{-1} \Delta\mu) \\ & - (\Delta\mu^\top C^{-1} B) + (\Delta\mu^\top C^{-1} \Delta\mu). \end{aligned} \tag{10}$$

The final contribution of the Gaussians within the local region is computed as:

$$\Gamma = e^{-\frac{1}{2} D^2_{n,w_0,h_0,c_0}} \cdot I, \tag{11}$$

where $\Gamma \in \mathbb{R}^{n \times w_0 \times h_0 \times c_0}$ represents the contributions of n Gaussians within the local region. This decomposition allows us to calculate the contributions in a highly efficient manner, reducing time complexity without increasing space complexity. Specifically, $B_{w_0,h_0,c_0,d}$ is only $\frac{1}{n}$ the size of $\lfloor B \rfloor_{n,w_0,h_0,c_0,d}$, enabling far more efficient computations. The quantitative evidence supporting this efficiency gain is presented in Table 5.

Then we aggregate the contribution of each Gaussian into its corresponding voxels within the volume. Specifically, for each contribution $\Gamma_{i,x,y,z}$ from the i-th Gaussian to the voxel located at position (x, y, z), the volume update proceeds as:

$$V_{x,y,z} \leftarrow V_{x,y,z} + \Gamma_{i,x,y,z}. \tag{12}$$

This aggregation is performed for all Gaussians, guaranteeing the precise accumulation of their contributions at the appropriate locations within the reconstructed volume. A highly parallelized implementation of this update process is available in our codebase for further reference.

Notably, our Fast Volume Reconstruction is a plug-and-play module that can be seamlessly integrated into existing methods leveraging 3D Gaussians. For instance, while X-Gaussian [3] originally focuses on novel view synthesis without volume reconstruction, integrating our module into its pipeline prior to loss computation enables it to perform volume reconstruction, as demonstrated in Table 1.

4.3. Global Optimization

Subsequent to reconstructing the 3D volume from the Gaussians, we adopt a joint optimization of the Gaussian param-

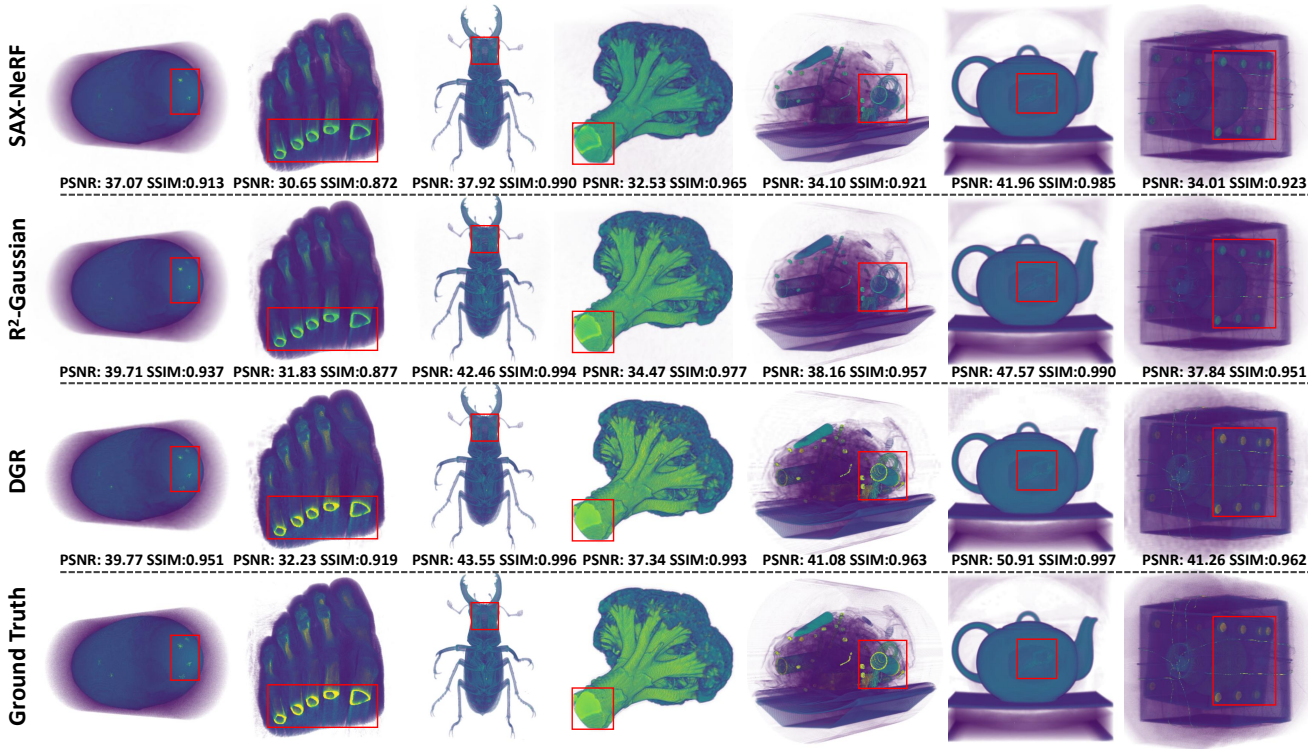


Figure 3. Qualitative Comparison of our DGR with previous state-of-the-art instance reconstruction methods SAX-NeRF and R^2-Gaussian, with major differences highlighted in **Red** boxes. *Please zoom in for better visibility.*

eters within the projection space. Diverging from conventional 3DGS and its variants, which typically optimize parameters per selected view [3, 19] or local blocks [41], our global optimization strategy achieves convergence in fewer than 1K iterations, representing a substantial improvement over standard approaches (e.g., requiring 30K iterations).

The initial phase of this optimization involves projecting the reconstructed 3D volume back into the projection domain. To accomplish this, we define a general geometry projection transformation $\mathcal{T}$:

$$\hat{P} = \mathcal{T}(V), \tag{13}$$

where $\hat{P} \in \mathbb{R}^{m \times n \times p}$ denotes the estimated projection.

This concise formulation is independent of specific CT acquisition geometries, demonstrating strong compatibility with various CT settings, including Fan-Beam or Cone-Beam CT, as well as Sparse-View or Limited-Angle CT. In each iteration, we optimize the Gaussian parameters to minimize the discrepancy between the estimated projection $\hat{P}$ and the reference projection P. This optimization is guided by an objective function combining $\mathcal{L}_1$ loss, SSIM [36] loss ($\mathcal{L}_{SSIM}$), and total variation [33] loss ($\mathcal{L}_{TV}$):

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_1(\hat{P}, P) + \lambda_2 \mathcal{L}_{\text{SSIM}}(\hat{P}, P) + \lambda_3 \mathcal{L}_{\text{TV}}(V), \tag{14}$$

where $\lambda_1 = 0.6$, $\lambda_2 = 0.2$, and $\lambda_3 = 1$ are the respective weights for the $\mathcal{L}_1$, $\mathcal{L}_{\text{SSIM}}$, and $\mathcal{L}_{\text{TV}}$. We discuss the definition and usage of these loss functions in Appendix D.1.

To further enhance the optimization process, we adopt the well-established adaptive density control for 3D Gaussians [19]. This involves cloning under-reconstructed Gaussians, splitting over-reconstructed ones into smaller Gaussians, and pruning those with near-zero gradient magnitudes. As this process is standard and not our core contribution, we provide full details in Appendix F.

5. Experiments

5.1. Experiment Settings

Datasets and Evaluation Setup We conducted experiments using three distinct datasets: FIPS [34], AAPM-Mayo LDCT [30], and FUMPE [27]. To ensure a fair and reproducible evaluation, we maintained consistent settings for all compared methods and grouped them by task, as detailed below: For Cone-Beam Sparse-View CT, we evaluated on the FIPS [34] dataset, following the setup used by R^2-Gaussian [41]. This dataset includes both real-world and synthetic data. The real-world evaluation utilized three cases with 721 real projections, while the synthetic portion involved 15 real CT volumes encompassing a range of subjects from organisms to artificial objects. For Fan-Beam Sparse-View CT and Limited-Angle CT, we followed the evaluation protocols of SWORD [37] and Diffusion-MBIR [6], using the widely adopted AAPM-Mayo LDCT dataset. This dataset comprises 5388 slices with a 1mm

Table 1. Comparisons of 75/50/25-view Sparse-View CT on FIPS dataset. Best in **Bold.**

Methods	75-view			50-view			25-view		
	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓
Real-World Dataset									
FDK [7]	30.03	0.535	-	27.38	0.449	-	23.30	0.335	-
IntraTomo [39]	36.79	0.858	2h25m	36.99	0.854	2h19m	**35.85**	0.835	2h18m
NAF [40]	38.58	0.848	51m28s	36.44	0.818	51m31s	32.92	0.772	51m24s
SAX-NeRF [2]	34.93	0.854	13h21m	34.89	0.840	13h23m	33.49	0.793	13h25m
X-Gaussian* [3]	38.27	0.894	10m21s	37.80	0.881	10m5s	35.12	0.859	9m55s
R^2-Gaussian (iter=10k) [41]	38.10	0.872	3m39s	37.52	0.866	3m37s	35.10	0.840	3m23s
R^2-Gaussian (iter=30k) [41]	39.40	0.875	14m16s	38.24	0.864	13m52s	34.83	0.833	12m56s
DGR (iter=300)	39.91	0.937	**3m36s**	38.66	0.929	**3m24s**	35.16	**0.883**	**2m59s**
DGR (iter=1000)	**41.28**	**0.952**	13m14s	**39.27**	**0.940**	11m48s	34.58	0.870	8m54s
Synthesis Dataset									
FDK [7]	28.63	0.497	-	26.50	0.422	-	22.99	0.317	-
IntraTomo [39]	35.42	0.924	2h7m	35.25	0.923	2h9m	34.68	0.914	2h19m
NAF [40]	37.84	0.945	30m43s	36.65	0.932	32m4s	33.91	0.893	31m1s
SAX-NeRF [2]	38.07	0.950	13h5m	36.86	0.938	13h5m	34.33	0.905	13h3m
X-Gaussian* [3]	38.22	0.955	9m58s	37.50	0.946	9m50s	35.18	0.924	9m51s
R^2-Gaussian (iter=10k) [41]	38.29	0.954	2m38s	37.63	0.949	2m35s	35.08	0.922	2m35s
R^2-Gaussian (iter=30k) [41]	38.88	0.959	8m21s	37.98	0.952	8m14s	35.19	0.923	8m28s
DGR (iter=300)	38.71	0.957	**2m24s**	38.05	0.954	**2m17s**	35.76	0.930	**2m14s**
DGR (iter=1000)	**39.63**	**0.964**	7m27s	**38.74**	**0.960**	7m13s	**36.55**	**0.947**	6m53s

thickness, commonly used in CT reconstruction research. Additionally, we performed evaluations on the FUMPE [27] dataset, which contains 8792 chest CT images. Following standard practice, we adopted Peak Signal-to-Noise Ratio (PSNR) [14] and Structural Similarity Index Measure (SSIM) [36] as our image quality metrics.

Implementation Details We implemented DGR using PyTorch. Most experiments were trained on a single NVIDIA RTX 6000 GPU. However, for the 75/50/25-view Sparse-View CT experiments on the FIPS dataset, we utilized a single NVIDIA RTX 3090 GPU to ensure a fair comparison by maintaining the same training environment as R^2-Gaussian. We initialized the number of Gaussians to 150K and set the local box size to $17\times17\times17$. The Adam optimizer [20] was employed with an initial learning rate of 3×10^{-4}, which decayed to 3×10^{-5} by the end of training. For Cone-Beam Sparse-View CT, the training process was limited to 1K iterations. For Fan-Beam Sparse-View CT, we did not set the iteration limit, and the training process is ceased when the validation loss converges in the projection domain.

5.2. Experimental Results and Analysis

5.2.1. Cone-Beam Sparse-View CT Reconstruction

In Table 1, we present a comparison of DGR with advanced instance reconstruction methods on the FIPS dataset. Baselines include NeRF-based methods (IntraTomo [39], NAF [40], SAX-NeRF [2]) and 3DGS-based methods (X-Gaussian [3], R^2-Gaussian [41]), with X-Gaussian specifically implemented using our Fast Volume Reconstruction module. DGR demonstrates superior image quality and computational efficiency across all view counts, significantly outperforming previous instance reconstruction methods. Even with just 300 iterations, DGR surpasses most advanced methods while completing reconstruction in approximately 3 minutes.

The faster reconstruction time achieved by DGR is attributed to the discretized representation, which is inherently well-aligned with the reconstruction task and avoids redundant computation. Furthermore, DGR's lightweight framework enables the joint optimization of all Gaussians, significantly reducing computation time without compromising reconstruction quality, as illustrated in Figure 3.

5.2.2. Fan-Beam Sparse-View CT Reconstruction

In Table 2, we compare DGR with leading DLR methods on the AAPM-Mayo LDCT dataset [30]. The term 'Data' in these tables refers to the number of images used to train the DLR methods, whereas our DGR optimizes without relying on additional training data. The baseline methods include traditional CNN-based approaches such as FBPConvNet [16], U-Net [23], and PLANet [38], alongside advanced diffusion-based methods like GMSD [10],

Table 2. Comparisons of 180/120/90/60-view Sparse-View CT on AAPM-Mayo LDCT dataset. Best in **Bold.**

Method	Extra Data	180-view		120-view		90-view		60-view	
		PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
FBP [1]	0	31.69	0.882	28.30	0.787	26.20	0.701	23.18	0.595
FBPConvNet [16]	4839	42.23	0.988	39.45	0.983	37.11	0.976	35.63	0.966
U-Net [23]	4839	38.37	0.985	35.58	0.977	30.09	0.947	28.83	0.937
PLANet [38]	4176	42.76	0.965	41.67	0.962	40.99	0.957	38.97	0.941
GMSD [10]	4839	41.44	0.988	39.41	0.981	37.25	0.974	34.31	0.958
SWORD [37]	4839	45.08	0.994	42.49	0.990	41.27	0.986	38.49	0.978
DGR	0	**46.13**	**0.997**	**44.64**	**0.994**	**43.23**	**0.992**	**40.25**	**0.985**

Table 3. Fan-Beam Limited-Angle CT 90° reconstruction.

Method	Data	Axial		Coronal		Sagittal	
		PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
FBP [1]	0	14.91	0.397	17.07	0.411	15.46	0.403
FBPConvNet [16]	3142	26.76	0.879	25.77	0.874	22.92	0.841
ADMM-TV [28]	3142	23.19	0.793	22.96	0.758	19.95	0.782
MCG [5]	3839	26.01	0.838	24.55	0.823	21.59	0.706
Lahiri *et al.* [22]	3142	28.08	0.931	26.02	0.856	23.24	0.812
DiffusionMBIR [6]	3142	34.92	0.956	32.48	0.947	28.82	0.832
DGR	0	**38.22**	**0.970**	**39.32**	**0.969**	**38.35**	**0.970**

and SWORD [37]. To ensure a fair comparison for 180/120/90/60-view Sparse-View CT, we replicated the settings used in SWORD [37]. Notably, DGR surpasses advanced DLR methods that incorporate prior knowledge in Fan-Beam Sparse-View CT reconstruction, despite requiring zero training data. This validates DGR's ability to address diverse CT reconstruction tasks without architectural modifications while achieving superior performance.

5.2.3. Fan-Beam Limited-Angle CT Reconstruction

In Table 3, we report the performance of Limited-Angle CT reconstruction experiments conducted on the AAPM-Mayo LDCT dataset [30], following the methodology of DiffusionMBIR [6]. The baseline methods include traditional CNN-based techniques such as FBPConvNet [16] and Lahiri *et al.* [22], alongside advanced diffusion-based approaches like MCG [5] and DiffusionMBIR [6]. For these evaluations, the angular data acquisition was restricted to a 90° range within a total of 180°. Without any prior knowledge, DGR still demonstrates a substantial improvement over these advanced DLR methods.

5.2.4. Additional Evaluations and Visualizations

Our DGR is further evaluated on the FUMPE dataset [27] for Fan-Beam 180/120/90/60-view chest CT reconstruction, with the quantitative results detailed in Appendix Table 9. Furthermore, iterative reconstruction visualizations of DGR are showcased in Appendix Figures 4, 5, and 6.

5.3. Ablation Study

In Table 4, we evaluate the impact of local box size on 60-view Sparse-View CT reconstruction using the AAPM-Mayo LDCT dataset. Each configuration was assessed over 1K iterations to ensure stable measurements of training time and video memory usage. We observe that performance improves with increasing box size; however, this comes at the cost of increased time consumption and video memory. Consequently, we chose $17 \times 17 \times 17$ as the default box size for our experiments, as it achieves an optimal balance between performance and efficiency.

Table 4. Effect of Box Size on 60-view Sparse-View CT

Box-Size	$13 \times 13 \times 13$	$15 \times 15 \times 15$	$17 \times 17 \times 17$	$19 \times 19 \times 19$
Time (minutes)	7.92	11.07	16.58	26.99
V-Ram (GiB)	10.70	13.29	16.87	21.32
PSNR / SSIM	35.99 / 0.960	38.90 / 0.973	40.25 / 0.985	40.98 / 0.987

Effect of Fast Volume Reconstruction In Table 5, we compare the video memory and time consumption of our Fast Volume Reconstruction (denoted as FVR below) against the Direct reconstruction (Equation 4). We observe that Fast Volume Reconstruction achieves a significant reduction in video memory consumption. The decomposition further reduces the reconstruction time from 1.05s to 0.09s, thereby enabling the highly efficient reconstruction capabilities.

Table 5. Video-Memory and Time Per Iteration

Method	VRAM (GiB)	Time (s)
Direct Reconstruction (estimated)	16662.50	/
FVR w/o Decomposition	16.87	1.05
FVR w/ Decomposition	16.87	0.09

Analysis of Loss Functions In Appendix Tables 6, 7, and 8, we discuss the usage of various loss functions. Results indicate that a combined use of $\mathcal{L}_1$ loss, SSIM loss ($\mathcal{L}_{SSIM}$), and total variation loss ($\mathcal{L}_{TV}$) yields the best performance.

Analysis of Isotropic Gaussians and Voxelizer As detailed in Appendix Sections D.2 and D.3, our ablation studies demonstrate that DGR surpasses alternative designs in both image quality and reconstruction speed.

6. Conclusion

We present DGR, a novel framework that rethinks CT reconstruction through representation, reconstruction, and optimization. Evaluated on multiple datasets against both Deep Learning Reconstruction and instance reconstruction baselines, DGR achieves superior image quality and speed, demonstrating significant potential for clinical application.

Acknowledgments

This research is supported by the Fundamental Research Funds for the Central Universities (project number YG2024ZD06), NSFC (No. 62176155), and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102).

References

[1] Ronald Newbold Bracewell and ACf Riddle. Inversion of fan-beam scans in radio astronomy. *Astrophysical Journal, vol. 150, p. 427*, 150:427, 1967. 2, 3, 8, 12

[2] Yuanhao Cai, Jiahao Wang, Alan Yuille, Zongwei Zhou, and Angtian Wang. Structure-aware sparse-view x-ray 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11174–11183, 2024. 2, 3, 7

[3] 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 *European Conference on Computer Vision*, pages 283–299. Springer, 2025. 1, 2, 3, 5, 6, 7

[4] Hu Chen, Yi Zhang, Mannudeep K Kalra, Feng Lin, Yang Chen, Peixi Liao, Jiliu Zhou, and Ge Wang. Low-dose ct with a residual encoder-decoder convolutional neural network. *IEEE transactions on medical imaging*, 36(12):2524–2535, 2017. 1, 2

[5] Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022. 2, 8, 12

[6] Hyungjin Chung, Dohoon Ryu, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Solving 3d inverse problems using pre-trained 2d diffusion models. ieee. In *CVF Conference on Computer Vision and Pattern Recognition*, page 6, 2023. 2, 6, 8, 11, 12

[7] Lee A Feldkamp, Lloyd C Davis, and James W Kress. Practical cone-beam algorithm. *Josa a*, 1(6):612–619, 1984. 7

[8] Xueming Fu, Yingtai Li, Fenghe Tang, Jun Li, Mingyue Zhao, Gao-Jun Teng, and S Kevin Zhou. 3dgr-car: Coronary artery reconstruction from ultra-sparse 2d x-ray views with a 3d gaussians representation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 14–24. Springer, 2024. 2, 3

[9] Richard Gordon, Robert Bender, and Gabor T Herman. Algebraic reconstruction techniques (art) for three-dimensional electron microscopy and x-ray photography. *Journal of theoretical Biology*, 29(3):471–481, 1970. 3

[10] Bing Guan, Cailian Yang, Liu Zhang, Shanzhou Niu, Minghui Zhang, Yuhao Wang, Weiwen Wu, and Qiegen Liu. Generative modeling in sinogram domain for sparse-view ct reconstruction. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 2023. 7, 8

[11] Yoseob Han and Jong Chul Ye. Framing u-net via deep convolutional framelets: Application to sparse-view ct. *IEEE transactions on medical imaging*, 37(6):1418–1429, 2018. 2

[12] Ji He, Yongbo Wang, and Jianhua Ma. Radon inversion via deep learning. *IEEE transactions on medical imaging*, 39 (6):2076–2087, 2020. 2

[13] Gabor T Herman, Arnold Lent, and Stuart W Rowland. Art: mathematics and applications: a report on the mathematical foundations and on the applicability to real data of the algebraic reconstruction techniques. *Journal of theoretical biology*, 42(1):1–32, 1973. 3

[14] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010. 7, 11

[15] Dianlin Hu, Jin Liu, Tianling Lv, Qianlong Zhao, Yikun Zhang, Guotao Quan, Juan Feng, Yang Chen, and Limin Luo. Hybrid-domain neural network processing for sparse-view ct reconstruction. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 5(1):88–98, 2020. 2

[16] Kyong Hwan Jin, Michael T McCann, Emmanuel Froustey, and Michael Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE transactions on image processing*, 26(9):4509–4522, 2017. 1, 2, 7, 8

[17] Avinash C Kak and Malcolm Slaney. *Principles of computerized tomographic imaging*. SIAM, 2001. 4, 12

[18] Eunhee Kang, Junhong Min, and Jong Chul Ye. A deep convolutional neural network using directional wavelets for low-dose x-ray ct reconstruction. *Medical physics*, 44(10):e360–e375, 2017. 2

[19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 2, 3, 6, 13

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[21] Lennart Koetzier, Domenico Mastrodicasa, Timothy Szczykutowicz, Niels Werf, Adam Wang, Veit Sandfort, Aart van der Molen, Dominik Fleischmann, and Martin Willemink. Deep learning image reconstruction for ct: Technical principles and clinical prospects. *Radiology*, 306, 2023. 1

[22] Anish Lahiri, Gabriel Maliakal, Marc L Klasky, Jeffrey A Fessler, and Saiprasad Ravishankar. Sparse-view cone beam ct reconstruction using data-consistent supervised and adversarial learning from scarce training data. *IEEE Transactions on Computational Imaging*, 9:13–28, 2023. 8

[23] Hoyeon Lee, Jongha Lee, Hyeongseok Kim, Byungchul Cho, and Seungryong Cho. Deep-neural-network-based sinogram synthesis for sparse-view ct image reconstruction. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 3(2):109–119, 2018. 7, 8

[24] Yingtai Li, Xueming Fu, Shang Zhao, Ruiyang Jin, and S Kevin Zhou. Sparse-view ct reconstruction with 3d gaussian volumetric representation. *arXiv preprint arXiv:2312.15676*, 2023. 3

[25] 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*, page 103585, 2025. 3

[26] Yiqun Lin, Hualiang Wang, Jixiang Chen, and Xiaomeng Li. Learning 3d gaussians for extremely sparse-view cone-beam ct reconstruction, 2024. 3

[27] Mojtaba Masoudi, Hamid-Reza Pourreza, Mahdi Saadatmand-Tarzjan, Noushin Eftekhari, Fateme Shafiee Zargar, and Masoud Pezeshki Rad. A new dataset of computed-tomography angiography images for computer-aided detection of pulmonary embolism. *Scientific data*, 5 (1):1–9, 2018. 2, 6, 7, 8, 11

[28] Michael T McCann, Masih Nilchian, Marco Stampanoni, and Michael Unser. Fast 3d reconstruction method for differential phase contrast x-ray ct. *Optics express*, 24(13):14564–14581, 2016. 4, 8

[29] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2, 3

[30] Taylor R Moen, Baiyu Chen, David R Holmes III, Xinhui Duan, Zhicong Yu, Lifeng Yu, Shuai Leng, Joel G Fletcher, and Cynthia H McCollough. Low-dose ct image and projection dataset. *Medical physics*, 48(2):902–911, 2021. 2, 6, 7, 8, 11

[31] Johann Radon. On the determination of functions from their integral values along certain manifolds. *IEEE transactions on medical imaging*, 5(4):170–176, 2007. 3

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 2, 3

[33] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992. 3, 4, 6

[34] The Finnish Inverse Problems Society. X-ray tomographic datasets, 2024. 2, 6, 10, 11

[35] Jeannot Trampert and Jean-Jacques Leveque. Simultaneous iterative reconstruction technique: Physical interpretation based on the generalized least squares solution. *Journal of Geophysical Research: Solid Earth*, 95(B8):12553–12559, 1990. 3

[36] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6, 7, 11

[37] Kai Xu, Shiyu Lu, Bin Huang, Weiwen Wu, and Qiegen Liu. Stage-by-stage wavelet optimization refinement diffusion model for sparse-view ct reconstruction. *IEEE Transactions on Medical Imaging*, 2024. 2, 6, 8, 11, 12

[38] Liutao Yang, Rongjun Ge, Shichang Feng, and Daoqiang Zhang. Learning projection views for sparse-view ct reconstruction. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 2645–2653, 2022. 2, 7, 8

[39] Guangming Zang, Ramzi Idoughi, Rui Li, Peter Wonka, and Wolfgang Heidrich. Intratomo: self-supervised learning-based tomography via sinogram synthesis and prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1960–1970, 2021. 2, 3, 7

[40] Ruyi Zha, Yanhao Zhang, and Hongdong Li. Naf: neural attenuation fields for sparse-view cbct reconstruction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 442–452. Springer, 2022. 2, 3, 7

[41] 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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 1, 2, 3, 6, 7, 10

[42] Bo Zhu, Jeremiah Z Liu, Stephen F Cauley, Bruce R Rosen, and Matthew S Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555(7697):487–492, 2018. 2

Appendix

A. Overview

Thank you for reading the Appendix of our research. This appendix is organized as follows:

- Section B provides a detailed explanation of the experimental setup.
- Section C describes the evaluation metrics used in our study.
- Section D discusses the application of different loss functions, types of Gaussians, and voxelizers.
- Section E elaborates on the Fast Volume Reconstruction method and analyzes its computational complexity.
- Section F details the implementation of adaptive density control in this work.
- Section G presents additional visualizations of DGR reconstruction.

B. Experiment Details

B.1. Code and Reproducibility

Our code is publicly available at `https://github.com/wskingdom/DGR` and is meticulously organized for readability. Experiments are categorized into three groups, each with corresponding code: (1) Cone-Beam Sparse-View CT, (2) Fan-Beam Sparse-View CT, and (3) Fan-Beam Limited-Angle CT. To reproduce our results, please refer to the **Readme.md** file in our codebase.

B.2. Organization of Experiments

To accommodate diverse experimental settings in related research (e.g., varying projection numbers and scanning geometry), our experiments are divided into three groups:

Cone-Beam Sparse-View CT: Experiments with 75, 50, and 25 views were conducted on the FIPS dataset [34], aligning with the settings of R^2-Gaussian [41].

Table 6. Usage of Different Loss Functions

Cone-Beam Sparse-View CT (50 view, 300 iter)				
$\mathcal{L}_1$ Loss	SSIM Loss	TV Loss	PSNR	SSIM
✓			38.66	0.929
✓	✓		38.94	0.933
✓	✓	✓	**39.65**	**0.939**

Table 7. Usage of Different Loss Functions

Fan-Beam Sparse-View CT (60 view, 300 iter)				
$\mathcal{L}_1$ Loss	SSIM Loss	TV Loss	PSNR	SSIM
✓			38.82	0.931
✓	✓		38.88	0.933
✓	✓	✓	**39.15**	**0.936**

Fan-Beam Sparse-View CT: Experiments with 180, 120, 90, and 60 views were performed on the AAPM-Mayo LDCT dataset [30] and the FUMPE dataset [27], following the methodology of the advanced Deep Learning Reconstruction (DLR) method, SWORD [37].

Fan-Beam Limited-Angle CT: Experiments on 90° Limited-Angle CT were conducted on the AAPM-Mayo LDCT dataset [30], consistent with DiffusionMBIR [6].

C. Evaluation Metrics

C.1. Peak Signal-to-Noise Ratio (PSNR)

PSNR [14] is a widely used metric to evaluate the quality of the reconstructed images. It is defined as:

$$\text{PSNR}(x, y) = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}(x, y)} \right), \quad (15)$$

where MAX is the maximum possible pixel value of the image and $\text{MSE}(x, y)$ is the mean squared error between the original and reconstructed images.

C.2. Structural Similarity Index (SSIM)

SSIM [36] is a metric that measures the similarity between two images. It is defined as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (16)$$

where μ_x and μ_y are the mean values of the images x and y, σ_x^2 and σ_y^2 are the variances of the images, σ_{xy} is the covariance of the images, and C_1 and C_2 are constants to stabilize the division when the denominator is small.

C.3. Evaluation Details

For the Cone-Beam 75/50/25-view Sparse-View CT experiments, we assess performance on the FIPS dataset [34] and maintain its original data split for consistency.

Table 8. Usage of Different Loss Functions

Fan-Beam Limited-Angle CT (90°, 300 iter)				
$\mathcal{L}_1$ Loss	SSIM Loss	TV Loss	PSNR	SSIM
✓			37.69	0.932
✓	✓		37.81	0.934
✓	✓	✓	**38.02**	**0.936**

In the Fan-Beam 180/120/90/60-view Sparse-View CT experiments, we evaluate the performance of the reconstruction methods over the entire 3D volume, as established in SWORD [37]. In this context, x and y in the PSNR and SSIM metrics refer to the reconstructed 3D volume and the ground truth 3D volume, respectively.

For the 90° Limited-Angle CT experiments, we evaluate performance across axial, coronal, and sagittal slices following DiffusionMBIR [6]. Here, x and y in the PSNR and SSIM metrics denote the reconstructed 2D slice and the ground truth 2D slice, respectively. The mean PSNR and SSIM values are then computed across all slices.

D. Discussions

D.1. Ablation on Loss Functions

In the main text, we use different loss functions together to enhance reconstruction quality. Specifically, the SSIM loss is computed in the projection domain to preserve structural information, while the TV loss is computed in the volume domain to promote sparsity in the reconstructed volume.

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_1(\hat{P}, P) + \lambda_2 \mathcal{L}_{\text{SSIM}}(\hat{P}, P),$$

where $\lambda_1 = 0.8$ and $\lambda_2 = 0.2$ are the weights of the $\mathcal{L}_1$ Loss and SSIM loss, respectively.

A combined use of $\mathcal{L}_1$ Loss, SSIM loss, and TV loss is as follows:

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_1(\hat{P}, P) + \lambda_2 \mathcal{L}_{\text{SSIM}}(\hat{P}, P) + \lambda_3 \mathcal{L}_{\text{TV}}(V),$$

where $\lambda_1 = 0.6$, $\lambda_2 = 0.2$, and $\lambda_3 = 1$ are the weights of the $\mathcal{L}_1$ Loss, SSIM loss, and TV loss, respectively.

The results for these loss combinations are presented in Table 6, Table 7, and Table 8, corresponding to Cone-Beam Sparse-View CT, Fan-Beam Sparse-View CT, and Fan-Beam Limited-Angle CT experiments on the real-world FIPS dataset [34]. The results demonstrate that the combined use of $\mathcal{L}_1$ Loss, SSIM loss, and TV loss achieves the best performance in terms of PSNR and SSIM metrics.

D.2. Isotropic Gaussians vs Anisotropic Gaussians

The use of isotropic Gaussians is justified by the generally isotropic nature of tissue attenuation properties in CT, especially for soft tissues, which exhibit minimal directional

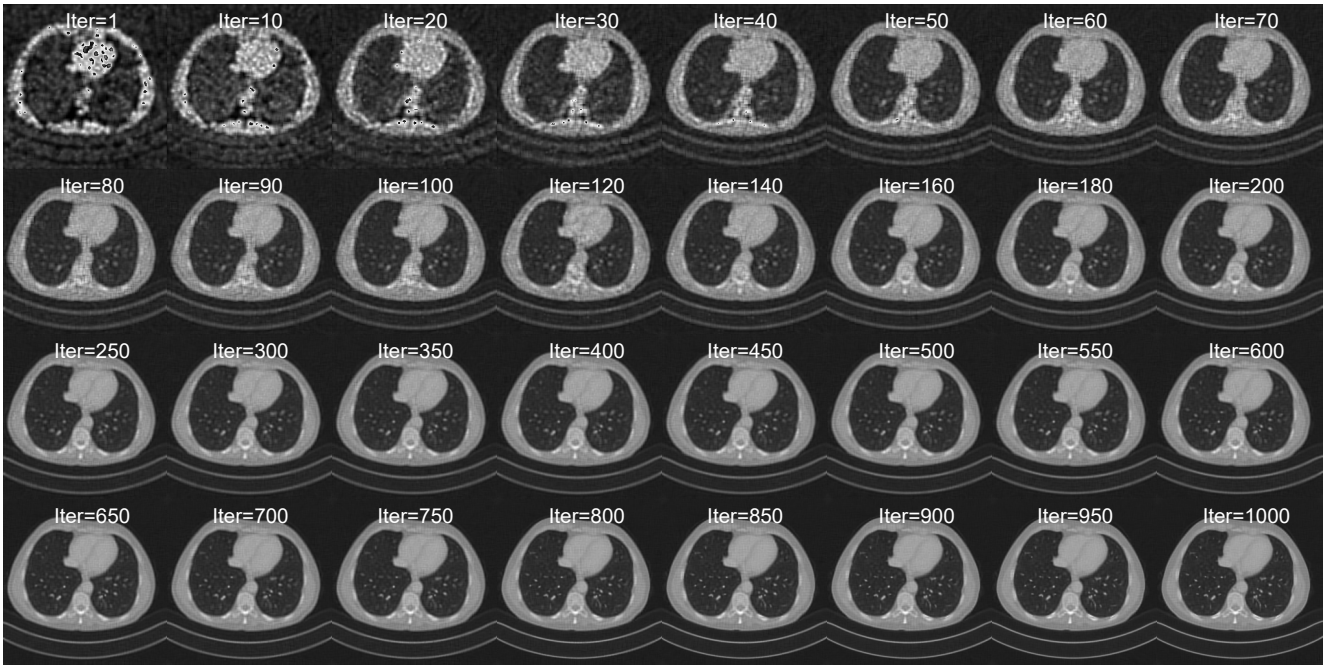


Figure 4. This figure illustrates the reconstruction process by iterations of Fan-Beam 60-view CT. The positions of Gaussians are initialized using Filtered Back Projection, as described in the experimental settings.

Table 9. Comparisons of 180/120/90/60-view Sparse-View CT on FUMPE dataset. Best in **Bold.**

Method	Extra Data	180-view		120-view		90-view		60-view	
		PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
FBP [1]	0	33.74	0.859	30.65	0.803	28.52	0.757	25.59	0.592
MCG [5]	8341	36.93	0.898	37.05	0.899	37.03	0.899	35.46	0.900
DiffusionMBIR [6]	8341	36.78	0.914	36.50	0.907	36.79	0.918	34.86	0.918
SWORD [37]	8341	41.50	0.967	39.69	0.959	37.77	0.948	31.64	0.910
DGR	0	**41.78**	**0.978**	**40.13**	**0.969**	**38.13**	**0.961**	**36.48**	**0.923**

dependence. As noted in Principles of Computerized Tomographic Imaging [17], small elements of a distributed source can be treated as isotropic, reflecting tissue behavior.

To further validate this point, we conduct ablation studies on 50-view Sparse-View CT of Table 1 (Synthetic dataset), repeating experiments 10 times and reporting Mean ± SD in Table 10 to evaluate the result of substituting our isotropic Gaussians with anisotropic Gaussians. Results also validate that using isotropic Gaussians can not only achieve better image quality and stability but also reduce training time.

Table 10. Replace isotropic Gaussians by Anisotropic Gaussians

	PSNR↑	SSIM↑	Time↓
DGR	**38.74± 0.07**	**0.960± 0.01**	**7m13s**
w/ anisotropic Gaussian	38.71± 0.17	0.960± 0.02	8m05s

D.3. Comparison with Other Voxelizer

We also conduct ablation studies on 50-view Sparse-View CT of Table 1 (Synthetic dataset) and repeat experiments 10 times and reporting Mean ± SD in Table 11 to evaluate the result of substituting our Fast Volume Reconstruction with the R^2-Gaussian voxelizer. Results show that our reconstruction module is faster than R^2-Gaussian voxelizer. Besides, our reconstruction module boosts the Gaussian optimization jointly, which achieves better performance.

Table 11. Replace our voxelizer by R^2-Gaussian Voxelizer

	PSNR↑	SSIM↑	Time↓
DGR	**38.74± 0.07**	**0.960± 0.01**	**7m13s**
w/ R^2-Gaussian Voxelizer	38.02± 0.22	0.956± 0.02	8m37s

E. Fast Volume Reconstruction

E.1. Highly Parallelized Implementation

We show the detailed implementation of our Fast Volume Reconstruction In Algorithm 1 and our code. The algorithm takes the mean μ, covariance C, and intensity I of the Gaussians as input and reconstructs the 3D volume in a fast and efficient manner. Notably, this process is implemented in a highly parallelized manner to accelerate the computation without sacrificing the reconstruction quality. More details can be found in our implementation code.

E.2. Analysis of Complexity

Method	VRAM (GiB)	Time (s)
Direct Reconstruction (estimated)	16662.50	/
FVR w/o Decomposition	16.87	1.05
FVR w/ Decomposition	16.87	0.09

As mentioned in the ablation study, the proposed fast volume reconstruction saves both time and space consumption. This difference in complexity primarily stems from the computation of the Mahalanobis distance, D^2.

Direct Reconstruction In the direct reconstruction approach, the Mahalanobis distance is calculated as $D^2 = (P-\mu)^\top \Sigma^{-1}(P-\mu)$. Here, $P \in \mathbb{R}^{w\times h\times c\times d}$ represents the position of a voxel in the volume, where $(d=3)$ indicates the spatial dimension. $\mu \in \mathbb{R}^{n\times d}$ is the set of means for n Gaussians, and $\Sigma \in \mathbb{R}^{n\times d\times d}$ is the covariance matrix for these Gaussians. The computational complexity for direct reconstruction is $O(n\cdot w\cdot h\cdot c\cdot d^2)$. Evidently, this method is not feasible for a large number of Gaussians due to its prohibitive memory consumption.

Fast w/o Decomposition In this condition, the Mahalanobis distance is calculated as $D^2 = \sum_d B''_{n,w_0,h_0,c_0,d}\ C^{-1}_{n,d,d}\ B''_{n,w_0,h_0,c_0,d}$. Here, w_0, h_0, and c_0 denote the dimensions of the local region. The computational complexity for this approach is $O(n\cdot w_0\cdot h_0\cdot c_0\cdot d^2)$. Compared to direct reconstruction, the local region size $(w_0\cdot h_0\cdot c_0)$ is significantly smaller than the full volume size $(w\cdot h\cdot c)$, leading to a substantial reduction in memory consumption.

Fast Volume Reconstruction with Decomposition The decomposed form of Fast Volume Reconstruction yields the Mahalanobis distance as:

$$D^2 = \sum_d \left(B'_{w_0,h_0,c_0,d} - \Delta\mu_{n,1,1,d}\right) C^{-1}_{n,d,d} \left(B'_{w_0,h_0,c_0,d} - \Delta\mu_{n,1,1,d}\right)$$

The space complexity remains equivalent to that of fast volume reconstruction without decomposition. However, the time complexity is significantly reduced to $O(w_0\cdot h_0\cdot c_0\cdot d)$, a benefit derived from the decomposition of the large matrix multiplication.

F. Adaptive Densification

Inspired by the adaptive density control utilized in 3D Gaussian Splatting [19], we implement this mechanism for our DGR, aiming to densify the Gaussians within the discretized 3D volume, employing techniques including cloning, splitting, and pruning. The densification strategy is periodically applied throughout the optimization process, striking a balance between the number of Gaussians and the reconstruction quality.

We clone the Gaussian in under-reconstructed regions into two Gaussians. Specifically, the original and cloned Gaussians share identical positions μ and covariances Σ, but their intensities are halved to preserve the total intensity. This process helps to capture the fine details in the under-reconstructed regions. While the gradients of the original Gaussians remain unchanged, the gradients of the cloned Gaussians are set to zero.

Conversely, in over-reconstructed regions, we split the Gaussian into two smaller Gaussians, keeping their scale σ in proportion to the original one. The positions μ of these new Gaussians are derived from the probability density function (PDF) of the original Gaussian.

Furthermore, we implement a pruning mechanism to eliminate Gaussians that either possess near-zero gradient magnitudes or exhibit a scale σ exceeding three times the local box dimension. For these Gaussians, we simply remove them from the reconstruction volume to reduce the computational cost and prevent overfitting. More details of the algorithm are provided in Algorithm 2 and in our code.

G. Visualization

Figures 4, 5, and 6 visually depict the reconstruction process across various iterations, with the corresponding iteration number indicated for each image. For comparative purposes, Gaussians for Sparse-View CT are initialized as detailed in the main text, whereas those for Limited-Angle CT are initialized randomly. As the iterations progress, the reconstructed volumes consistently show steady improvement, demonstrating the effectiveness of DGR.

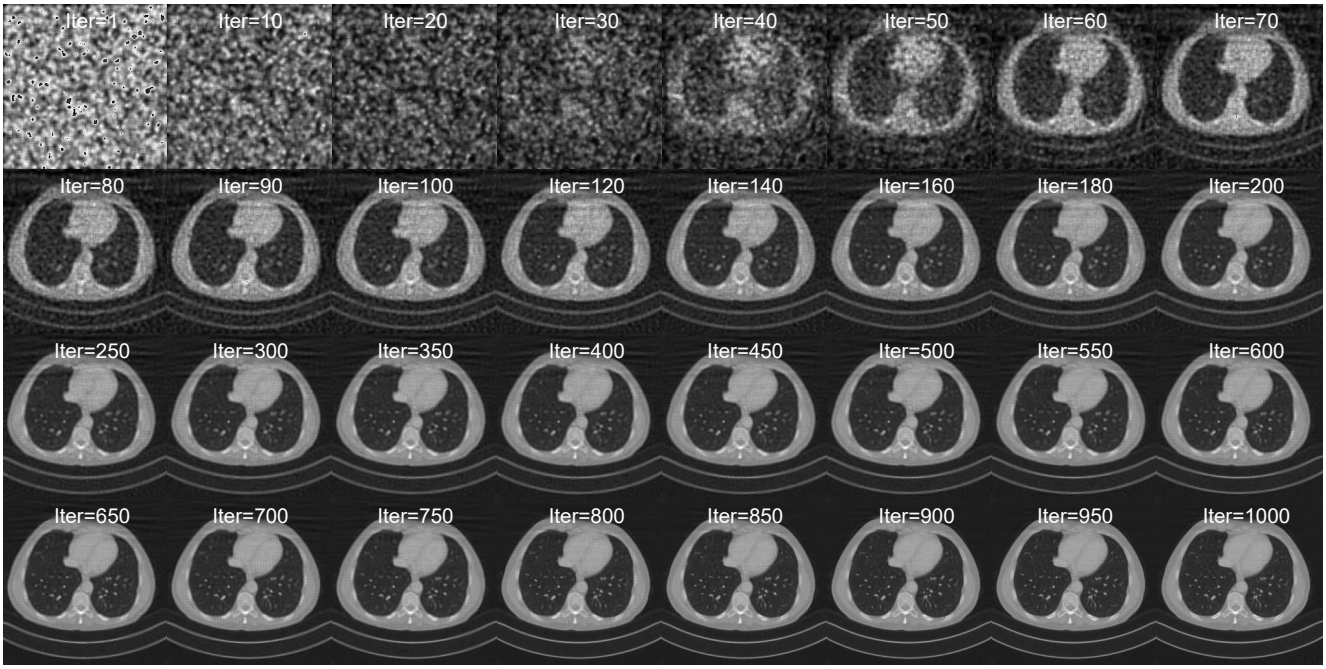


Figure 5. This figure illustrates the reconstruction process by iterations of Fan-Beam 90° Limited-Angle CT. The positions of Gaussians are randomly initialized for visualization comparison.

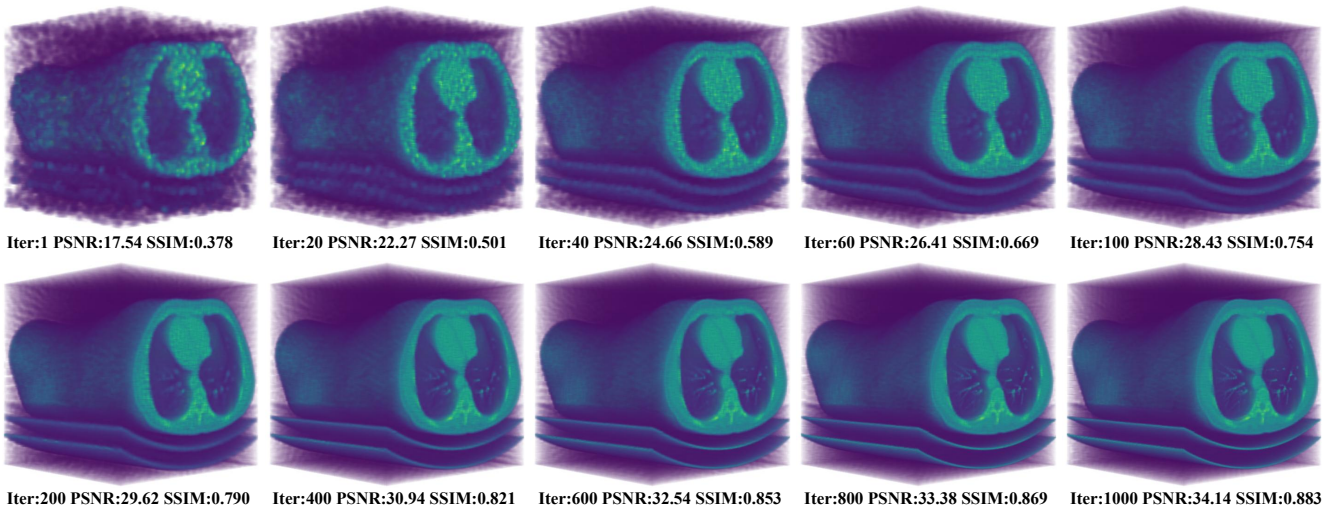


Figure 6. This figure illustrates the reconstruction process by iterations of Fan-Beam 60-view Sparse-View CT by 3D volume visualization.

Algorithm 1 Fast Volume Reconstruction

Require: $\mu \in \mathbb{R}^{n \times d}$: mean of Gaussians ($d = 3$ for 3D space)
Require: $C \in \mathbb{R}^{n \times d \times d}$: covariance of Gaussians
Require: $I \in \mathbb{R}^{n \times 1}$: intensity of Gaussians
$V \leftarrow \mathbf{O}_{w,h,c}$ ▷ Initialize target volume with zeros
$B'_{w_0,h_0,c_0,d} \leftarrow \texttt{meshgrid}(\{-\frac{w_0-1}{2}, \ldots, \frac{w_0-1}{2}\}, \{-\frac{h_0-1}{2}, \ldots, \frac{h_0-1}{2}\}, \{-\frac{c_0-1}{2}, \ldots, \frac{c_0-1}{2}\})$ ▷ Initialize the shift of local regions
$\Delta\mu \leftarrow \mu - \lfloor \mu \rfloor$ ▷ Align μ to discrete voxel grid
$B'^T C^{-1} B'_{n,w_0,h_0,c_0} \leftarrow \sum_d B'_{w_0,h_0,c_0,d}\, C^{-1}_{n,d,d}\, B'_{w_0,h_0,c_0,d}$ ▷ Decompose large matrix multiplication
$B'^T C^{-1} \Delta\mu_{n,w_0,h_0,c_0} \leftarrow \sum_d B'_{w_0,h_0,c_0,d}\, C^{-1}_{n,d,d}\, \Delta\mu_{n,1,1,d}$
$\Delta\mu^T C^{-1} B'_{n,w_0,h_0,c_0} \leftarrow \sum_d \Delta\mu_{n,1,1,d}\, C^{-1}_{n,d,d}\, B'_{w_0,h_0,c_0,d}$
$\Delta\mu^T C^{-1} \Delta\mu_{n,1,1,1} \leftarrow \sum_d \Delta\mu_{n,1,1,d}\, C^{-1}_{n,d,d}\, \Delta\mu_{n,1,1,d}$
$\Gamma \leftarrow e^{-\frac{1}{2}(B'^T C^{-1} B' + B'^T C^{-1} \Delta\mu + \Delta\mu^T C^{-1} B' + \Delta\mu^T C^{-1} \Delta\mu)} \cdot I$ ▷ Compute the Gaussian contributions
$P_{n,w_0,h_0,c_0,d} \leftarrow \lfloor \mu \rfloor_{n,1,1,1,d} + B'$ ▷ Compute the voxel positions that the Gaussians will impact
$Valid_{n,w_0,h_0,c_0} \leftarrow (P_{n,w_0,h_0,c_0,0} \geq 0) \wedge (P_{n,w_0,h_0,c_0,0} < w) \wedge (P_{n,w_0,h_0,c_0,1} \geq 0) \wedge (P_{n,w_0,h_0,c_0,1} < h) \wedge (P_{n,w_0,h_0,c_0,2} \geq 0) \wedge (P_{n,w_0,h_0,c_0,2} < c)$ ▷ Get valid indices that are within the volume
$V \leftarrow \texttt{scatter_add}(V, P[Valid], \Gamma[Valid])$ ▷ Accumulate the contributions at the valid indices in parallel
return V ▷ Reconstructed CT volume

Algorithm 2 Adaptive Densification

Require: $\mu \in \mathbb{R}^{n \times d}$: mean of Gaussians
Require: $\sigma \in \mathbb{R}^{n \times 1}$: standard deviation of Gaussians
Require: $I \in \mathbb{R}^{n \times 1}$: intensity of Gaussians
Require: n_{max}: Maximum allowed quantity of Gaussians (500K by default)
Require: τ: Minimum gradient value (2e-4 by default)
Require: θ: Threshold determining Gaussian classification as small or large (0.005 of body diagonal length by default)
Require: $size$: Box size of each Gaussian (17 by default)
for $iteration \leftarrow 100$ **to** max_iter **step** 100 **do**
$avg_grad \leftarrow \mu.grad / iteration$
$mask_{clone} \leftarrow (avg_grad \geq \tau) \wedge (\sigma \leq \theta)$
$available_gaussians \leftarrow n_{max} - n$ ▷ Ensure the total number of Gaussians does not exceed the limit
$n_{clone} \leftarrow \min(available_gaussians, \sum_{mask_{clone}})$
$sorted_indices \leftarrow \texttt{argsort}(avg_grad[mask_{clone}])$
$mask_{clone} \leftarrow mask_{clone} \wedge \texttt{top_k}(sorted_indices, n_{clone})$ ▷ Select top n_{clone} Gaussians to clone
$I[mask_{clone}] \leftarrow I[mask_{clone}]/2$ ▷ Halve the intensity of the cloned Gaussians to maintain the total intensity
$\mu_{clone} \leftarrow no_grad(\mu[mask_{clone}]),\ \sigma_{clone} \leftarrow no_grad(\sigma[mask_{clone}]),\ I_{clone} \leftarrow no_grad(I[mask_{clone}])$
$\mu \leftarrow \mu \cup \mu_{clone},\ \sigma \leftarrow \sigma \cup \sigma_{clone},\ I \leftarrow I \cup I_{clone}, n \leftarrow n + n_{clone}$
$mask_{split} \leftarrow (avg_grad \geq \tau) \wedge (\sigma > \theta)$
$available_gaussians \leftarrow n_{max} - n$
$n_{split} \leftarrow \min(available_gaussians, \sum_{mask_{split}})$
$sorted_indices \leftarrow \texttt{argsort}(avg_grad[mask_{split}])$
$mask_{split} \leftarrow mask_{split} \wedge \texttt{top_k}(sorted_indices, n_{split})$ ▷ Select top n_{split} Gaussians to split
$\mu_{new} \leftarrow PDF(\mu[mask_{split}], \sigma[mask_{split}], 2)$ ▷ Initialize new Gaussians by PDF sampling
$\sigma_{new} \leftarrow \sigma[mask_{split}]/\sqrt[3]{2}, I_{new} \leftarrow I[mask_{split}]$ ▷ Divide the standard deviation by $\sqrt[3]{2}$ to maintain the total volume
$\mu \leftarrow \mu \cup \mu_{new} - \mu_{split},\ \sigma \leftarrow \sigma \cup \sigma_{new} - \sigma_{split},\ I \leftarrow I \cup I_{new} - I_{split}, n \leftarrow n + n_{split}$
$mask_{prune} \leftarrow (avg_grad) \leq \tau \vee (\sigma > 3 \times size)$
$\mu \leftarrow \mu[\neg mask_{prune}],\ \sigma \leftarrow \sigma[\neg mask_{prune}],\ I \leftarrow I[\neg mask_{prune}]$
end for
return μ, σ, I