

LoopSparseGS: Loop Based Sparse-View Friendly Gaussian Splatting

Zhenyu Bao^{*1,2}, Guibiao Liao^{†1,2}, Kaichen Zhou¹, Kanglin Liu², Qing Li^{†2}, Guoping Qiu³

¹Peking University

²Pengcheng Laboratory

³University of Nottingham

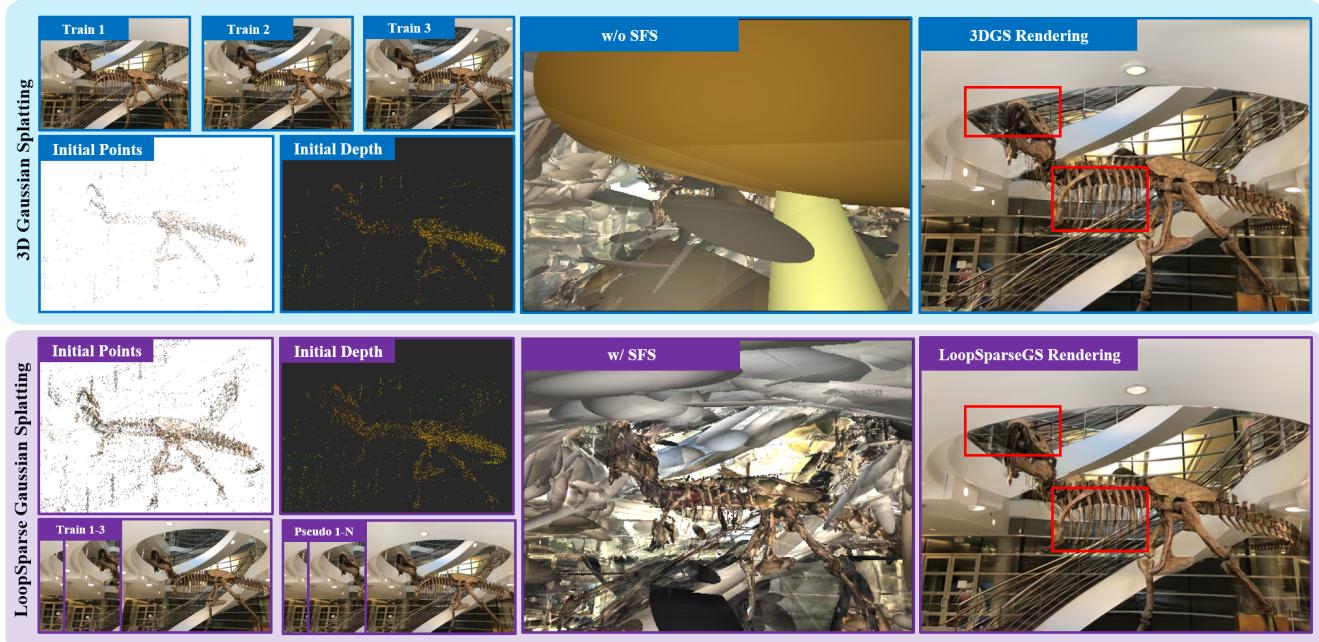


Figure 1. In scenarios with limited input data, the standard 3D Gaussian Splatting (3DGS) method generates insufficient points and minimal depth constraints for training. Our LoopSparseGS employs additional pseudo-cameras to produce more comprehensive initialization points and richer depth information for 3DGS training. Additionally, we found that the excessively large ellipsoids, damage view rendering quality through blurring. To mitigate the issue, we propose a Sparse Friendly Sampling (SFS) strategy to split oversized ellipsoids. The results are presented in the third column, demonstrating the effectiveness of our method.

Abstract

Despite the photorealistic novel view synthesis (NVS) performance achieved by the original 3D Gaussian splatting (3DGS), its rendering quality significantly degrades with sparse input views. This performance drop is mainly caused by the limited number of initial points generated from the sparse input, insufficient supervision during the training process, and inadequate regularization of the oversized Gaussian ellipsoids. To handle these issues, we propose the LoopSparseGS, a loop-based 3DGS framework for the sparse novel view synthesis task. In specific, we propose a loop-based Progressive Gaussian Initialization (PGI)

strategy that could iteratively densify the initialized point cloud using the rendered pseudo images during the training process. Then, the sparse and reliable depth from the Structure from Motion, and the window-based dense monocular depth are leveraged to provide precise geometric supervision via the proposed Depth-alignment Regularization (DAR). Additionally, we introduce a novel Sparse-friendly Sampling (SFS) strategy to handle oversized Gaussian ellipsoids leading to large pixel errors. Comprehensive experiments on four datasets demonstrate that LoopSparseGS outperforms existing state-of-the-art methods for sparse-input novel view synthesis, across indoor, outdoor, and object-level scenes with various image resolutions.

^{*} Github: <https://github.com/pcl3dv/LoopSparseGS>

[†] Corresponding authors

1. Introduction

Novel view synthesis (NVS) aims to generate photorealistic images of 3D scenes from perspectives that were not originally captured [1, 8, 19, 22, 24, 25, 40, 46], which is an essential task in computer vision and graphics field. Recently, 3D Gaussian Splatting (3DGS) [15] has emerged as a promising technique for NVS, as it can efficiently model the highly detailed appearance and geometry of 3D scenes. Such superior performance is usually obtained when large amounts of input images are available. However, in many real-world applications [11, 13, 20, 45], only a few sparse input images are available such as in sports event broadcasting and robotics, where acquiring dense views is often time-consuming and expensive, even impossible. These sparse inputs introduce several challenges to 3DGS. Firstly, given the sparse input views, the initial Gaussian points provided by Structure from Motion (SfM) [31] can be sparse and inadequate, as shown in Fig. 1 (top left). Secondly, reconstructing the appearance and geometry of scenes becomes an under-constrained and ill-posed issue with insufficient inputs with only the image reconstruction constraints. Thirdly, the scales of some Gaussians grow to be very large during the optimization process, and these oversized Gaussian ellipsoids result in the overfitting problem, thus producing unsatisfactory results at novel viewpoints as illustrated in Fig. 1 (top middle)..

Recent studies [5, 18, 48] have attempted to address the aforementioned issues of sparse-input 3DGS. To handle the issue of sparse initialized Gaussian points, FSGS [48] introduces a proximity-guided Gaussian unpooling technique, which generates new Gaussians by measuring the proximity of existing Gaussians with their neighbours during training. This densification strategy, however, is sensitive to noisy points, potentially leading to the generation of invalid points. To mitigate the under-constrained problem, DNGaussian [18] adopts monocular depth, obtained from the pre-trained monocular depth estimator [28], to constrain the depth rendered by 3DGS in a global-local normalization manner. However, these monocular depth maps are often scale-inconsistent across different views, posing a challenge for effective depth regularization. Additionally, previous studies ignore the issue of excessively large Gaussians in sparse-input scenarios, limiting the quality of novel view synthesis.

In this paper, we present the **LoopSparseGS**, a novel 3DGS framework for precise and robust sparse-input novel view synthesis. LoopSparseGS is built upon a looping mechanism and incorporates a sparse-friendly Gaussian densification strategy with the following considerations.

As shown in Fig. 1 (bottom left), we observe that those rendered views close to the training views exhibit high visual quality even with sparse input. This observation motivates us to integrate these pseudo images, i.e., rendered

novel images, with training images to generate additional initialized 3D points using SfM. This process is developed in a looping mechanism to increase the number of initialized points. Consequently, we propose a **Progressive Gaussian Initialization (PGI)** strategy, which leverages both rendered images and training images to iteratively increase initialized Gaussian points, resulting in more comprehensive scene coverage.

Moreover, increased initialized 3D points provide additional precise but sparse depth constraints for 3DGS optimization. Dense monocular depth from the pre-trained model provides dense but scale-invariant depth constraints. To effectively utilize the two constraints, we develop a **Depth-alignment Regularization (DAR)** approach to generate smoother and more precise rendered depths as illustrated in (d) of Fig. 4.

Furthermore, to address the issue of excessively large Gaussian ellipsoids, we propose a **Sparse-friendly Sampling (SFS)** strategy guided by the pixel error. Specifically, SFS identifies and splits the Gaussian ellipsoids associated with high-error pixels that have the largest weights, which could effectively produce more detailed geometric and rendering results, as shown in Fig. 1 (bottom middle). The main contributions of this work are as follows:

- We present the LoopSparseGS, a novel 3DGS-based framework for sparse-input novel view synthesis, featuring a looping mechanism to provide denser Gaussian initialization and precise geometry constraints, and a sparse-friendly sampling strategy to address the oversized Gaussian ellipsoids.
- We develop a Progressive Gaussian Initialization (PGI) method to produce dense 3D Gaussian points by incorporating iteratively rendered images with training images into SfM.
- We propose a Depth Alignment Regularization (DAR) approach that aligns dense relative-scale monocular depths with absolute-scale sparse SfM-derived depths, to provide effective geometric constraints.
- We introduce a Sparse-friendly Sampling (SFS) strategy to address the issue of excessively large Gaussian ellipsoids unique to sparse-input scenes, thus further enhancing the view synthesis quality of the scene.
- Comprehensive experimental results on four datasets demonstrate that our proposed approach outperforms existing state-of-the-art methods in novel view synthesis with sparse-input data, across indoor scenes, outdoor scenes, and object-level scenes.

2. Related Work

2.1. Novel View Synthesis using Radiance Fields

Novel view synthesis techniques typically utilize one or more input views to generate images from novel perspec-

tives. Recent advancements in this field have concentrated on employing radiance fields and achieved encouraging progress. For example, Mildenhall et al. [24] introduce Neural Radiance Field (NeRF) that enables novel view synthesis using coordinate-based neural networks. Tremendous following efforts concentrate on improving its rendering quality [2, 3, 34], efficiency [4, 7, 26, 42], scene understanding [16, 21, 43, 44], and 3D generation [9, 10]. Particularly, Mip-NeRF [2] employs conical frustum instead of single rays to reduce aliasing. Mip-NeRF 360 [3] extends this approach to handle unbounded scenes. Recently, Kerbl et al. [15] achieve a significant breakthrough with the 3D Gaussian Splatting (3DGS) method, which enhances rendering efficiency using explicit Gaussian representations and the differentiable rasterization technique. Building upon its high efficiency in novel view synthesis, several works attempt to extend 3DGS to various tasks. Wu et al. [37] propose an explicit representation method for dynamic scenes utilizing 3D Gaussian and 4D neural voxels. Tang et al. [33] presents a generative 3D Gaussian Splatting model for efficient text-to-3D content creation. Zhou et al. [47] introduce DrivingGaussian for efficient dynamic autonomous driving scene reconstruction.

Although the methods mentioned above demonstrate excellent performance in novel view synthesis, they typically require dense input views for training the radiance fields. When provided with sparse training views, these methods tend to overfit the available training views, resulting in a significant performance drop in novel views.

2.2. Sparse Novel View Synthesis

In recent years, several NeRF-based studies have been proposed to address sparse-input novel view synthesis. Specifically, RegNeRF [27] introduces geometry and color regularization from unobserved viewpoints, enhancing the quality of sparse-input novel view synthesis. It employs a 2D consistency loss on the depth and color of image patches, ensuring that neighboring regions have similar geometry and appearance. InfoNeRF [17] enhances sparse-input view synthesis by employing regularization techniques based on information theory. Specifically, it applies a sparsity constraint on the density distribution of the ray by minimizing entropy. DS-NeRF [6] utilizes sparse depth cues generated by SfM, to impose depth supervision for sparse NeRF. ViP-NeRF [32] enhances the traditional NeRF framework by incorporating the visibility prior, which enforces multi-view constraints during optimization. This modification involves calculating the visibility of a point and using these results to regularize the visibility and alpha-blended depth across different views. FreeNeRF [39] incorporates a frequency regularization strategy designed to train the sparse-input NeRF, aiming to regularize the frequency range of NeRF’s inputs, and the other to penalize the near-camera density

fields. SparseNeRF [35] utilizes a pre-trained depth estimation model to generate pseudo-ground truth depth maps, which are employed for a local depth ranking loss. Besides, SparseNeRF applies a depth smoothness loss to ensure that the rendered depth maps exhibit patch-wise smoothness.

In addition, some 3DGS-based approaches try to tackle the sparse-input novel view synthesis. For example, Yu et al. [5] align sparse depth from SfM with dense depth from a monocular depth estimation model [28] to guide the geometry for sparse-input 3DGS. FSGS [48] integrates estimated monocular depth and employs a Pearson correlation depth distribution loss to train sparse 3D Gaussian Splatting. Likewise, DNGaussian [18] introduces a monocular depth loss and incorporates global-local depth normalization to optimize the parameters of Gaussians. SparseGS [38] proposes generative constraints from a pre-trained diffusion model [30], which guides the 3D Gaussian representation in novel views via Score Distillation Sampling.

Unlike previous methods that utilize scale-inconsistent monocular depth across different views for regularization, our work introduces a Depth-alignment Regularization (DAR) approach. DAR extract accurate and reliable depth values from the SfM points and aligns them with monocular depths using a sliding-window mechanism, providing more effective geometric supervision. Furthermore, our work proposes a loop-based Gaussian initialization, resulting in a denser point cloud. This not only offers more precise depth values for the DAR but also facilitates the training convergence quality of Gaussians. Additionally, we present a sparse-friendly sampling strategy to further enhance Gaussian densification.

3. Method

LoopSparseGS facilitates scene novel view synthesis given sparse input images, and the framework is illustrated in Fig. 2. First, initialized sparse point clouds and camera parameters are obtained using Structure from Motion (SfM). We then introduce a loop-based initialization strategy using pseudo-view rendering results to progressively provide denser Gaussian initialization (Section 3.2). Second, during the Gaussian optimization, we incorporate depth-alignment regularization to impose additional and precise geometric constraints (Section 3.3). Lastly, we adopt a sparse-friendly Gaussian densification approach to sample effective Gaussians for sparse-input reconstruction quality enhancement (Section 3.4). Before introducing our method, we briefly revisit 3D Gaussian Splatting in Section 3.1.

3.1. Preliminaries

3D Gaussian Splatting (3DGS) [15] represents a 3D scene using a set of anisotropic 3D Gaussian primitives, enabling efficient and differentiable rendering via α -blending. The properties of i -th Gaussian primitive can be described as

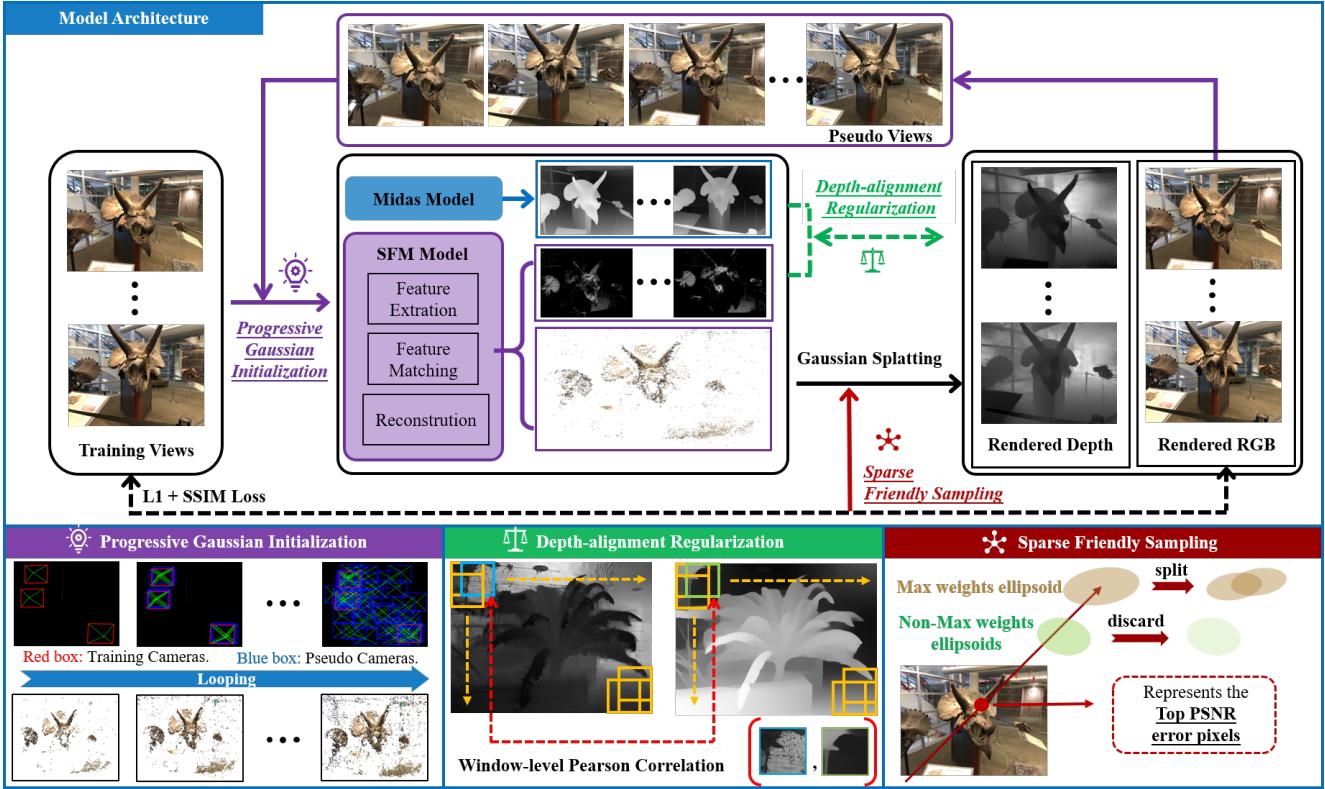


Figure 2. Overview of the proposed **LoopSparseGS**. The LoopSparseGS features three key components: Progressive Gaussian Initialization, Depth Alignment Regularizer and Sparse-friendly sampling. Progressive Gaussian Initialization leverages the training view and high-quality pseudo views near the training view to increase the number of the Gaussian initialized points. Depth Alignment Regularizer incorporates the precise SfM depth and monocular depth and provides a sliding window-based manner to align the two scale-invariant depth regularizers. Sparse-friendly sampling slit large Gaussian ellipsoids of large pixels errors to enhance the representation capacity of large pixel areas.

$\Theta_i = \{u_i, o_i, s_i, q_i, c_i\}$, where $u_i \in \mathbb{R}^3$ is the center, $o_i \in \mathbb{R}$ is the opacity, $s_i \in \mathbb{R}^3$ is the scaling factor, $q_i \in \mathbb{R}^4$ is the rotation, and $c_i \in \mathbb{R}^3$ is the color. To compute the pixel color C , 3DGS employs differentiable α -blending point-based rendering by blending \mathcal{N} Gaussian points in the front-to-back depth order, which can be written as:

$$C = \sum_{i \in \mathcal{N}} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (1)$$

where \mathcal{N} denotes the set of Gaussian points that overlap with the given pixel, and α_i is calculated by $\alpha_i = o_i f_i^{2D}$, where f_i^{2D} represents the projection function of the i -th Gaussian onto the 2D plane.

3DGS is optimized by projecting 3D Gaussians onto the 2D image plane and employing gradient-based color supervision to minimize the distance between the rendered image \tilde{I} and the ground truth image I . This process is as follows:

$$\mathcal{L}_{color} = (1 - \lambda) \mathcal{L}_1(\tilde{I}, I) + \lambda \mathcal{L}_{D-SSIM}(\tilde{I}, I), \quad (2)$$

where λ is set to 0.2 as per [15].

3.2. Progressive Gaussian Initialization (PGI)

Initialization of Gaussian points is crucial for 3DGS-based novel view synthesis, as it significantly impacts the training convergence quality and speed. The original 3DGS relies on dense input images to generate enough initial Gaussian points. However, in scenarios with limited views, the number of 3D points drops dramatically, potentially compromising reconstruction quality. Considering that rendered views close to the training views exhibit satisfactory visual quality, as illustrated in Fig. 1 (bottom left), we develop a **Progressive Gaussian Initialization (PGI)** approach, which combines the rendered images with the original training images to generate additional initialized points. Instead of generating images once, we rely on the iteratively refined 3DGS to produce the high-quality pseudo images progressively. The detailed process is shown in Fig. 2.

Before starting a new loop, we generate 4 new pseudo-views around each training view, resulting in a total of $P \times 4$ new pseudo-views per loop iteration, where P denotes the number of training views. The camera location of the

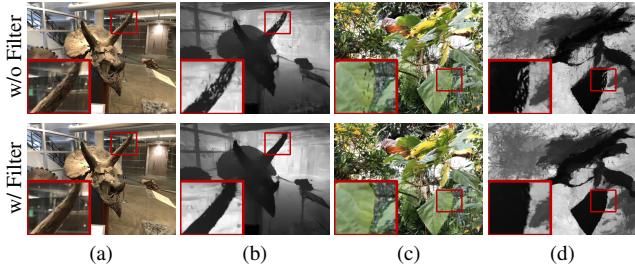


Figure 3. Illustration of the rendered RGB and depth maps without using filter strategy (“w/o” Filter) and utilizing filter strategy (“w” Filter). (a) Rendered image of *Horns*. (b) Rendered depth of *Horns*. (c) Rendered image of *leaves*. (d) Rendered depth of *leaves*. Without filtering, the rendered depth shows significant holes in the edges of the horn and leaves, while the holes are filled up when using our custom filter strategy.

pseudo-views is obtained by adding Gaussian noise to the training camera positions. The locations of the generated pseudo views are computed as follows:

$$z_{gl}^{ij} = N(z_{tj}, \varepsilon + \delta \times l)^i, j \in [1, P], i \in [1, 4] \quad (3)$$

Where z_g and z_t are locations of the pseudo views and training views, respectively. N is Gaussian noise, j denotes the j -th training view, and i denotes the i -th pseudo-view generated from one of the training views for each loop. To ensure the quality of the generated pseudo-view, ε starts from a small value. It gradually enlarges by a rate of δ with loop iterating (l). ε and δ are set to 0.02 and 0.1, respectively. As the number of loops increases, the pseudo view gradually expands the coverage around the training views. To force the view range of the pseudo-images under the coverage of the training views, the locations of pseudo-images should not be out of the bounding box determined by the locations of the training views. The orientation of pseudo-images is the averages of two adjacent training views as in [48]. For each loop, pseudo-images generated from previous loops are accumulated to train the Gaussians.

3.3. Depth-alignment Regularization (DAR)

Beyond providing denser initialized Gaussian points, our loop-based method offers an additional advantage: valuable and reliable depth information derived from these initialized 3D points. Given the inherent inaccuracies and sparsity in the depth supervision from these 3D points, we propose a Depth-alignment Regularization (DAR) strategy, which comprises a Filter Enhancement and a Sliding window-based Alignment. The former aims to enhance the accuracy of SfM-derived depth while the latter incorporates dense monocular depth cues to improve depth regularization for sparse-input 3D Gaussian optimization.

Filter Enhancement. We observed that inaccurate depth information directly derived from SfM can result in erroneous rendering, as illustrated in Fig. 3. The *horns* and *leaves* scenes exhibit significant holes at their edges. To enhance depth reliability, we implemented three filtering strategies according to the reliability and visibility of matched points from certain perspectives.

Filter Strategy 1: Firstly, we ignore the depth of 3D points with large match errors according to the SfM key point match report. We use the average pixel match error to find coarse points. The threshold is set to 2. This filter strategy can be described with the following equation.

$$D(p) = \begin{cases} d, & \text{if } \Upsilon(p) < 2, \\ 0, & \text{if } \Upsilon(p) > 2, \end{cases} \quad (4)$$

where Υ is the average pixel match error and d is the depth value. Note that such points are not used for depth map generation but are yet kept for Gaussian initialization. Although such coarse points can not produce precise depth information they are accurate enough to initialize Gaussian.

Filter Strategy 2: Secondly, considering that pseudo-view images may coincidentally have erroneous regions that satisfy the key points match, we discard points generated solely from pseudo-view. This is illustrated as follows:

$$p = \begin{cases} \text{Keep, if } p \subset M(C_i^t, C_j^t) \cap M(C_i^t, C_j^g), \\ \text{Discard, if } p \subset M(C_i^g, C_j^g), \end{cases} \quad (5)$$

where p is the matched 3D point. C^t and C^g are the training views and pseudo views, respectively. The M indicates that the point is computed from these views. Points are unreliable when they are produced from pseudo-images only rendered by the preliminary trained 3DGs.

Filter Strategy 3: Lastly, considering that the foreground points block the background points in 3D space, we select 3D points to produce the depth images based on visibility provided by the RGB images. For certain view C_i , its corresponding depth D_i of certain 3D point p_j is d , only if p_j derived from view C_i . This can be described under the following conditions:

$$D_i(p_j) = \begin{cases} d, & \text{if } p_j \subset M(C_i, C_k), \\ 0, & \text{other situations,} \end{cases} \quad (6)$$

where $D_i(p_j)$ represents that the depth of point p_j in i -th camera. C_i is the i -th camera. d represents the distance from point p_j to camera C_i .

Sliding window-based Alignment. To impose dense depth regularization, an intuitive approach is to employ the image-level *Pearson* distribution loss between the rendered depth and the monocular depth predicted by the Mi-DaS model [28]. However, directly combining SfM-derived sparse depth and dense monocular depth constraints leads to

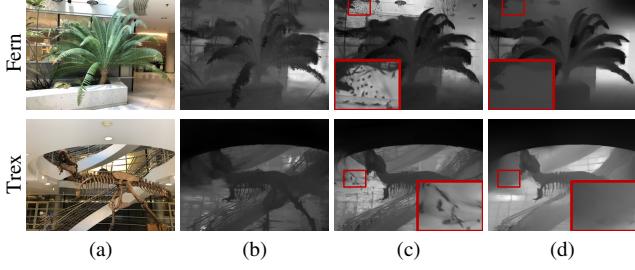


Figure 4. Illustration of the rendered depth maps using different depth supervision. (a) GT image. (b) Using SfM-derived depth supervision. (c) Using SfM-derived depth and Monocular depth supervision without depth alignment. (d) Using SfM-derived depth and Monocular depth with our depth-alignment strategy.

a misalignment issue, manifesting as erroneous black points in the rendered depth maps, as illustrated in (c) of Fig. 4.

This phenomenon is attributed to the limitations of the relative-scale and image-level Pearson constraints, which struggle to impose sufficient constraints on local regions, thereby hindering the alignment of absolute-scale and sparse depths derived from SfM.

To address this misalignment issue, we develop a sliding window-based depth regularization, as illustrated in Fig. 5. We slide the window from left to right, and from the top to down with over the monocular depth and render depth. We compute the Pearson distribution loss of the window-covered region rather than across the entire image. This region-based Person distribution loss can effectively work with the L_1 loss generated from sparsely distributed depth values from SfM as they both are effective locally. Therefore, the final loss function (L_a) is formulated as follows:

$$L_a = \lambda_d L_1(D_p, D_c) + \sum_{w=1}^W \lambda_p L_p[\chi_w(D_p), \chi_w(D_m)], \quad (7)$$

where D_p , D_c , and D_m denote the rendered depth, SfM-derived depth, and mono-depth, respectively. χ_w represents the w -th window, and W is the total number of windows.

This local window-based depth-alignment constraint approach not only effectively enforces the absolute scale depth constraint, but also allows the depth within the window to satisfy the relative scale distribution, thus providing depth-aligned depth constraints for more precise rendered depth as shown in (d) of Fig. 4.

3.4. Sparse-friendly Sampling (SFS)

The Gaussian ellipsoid densification scheme in 3DGS performs effectively with an large number of training images. However, with extremely sparse input data, some Gaussian ellipsoids may grow excessively large with extremely sparse input data and lead to inferior rendered results, as illustrated in (a) of Fig. 6. This is caused by two primary factors: 1)

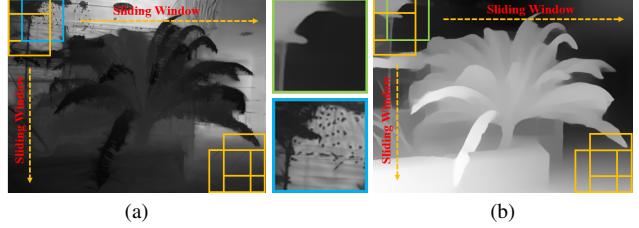


Figure 5. Illustration of sliding window-based sampling strategy in DAR. (a) Rendered depth map. (b) Monocular depth map provided by Midas. Our method begins by sliding a window to obtain the rendering depth and mono depth with the specified window size. Instead of computing the *Pearson* loss over the whole image, we compute the region of sliding window areas, which enlarge the *Pearson* loss in the misaligned regions between SfM-derived depth and mono-depth, as illustrated in the blue box of (a) and the green box of (b) of the middle area.

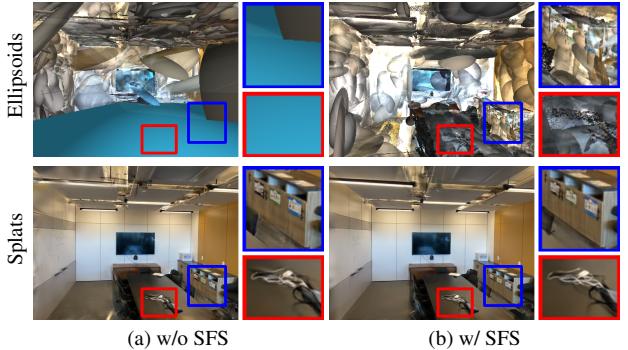


Figure 6. Illustration of ellipsoids and their corresponding rendered results.

The initial scale or size of Gaussian ellipsoids is determined by the average distance from its three nearest neighbours. In cases where the sparse initialized point cloud fully covers the entire space, this method can lead to irrationally large scales for some ellipsoids. 2) The extreme sparsity of input data can cause rapid increases in some ellipsoids' scale along certain directions. This leads to overfitting the training views while significantly deteriorating the performance of novel viewing perspective.

One obvious solution is to increase the frequency of Gaussian densification or lower the Gaussian densification threshold. However, these direct strategies would exacerbate the overfitting of Gaussian Splatting in sparse input settings. To address it, we introduce a non-trivial strategy, named **Sparse-friendly Sampling**, which selectively applies densification to Gaussian ellipsoids that adversely affect rendering. In this way, we enhance the representation of Gaussian ellipsoids of large PSNR error area by splitting the oversized Gaussian ellipsoids to produce more Gaussian ellipsoids.

Figure. 6 (a) shows that the oversized ellipsoids could

lead to the pixels blurring, resulting in lower PSNR values. Based on such observations, we traverse all rendered pixels of the training views and collect pixels with the highest PSNR errors. Since the color of each pixel is derived from multiple ellipsoid primitives through α -blending, we identify the ellipsoid with the largest weight ω (calculated by $\omega = \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)$) as the primary determinant of the pixel's final color. Subsequently, we apply a splitting procedure to the Gaussian primitives, akin to the method employed in the original 3D Gaussian Splatting (3DGS). This process subdivides large ellipsoids into m smaller ellipsoids to enhance the representation of fine details (in our experiments, $m = 2$). Besides, we introduce opacity regularization during the training process to encourage non-maximum weight ellipsoids along a camera ray to be more transparent. These operations can reduce the number of excessively large ellipsoids and enhance detailed geometric and rendered results as shown in (b) of Fig. 6.

3.5. Optimization

We summarize our training constraints as follows:

$$\mathcal{L} = \lambda_1 L_1(C_p, C_{gt}) + \lambda_2 L_{\text{D-SSIM}}(C_p, C_{gt}) + L_a + L_o, \quad (8)$$

where C_p and C_{gt} denote the rendered and GT images, respectively. L_1 and $L_{\text{D-SSIM}}$ represent the photometric and SSIM loss. L_a is the depth-alignment loss computed as Eq. 7. L_o is the non-maximum weight regularization:

$$L_o = \frac{\lambda_o}{N} \sum_{n=1}^N |\alpha_n|, \quad (9)$$

where N denotes the total number of non-maximum weighted ellipses hit by all pixels in an image. In addition, we compute L_1 , $L_{\text{D-SSIM}}$ and L_o for training views and L_a for both training views and pseudo views.

4. Experiment

4.1. Experimental Settings

Datasets. To evaluate our sparse-input method, we conduct experiments on four widely used sparse-view datasets for novel view synthesis: LLFF [23], DTU [14], Mip-NeRF360 [3], and Blender [24]. The LLFF dataset [23] includes eight forward-facing scenes. Following previous methods [18, 48], we select every eighth image as the held-out testing view, and evenly sample sparse views from the remaining images as the training views. For each scene, three views are utilized to train all the approaches. During evaluation, the image resolution are set to 1008×756 and 504×378 .

The Mip-NeRF360 dataset [3] consists of nine scenes, each containing a complex central area or object against an intricate background. As per the protocol in [48], we utilize seven of these scenes for our experiments, employing

24 views with images downsampling rates of 4 and 8 for training all methods.

The DTU dataset [14] is a comprehensive object-level dataset. In line with [18, 35], we use the same 15 scenes with three views for training in the experiments. Consistent with the evaluation protocol of prior research [18, 27], object masks are employed to exclude the background during inference, as evaluating the entire image introduces bias due to background elements.

The Blender dataset [24] includes eight objects rendered with photorealistic images using Blender. Following [27, 48], we use 8 images for training and 25 unseen images for testing.

Evaluation Metrics. For quantitative comparisons, we adopt three evaluation metrics: peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) [36], learned perceptual image patch similarity (LPIPS) to assess the visual quality[41].

Implementation Details. Our approach is implemented using the PyTorch framework, and utilizes the pre-trained Midas Model [28, 29] for zero-shot monocular depth estimation and the Colmap model for initial camera poses and 3D points. We set the total number of loop iterations as 3 and the number of pseudo cameras generated around one training camera l as 4. In a single loop iteration, we start the densification of Gaussian ellipsoids after 1000 iterations, and the frequency of densification is set to 200 iterations. The 2D grad threshold of densification is set to 0.0005. We set the window length to 32 and the step size of the window sliding to 4 for all datasets. For each loop, 3DGS are trained with 10,000 iterations. The weights of the loss function $\lambda_1, \lambda_2, \lambda_o, \lambda_d, \lambda_p$ are set to 0.8, 0.2, 0.05, 0.005 and 0.05, respectively.

Efficiency. Our method achieves average rendering speeds (ARS) of 418, 490, 831, and 720 FPS for the LLFF (3 training views), Mip-NeRF360 (24 training views), DTU (3 training views), and Blender (8 training views) datasets, respectively. Here, the average speed is calculated by $ARS = \frac{1}{20} \sum_{i=1}^{20} \frac{M}{T}$, where T is the total inference time of all test images across all scenes in one dataset, the M is the number of all test images, and we tested 20 times to eliminate random error. The training time of the model depends on the number of loops, with a training time of about 10 min per loop and a total training time of about 35-45 min on all datasets. All experiment times were evaluated on a 3090 NVIDIA GPU.

4.2. Comparison With Existing Methods

Comparisons on LLFF. As shown in Table 1, our proposed method outperforms other state-of-the-art approaches across different image resolutions and sparse-input settings. Specifically, our approach surpasses the second-best FSGS by 0.42 and 0.38 in PSNR at two test resolutions when only

Table 1. Quantitative comparison on LLFF dataset [23].

Method	3 Views (1/8 Resolution)			3 Views (1/4 Resolution)			6 Views			9 Views		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF [2]	16.11	0.401	0.460	15.22	0.351	0.540	22.91	0.756	0.213	24.88	0.826	0.170
DietNeRF [12]	14.94	0.370	0.496	13.86	0.305	0.578	21.75	0.717	0.248	24.28	0.801	0.183
RegNeRF [27]	19.08	0.587	0.336	18.66	0.535	0.411	23.10	0.760	0.206	24.86	0.820	0.161
FreeNeRF [39]	19.63	0.612	0.308	19.13	0.562	0.384	25.13	0.779	0.195	25.13	0.827	0.160
SparseNeRF [35]	19.86	0.624	0.328	19.07	0.564	0.392	24.97	0.784	0.202	24.97	0.834	0.158
3DGSS [15]	17.83	0.582	0.321	16.94	0.488	0.402	22.87	0.732	0.204	24.65	0.813	0.159
DNGaussian [18]	19.12	0.591	0.294	18.47	0.578	0.330	22.18	0.755	0.198	23.17	0.788	0.180
FSGS [48]	20.43	0.682	0.248	19.71	0.642	0.283	24.20	0.811	0.173	25.32	0.856	0.136
Ours	20.85	0.717	0.205	20.19	0.680	0.274	24.58	0.827	0.125	25.86	0.862	0.103

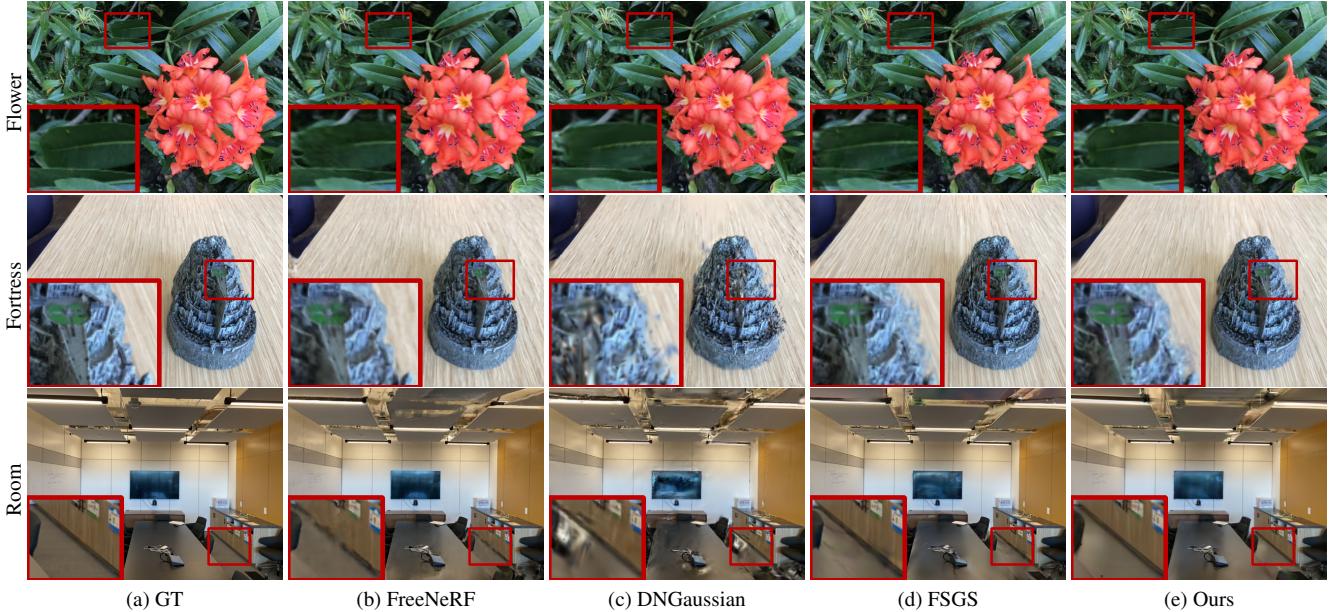


Figure 7. Qualitative Results on LLFF Datasets. Our method can produce photorealistic results with finer details.

using 3 sparse-input views for training. Moreover, we can see that more training views can bring better reconstruction quality and our LoopSparseGS delivers superior performance compared to all other methods, validating the effectiveness of our proposed strategies. We show the qualitative results in Fig. 7. Existing methods tend to produce artifact and blurry rendered results. In comparison, our approach exhibits fine-grained details such as the leaves (Scens: *Flower*) and the floor (Scens: *Room*).

Comparisons on Mip-NeRF360. Table 2 presents the quantitative results in complex scenes from Mip-NeRF360. It can be seen that our method also outperforms other state-of-the-art approaches in terms of various metrics across different image resolutions. Compared to Mip-NeRF requiring dense-input, although methods using regularizations or depth information for sparse-input, such as FreeNeRF and SparseNeRF, enhance rendering quality to some extent,

they still encounter a performance bottleneck. Compared to FSGS that incorporates Gaussian unpooling densification technique and monocular depth maps, our proposed method significantly outperforms it with an improvement of 0.39 and 1.02 in PSNR across two resolutions. These demonstrate the effectiveness of our proposed loop-based mechanism and importance-guided sampling strategy. Moreover, we provide qualitative comparison in Fig. 8. It can be seen that existing methods tend to produce blurred rendered results with incomplete structure, as highlighted by the red boxes around the “plate,” “switch,” and “window.” In contrast, our proposed loop-based and sparse-friendly sampling strategies yield denser initialized Gaussians with effective geometric constraints, resulting in more complete structures and finer details.

Comparisons on DTU. As shown in Table 3, we present the quantitative results on the DTU 3-view sparse-input set-

Table 2. Quantitative comparison on Mip-NeRF360 dataset [3].

Method	24 Views (1/8 Resolution)			24 Views (1/4 Resolution)		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF [2]	21.23	0.613	0.351	19.78	0.530	0.431
DietNeRF [12]	20.21	0.557	0.387	19.11	0.482	0.452
RegNeRF [27]	22.19	0.643	0.335	20.55	0.546	0.398
FreeNeRF [39]	22.78	0.689	0.323	21.39	0.587	0.377
SparseNeRF [35]	22.85	0.693	0.315	21.43	0.604	0.389
3DGS [15]	20.89	0.633	0.317	19.93	0.588	0.401
DNGaussian [18]	22.00	0.683	0.287	21.93	0.668	0.337
FSGS [48]	23.70	0.745	0.230	22.52	0.673	0.313
Ours	24.09	0.755	0.226	23.54	0.722	0.288



Figure 8. Qualitative Results on Mip-NeRF360 Datasets. Our approach can render photorealistic results with more complete structures and finer details.

ting. For the object-level scenes, our method also achieve the best rendering quality in terms of different metrics, with the significant improvement of 0.76 in PSNR. In Fig. 9, we present the visual comparisons. Compared to other methods that produce blurry renderings, our approach can capture color details much closer to the ground truth, demonstrating its effectiveness on object-level real-world scenes.

Comparisons on Blender. Table 4 shows the quantitative results on the Blender dataset with an 8-view sparse-input setting. Our method also significantly outperforms other approaches on the synthesis scenes, achieving a 0.92 improvement in PSNR compared to the second-best method. Furthermore, Fig. 10 illustrates the rendered results. DNGaussian struggles to achieve precise texture and illumination, whereas our approach accurately captures the geometry of objects and authentic reflections. This demonstrates the effectiveness of our proposed strategies for object-level synthesis scenes.

Table 3. Quantitative comparison on DTU dataset [14].

Method	3 Views		
	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF [2]	8.68	0.571	0.353
DietNeRF [12]	11.85	0.633	0.314
RegNeRF [27]	18.89	0.745	0.190
FreeNeRF [39]	19.92	0.787	0.182
SparseNeRF [35]	19.55	0.769	0.201
3DGS [15]	10.99	0.585	0.313
DNGaussian [18]	18.91	0.790	0.176
Ours	20.68	0.856	0.125

4.3. Ablation Experiments

In this section, we ablate our method on the LLFF 3-view with 503×381 image resolution setting. The quantitative results are presented in Tables 5 to 10.

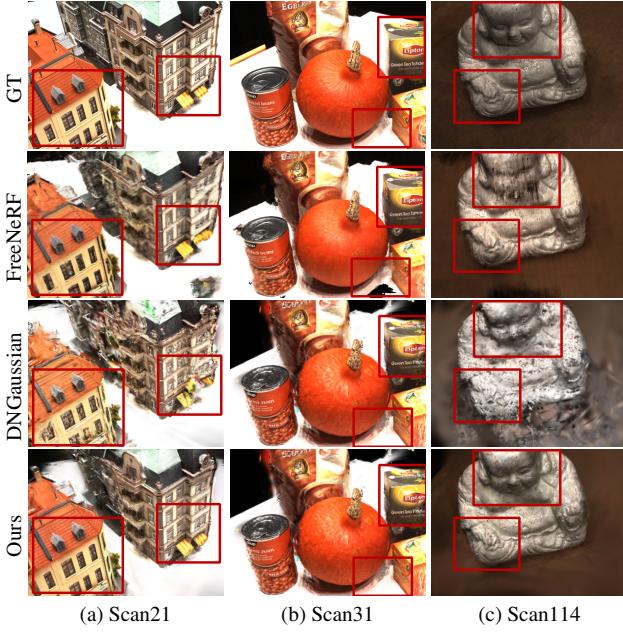


Figure 9. Qualitative results on DTU datasets.

Table 4. Quantitative comparison on Blender dataset [24].

Method	8 Views		
	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF [2]	20.89	0.830	0.168
DietNeRF [12]	22.50	0.823	0.124
RegNeRF [27]	23.86	0.852	0.105
FreeNeRF [39]	24.26	0.883	0.098
SparseNeRF [35]	24.04	0.876	0.113
3DGS [15]	21.56	0.847	0.130
DNGaussian [18]	24.31	0.886	0.088
FSGS [48]	24.64	0.895	0.095
Ours	25.56	0.906	0.075

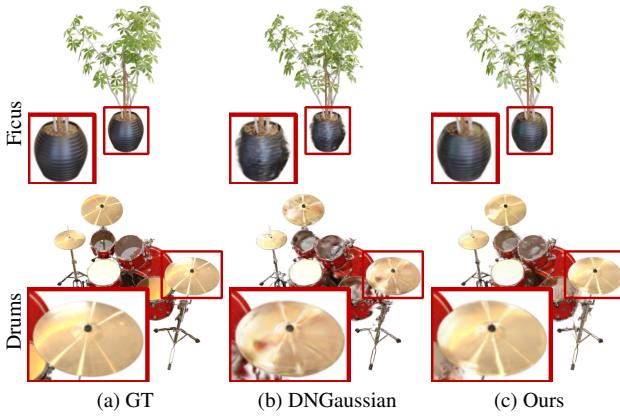


Figure 10. Qualitative results on Blender datasets.

Table 5. Ablation study for our LoopSparseGS.

Index	DAR	PGI	SFS	PSNR↑	SSIM↑	LPIPS↓
(a)				19.137	0.637	0.247
(b)	✓			19.466	0.658	0.236
(c)		✓		19.940	0.682	0.217
(d)			✓	19.419	0.638	0.250
(e)	✓	✓		20.565	0.710	0.208
(f)	✓		✓	20.203	0.677	0.225
(g)		✓	✓	20.158	0.692	0.214
(h)	✓	✓	✓	20.846	0.717	0.205

Table 6. Ablation study for the Depth-alignment Regularization.

Setting	PSNR↑	SSIM↑	LPIPS↓
wo Depth Loss	20.158	0.692	0.214
SfM Depth Loss	20.205	0.690	0.210
Monocular Depth Loss	20.620	0.706	0.215
SfM + Monocular Depth Loss	20.720	0.709	0.207
Proposed Depth-alignment Loss	20.846	0.717	0.205

4.3.1 Effectiveness of architecture modules

In Table 5, we present the ablation results obtained by progressively applying our Depth-alignment Regularization (DAR), Progressive Gaussian Initialization (PGI), and Sparse-friendly Sampling (SFS) strategies. Compared to the baseline (a), each proposed module contributes to improved reconstruction quality, as shown in (b)-(d). Specifically, DAR provides additional depth information to mitigate the under-constrained effect caused by sparse-input, thereby enhancing reconstruction quality. PGI introduces a loop-based initialization strategy and offers denser Gaussians for scene modeling, resulting in a 0.803 increase in PSNR. SFS provides a Gaussian densification strategy suitable for sparse-view training and exhibits a gain of 0.282 PSNR. Furthermore, as shown in (e)-(g), the combination of these strategies yields better results, with all three working together to produce the best performance, as demonstrated in (h).

4.3.2 Effectiveness of depth-alignment loss

Table 6 presents the ablation of our depth-alignment loss through various depth loss configurations. The experiments indicate that both SfM depth loss and monocular depth loss enhance the sparse-input reconstruction quality, with their combination yielding even better results. Moreover, applying our depth-alignment regularization with window-level Pearson correlation loss can further boost the performance. This improvement is attributed to that our window-level operation can effectively align the absolute depth and relative depth constraints and rectify incorrect depth constraints.

Table 7. Ablation study for different filter strategies in PGI.

Index	Filter1	Filter2	Filter3	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
(a)				20.240	0.697	0.222
(b)	✓			20.404	0.700	0.221
(c)		✓		20.661	0.712	0.205
(d)			✓	20.386	0.701	0.217
(e)	✓	✓		20.744	0.715	0.205
(f)	✓		✓	20.629	0.706	0.216
(g)		✓	✓	20.700	0.716	0.202
(h)	✓	✓	✓	20.846	0.717	0.205

Table 8. Ablation study for Sparse-friendly Sampling.

Setting	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
w/o Non-maximum regularization	20.674	0.716	0.192
w/o Maximum split	20.814	0.713	0.217
Proposed Sparse-friendly sampling	20.846	0.717	0.205

4.3.3 Effectiveness of filter strategies

As shown in Table 7, we present the effectiveness of the proposed filter strategies through various configurations. It can be seen that each filter strategy contributes to improved reconstruction quality, as shown in (b)-(d). Moreover, their combination can achieve better results, demonstrating their effectiveness in eliminating unreliable geometric constraints and providing informative supervision for sparse-input view synthesis.

4.3.4 Effectiveness of sparse-friendly sampling

In Table 8, we investigate the effectiveness of the non-max weight regularization and max weight densification operations in SFS. The experiments show that both operations enhance the representation of Gaussian ellipsoids, leading to improved PSNR.

4.3.5 Number of looping

To investigate the number of looping in the Progressive Gaussian Initialization (PGI) strategy, we present comparison results in Table 9. Increasing the number of loops results in more initialization points, which can enhance rendering performance. However, excessive looping may lead to performance degradation. This is primarily because pseudo-views added at later stages may be distant from the training views, leading to inaccurate initialization points and ultimately impairing performance. Therefore, we selected three loops as the final setting for our experiments.

4.3.6 Number of pseudo views

To examine the impact of the number of pseudo views in the Progressive Gaussian Initialization (PGI) strategy,

Table 9. Ablation study for the number of loop in PGI.

Loop Number	Point Number	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
0	1921	20.203	0.677	0.225
1	6370	20.773	0.711	0.209
2	8531	20.773	0.714	0.207
3	9903	20.846	0.717	0.205
4	11078	20.717	0.715	0.206
5	11922	20.761	0.716	0.205

Table 10. Ablation study for the number of pseudo views.

Number of Pseudo Views	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
3	20.812	0.714	0.207
6	20.838	0.715	0.205
12	20.846	0.717	0.205
24	20.764	0.717	0.204
48	20.814	0.716	0.204

we conducted ablation experiments, as shown in Table 10. The results indicate that adding 12 pseudo images at each Gaussian initialization, equivalent to four times the training views produces better results. Thus, we adopt this as the final configuration.

5. Concluding remarks

This paper proposes LoopSparseGS, an innovative 3DGS-based framework for novel view synthesis using sparse input data. In LoopSparseGS, the proposed loop-based strategies, including Progressive Gaussian Initialization (PGI) and Depth-alignment Regularization (DAR), provide denser Gaussians and precise geometric information for effective scene coverage. Additionally, the proposed Sparse-friendly Sampling (SFS) strategy enhances Gaussian densification unique to sparse-input scenes, facilitating the generation of photo-realistic images. Extensive experimental results demonstrate that our approach outperforms existing SOTA methods in sparse-input novel view synthesis across indoor, outdoor, and object-level scenes at various image resolutions.

References

- [1] Zhenyu Bao, Guibiao Liao, Zhongyuan Zhao, Kanglin Liu, Qing Li, and Guoping Qiu. 3d reconstruction and new view synthesis of indoor environments based on a dual neural radiance field. *arXiv preprint arXiv:2401.14726*, 2024. 2
- [2] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 3, 8, 9, 10
- [3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded

- anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 3, 7, 9
- [4] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Proceedings of the European Conference on Computer Vision*, pages 333–350. Springer, 2022. 3
- [5] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3d gaussian splatting in few-shot images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 811–820, 2024. 2, 3
- [6] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022. 3
- [7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 3
- [8] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022. 2
- [9] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. *arXiv preprint arXiv:2110.08985*, 2021. 3
- [10] Lukas Höllerin, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7909–7920, 2023. 3
- [11] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 10608–10615, 2023. 2
- [12] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021. 8, 9, 10
- [13] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Obama, Ganesh Iyer, Soroush Saryazdi, Tao Chen, Alaa Maalouf, Shuang Li, Nikhil Varma Keetha, Ayush Tewari, Joshua Tenenbaum, Celso de Melo, Madhava Krishna, Liam Paull, Florian Shkurti, and Antonio Torralba. Conceptfusion: Open-set multimodal 3d mapping. In *Proceedings of Robotics: Science and Systems*, 2023. 2
- [14] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 406–413, 2014. 7, 9
- [15] 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. 2, 3, 4, 8, 9, 10
- [16] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19729–19739, 2023. 3
- [17] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022. 3
- [18] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20775–20785, 2024. 2, 3, 7, 8, 9, 10
- [19] Guibiao Liao, Jiankun Li, Zhenyu Bao, Xiaoqing Ye, Jingdong Wang, Qing Li, and Kanglin Liu. Clip-gs: Clip-informed gaussian splatting for real-time and view-consistent 3d semantic understanding. *arXiv preprint arXiv:2404.14249*, 2024. 2
- [20] Guibiao Liao, Jiankun Li, and Xiaoqing Ye. Vlm2scene: Self-supervised image-text-lidar learning with foundation models for autonomous driving scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3351–3359, 2024. 2
- [21] Guibiao Liao, Kaichen Zhou, Zhenyu Bao, Kanglin Liu, and Qing Li. Ov-nerf: Open-vocabulary neural radiance fields with vision and language foundation models for 3d semantic understanding. *arXiv preprint arXiv:2402.04648*, 2024. 3
- [22] Tao Lu, Mulin Yu, Lining Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 2
- [23] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics*, 38(4):1–14, 2019. 7, 8
- [24] 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 *Proceedings of the European Conference on Computer Vision*, pages 405–421. Springer, 2020. 2, 3, 7, 10
- [25] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, 2022. 2
- [26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics*, 41(4):1–15, 2022. 3
- [27] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Reg-nerf: Regularizing neural radiance fields for view synthesis

- from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022. 3, 7, 8, 9, 10
- [28] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020. 2, 3, 5, 7
- [29] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021. 7
- [30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 3
- [31] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016. 2
- [32] Nagabhushan Somraj and Rajiv Soundararajan. Vip-nerf: Visibility prior for sparse input neural radiance fields. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023. 3
- [33] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023. 3
- [34] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705, 2023. 3
- [35] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9065–9076, 2023. 3, 7, 8, 9, 10
- [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. 7
- [37] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024. 3
- [38] Haolin Xiong, Sairisheek Muttukuru, Rishi Upadhyay, Pradyumna Chari, and Achuta Kadambi. Sparsesgs: Real-time 360 $\{\backslash\deg\}$ sparse view synthesis using gaussian splatting. *arXiv preprint arXiv:2312.00206*, 2023. 3
- [39] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8254–8263, 2023. 3, 8, 9, 10
- [40] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2
- [41] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 7
- [42] Wenyuan Zhang, Ruofan Xing, Yunfan Zeng, Yu-Shen Liu, Kanle Shi, and Zhizhong Han. Fast learning radiance fields by shooting much fewer rays. *IEEE Transactions on Image Processing*, 2023. 3
- [43] Xiaoyun Zheng, Liwei Liao, Jianbo Jiao, Feng Gao, and Ronggang Wang. Surface-sos: Self-supervised object segmentation via neural surface representation. *IEEE Transactions on Image Processing*, 2024. 3
- [44] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15838–15847, 2021. 3
- [45] Kaichen Zhou. Neural surface reconstruction from sparse views using epipolar geometry. *arXiv preprint arXiv:2406.04301*, 2024. 2
- [46] Kaichen Zhou, Jia-Xing Zhong, Sangyun Shin, Kai Lu, Yiyuan Yang, Andrew Markham, and Niki Trigoni. Dynpoint: Dynamic neural point for view synthesis. *Advances in Neural Information Processing Systems*, 36, 2024. 2
- [47] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21634–21643, 2024. 3
- [48] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *Proceedings of the European Conference on Computer Vision*, 2024. 2, 3, 5, 7, 8, 9, 10