

FSGS: Real-Time Few-shot View Synthesis using Gaussian Splatting

Zehao Zhu^{1*} Zhiwen Fan^{1*} Yifan Jiang¹ Zhangyang Wang¹

¹University of Texas at Austin

Abstract

*Novel view synthesis from limited observations remains an important and persistent task. However, high efficiency in existing NeRF-based few-shot view synthesis is often compromised to obtain an accurate 3D representation. To address this challenge, we propose a Few-Shot view synthesis framework based on 3D Gaussian Splatting that enables real-time and photo-realistic view synthesis with as few as three training views. The proposed method, dubbed **FSGS**, handles the extremely sparse initialized SfM points with a thoughtfully designed Gaussian Unpooling process. Our method iteratively distributes new Gaussians around the most representative locations, subsequently infilling local details in vacant areas. We also integrate a large-scale pre-trained monocular depth estimator within the Gaussians optimization process, leveraging online augmented views to guide the geometric optimization towards an optimal solution. Starting from sparse points observed from limited input viewpoints, our FSGS can accurately grow into unseen regions, comprehensively covering the scene and boosting the rendering quality of novel views. Overall, FSGS achieves state-of-the-art performance in both accuracy and rendering efficiency across diverse datasets, including LLFF, Mip-NeRF360, and Blender. Project website: <https://zehaozhu.github.io/FSGS/>.*

1. Introduction

Novel view synthesis (NVS) from a set of view collections, as demonstrated by recent works [16, 35, 46], has played a critical role in the domain of 3D vision and is pivotal in many applications, e.g., VR/AR and autonomous driving. Despite its effectiveness in photo-realistic rendering, the requirement of dense support views has hindered its practical usages [33]. Previous studies have focused on reducing the view requirements by leveraging Neural Radiance Field (NeRF)[30], a powerful implicit 3D representation that captures scene details, combined with volume rendering techniques[13]. Depth regulariza-

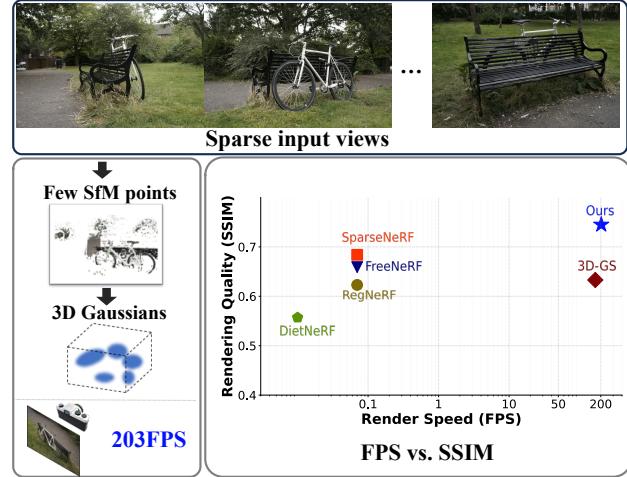


Figure 1. **Real-Time Few-shot Novel View Synthesis.** We present a point-based framework that is initialized from extremely sparse SfM points, achieving a significantly faster rendering speed ($2900\times$) while enhancing the visual quality (from 0.684 to 0.745, in SSIM) compared to the previous SparseNeRF [50].

tions [11, 33, 47, 54] within the density field, additional supervision from 2D pre-trained models [23, 50], large-scale pre-training [8, 59], and frequency annealings [55] have been proposed and adopted to address the challenge of few-shot view synthesis. However, NeRF-based methods incur high training and rendering costs to achieve satisfactory rendering quality. Though some NeRF follow-ups have reduced the training time in real-world scenes from days to hours [40, 43, 51, 58], or even minutes [31], a significant gap remains between achieving real-time rendering speed and photo-realistic high-resolution rendering quality.

3D Gaussian Splatting (3D-GS)[26] has recently emerged as an efficient representation to model the 3D scenes from a dense collection of camera viewpoints. It represents the scene as a combination of 3D Gaussians with attributes for modeling intricate shape and appearances, and 2D images are rendered via splatting-based rasterization[57]. By replacing volume rendering with the efficient differentiable splatting, 3D Gaussian Splatting attains real-time rendering speed while maintaining the abil-

*Equal contribution

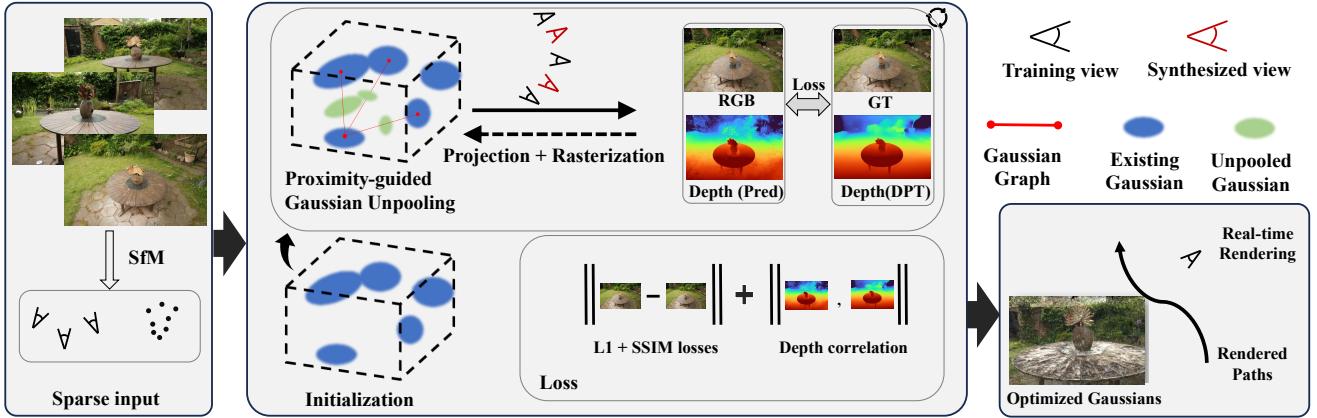


Figure 2. **FSGS Pipeline.** 3D Gaussians are initialized from SfM, with a few images (**black cameras**). For the sparsely placed Gaussians, we propose densifying new Gaussians to enhance scene coverage by unpooling existing Gaussians into new ones, with properly initialized Gaussian attributes. Monocular depth priors, enhanced by sampling unobserved views (**red cameras**), guide the optimization of grown Gaussians towards a reasonable geometry. The final loss consists of a photometric loss term, and a geometric regularization term calculated as depth correlation.

ity to render photo-realistic images from any new viewpoint. However, 3D-GS is initialized from Structure-from-Motion (SfM) points, and its performance strongly relies on both the quantity and accuracy of the initialized points. Although the subsequent Gaussian densification [26] can increase the number of Gaussians in both under-reconstructed and over-reconstructed regions, this straightforward strategy falls short in few-shot settings: it suffers from inadequate initialization, leading to oversmoothed outcomes and a tendency to overfit on training views.

Our goal is to construct a compact and efficient representation while preserving the visual fidelity from limited observations. To this end, we present **FSGS**, to model large-scale scenes from Sparse viewpoint inputs using 3D Gaussians as representation. To address the challenge of mitigating the gap in dense coverage from sparse input, the first challenge is how to move the locations of Gaussians effectively to cover and represent the 3D scene. Therefore, we propose *Proximity-guided Gaussian Unpooling*, where it grows new Gaussians by measuring the proximity of existing Gaussians with their neighbors. By placing new Gaussians in representative locations and initializing them with the information from existing Gaussians, it effectively increases the Gaussian count for comprehensive scene coverage. The next challenge is how to ensure the grown Gaussians correctly represent the geometry of scenes, even when there are not enough multi-view cues? Thus it becomes essential to leverage extra priors to control and regularize the optimization of the new Gaussians. Specifically, we propose to utilize rich depth priors with pre-trained monocular depth estimators on both training and online augmented pseudo views, which guide the Gaussian Unpooling to converge to a reasonable solution and en-

sure the geometric smoothness of the scenes. The back-propagation of the integrated depth prior is achieved by implementing a differentiable depth rasterizer. Validations on few-shot NVS benchmarks: scene-level **LLFF**, **Mip-NeRF360**, and object-level **Blender** datasets, demonstrates that FSGS achieves the state-of-the-art rendering quality, while can run at 203 FPS, a practical speed for real-world applications. Our contributions are outlined below:

- We propose a novel point-based framework for few-shot view synthesis, featuring Proximity-guided Gaussian Unpooling to densify Gaussians for comprehensive scene coverage.
- Our framework integrates monocular depth priors, enhanced by virtually sampled training views, to guide the Gaussian optimization toward an optimal solution.
- Equipped with an enhanced training paradigm, FSGS not only achieves real-time rendering speed (200+FPS) but also improves visual quality, paving the way for practical usage in real-world scenarios.

2. Related Works

2.1. Neural Representations for 3D Reconstruction

The recent advancement of neural rendering techniques, such as Neural Radiance Fields (NeRFs) [30], has shown encouraging progress for novel view synthesis. NeRF learns an implicit neural scene representation that utilizes a MLP to map 3D coordinates (x, y, z) and view dependency (θ, ϕ) to color and density through a volume rendering function. Tremendous works focus on improving its efficiency [7, 15, 17, 26, 31, 38, 43], quality [1, 3, 4, 9, 20, 42, 48, 52], generalizing to unseen scenes [8, 10, 24, 44, 53, 59], applying artistic effects [14, 22, 49, 60] and 3D genera-

tion [5, 6, 18, 21, 25, 27, 28, 34, 41, 45]. In particular, Reiser *et al.* [38] accelerate NeRF’s training by splitting a big MLP into thousands of tiny MLPs. MVSNeRF [8] constructs a 3D cost volume [19, 56] and renders high-quality images from novel viewpoints. Moreover, Mip-NeRF [1] adopts conical frustum rather than a single ray in order to mitigate aliasing. Mip-NeRF 360 [2] further extends it to the unbounded scenes. While these NeRF-like models present strong performance on various benchmarks, they generally require several hours of training time. Muller *et al.* [31] adopt a multiresolution hash encoding technique that reduces the training time to 5 seconds. Kerbl *et al.* [26] propose to use a 3D Gaussian Splatting pipeline that achieves real-time rendering for either objects or unbounded scenes. The proposed FSGS approach is based on the 3D Gaussian Splatting framework but largely reduces the required training views.

2.2. Novel-View Synthesis Using Sparse Views

The original neural radiance field takes more than one hundred images as training views, which largely prohibits its practical usage. To tackle this issue, several works have attempted to reduce the number of training views. Specifically, Deng *et al.* [12] applies additional depth supervision to improve the rendering quality. Niemeyer *et al.* [32] proposes a depth smoothness loss as geometry regularization to stabilize training. DietNeRF [23] adds supervision on high-dimensional semantic space, the CLIP embedding space [36], to constraint the rendered unseen views. Pixel-NeRF [59] trains a convolution encoder to capture context information and learns to predict 3D representation from a single input. More recently, FreeNeRF [55] proposes a dynamic frequency controlling module that successfully trains a NeRF using sparse views with minimal modification. SparseNeRF [50] proposes a new spatial continuity loss to distill spatial coherence from monocular depth estimators. Compared to the aforementioned approaches, our method achieves real time rendering in novel views, and significantly improves the rendering quality.

3. Method

An overview of the FSGS framework is provided in Fig. 2. FSGS inputs with a limited number of images which are captured within a static scene. The camera poses and sparse point clouds are calculated from Structure-from-Motion(SfM) [39]. The initial 3D Gaussians used for further training are initialized from SfM points, with attributes of color, position and shape. The challenge in extremely sparse SfM points and insufficient observations is addressed by adopting *Proximity-guided Gaussian Unpooling* to densify Gaussians and fill the empty space by measuring the proximity between existing Gaussians and strategically placing new ones to the most representative loca-

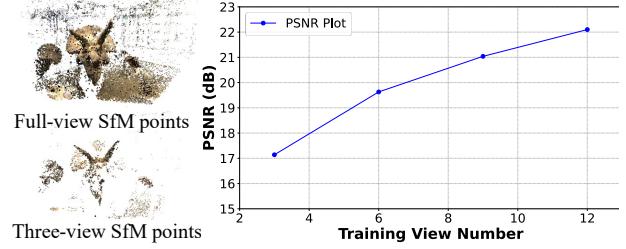


Figure 3. **Sparse SfM Points vs. Synthesized Quality.** The SfM points from COLMAP using 3-views (Bottom Left) is significantly sparse than full-view(Top Left). 3D-GS with sparse SfM points will decrease its quality when the training view number decreases.

tion, to increase the capacity of handling details. To guarantee that the densified Gaussians can be optimized toward the correct scene geometry, we leverage the priors from 2D monocular depth estimator, enhanced by pseudo view generation which prevents our model from overfitting to sparse input viewpoints.

3.1. Preliminary and Problem Formulation

3D Gaussian Splatting (3D-GS), as delineated in Kerbl et al. [26], represents an 3D scene explicitly through a collection of 3D Gaussians, with attributes: a position vector $\mu \in \mathbb{R}^3$ and a covariance matrix $\Sigma \in \mathbb{R}^{3 \times 3}$. Each Gaussian influences a point x in 3D space following the 3D Gaussian distribution:

$$G(x) = \frac{1}{(2\pi)^{3/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)} \quad (1)$$

To ensure that Σ is positive semi-definite and holds practical physical significance, Σ is decomposed into two learnable components by $\Sigma = RSS^T R^T$, where R is a quaternion matrix representing rotation and S is a scaling matrix

In addition, each Gaussians store an opacity logit $o \in \mathbb{R}$ and the appearance feature represented by n spherical harmonic (SH) coefficients $\{c_i \in \mathbb{R}^3 | i = 1, 2, \dots, n\}$ where $n = D^2$ is the number of coefficients of SH with degree D . To render the 2D image, 3D-GS orders all the Gaussians that contributes to a pixel and blends the ordered Gaussians overlapping the pixels using the following function:

$$c = \sum_{i=1}^n c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (2)$$

where c_i is the color computed from the SH coefficients of the i^{th} Gaussian. α_i is given by evaluating a 2D Gaussian with covariance $\Sigma' \in \mathbb{R}^{2 \times 2}$ multiplied by the opacity. The 2D covariance matrix Σ' is calculated by $\Sigma' = JW\Sigma W^T J^T$, projecting the 3D covariance Σ to the camera coordinates. Here, J denotes the Jacobian of the affine approximation of the projective transformation, W is the view transformation matrix.

A heuristic Gaussian densification scheme is introduced in 3D-GS [26], where Gaussians are densified based on an average magnitude of view-space position gradients which exceed a threshold. Although this method is effective when initialized with comprehensive SfM points, it is insufficient for fully covering the entire scene with an extremely sparse point cloud, from sparse-view input images. Additionally, some Gaussians tend to grow towards extremely large volumes, leading to results that overfit the training views and generalize badly to novel viewpoints (See Fig. 3).

3.2. Proximity-guided Gaussian Unpooling

The granularity of the modeled scene depends heavily on the quality of the 3D Gaussians representing the scene; therefore, addressing the limited 3D scene coverage is crucial for effective sparse-view modeling.

Proximity Score and Graph Construction During Gaussian optimization, we construct a directed graph, referred to as the proximity graph, to connect each existing Gaussian with its nearest K neighbors by computing the Euclidean distance. We denote the originating Gaussian at the head as the “source” Gaussian, while the one at the tail as the “destination” Gaussian, which is one of the source’s K neighbors. The proximity score assigned to each Gaussian is calculated as the average distance to its K nearest neighbors. The proximity graph is updated following the densification or pruning process during optimization. We set K to 3 in practice.

Gaussian Unpooling Inspired by the vertex-adding strategy of the mesh subdivision algorithm [61] which is widely used in computer graphics, we propose unpooling Gaussians based on the proximity graph and the proximity score of each Gaussian. Specifically, if the proximity score of a Gaussian exceeds the threshold t_{prox} , our method will grow a new Gaussian at the center of each edge, connecting the “source” and “destination” Gaussians, as shown in Fig. 4. The attributes of scale and opacity in the newly created Gaussians are set to match those of the “destination” Gaussians. Meanwhile, other attributes such as rotation and SH coefficients are initialized to zero. The Gaussian unpooling strategy encourages the newly densified Gaussians to be distributed around the representative locations and progressively fill observation gaps during optimization.

3.3. Geometry Guidance for Gaussian Optimization

Having achieved dense coverage by unpooling Gaussians, a photometric loss with multi-view clues is applied for optimizing Gaussians. However, the insufficient observations in the sparse-view setting limit the capacity for learning coherent geometry, leading to a high risk of overfitting on training views and poor generalization to novel views. This necessitates the incorporation of additional regularization

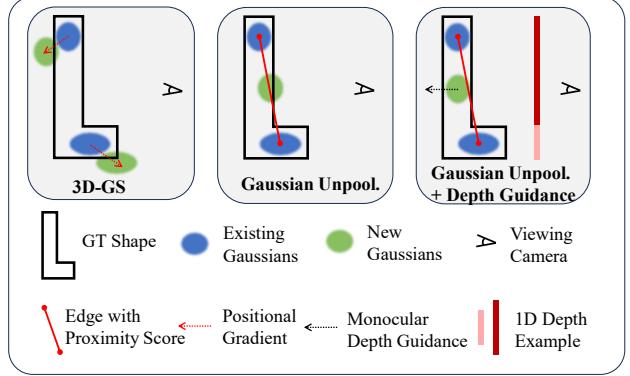


Figure 4. **Gaussian Unpooling Illustration.** We show a 2D toy case for visualizing Gaussian Unpooling with depth guidance, where the example 1D depth can provide priors on the relative distance of the Gaussians from the viewing direction, guide the Gaussian deformation toward a better solution.

and priors to guide the Gaussian optimization. Specifically, we seek help from depth priors produced by a well-trained monocular depth estimator to guide the geometry of Gaussians towards a reasonable solution.

Injecting Geometry Coherence from Monocular Depth

We generate the monocular D_{est} depth maps at training views by using the pre-trained Dense Prediction Transformer (DPT), trained with 1.4 million image-depth pairs as a handy yet effective choice. To mitigate the scale ambiguity between the true scene scale and the estimated depth, we introduce a relaxed relative loss, *Pearson* correlation, on the estimated and rendered depth maps. It measures the distribution difference between 2D depth maps and follows the below function:

$$\text{Corr}(\hat{D}_{\text{ras}}, \hat{D}_{\text{est}}) = \frac{\text{Cov}(\hat{D}_{\text{ras}}, \hat{D}_{\text{est}})}{\sqrt{\text{Var}(\hat{D}_{\text{ras}}) \text{Var}(\hat{D}_{\text{est}})}} \quad (3)$$

This soften constraint allows for the alignment of depth structure without being hindered by the inconsistencies in absolute depth values.

Differentiable Depth Rasterization To enable the back-propogation from depth prior to guide Gaussian training, we implement a differentiable depth rasterizer, allowing for receiving the error signal between the rendered depth D_{ras} and the estimated depth D_{est} . Specifically, we utilize the alpha-blending rendering in 3D-GS for depth rasterization, where the z-buffer from the ordered Gaussians contributing to a pixel is accumulated for producing the depth value:

$$d = \sum_{i=1}^n d_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (4)$$

Here d_i represents the z-buffer of the i^{th} Gaussians and α is identical to that in Eq. 2. The fully differentiable implementation enables the depth correlation loss, further improving the similarity between the rendered and estimated depths.

Synthesize Pseudo Views To address the inherent issue of overfitting to sparse training views, we employ unobserved (pseudo) view augmentation to incorporate more prior knowledge within the scene derived from a 2D prior model. The synthesized view is sampled from the two closest training views in Euclidean space, calculating the averaged camera orientation and interpolating a virtual one between them. A random noise is applied to the 3 degrees-of-freedom (3DoF) camera location as shown in Eq. 5, and then images are rendered.

$$\mathbf{P}' = (\mathbf{t} + \boldsymbol{\varepsilon}, \mathbf{q}), \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \delta) \quad (5)$$

Here, $\mathbf{t} \in \mathbf{P}$ denotes camera location, while \mathbf{q} is a quaternion representing the rotation averaged from the two cameras. This approach of synthesizing online pseudo-views enables dynamic geometry updates, as the 3D Gaussians will update progressively, reducing the risk of overfitting.

3.4. Optimization

Combining all together, we can summarize the training loss:

$$\begin{aligned} \mathcal{L}(\mathbf{G}, \mathbf{C}) = & \underbrace{\lambda_1 \|\mathbf{C} - \hat{\mathbf{C}}\|_1}_{\mathcal{L}_1} + \underbrace{\lambda_2 \text{D-SSIM}(\mathbf{C}, \hat{\mathbf{C}})}_{\mathcal{L}_{\text{ssim}}} \\ & + \lambda_3 \underbrace{\|\text{Corr}(\mathbf{D}_{\text{ras}}, \mathbf{D}_{\text{est}})\|_1}_{\mathcal{L}_{\text{prior}}} \end{aligned} \quad (6)$$

where \mathcal{L}_1 , and $\mathcal{L}_{\text{ssim}}$ stands for the photometric loss term between predicted image $\hat{\mathbf{C}}$ and ground-truth image \mathbf{C} . $\mathcal{L}_{\text{prior}}$ represents the geometric regularization term on the training views and synthesized pseudo views. We set λ_1 , λ_2 , λ_3 as 0.8, 0.2, 0.05 respectively by grid search. The pseudo views sampling is enabled after 2,000 iterations to ensure the Gaussians can roughly represent the scene.

4. Experiments

4.1. Experimental Settings

LLFF Datasets [29] consist of eight forward-facing real-world scenes. Following RegNeRF [33], we select every eighth image as the test set, and evenly sample sparse views from the remaining images for training purposes. We use 3 views to train all the methods, and evaluate them at resolutions of 1008×756 and 504×378 .

Mip-NeRF360 Datasets [2] consist of nine scenes, each featuring a complex central object or area against a detailed background. We utilize the published seven scenes for comparison, using 24 training views with images downsampled to $4 \times$ and $8 \times$. Test images are selected following the same convention with LLFF Datasets. To the best of our knowledge, we are the first to attempt few-shot novel view synthesis in unbounded scenes like Mip-NeRF360. We aim to establish this simple baseline for the challenge of few-shot novel view synthesis in complex large-scale scenarios.

Blender datasets [30] have eight objects with realistic images synthesized by Blender. We align with DietNeRF [23], where we use 8 images for training and 25 for testing. We evaluate all methods at resolution of 400×400 .

Baselines We compare FSGS with several few-shot NVS methods on these three dataset, including DietNeRF [23], RegNeRF [33], FreeNeRF [55], and SparseNeRF [47]. Additionally, we include comparisons with the high-performing MLP-based NeRF, Mip-NeRF [3], which is primarily designed for dense-view training, and point-based 3D-GS, following its original dense-view training recipe. Following [26, 33, 50], we report the average PSNR, SSIM, LPIPS scores and FPS for all the methods.

Implementation Details We implemented FSGS using the PyTorch framework, with initial camera poses and point clouds computed from SfM, based on the specified number of training views. During optimization, we densify the Gaussians every 100 iterations and perform densification after 500 iterations. The total optimization steps are set at 10,000 for all datasets, requiring approximately 9.5 minutes on Blender and LLFF datasets, and ~ 24 minutes on Mip-NeRF360 datasets. We set proximity threshold t_{prox} to 10, and the pseudo views are sampled after 2,000 iterations, with σ set as 0.1. We utilize the pre-trained Dense Prediction Transformer (DPT) model [37] for zero-shot monocular depth estimation. All results are obtained using an NVIDIA A6000 GPU.

4.2. Comparisons to other Few-shot Methods

Comparisons on LLFF Datasets As shown in Tab. 1, our method FSGS, despite trained from sparse SfM point clouds, provides the best quantitative results and effectively addresses the insufficient scene coverage in the initialization. Our method surpasses SparseNeRF by 0.57 and 0.64 in PSNR at both test resolutions, while operating 2180 times faster. FSGS also outperforms 3D-GS by 2.60 in PSNR and boost the FPS from 385 to 458, demonstrating that our refined Gaussians are more compact and precise for scene representation from sparse views. The real-time rendering speed makes FSGS a viable choice for practical usages.

The qualitative analysis, as presented in Fig. 5, demonstrates that Mip-NeRF and 3D-GS struggle with the extreme sparse view problem; Mip-NeRF leads to degraded geometric modeling, and 3D-GS tends to produce blurred results in areas of complex geometry. The geometry field regularization in RegNeRF and frequency annealing in FreeNeRF do improve the quality to some extent, but still exhibit insufficient visual quality. SparseNeRF, which also utilizes the monocular depth maps for optimization, manifests as less textural details. In contrast, our proposed Gaussian unpooling, pulls more Gaussians to the unobserved regions and thus recovers more structural details in novel views.

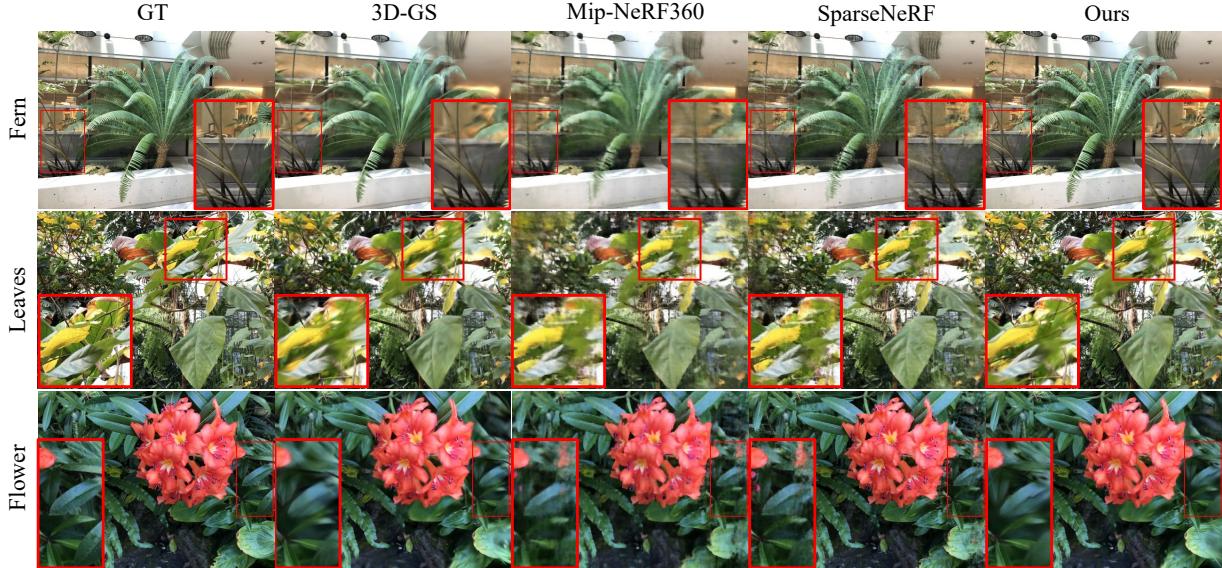


Figure 5. **Qualitative Results on LLFF Datasets.** We demonstrate novel view results produced by 3D-GS [26], Mip-NeRF360 [2], SparseNeRF [50] and our approach for comparison. We can observe that NeRF-based methods generate floaters (Scene: *Flower*) and show aliasing results (Scene: *Leaves*) due to limited observation. 3D-GS produces oversmoothed results, caused by overfitting on training views. Our method produces pleasing appearances while demonstrating detailed thin structures.

Methods	503 × 381 Resolution				1006 × 762 Resolution			
	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF	0.21	16.11	0.401	0.460	0.14	15.22	0.351	0.540
3D-GS	385	17.83	0.582	0.321	312	16.94	0.488	0.402
DietNeRF	0.14	14.94	0.370	0.496	0.08	13.86	0.305	0.578
RegNeRF	0.21	19.08	0.587	0.336	0.14	18.66	0.535	0.411
FreeNeRF	0.21	19.63	0.612	0.308	0.14	19.13	0.562	0.384
SparseNeRF	0.21	19.86	0.624	0.328	0.14	19.07	0.564	0.392
Ours	458	20.43	0.682	0.248	351	19.71	0.642	0.283

Table 1. **Quantitative Comparison in LLFF Datasets, with 3 Training Views.** FSGS achieves the best performance in terms of rendering accuracy and inference speed across all resolutions. Significantly, FSGS runs 2,180× faster than the previous best, SparseNeRF, while improving the SSIM from 0.624 to 0.682, at the resolution of 503 × 381. We color each cell as **best**, **second best**, and **third best**.

Comparisons on Mip-NeRF360 Datasets As shown in Tab. 2, methods requiring dense view coverage (Mip-NeRF360, 3D-GS) are outperformed by ours in terms of rendering speed and metrics, across the two resolutions. Methods employing regularizations from their respective geometry and appearance fields (DietNeRF, RegNeRF, FreeNeRF) still suffer from rendering quality. SparseNeRF, by incorporating depth ranking information into NeRF, enhances quality to a degree, but remains far from achieving real-time speed. Our FSGS significantly outperforms NeRF-based approaches, boosting PNSR by 0.85 and improving FPS from 0.07 to 290 at 1/8 resolution. We provide a qualitative comparison in Fig. 9, where we observe that Mip-NeRF360 and SparseNeRF fail to capture the intricate details of scenes and tend to overfit on sparse training views, most notably in areas far away from cameras. In

comparison, FSGS recovers the fine-grained details such as the leaves on the ground (Scene: *Stump*) and the piano keys (Scene: *Bonsai*), aligning well with the ground truth.

Comparisons on Blender Datasets Tab. 3 presents the quantitative results on the Blender datasets. Here, our method significantly outperforms the baselines on object-level datasets, with an improvement of 0.40 in PSNR compared to FreeNeRF, although primarily designed for scene-level scenarios with complex geometry. Fig. 7 visualizes the rendered image. We find that DietNeRF hallucinates geometric details, and FreeNeRF exhibits noticeable aliasing effects leading to worse geometry. 3D-GS suffers from excessive blurriness and distorts the edges of the objects. In contrast, our model not only captures the precise geometry of objects but also accurately simulates the shading effects.



Figure 6. **Qualitative Results on Mip-NeRF360 Datasets.** Comparisons were conducted with 3D-GS [26], Mip-NeRF360 [2], and SparseNeRF [50]. Our method continues to produce visually pleasing results with sharper details than other methods in large-scale scenes.

Methods	1/8 Resolution				1/4 Resolution			
	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF360	0.12	21.23	0.613	0.351	0.07	19.78	0.530	0.431
3D-GS	223	20.89	0.633	0.317	145	19.93	0.588	0.401
DietNeRF	0.05	20.21	0.557	0.387	0.03	19.11	0.482	0.452
RegNeRF	0.07	22.19	0.643	0.335	0.04	20.55	0.546	0.398
FreeNeRF	0.07	22.78	0.689	0.323	0.04	21.39	0.587	0.377
SparseNeRF	0.07	22.85	0.693	0.315	0.04	21.43	0.604	0.389
Ours	290	23.70	0.745	0.230	203	22.52	0.673	0.313

Table 2. **Quantitative Comparison in Mip-NeRF360 Datasets, with 24 Training Views.** Our FSGS shows obvious advantages over NeRF-based methods, with an improvement of more than 0.05 in SSIM and running $4,142\times$ faster. Additionally, our method not only performs better than 3D-GS in rendering metrics but also shows improvement in FPS (from 223 to 290), thanks to the Gaussian unpooling which motivates Gaussians to expand to unseen regions more accurately.

4.3. Ablation Studies

In Tab. 4, we ablate our design choices on the the LLFF dataset under the 3-view setting.

Effectiveness of Gaussian Unpooling As shown in the second row of Tab. 4, our Gaussian Unpooling expands the scene geometry caused by limited training views, resulting in a PSNR improvement of 0.81 compared to 3D-GS. We also visualize its visual effects in Fig. 8 (2nd row). The heuristic Gaussian densification leads to blurring results, particularly noticeable in areas like bush and grass, our approach enriches structural and visual details.

Impact of Monocular Depth Prior Tab. 4 (3rd row)

demonstrates the improvement by introducing depth priors, guiding the Gaussian unpooling towards more plausible geometry. In Fig. 8, we observe that the depth regularization effectively eliminates the artifacts in grassy regions, and enforces more consistent and solid surfaces with geometric coherence. We also display the rendered depth map, where depth regularization leads to depths aligning better with the actual geometric structures.

Pseudo-view Matters in Few-shot Modeling Tab. 4 (4th row) validates the impact of synthesizing more unseen views during training, which anchors the Gaussians to a plausible geometry and further enhances the modeling quality when the geometry in densification is not accurate.

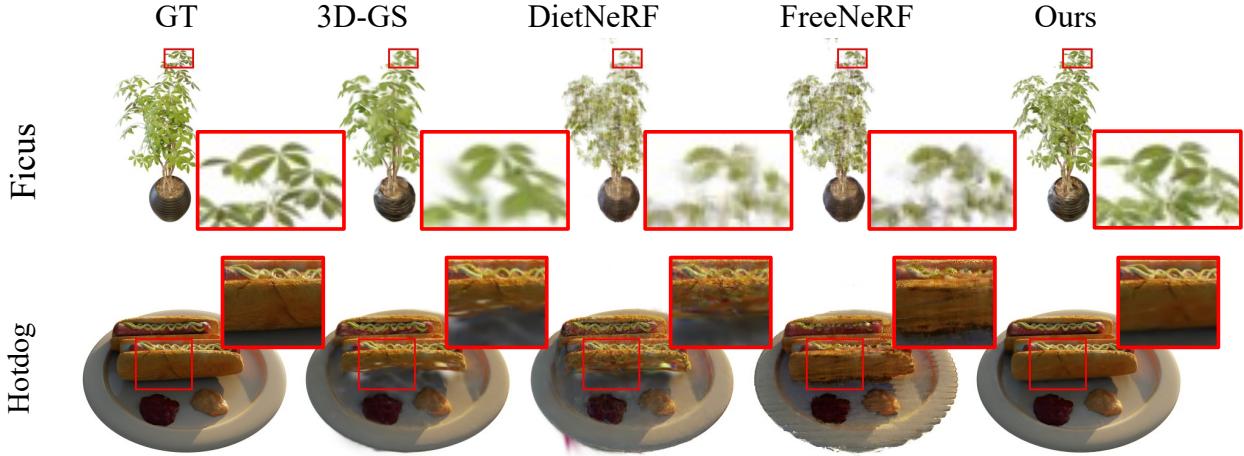


Figure 7. **Qualitative Results on Blender Datasets.** Our method consistently outperforms other baselines in the task of novel view synthesis for object-centric datasets.

Method	FPS↓	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF	0.22	20.89	0.830	0.168
3D-GS	332	21.56	0.847	0.130
DietNeRF	0.14	22.50	0.823	0.124
RegNeRF	0.22	23.86	0.852	0.105
FreeNeRF	0.22	24.26	0.883	0.098
SparseNeRF	0.22	24.04	0.876	0.113
Ours	467	24.64	0.895	0.095

Table 3. **Quantitative Comparison in Blender Datasets, with 8 Training Views.** FSGS outperforms existing few-shot methods and 3D-GS across all metrics, validating the generalization of the proposed techniques to handheld object-level 3D modeling.

Gaussian Unpooling	Geometry Guidance	Pseudo Views	PSNR↑	SSIM↑	LPIPS↓
✗	✗	✗	17.83	0.582	0.321
✓	✗	✗	18.64	0.603	0.311
✓	✓	✗	19.93	0.644	0.283
✓	✓	✓	20.43	0.682	0.248

Table 4. **Ablation Study.** Starting from 3D-GS [26] (1st row), we find that our proposed *Gaussian Unpooling* (2nd row) is more effective than the densification scheme in 3D-GS for few-shot view synthesis. Applying additional supervision from a monocular depth estimator further regularizes the Gaussian optimization towards a better solution (3rd row). Introducing pseudo-view augmentation to apply additional regularization when optimizing Gaussians further enhances the results in a few-shot scenario.

5. Conclusion

In this work, we present a real-time few-shot framework, FSGS, for novel views synthesis within an insufficiently view overlapping. Starting from extremely sparse SfM point cloud, FSGS adopts the point-based representation

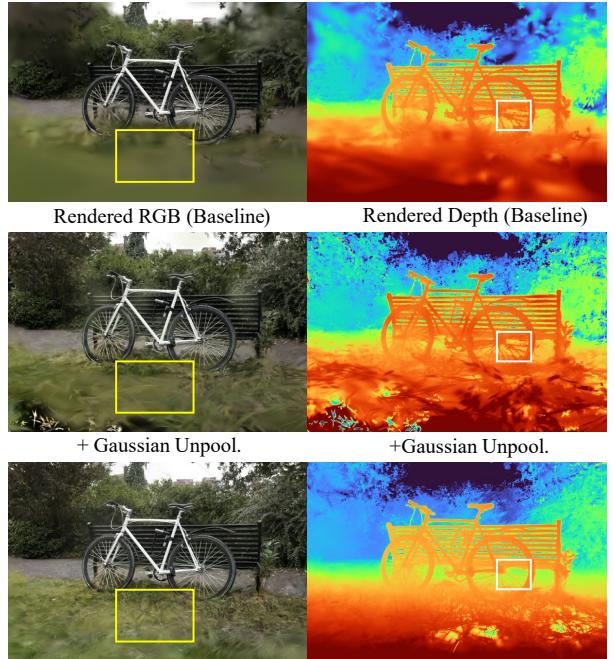


Figure 8. **Ablation Study by Visualization.** 3D-GS [26] (1st row) shows that the baseline method is significantly degraded when the view coverage is insufficient. *Gaussian Unpooling* provides extra capacity to 3D Gaussians to model the scene, but the learned geometry may not be accurate. Adding *Depth Correlation* regularization (3rd row) can further improve the modeled details.

and proposes an effective Gaussian unpooling method by measuring the proximity of each Gaussian to its neighbor. The adoption of monocular depth priors has proven effective to guide the expanded scene geometry toward a better solution. FSGS is capable of generating photo-realistic images with as few as three images, and perform inference at more than 200FPS, offering new avenues for real-time rendering and more cost-effective capture methods.

A. Appendix Part I: More Technical Details

A.1. Initialization

Similar to 3D Gaussian Splatting [26], we start our pipeline from unstructured multi-view images, and calibrate the images using Structure-from-Motion [39]. Next, we will continue the dense stereo matching under COLMAP with the function “patch_match_stereo” and utilize the fused stereo point cloud from “stereo_fusion”. We then initialize the SH coefficients at degree 0 and the positions of the 3D Gaussians based on the fused point cloud. Additionally, we remain the rest coefficients and rotation to 0. We also initialize the opacity to 0.1 and set the scale to match the average distance between points.

A.2. Training

During training, we start with a SH degree of 0 for a basic lighting representation, incrementing by 1 every 500 iterations up to a degree of 4 to increase complexity over time. We set the learning rate of position, SH coefficients, opacity, scaling, and rotation to 0.00016, 0.0025, 0.05, 0.005, and 0.001 respectively. At iterations 2000, 5000, and 7000, the opacity for all Gaussians is reset to 0.05 to eliminate the low-opacity floaters. In the Blender dataset [30], the Pearson correlation is only computed in pixels where the depth values are greater than 0. Additionally, we utilize an open-source code¹ to estimate the inverse depth map for both the input images and the rendered images from pseudo views. We detail the procedures of our proposed FSGS in Algorithm 1.

B. Appendix Part II: More Experiments

B.1. Effects of Training Views

We demonstrate the quantitative results of FSGS on LLFF datasets under 3, 6, 9 views in Tab. 5. We can observe that more views consistently exhibit higher visual quality. In Fig. 9, we present a qualitative comparison. We can observe that training with fewer views often leads to overfitting on sparse training data. In contrast, training with 9 views continues to capture the fine-grained structures of the scenes. More views provide a more comprehensive coverage of the scene, capturing more details of the scenes. This richness in data boosts supervision signals during optimization, leading to more detailed and structural texture. Across all the settings, FSGS delivers superior performance compared to all the baselines, affirming the effectiveness of our proposed depth prior.

¹https://pytorch.org/hub/intelisl_midas_v2/

Algorithm 1 The training pipeline of FSGS

```

1: Training view images  $\mathcal{I} = \{I_i \in \mathbb{R}^{H \times W \times 3}\}_{i=1}^N$  and
   their associated camera poses  $\mathcal{P} = \{\phi_i \in \mathbb{R}^{3 \times 4}\}_{i=1}^N$ .
2: Run SfM with the input images and camera poses and
   obtain an initial point cloud  $\mathcal{P}$ , used to define 3D Gaussians function  $\mathcal{G} = \{G_i(\mu_i, \sigma_i, c_i, \alpha_i)\}_{i=1}^K$ .
3: Leverage pretrained depth estimator  $\mathcal{E}$  to predict the
   depth map  $D_i = \mathcal{E}(I_i)$ .
4: Synthesize pseudo views  $\mathcal{P}^\dagger = \{\phi_i^\dagger \in \mathbb{R}^{3 \times 4}\}_{i=1}^M$  from
   input camera poses  $\mathcal{P}$ .
5: while until convergence do
6:   Randomly sample an image  $I_i \in \mathcal{I}$  and the corre-
      sponding camera pose  $\phi_i$ 
7:   Rasterize the rgb image  $\hat{I}_i$  and the depth map  $\hat{D}_i$ 
      with camera pose  $\phi_i$ 
8:    $\mathcal{L} = \lambda_1 \|I_i - \hat{I}_i\|_1 + \lambda_2 \text{D-SSIM}(I_i, \hat{I}_i) +$ 
       $\lambda_3 \text{Pearson}(D_i, \hat{D}_i)$ 
9:   if iteration >  $t_{iter}$  then
10:    Sample a pseudo camera pose  $\phi_j^\dagger \in \mathcal{P}^\dagger$ .
11:    Rasterize the rgb image  $\hat{I}_j$  and the depth  $\hat{D}_j$ 
12:    Compute the estimated depth as  $D_j^\dagger = \mathcal{E}(\hat{I}_j)$ 
13:     $\mathcal{L} = \mathcal{L} + \lambda_4 \text{Pearson}(D_j^\dagger, \hat{D}_j)$ 
14:   end if
15:   if IsRefinement(iteration) then
16:     for  $G_i(\mu_i, \sigma_i, c_i, \alpha_i) \in \mathcal{G}$  do
17:       if  $\alpha_i > \varepsilon$  or IsTooLarge( $\mu_i, \sigma_i$ ) then
18:         RemoveGaussian()
19:       end if
20:       if  $\nabla_p \mathcal{L} > t_{pos}$  then
21:         GaussianDensify()
22:       end if
23:       if NoProximity( $\mathcal{G}$ ) then
24:         GaussianUnpooling()
25:       end if
26:     end for
27:   end if
28:   Update Gaussians parameter  $\mathcal{G}$  via  $\nabla_{\mathcal{G}} \mathcal{L}$ .
29: end while

```

B.2. FSGS on Mobile Phones Data

To validate the generalization capability of FSGS in various real-world settings, we created a new dataset using only a consumer smartphone, the iPhone 15 Pro. This dataset contains three scenes, comprising two indoor scenes and one outdoor scene. Each scene consists of a collection of RGB images under 5712×4284 resolution, with the viewpoint number ranging from 20 to 40. Our data calibration pipeline follows the same process procedures as the LLFF datasets [29], and we also select every 8-th image as the novel views for evaluation. For training, we evenly sam-

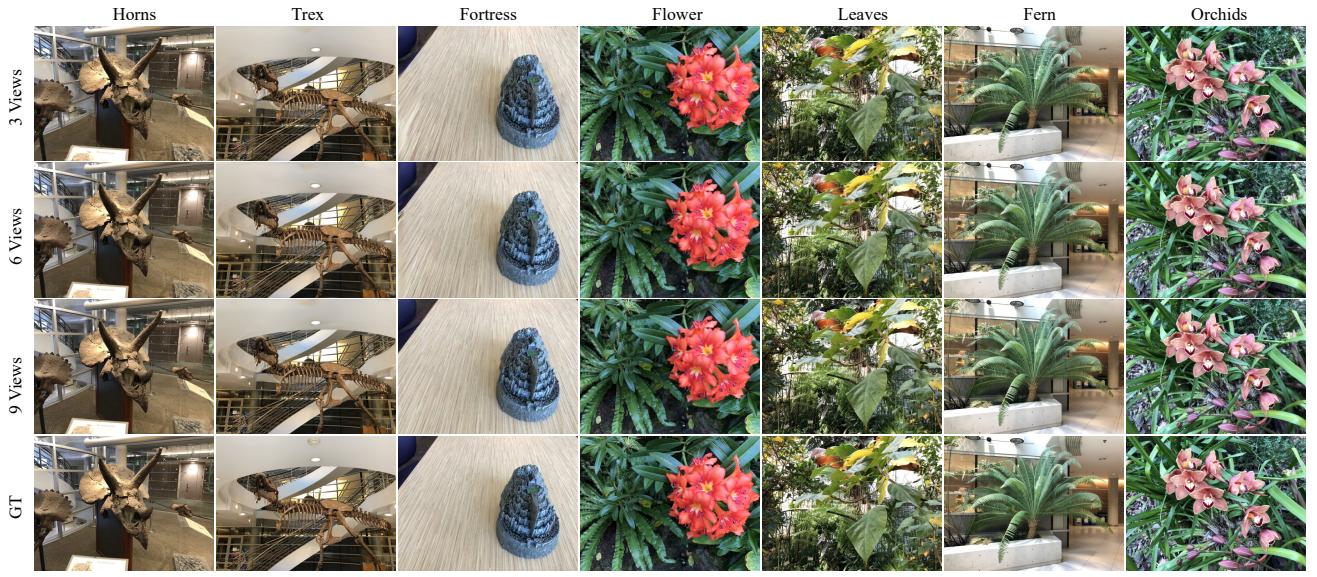


Figure 9. **Qualitative Results on the Effects of Training Views.** We visualize the rendered images produced by all methods with training views ranging from three to nine.

Methods	PSNR			SSIM			LPIPS		
	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view
Mip-NeRF	16.10	22.91	24.88	0.401	0.756	0.826	0.460	0.213	0.170
3D-GS	17.83	22.87	24.65	0.582	0.732	0.813	0.321	0.204	0.159
DietNeRF	14.94	21.75	24.28	0.370	0.717	0.801	0.496	0.248	0.183
RegNeRF	19.08	23.10	24.86	0.587	0.760	0.820	0.336	0.206	0.161
FreeNeRF	19.63	23.73	25.13	0.612	0.779	0.827	0.308	0.195	0.160
SparseNeRF	19.86	23.64	24.97	0.624	0.784	0.834	0.328	0.202	0.158
Ours	20.31	24.20	25.32	0.682	0.811	0.856	0.248	0.173	0.136

Table 5. **Quantitative Analysis on the Effects of Training Views.** We conduct experiments by using different training views (from 3 to 9) to test the adopted baseline methods. Our FSGS consistently outperforms other methods across all metrics.

ple 3 images from the remaining views. These images are then downsampled to $4\times$ and $8\times$ for both training and evaluation. We utilize the pretrained monocular depth estimator to predict the depth for input images, and the poses are computed via COLMAP. We compare our method with FreeNeRF [55], SparseNeRF [50], and 3D-GS [26].

Tab.6 presents the quantitative results, where FSGS outperforms SparseNeRF with over 0.75 higher PSNR and runs $3,757\times$ faster, a significant leap that underscores its potential for practical, real-world applications where speed is crucial and the environment is intricate. We also visualize the qualitative results in Fig.10, where FSGS significantly improves the visual quality of the scenes over 3D-GS, particularly in the realm of geometry reconstruction. FreeNeRF and SparseNeRF are constrained by geometric continuity

from unseen perspectives and do not fully capitalize on the available depth information.

B.3. Visual Comparisons of the Rendered Depth

We demonstrate the qualitative results of the predicted depth for each methods, as shown in Fig. 11. We compare our method with 3DGS [26], FreeNeRF [55] and SparseNeRF [50]. In the left we visualize the ground truth of the images. FSGS significantly outperforms the three baselines in terms of depth quality and details. The depth maps produced by FSGS are not only more accurate but also exhibit a higher level of detail, showcasing its robustness in reconstructing complex scenes. In contrast, both FreeNeRF and SparseNeRF exhibit limitations in geometric modeling and struggle to accurately learn complex geometries, lead-



Figure 10. **Qualitative Results on Datasets Collected by Mobile Phones.** We test the generalization capacity of all methods on self-captured iPhone images, with a calibration process from COLMAP. Our method reveals the majority of scene details despite only adopting three views in training.

Methods	1/8 Resolution				1/4 Resolution			
	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF	0.07	14.74	0.337	0.602	0.14	13.84	0.284	0.631
3D-GS	225	16.29	0.408	0.439	127	15.83	0.413	0.530
DietNeRF	0.05	13.62	0.263	0.598	0.08	12.57	0.228	0.722
RegNeRF	0.07	17.41	0.517	0.440	0.14	16.44	0.443	0.504
FreeNeRF	0.07	18.07	0.497	0.426	0.14	17.14	0.462	0.509
SparseNeRF	0.07	18.79	0.539	0.441	0.04	17.82	0.472	0.524
Ours	263	19.54	0.539	0.403	190	18.41	0.493	0.471

Table 6. **Quantitative Comparison in Mobile Phone Datasets, with 3 Training Views.** Our method continues to achieve the best performance in the challenging mobile phone dataset.

ing to a distorted scene representation. Although SparseNeRF leverage depth prior, it still does not fully capture the fine-grained structures in real-world structures, resulting in a noticeable drop in quality compared to FSGS. 3D-GS, on the other hand, tends to lose fine details in the areas far away the camera, leading to a diminished overall quality in the depth and texture of distant objects.

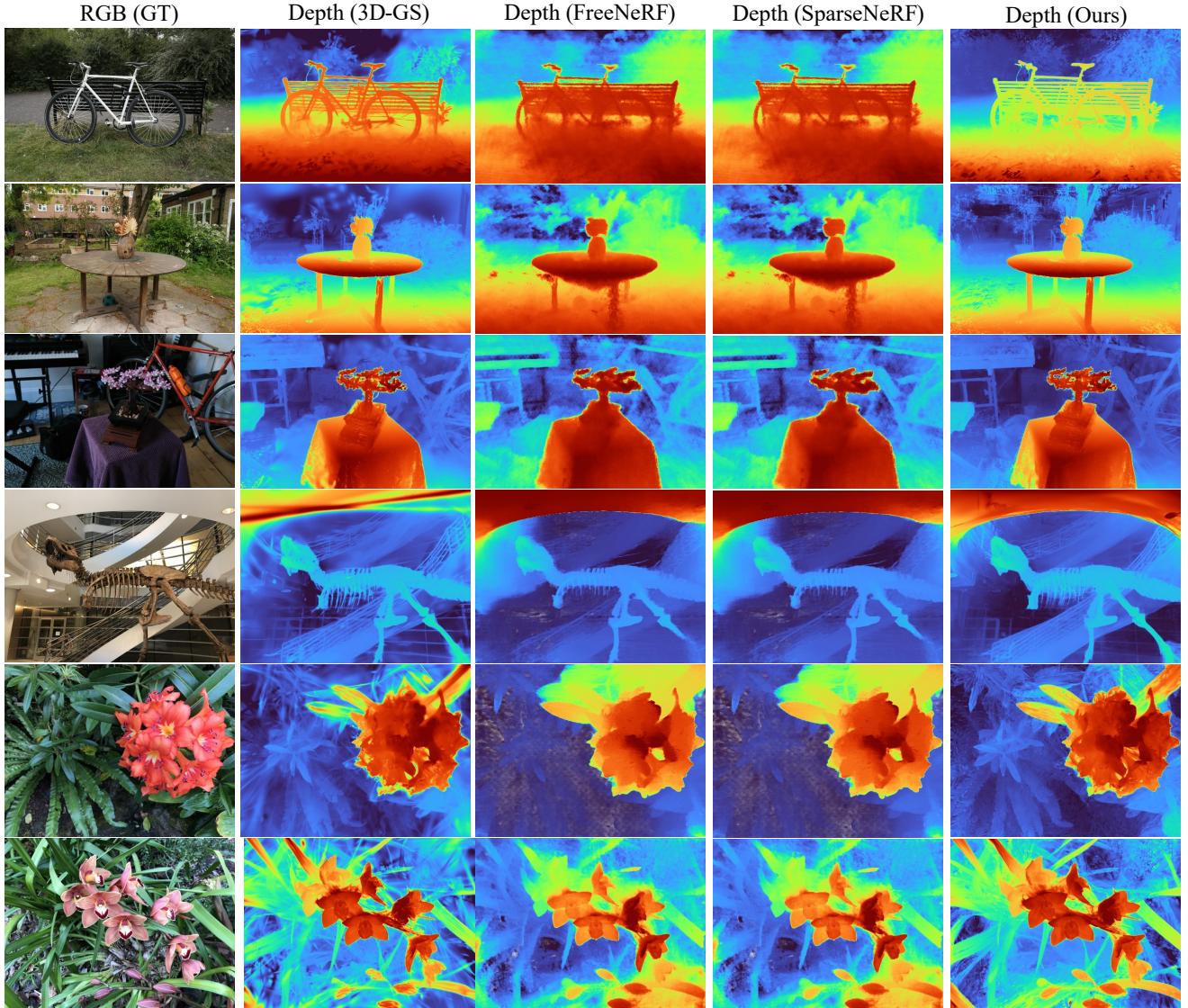


Figure 11. Visual Comparisons of Predicted Depth. We visualize the estimated scene depth from all baselines. Noticeable NeRF alias artifacts are found in SparseNeRF and FreeNeRF. 3D-GS produces an oversmoothed scene geometry, while our method demonstrates visually pleasing geometric details.

References

- [1] 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. [2](#) [3](#)
- [2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. 2022 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5460–5469, 2022. [3](#) [5](#) [6](#) [7](#)
- [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. [2](#) [5](#)
- [4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023. [2](#)
- [5] Yukang Cao, Yan-Pei Cao, Kai Han, Ying Shan, and Kwan-Yee K Wong. Dreamavatar: Text-and-shape guided 3d human avatar generation via diffusion models. *arXiv preprint arXiv:2304.00916*, 2023. [3](#)
- [6] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *arXiv*, 2020. [3](#)
- [7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorrf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. [2](#)
- [8] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021. [1](#) [2](#) [3](#)
- [9] Tianlong Chen, Peihao Wang, Zhiwen Fan, and Zhangyang Wang. Aug-nerf: Training stronger neural radiance fields with triple-level physically-grounded augmentations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15191–15202, 2022. [2](#)
- [10] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis from sparse views of novel scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2021. [2](#)
- [11] Congyue Deng, Chiyu Jiang, Charles R Qi, Xinchen Yan, Yin Zhou, Leonidas Guibas, Dragomir Anguelov, et al. Nerdi: Single-view nerf synthesis with language-guided diffusion as general image priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20637–20647, 2023. [1](#)
- [12] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791*, 2021. [3](#)
- [13] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4):65–74, 1988. [1](#)
- [14] Zhiwen Fan, Yifan Jiang, Peihao Wang, Xinyu Gong, Dejia Xu, and Zhangyang Wang. Unified implicit neural stylization. In *European Conference on Computer Vision*, pages 636–654. Springer, 2022. [2](#)
- [15] 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. [2](#)
- [16] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review, 2023. [1](#)
- [17] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021. [2](#)
- [18] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylererf: A style-based 3d aware generator for high-resolution image synthesis. In *International Conference on Learning Representations*, 2022. [3](#)
- [19] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2495–2504, 2020. [3](#)
- [20] Yuan-Chen Guo, Di Kang, Linchao Bao, Yu He, and Song-Hai Zhang. Nerfren: Neural radiance fields with reflections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18409–18418, 2022. [2](#)
- [21] 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 (ICCV)*, pages 7909–7920, October 2023. [3](#)
- [22] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 867–876, 2022. [2](#)
- [23] 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. [1](#) [3](#) [5](#)
- [24] M. M. Johari, Y. Lepoittevin, and F. Fleuret. Geonerf: Generalizing nerf with geometry priors. *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. [2](#)
- [25] Animesh Karnewar, Andrea Vedaldi, David Novotny, and Niloy Mitra. Holodiffusion: Training a 3D diffusion model using 2D images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023. [3](#)

- [26] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- [27] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3
- [28] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 3
- [29] 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 (TOG)*, 38(4):1–14, 2019. 5, 9
- [30] 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. 1, 2, 5, 9
- [31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 1, 2, 3
- [32] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. *arXiv preprint arXiv:2112.00724*, 2021. 3
- [33] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: 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. 1, 5
- [34] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 3
- [35] AKM Shahriar Azad Rabby and Chengcui Zhang. Beyondpixels: A comprehensive review of the evolution of neural radiance fields, 2023. 1
- [36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 3
- [37] 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. 5
- [38] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021. 2, 3
- [39] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 3, 9
- [40] Katja Schwarz, Axel Sauer, Michael Niemeyer, Yiyi Liao, and Andreas Geiger. Voxgraf: Fast 3d-Aware Image Synthesis With Sparse Voxel Grids. *ArXiv Preprint ArXiv:2206.07695*, 2022. 1
- [41] Junyoung Seo, Wooseok Jang, Min-Seop Kwak, Jaehoon Ko, Hyeonsu Kim, Junho Kim, Jin-Hwa Kim, Jiyoung Lee, and Seungryong Kim. Let 2d diffusion model know 3d-consistency for robust text-to-3d generation. *arXiv preprint arXiv:2303.07937*, 2023. 3
- [42] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8269–8279, 2022. 2
- [43] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct Voxel Grid Optimization: Super-Fast Convergence for Radiance Fields Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 1, 2
- [44] Mukund Varma T, Peihao Wang, Xuxi Chen, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. Is attention all that neRF needs? In *The Eleventh International Conference on Learning Representations*, 2023. 2
- [45] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior, 2023. 3
- [46] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in neural rendering, 2022. 1
- [47] Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. Sparf: Neural radiance fields from sparse and noisy poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4190–4200, 2023. 1, 5
- [48] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. *arXiv preprint arXiv:2112.03907*, 2021. 2
- [49] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 2
- [50] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. *arXiv preprint arXiv:2303.16196*, 2023. 1, 3, 5, 6, 7, 10

- [51] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. Fourier PlenOctrees for Dynamic Radiance Field Rendering in Real-Time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13524–13534, 2022. 1
- [52] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. *CVPR*, 2023. 2
- [53] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021. 2
- [54] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Humphrey Shi, and Zhangyang Wang. Sinnerf: Training neural radiance fields on complex scenes from a single image. In *European Conference on Computer Vision*, pages 736–753. Springer, 2022. 1
- [55] 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. 1, 3, 5, 10
- [56] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pages 767–783, 2018. 3
- [57] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019. 1
- [58] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxtels: Radiance Fields Without Neural Networks. 2022 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5491–5500, 2022. 1
- [59] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 1, 2, 3
- [60] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields, 2022. 2
- [61] Denis Zorin, Peter Schröder, and Wim Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 189–192, 1996. 4