FrugalNeRF: Fast Convergence for Extreme Few-shot Novel View Synthesis without Learned Priors

Supplementary Material

This supplementary material presents additional results to complement the main manuscript. First, we discuss the difference between competing methods in Suppl. A Second, we explain the implementation details in calculating reprojection errors in Suppl. B. Then, we provide all the training losses in our training process in Suppl. C. Next, we provide additional experiments in Suppl. D. Moreover, we describe the experimental setup, including the dataset and training time measurement of compared methods in our evaluations in Suppl. E. In addition to this document, we provide an interactive HTML interface to compare our video results with state-of-the-art methods and show ablation videos and failure cases. We also attach the source code of our implementation for reference and will make it publicly available for reproducibility.

A. Discussions on Competing Models

GeCoNeRF. GeCoNeRF [5] is a few-shot NeRF that uses warped features as pseudo labels, which is sufficiently different from our method. Our method primarily focuses on cross-scale geometric adaptation, selecting render depths with minimal reprojection error across different scales as pseudo labels to adaptively learn the most suitable geometry for each scale. In contrast, GeCoNeRF, besides requiring a pre-trained feature extractor, directly optimizes warped features, making it highly sensitive to geometric noise and resulting in many floaters in its rendering result as shown in our supplementary videos. Our approach, on the other hand, is more robust due to our proposed multi-scale voxels. Low-resolution voxels represent coarse geometry, which is less likely to produce floaters. Using this as supervision effectively suppresses the generation of floaters.

ZeroRF. ZeroRF [10] is a concurrent work to ours, also aimed at training NeRF with sparse input views and achieving fast training times. Unlike TensoRF [2], which directly optimizes the decomposed feature grid, ZeroRF parameterizes the feature grids with a randomly initialized deep neural network (generator). This decision is based on the belief in the higher resilience to noise and artifacts ability of deep neural networks. Although ZeroRF claims to achieve fast convergence stemming from its voxel representation, the need to train the generator results in slower training speeds compared to ours (refer to the main paper Table 2). Our method directly optimizes the feature grid and utilizes cross-scale geometry adaptation to avoid overfitting under sparse views, without requiring a generator that slows down convergence to form decomposed tensorial feature volumes. Additionally, we found that ZeroRF is not suitable for scenes with a background (*e.g.*, LLFF [6]) or datasets like the DTU [4] Dataset, where ZeroRF must extensively use object masks for training. These object masks are not provided directly in these two datasets. Otherwise, ZeroRF may produce many artifacts and floaters, or the feature volume may be filled up to fit the background, leading to severe memory consumption issues causing training failures due to out-of-memory errors.

SparseNeRF. SparseNeRF [14] proposes a spatial continuity regularization that distills depth continuity priors, but it requires a pre-trained depth prior and is extremely slow by using MLP representation. Additionally, because monocular depth prediction results lack detail, SparseNeRF's rendered results tend to be blurry and lack detail. In contrast, our proposed cross-scale geometric adaptation does not rely on pre-trained priors and ensures the generation of overall geometry while paying attention to details.

SimpleNeRF. SimpleNeRF [12] introduces a data augmentation method for few-shot NeRF, employing an MLP with fewer positional encoding frequencies for augmentation, but this simultaneously increases the training time. In contrast, we propose an efficient cross-scale geometric adaptation that achieves multi-scale representation through shared-weight voxels, eliminating the need for an additional model to reconstruct the same scene. This approach yields better results with lower costs.

FreeNeRF. FreeNeRF [16] is an MLP-based few-shot NeRF model. FreeNeRF proposes using a scheduling mechanism to gradually increase input frequency, allowing the model to learn low-frequency geometry during the early stages of training and then ramp up positional encoding to enable the model to learn more detailed geometry later on. However, our approach takes advantage of the explicit voxel representation, which converges faster and allows for direct cross-scaled geometry operations. Additionally, because we employ cross-scale geometry adaptation, our model dynamically determines which frequency of geometry to learn at different training stages. We do not require the complex frequency scheduling of FreeNeRF, nor are we limited to learning only high-frequency components in the later stages of training like FreeNeRF. This makes our method simpler, more general, and more robust.

VGOS. VGOS [13] introduces an incremental voxel training strategy and a voxel smoothing method for Few-shot NeRF, aimed at reducing training time. It employs a complex scheduling strategy to freeze the outer part of the voxel, leading to a leaky reconstruction of the background scene. Additionally, VGOS requires ground truth poses for novel pose sampling, which results in a quality drop when using random sampling. However, while VGOS's training time is shorter than ours, its performance significantly lags behind. Our cross-scale geometric adaptation strategy eliminates the need for complex scheduling and ground truth pose sampling.

FSGS. FSGS [19] addresses the challenge of limited 3D Gaussian splatting (3DGS) by introducing Proximity-guided Gaussian Unpooling, which adaptively densifies the Gaussians between existing points. Although this method mitigates the issue of insufficient GS, it still relies on a sufficient initial set of Gaussians to perform effectively. In few-shot scenarios, the initial number of GS can be extremely sparse, leading to suboptimal results. Furthermore, FSGS frequently requires novel view inference using monocular depth models during training, which significantly increases the training time. In contrast, our cross-scale geometric adaptation approach ensures rapid convergence without relying on novel view inference or monocular depth models, providing efficient and robust performance even with minimal initial data.

ReVoRF. ReVoRF [15] introduces a voxel-based framework that strategically utilizes unreliable regions in pseudo-novel view synthesis for few-shot NeRF. By leveraging a bilateral geometric consistency loss and reliability-aware voxel smoothing, ReVoRF achieves significant improvements in reconstruction quality and training efficiency. However, its reliability mask requires rendering the entire frame to infer the monocular depth model, which limits the frequency of updates. Additionally, its smoothing process does not account for the balance between high- and low-frequency details, resulting in reconstructions that lack fine details. In contrast, FrugalNeRF employs cross-scale geometric adaptation to preserve high-frequency details while preventing floaters. Its high efficiency allows geometric adaptation to be computed at every iteration, ensuring robust and detailed scene reconstruction.

B. Details of Calculating Reprojection Errors

Mathematically, let $\mathbf{p}_i$ be a 2D pixel coordinate in frame i, and $\widetilde{\mathbf{p}}_\mathbf{i}$ be its homogeneous augmentation. The depth $D_i^l(\mathbf{p}_i)$ at scale l obtained from volume rendering, and camera intrinsics K_i are used to reproject $\mathbf{p}_i$ onto the 3D point $\mathbf{x}_i^l$ in camera coordinate system of frame i. Subsequently, utilizing the rotation matrix R_i and translation matrix t_i of frame i, $\mathbf{x}_i^l$ are transformed into world coordinates system $\mathbf{x}^l$:

$$\mathbf{x}_i^l = D_i^l(\mathbf{p}_i) K_i^{-1} \widetilde{\mathbf{p}}_i \tag{1}$$

$$\mathbf{x}^l = R_i \mathbf{x}_i^l + t_i \tag{2}$$

We simplify the previous two equations because the position of the 3D point $\mathbf{x}^l$ in world coordinates can also be determined directly from the ray defined by the starting point $o_i(\mathbf{p}_i)$ and the direction $v_i(\mathbf{p}_i)$:

$$\mathbf{x}^l = o_i(\mathbf{p}_i) + D_i^l(\mathbf{p}_i) v_i(\mathbf{p}_i) \tag{3}$$

Following this, the 3D point $\mathbf{x}^l$ in the world coordinate system is transformed to the camera coordinate system of frame j using its rotation matrices R_j, and translation matrices T_j:

$$\mathbf{x}_{i\to j}^l = R_j^T \left(\mathbf{x}^l - t_j\right) \tag{4}$$

Finally, project it back to the 2D pixel coordinate system of frame j,

$$\widetilde{\mathbf{p}}_{i\to j}^l = \pi(K_j \mathbf{x}_{i\to j}^l) \tag{5}$$

where $\pi([x, y, z]^T) = \left[\frac{x}{z}, \frac{y}{z}\right]$. Using coordinates $\mathbf{p}_i$ and $\mathbf{p}_{i\to j}^l$ to index the RGB maps of frames i (denoted as C_i) and j (denoted as C_j), facilitating the computation of the reprojection error:

$$e^l(\mathbf{p}_i) = \left\| C_i(\mathbf{p}_i) - C_j(\mathbf{p}_{i\to j}^l) \right\|^2 \tag{6}$$

Therefore, for each ray sampled from the training view, the pseudo-GT depth of the scale with the minimum reprojection error is obtained,

$$D'(\mathbf{r}_{\text{train}}) = \arg\min_l (e^l(\mathbf{r}_{\text{train}})). \tag{7}$$

where the pseudo-GT depth is utilized to compute the geometric adaptation loss (MSE) $\mathcal{L}_{\text{geo}}$.

$$\mathcal{L}_{\text{geo}}(\mathbf{r}_{\text{train}}) = \sum_{l=0}^{L} \sum_{r_{\text{train}} \in \mathcal{R}_{\text{train}}} \left\| \hat{D}^l(\mathbf{r}_{\text{train}}) - D'(\mathbf{r}_{\text{train}}) \right\|^2. \tag{8}$$

This mechanism provides a supervisory signal for geometry, ensuring that the model can effectively maintain the geometric integrity of the scene across different scales, even in the absence of explicit depth ground truth. It is a pivotal part of the training process, allowing the model to adapt and refine its understanding of the scene's geometric structure in a self-adaptive manner. In our implementation, instead of using a single pixel to calculate reprojection error, we use a patch with 5×5 pixels to calculate reprojection error. This avoids warping noise caused by similar patterns in scenes, for example, in the case of the LLFF fortress and room. Furthermore, we set a threshold for reprojection error that allows us to ignore cases of image warping with occlusions and prevents crashes during initial training processes, which typically have high reprojection errors.

C. Losses

Voxel TV loss ($\mathcal{L}_{\text{tv}}$). We use the TV loss on voxel to smooth the result in voxel space.

Patch-wise depth smoothness loss ($\mathcal{L}_{\text{ds}}$). We sample patches of rays and calculate the total variance of depth to smooth the geometry in the depth space.

L1 sparsity loss ($\mathcal{L}_{\text{l1}}$). We suppress the voxel density in air space by introducing a density L1 regularization loss.

Distortion loss ($\mathcal{L}_{\text{dist}}$). We adopt the approach from Mip-NeRF 360 [1], integrating distortion loss to remove floaters from the novel views.

Occlusion loss ($\mathcal{L}_{\text{occ}}$). In the DTU dataset, we follow FreeNeRF [16] by incorporating an occlusion loss that utilizes black and white background priors to push floaters into the background.

Novel pose sampling form spiraling trajectory. We follow the implementation of a spiraling trajectory from TensoRF [2]. For the LLFF dataset, we sample 60 novel poses from the spiraling trajectory sampled from training views with 1 rotations, radius scale 1.0, and z_{rate} 0.5. For the DTU dataset, we sample 60 novel poses from the spiraling trajectory sampled from training views with 4 rotations, radius scale 0.5, and z_{rate} 0.5. For the RealEstate-10K dataset, we sample 60 novel poses from the spiraling trajectory sampled from training views with 2 rotations, radius scale 2.0, and z_{rate} 0.5.

C.1. Details of adding Pretrained Monocular Depth Prior

We utilize the pre-trained Dense Prediction Transformer (DPT) [8] to generate monocular depth maps from training views. DPT is trained on 1.4 million image-depth pairs, making it a convenient and effective choice for our setup. To address the scale ambiguity between the true scene scale and the estimated depth, we introduce a relaxed relative loss based on Pearson correlation between the estimated and rendered depth maps. This loss is applied at multiple scales, enhancing the monocular depth prior's constraint across different scales and improving the overall geometric consistency. Fig. 1 show the render depth on adding Pretrained Monocular Depth Prior.

D. Additional experiments.

D.1. Number of training views analysis.

We plot the number of training views experiment in Fig. 2 and Tab. 1, demonstrating that FrugalNeRF outperforms

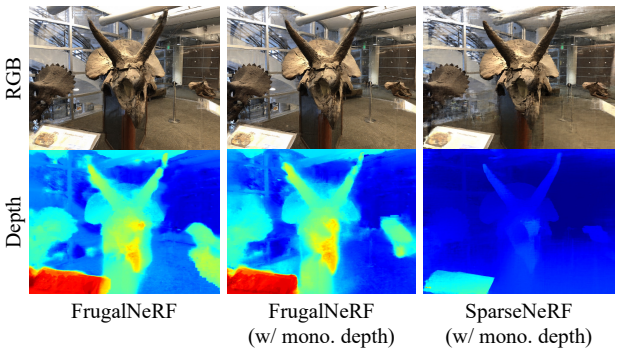


Figure 1. **Visual comparisons on adding a pre-trained monocular depth prior.**

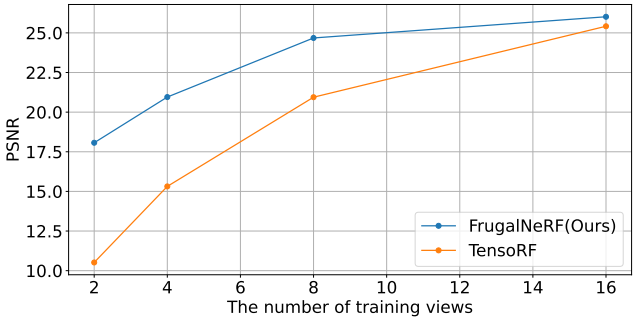


Figure 2. **Number of training views analysis.** FrugalNeRF significantly outperforms the base TensoRF on sparse views.

TensoRF on sparse views (2 to 8 views) and continues to lead as the number of views increases.

Table 1. **Number of training views analysis.**

Method	2-view PSNR ↑	2-view SSIM ↑	2-view LPIPS ↓	16-view PSNR ↑	16-view SSIM ↑	16-view LPIPS ↓	Dense view (17- to 54-view) PSNR ↑	Dense view (17- to 54-view) SSIM ↑	Dense view (17- to 54-view) LPIPS ↓
TensoRF	11.97	0.32	0.64	25.42	0.813	0.151	**26.73**	**0.839**	0.124
Ours	**18.07**	**0.54**	**0.35**	**25.82**	**0.822**	**0.128**	26.69	0.835	**0.114**

D.2. Downsampling Strategy.

We use nearest-neighbor downsampling for stability and to prevent floaters. In Tab. 2 and Tab. 3 we compare it to bilinear interpolation.

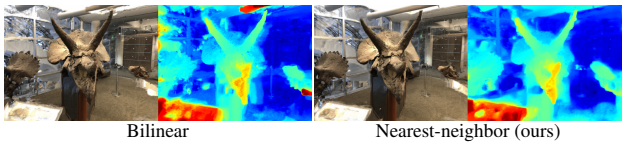
Table 2. **Ablation on downsampling strategy on LLFF dataset.**

Method	PSNR ↑	SSIM ↑	LPIPS ↓
Bilinear	17.08	0.49	0.36
Nearest-neighbor (ours)	**18.07**	**0.54**	**0.35**

E. Experimental Setup

We compare the result of few-shot NeRF on LLFF and DTU with $n = 2, 3, 4$ input views.

LLFF dataset. The LLFF dataset comprises 8 forward-facing unbounded scenes with variable frame counts at a



Bilinear Nearest-neighbor (ours)

Table 3. **Visual comparison on different downsampling strategies .**

resolution of 1008 × 756. In line with prior work [11], we use every 8th frame for testing in each scene. For training, we uniformly sample n views from the remaining frames.

DTU dataset. The DTU dataset is a large-scale multi-view collection that includes 124 different scenes. Follow the Pixel-NeRF [17] and ViP-NeRF [11] approach, we use the same test sets. However, because COLMAP will fail to generate sparse depth at scans 8, 30, and 110, we can only test on 12 scenes. Test scan IDs are 21, 31, 34, 38, 40, 41, 45, 55, 63, 82, 103, and 114. We use specific image IDs as input views and downsample images to 300 × 400 pixels for consistency with prior studies [11, 17].

RealEstate-10K dataset. RealEstate-10K is a comprehensive database of approximately 80,000 video segments, each with over 30 frames, widely utilized for novel view synthesis. For our study, we select five scenes from its extensive test set, following the approach outlined in ViP-NeRF [11]. We selected frames 0, 10, 20, and 30 for the training set with a resolution of 1024 × 576, in accordance with the SimpleNeRF [12] methodology, while testing on the same test set as SimpleNeRF [12] due to the unobserved region problem, which NeRF cannot handle, in some testing view.

E.1. Training Time Measurement and Time Complexity

RegNeRF. We use the official implementation of RegNeRF [7] and follow most of the default configuration, while the batch size or other hyperparameters might be adjusted due to the GPU memory issue. For the LLFF dataset, the training requires roughly 2.35 hours per scene with 69769 iterations and a batch size of 2,048. Note that RegNeRF samples 10000 random poses by its default configuration on the DTU dataset, leading to out-of-memory on a single NVIDIA RTX 4090 GPU. While reducing the number of random poses to about 1/8 could potentially resolve this issue, such a reduction is likely to adversely affect the performance, so we simply exclude this method from our experiments.

FreeNeRF. We use the official implementation of FreeNeRF [16] and follow most of the default configuration, while the batch size or other hyperparameters might be adjusted due to the GPU memory issue. For the LLFF dataset, the training requires roughly 1.5 hours per scene with 69,769 iterations and a batch size of 2,048. For the DTU dataset, the training requires about 1 hour per scene with 43,945 iterations and a batch size of 2,048.

SparseNeRF. We use the official implementation of SparseNeRF. [14] and follow most of the default configuration, while the batch size or other hyperparameters might be adjusted due to the GPU memory issue. For the LLFF dataset, the training requires roughly 1 hour per scene with 70,000 iterations and a batch size of 512. For the DTU dataset, the training requires about 30 minutes per scene with 70,000 iterations and a batch size of 256.

SimpleNeRF. We use the official implementation of SimpleNeRF [12] and follow most of the default configuration, while the batch size or other hyperparameters might be adjusted due to the GPU memory issue. For the LLFF dataset, we use the model weights released by the author directly. Since there's no official implemented dataloader for the DTU dataset, we use the dataloader and configuration from ViP-NeRF [11], which requires about 1.38 hours per scene with 25,000 iterations and batch size of 2,048.

VGOS. We furter provide VGOS result. We use the official implementation of VGOS [13] and follow most of the default configuration, while the batch size or other hyperparameters might be adjusted due to the GPU memory issue. Note that VGOS samples random poses directly from the entire dataset, which is unreasonable under the few-shot setting, so we replace the sampling with the interpolation from training poses implemented in the official repo. For the LLFF dataset, the training requires roughly 5 minutes per scene with 9,000 iterations and a batch size of 16,384. For the DTU dataset, the training requires about 3 minutes per scene with 9,000 iterations and a batch size of 16,384. Note that VGOS seems invalid on the DTU dataset (Fig. 4) and they does not evaluate the DTU dataset in their paper.

GeCoNeRF. As mentioned in GeCoNeRF [5]'s official github repo, their current code is unexecutable. To complete our experiment, we still try our best to implement their method based on the code provided. For the LLFF dataset, the training requires roughly 4 hours per scene with 85,000 iterations and a batch size of 1024. It is important to note that we utilized 2 GPUs for training this method, so the training time reported in our paper might be shorter than what is actually required.

ZeroRF. We use the official implementation of ZeroRF [10] and follow most of the default configurations. For the LLFF dataset, ZeroRF does not provide the dataloader for the LLFF, and their paper mentions its inability to be used

Table 4. **Comparison of the time complexity.**

Method	MFLOPs / pixel ↓
FreeNeRF [16]	288.57
ViP-NeRF [11]	149.26
SimpleNeRF [12]	303.82
SparseNeRF [14]	287.92
Ours	13.77

for unbounded scenes. Therefore, our primary testing was conducted on the DTU dataset. In the DTU dataset, the original implementation of ZeroRF necessitates masking out the background area of the input frame before training, which is incompatible with our evaluation benchmark. Consequently, we trained it without object masks. Training requires approximately 25 minutes per scene with 10,000 iterations and a batch size of 2^{14}.

FSGS. We use the official implementation of FSGS [19] and follow most of the default configurations. For the LLFF dataset, we adjust the input views to match the settings used in ViP-NeRF, which differs from the original FSGS paper. Training takes approximately 25 minutes per scene with 10,000 iterations. Since there is no official dataloader for the DTU dataset, we convert the DTU camera poses to the LLFF format and use the default LLFF configuration. Training on the DTU dataset requires around 20 minutes per scene with 10,000 iterations.

Time complexity. To verify the efficiency of our method, besides comparing the training time of various methods, we also calculated the MFLOPs per pixel in Tab. 4.

F. Complete Quantitative Evaluations

LLFF dataset. We show all 8 scenes of the quantitative comparisons with two, three, and four input views on the LLFF dataset in Tab. 5, Tab. 6, and Tab. 7, respectively.

DTU dataset. We show all 12 scenes of the quantitative comparisons with two, three, and four input views on the DTU dataset in Tab. 8, Tab. 9, Tab. 10, and Tab. 11, respectively.

RealEstate-10K dataset. We show all 12 scenes of the quantitative comparisons with two, three, and four input views on the RealEstate-10K dataset in Tab. 12, Tab. 13, Tab. 14, and Tab. 15.

G. Additional Visual Comparisons

LLFF dataset. We show additional visual comparisons on the LLFF dataset with two input views in Fig. 3 and Fig. 7.

DTU dataset. We show additional visual comparisons on the DTU dataset with two input views in Fig. 4 and Fig. 8.

RealEstate-10K dataset. We further present the qualitative comparisons of novel view synthesis on the RealEstate-10K dataset with two input views in Fig. 5 and Fig. 6. Compared to SimpleNeRF [12], which requires hours of training, FrugalNeRF needs only less than 20 minutes and can render comparable results, demonstrating FrugalNeRF's effectiveness in more in-the-wild scenes.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 3

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 1, 3, 9, 11

[3] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *CVPR*, 2022. 6, 7, 8, 11, 12

[4] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 1, 9, 10, 14, 17

[5] Minseop Kwak, Jiuhn Song, and Seungryong Kim. Geconerf: Few-shot neural radiance fields via geometric consistency. In *ICML*, 2023. 1, 4, 6, 7, 8

[6] 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)*, 2019. 1, 6, 7, 8, 13, 16

[7] 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 *CVPR*, 2022. 4, 6, 7, 8, 11, 12

[8] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 3

[9] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, 2022. 6, 7, 8, 11, 12

[10] Ruoxi Shi, Xinyue Wei, Cheng Wang, and Hao Su. Zerorf: Fast sparse view 360 {\deg} reconstruction with zero pretraining. In *CVPR*, 2024. 1, 4, 9, 10

[11] Nagabhushan Somraj and Rajiv Soundararajan. Vip-nerf: Visibility prior for sparse input neural radiance fields. In *ACM SIGGRAPH*, 2023. 4, 5, 6, 7, 8, 9, 10, 11, 12, 14

[12] Nagabhushan Somraj, Adithyan Karanayil, and Rajiv Soundararajan. Simplenerf: Regularizing sparse input neural radiance fields with simpler solutions. In *ACM SIGGRAPH Asia*, 2023. 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14

[13] Jiakai Sun, Zhanjie Zhang, Jiafu Chen, Guangyuan Li, Boyan Ji, Lei Zhao, and Wei Xing. Vgos: Voxel grid optimization for view synthesis from sparse inputs. In *IJCAI*, 2023. 2, 4, 6, 7, 8, 9, 10

Table 5. **Quantitative results on the LLFF [6] dataset with two input views. The three rows show LPIPS, SSIM, and PSNR scores, respectively.**

Method \ Scene	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex	Average
RegNeRF [7]	0.51	0.43	0.37	0.51	0.35	0.45	0.38	0.42	0.43
	0.45	0.51	0.46	0.42	0.37	0.30	0.74	0.54	0.49
	15.8	17.0	20.6	15.9	14.5	13.9	18.7	16.7	16.9
DS-NeRF [3]	0.50	0.43	0.30	0.49	0.47	0.43	0.35	0.41	0.42
	0.46	0.44	0.65	0.49	0.24	0.32	0.76	0.53	0.51
	16.4	16.1	23.0	16.6	12.4	13.7	18.9	15.7	16.9
DDP-NeRF [9]	0.44	0.46	0.17	0.46	0.52	0.41	0.30	0.43	0.39
	0.49	0.45	0.77	0.52	0.23	0.38	0.76	0.54	0.54
	17.2	16.2	22.7	17.1	12.6	15.1	18.7	15.7	17.2
FreeNeRF [16]	0.46	0.38	0.33	0.43	0.36	0.42	0.34	0.33	0.38
	0.49	0.55	0.53	0.53	0.38	0.35	0.76	0.60	0.54
	17.1	17.6	21.3	17.1	14.4	14.1	18.3	18.1	17.6
ViP-NeRF [11]	0.45	0.42	0.21	0.39	0.46	0.40	0.36	0.38	0.37
	0.45	0.43	0.71	0.54	0.21	0.36	0.72	0.54	0.52
	16.2	14.9	22.6	17.1	11.7	14.2	17.7	15.9	16.7
SimpleNeRF [12]	0.51	0.43	0.25	0.42	0.44	0.41	0.35	0.39	0.39
	0.50	0.53	0.67	0.54	0.30	0.37	0.77	0.58	0.55
	17.0	16.9	22.5	17.1	13.5	14.7	19.5	16.8	17.6
VGOS [13]	0.48	0.44	0.37	0.47	0.36	0.42	0.38	0.40	0.42
	0.51	0.55	0.53	0.55	0.38	0.40	0.77	0.59	0.55
	16.5	17.5	19.4	15.7	14.7	14.4	18.8	16.0	16.7
GeCoNeRF [5]	0.56	0.49	0.50	0.61	0.49	0.51	0.54	0.49	0.52
	0.47	0.49	0.43	0.41	0.28	0.29	0.68	0.52	0.45
	16.4	16.9	17.9	15.4	13.3	13.4	17.3	16.1	15.8
SparseNeRF [14]	0.48	0.55	0.40	0.52	0.52	0.55	0.29	0.37	0.45
	0.52	0.41	0.61	0.51	0.244	0.24	0.82	0.62	0.52
	18.2	15.4	21.7	17.4	13.4	13.3	22.8	18.6	18.0
FSGS [19]	0.46	0.45	0.35	0.42	0.33	0.41	0.38	0.45	0.41
	0.40	0.38	0.47	0.42	0.34	0.24	0.72	0.46	0.45
	15.0	14.8	16.9	16.2	14.2	12.6	17.6	13.8	15.3
FrugalNeRF (Ours)	0.41	0.41	0.27	0.36	0.32	0.42	0.34	0.32	0.35
	0.47	0.50	0.54	0.55	0.41	0.33	0.75	0.61	0.54
	17.4	17.5	20.3	18.5	15.5	15.0	19.2	18.6	18.1
FrugalNeRF w/ mono. depth (Ours)	0.40	0.40	0.27	0.37	0.33	0.39	0.32	0.35	0.35
	0.46	0.53	0.54	0.54	0.41	0.37	0.76	0.59	0.54
	17.7	17.9	20.9	18.5	15.4	15.6	19.6	18.2	18.3

[14] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *ICCV*, 2023. 1, 4, 5, 6, 7, 8, 9, 10

[15] Yingjie Xu, Bangzhen Liu, Hao Tang, Bailin Deng, and Shengfeng He. Learning with unreliability: Fast few-shot voxel radiance fields with relative geometric consistency. In *CVPR*, 2024. 2

[16] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *CVPR*, 2023. 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

[17] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 4

[18] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *ACM TOG*, 2018. 11, 12, 14, 15

[19] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *ECCV*, 2024. 2, 5, 6, 7, 8, 9

Table 6. **Quantitative results on the LLFF [6] dataset with three input views. The three rows show LPIPS, SSIM, and PSNR scores, respectively.**

Method \ Scene	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex	Average
RegNeRF [7]	0.47 0.48 17.9	0.27 0.58 19.6	0.31 0.64 22.7	0.44 0.53 18.2	0.39 0.37 14.6	0.44 0.31 14.2	0.25 0.81 21.0	0.36 0.63 18.4	0.36 0.57 18.7
DS-NeRF [3]	0.47 0.52 18.5	0.25 0.66 21.3	0.25 0.72 24.8	0.47 0.52 17.5	0.50 0.25 12.6	0.45 0.33 14.1	0.22 0.84 23.0	0.37 0.59 17.1	0.36 0.58 19.0
DDP-NeRF [9]	0.47 0.53 18.5	0.29 0.63 20.2	0.20 0.75 22.1	0.48 0.53 17.4	0.52 0.24 12.8	0.45 0.35 15.1	0.32 0.76 18.3	0.42 0.54 16.0	0.39 0.56 17.7
FreeNeRF [16]	0.40 0.54 18.9	0.28 0.61 20.7	0.32 0.60 22.0	0.41 0.58 18.7	0.40 0.40 15.0	0.41 0.37 14.7	0.22 0.85 22.6	0.33 0.64 19.0	0.34 0.60 19.3
ViP-NeRF [11]	0.51 0.49 17.3	0.24 0.65 20.8	0.19 0.76 24.5	0.42 0.57 18.2	0.44 0.25 12.4	0.41 0.34 14.2	0.27 0.81 21.7	0.32 0.62 18.1	0.34 0.59 18.9
SimpleNeRF [12]	0.43 0.52 18.2	0.24 0.66 20.7	0.17 0.78 24.7	0.42 0.57 18.4	0.42 0.38 14.8	0.39 0.38 15.0	0.26 0.83 22.0	0.34 0.66 18.9	0.33 0.62 19.5
VGOS [13]	0.40 0.58 19.0	0.31 0.61 20.0	0.33 0.69 23.0	0.46 0.58 17.0	0.40 0.40 15.0	0.41 0.40 15.2	0.31 0.83 21.8	0.35 0.66 18.0	0.37 0.61 18.8
GeCoNeRF [5]	0.57 0.46 17.0	0.36 0.57 19.5	0.45 0.53 20.6	0.60 0.44 15.8	0.50 0.32 13.8	0.51 0.30 13.6	0.34 0.80 21.1	0.43 0.59 18.1	0.47 0.50 17.4
SparseNeRF [14]	0.43 0.57 19.6	0.33 0.60 19.8	0.37 0.59 23.0	0.50 0.53 18.4	0.35 0.45 16.5	0.41 0.37 15.2	0.28 0.81 21.5	0.31 0.67 20.1	0.37 0.59 19.5
FSGS [19]	0.48 0.55 17.9	0.30 0.68 21.5	0.15 0.72 23.9	0.36 0.65 19.4	0.26 0.28 13.3	0.35 0.37 14.1	0.28 0.84 22.6	0.28 0.62 17.4	0.30 0.61 19.2
FrugalNeRF (Ours)	0.39 0.50 18.2	0.32 0.55 18.8	0.24 0.63 23.4	0.34 0.59 19.3	0.37 0.39 15.5	0.42 0.35 15.3	0.27 0.81 22.2	0.29 0.66 19.3	0.32 0.59 19.4
FrugalNeRF w/ mono. depth (Ours)	0.40 0.49 18.6	0.23 0.63 21.4	0.22 0.69 23.5	0.33 0.60 19.0	0.37 0.39 15.4	0.40 0.36 15.7	0.25 0.83 22.3	0.29 0.67 20.0	0.30 0.61 19.9

Table 7. **Quantitative results on the LLFF [6] dataset with four input views. The three rows show LPIPS, SSIM, and PSNR scores, respectively.**

Method \ Scene	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex	Average
RegNeRF [7]	0.35 0.63 20.8	0.29 0.64 19.8	0.37 0.55 22.4	0.34 0.64 20.1	0.32 0.44 15.9	0.43 0.34 14.8	0.19 0.87 23.9	0.32 0.66 18.9	0.32 0.62 19.9
DS-NeRF [3]	0.35 0.63 20.9	0.28 0.64 20.6	0.31 0.66 24.1	0.41 0.59 19.5	0.41 0.39 15.8	0.41 0.38 15.2	0.16 0.89 25.6	0.39 0.59 17.1	0.34 0.61 20.1
DDP-NeRF [9]	0.40 0.60 20.1	0.30 0.63 20.0	0.18 0.73 23.4	0.42 0.59 19.3	0.45 0.37 15.1	0.42 0.41 15.8	0.26 0.82 20.8	0.39 0.60 17.3	0.35 0.61 19.2
FreeNeRF [16]	0.37 0.64 21.1	0.30 0.64 20.5	0.35 0.60 23.2	0.37 0.63 20.4	0.35 0.47 16.6	0.42 0.37 14.9	0.19 0.88 24.8	0.31 0.68 19.6	0.33 0.63 20.5
ViP-NeRF [11]	0.39 0.58 18.2	0.27 0.63 19.5	0.25 0.70 23.3	0.38 0.60 19.0	0.36 0.40 14.8	0.40 0.39 14.8	0.23 0.85 23.2	0.32 0.64 18.6	0.32 0.62 19.3
SimpleNeRF [12]	0.33 0.65 21.1	0.27 0.67 20.8	0.28 0.69 24.3	0.38 0.63 19.7	0.35 0.46 16.3	0.36 0.42 15.7	0.19 0.88 24.3	0.32 0.68 19.3	0.31 0.65 20.4
VGOS [13]	0.40 0.64 19.6	0.35 0.63 20.3	0.40 0.64 22.7	0.43 0.62 18.6	0.34 0.49 16.6	0.41 0.43 15.8	0.28 0.86 23.6	0.35 0.68 18.7	0.37 0.64 19.7
GeCoNeRF [5]	0.45 0.61 20.5	0.36 0.61 19.9	0.44 0.51 21.2	0.47 0.59 19.6	0.44 0.40 15.5	0.51 0.30 13.9	0.27 0.85 23.5	0.40 0.63 19.0	0.42 0.56 19.1
SparseNeRF [14]	0.42 0.62 21.4	0.32 0.64 20.7	0.31 0.70 24.6	0.39 0.63 20.4	0.36 0.49 17.5	0.42 0.39 15.7	0.25 0.85 23.5	0.29 0.70 20.9	0.34 0.65 20.9
FSGS [19]	0.26 0.67 20.5	0.22 0.65 20.2	0.17 0.65 22.6	0.24 0.70 20.9	0.22 0.46 15.6	0.28 0.45 15.4	0.17 0.88 23.7	0.23 0.71 19.2	0.22 0.66 20.1
FrugalNeRF (Ours)	0.30 0.63 21.1	0.28 0.64 20.8	0.24 0.60 23.6	0.30 0.66 21.6	0.26 0.52 16.9	0.38 0.41 16.3	0.19 0.87 24.2	0.27 0.72 19.7	0.27 0.65 20.9
FrugalNeRF w/ mono. depth (Ours)	0.30 0.64 21.5	0.27 0.65 20.9	0.25 0.64 23.9	0.28 0.68 21.1	0.24 0.53 17.2	0.37 0.41 16.3	0.18 0.88 24.1	0.27 0.71 19.6	0.26 0.66 20.9

Table 8. **Quantitative results on the DTU [4] dataset.** FurgalNeRF synthesizes better images than most of the other baselines under extreme few-shot settings but with shorter training time and does not rely on any externally learned priors. Additionally, integrating monocular depth model regularization further improves quality while maintaining fast convergence. We follow SparseNeRF [14] to remove the background when computing metrics.

Method	Venue	Learned priors	2-view			3-view			4-view			Training time ↓
			PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	
TensoRF [2]	ECCV22	-	8.81	0.34	0.71	9.13	0.34	0.9.11	9.15	0.33	0.71	5 mins
FreeNeRF [16]	CVPR23	-	18.05	0.73	0.22	22.40	0.82	0.14	24.98	0.86	0.12	1 hrs
ViP-NeRF [11]	SIGGRAPH23	-	14.91	0.49	0.24	16.62	0.55	0.22	17.64	0.57	0.21	2.2 hrs
SimpleNeRF [12]	SIGGRAPH Asia23	-	14.41	0.79	0.25	14.01	0.77	0.25	13.90	0.78	0.26	1.38 hrs
ZeroRF [10]	CVPR24	-	14.84	0.60	0.30	14.47	0.61	0.31	15.73	0.67	0.28	25 mins
FrugalNeRF (Ours)	-	-	19.72	0.78	0.16	22.43	0.83	0.14	24.51	0.86	0.12	6 mins
SparseNeRF [14]	ICCV23	monocular depth	19.83	0.75	0.20	22.47	0.83	0.14	24.03	0.86	0.12	30 mins
FSGS [19]	ECCV24	monocular depth	16.82	0.64	0.27	18.29	0.69	0.21	20.08	0.75	0.16	20 mins
FrugalNeRF (Ours)	-	monocular depth	20.77	0.79	0.15	22.84	0.83	0.13	24.81	0.86	0.12	7 mins

Table 9. **Quantitative results on the DTU [4] dataset with two input views. The three rows show LPIPS, SSIM and PSNR scores, respectively.**

Method \ Scene	Scan21	Scan31	Scan34	Scan38	Scan40	Scan41	Scan45	Scan55	Scan63	Scan82	Scan103	Scan114	Average
FreeNeRF [16]	0.33	0.18	0.31	0.34	0.41	0.35	0.19	0.11	0.07	0.08	0.17	0.12	0.22
	0.51	0.75	0.63	0.61	0.58	0.63	0.76	0.80	0.93	0.90	0.82	0.85	0.73
	13.21	19.33	14.66	16.76	11.42	14.50	18.66	21.62	23.19	21.56	17.55	24.19	18.05
ViP-NeRF [11]	0.37	0.24	0.27	0.38	0.31	0.23	0.31	0.21	0.09	0.12	0.18	0.17	0.24
	0.26	0.49	0.52	0.43	0.47	0.58	0.37	0.39	0.63	0.57	0.65	0.49	0.49
	11.31	13.57	17.13	13.25	15.08	17.81	11.35	16.92	16.71	13.37	16.15	16.24	14.91
SimpleNeRF [12]	0.23	0.32	0.23	0.21	0.24	0.19	0.28	0.22	0.30	0.27	0.19	0.27	0.25
	0.73	0.71	0.76	0.77	0.77	0.84	0.70	0.88	0.75	0.79	0.81	0.82	0.79
	12.71	11.91	14.39	14.50	13.76	15.57	11.88	19.58	12.73	14.37	16.64	14.86	14.41
VGOS [13]	0.28	0.36	0.33	0.31	0.30	0.27	0.37	0.15	0.49	0.45	0.34	0.18	0.32
	0.69	0.67	0.69	0.71	0.73	0.78	0.64	0.90	0.56	0.57	0.73	0.85	0.71
	9.69	8.97	9.75	10.27	8.79	9.75	7.54	19.24	5.17	5.63	11.29	15.81	10.16
SparseNeRF [14]	0.39	0.22	0.26	0.33	0.24	0.21	0.20	0.14	0.08	0.08	0.15	0.13	0.20
	0.45	0.69	0.70	0.60	0.72	0.76	0.75	0.78	0.92	0.91	0.84	0.85	0.75
	14.25	17.95	20.65	17.93	16.33	20.13	18.22	22.29	20.70	23.46	21.70	24.40	19.83
ZeroRF [10]	0.45	0.27	0.35	0.44	0.29	0.28	0.39	0.25	0.13	0.18	0.25	0.29	0.30
	0.30	0.61	0.50	0.39	0.59	0.63	0.49	0.68	0.88	0.82	0.73	0.63	0.60
	10.99	14.40	13.93	12.16	15.41	16.73	11.24	17.08	20.39	15.36	16.23	14.12	14.84
FrugalNeRF (Ours)	0.25	0.16	0.20	0.24	0.24	0.17	0.16	0.13	0.09	0.07	0.13	0.11	0.16
	0.57	0.73	0.73	0.64	0.73	0.78	0.77	0.86	0.92	0.92	0.85	0.89	0.78
	14.67	17.86	19.47	17.66	14.51	19.74	16.94	24.87	21.21	22.67	21.45	25.60	19.72
FrugalNeRF w/ mono. depth (Ours)	0.25	0.15	0.19	0.21	0.23	0.16	0.15	0.12	0.08	0.07	0.10	0.10	0.15
	0.56	0.73	0.75	0.68	0.74	0.79	0.78	0.86	0.93	0.91	0.88	0.90	0.79
	14.14	18.46	21.27	19.40	15.56	20.53	18.05	25.65	23.46	22.72	23.76	26.25	20.77

Table 10. **Quantitative results on the DTU [4] dataset with three input views. The three rows show LPIPS, SSIM and PSNR scores, respectively.**

Method \ Scene	Scan21	Scan31	Scan34	Scan38	Scan40	Scan41	Scan45	Scan55	Scan63	Scan82	Scan103	Scan114	Average
FreeNeRF [16]	15.93	19.53	23.23	19.88	18.38	22.83	21.07	22.88	25.28	26.39	26.68	26.68	22.40
	0.58	0.76	0.80	0.70	0.80	0.84	0.84	0.80	0.94	0.94	0.92	0.90	0.82
	15.93	19.53	23.23	19.88	18.38	22.83	21.07	22.88	25.28	26.39	26.68	26.68	22.40
ViP-NeRF [11]	0.34	0.18	0.26	0.32	0.32	0.28	0.22	0.22	0.09	0.11	0.12	0.12	0.22
	0.33	0.58	0.58	0.53	0.47	0.55	0.50	0.43	0.66	0.65	0.77	0.60	0.55
	12.97	16.58	18.63	16.12	14.82	16.25	14.14	18.04	17.67	14.75	20.85	18.65	16.62
SimpleNeRF [12]	0.22	0.32	0.24	0.24	0.28	0.27	0.23	0.15	0.31	0.36	0.17	0.25	0.25
	0.74	0.68	0.74	0.75	0.75	0.77	0.79	0.90	0.77	0.67	0.84	0.81	0.77
	12.90	11.29	14.17	13.42	11.44	12.23	15.31	20.41	13.97	10.93	17.41	14.66	14.01
VGOS [13]	0.28	0.38	0.29	0.26	0.28	0.27	0.38	0.16	0.51	0.47	0.29	0.15	0.31
	0.69	0.65	0.71	0.76	0.74	0.76	0.62	0.90	0.58	0.58	0.75	0.87	0.72
	9.84	8.34	10.50	11.91	8.51	9.14	7.27	18.86	5.38	5.80	11.81	16.74	10.34
SparseNeRF [14]	0.23	0.12	0.15	0.37	0.14	0.14	0.12	0.14	0.04	0.04	0.11	0.08	0.14
	0.63	0.81	0.79	0.59	0.84	0.84	0.84	0.84	0.96	0.95	0.90	0.92	0.83
	17.14	21.11	24.88	12.36	22.25	23.05	20.85	19.75	27.52	28.98	23.74	28.00	22.47
ZeroRF [10]	0.45	0.36	0.41	0.45	0.29	0.30	0.33	0.27	0.19	0.19	0.24	0.30	0.31
	0.33	0.55	0.47	0.41	0.65	0.68	0.57	0.68	0.84	0.83	0.74	0.63	0.61
	11.55	12.43	11.81	12.84	15.66	16.01	12.77	16.50	17.81	15.34	16.64	14.25	14.47
FrugalNeRF (Ours)	0.19	0.14	0.18	0.22	0.21	0.13	0.13	0.12	0.06	0.05	0.10	0.11	0.14
	0.69	0.76	0.77	0.69	0.79	0.84	0.82	0.89	0.94	0.94	0.89	0.90	0.83
	17.38	19.06	22.38	18.96	17.77	24.01	20.35	26.11	24.57	25.85	25.43	27.28	22.43
FrugalNeRF w/ mono. depth (Ours)	0.19	0.13	0.17	0.21	0.20	0.13	0.13	0.12	0.06	0.05	0.08	0.10	0.13
	0.68	0.78	0.78	0.73	0.79	0.84	0.82	0.88	0.95	0.93	0.91	0.91	0.83
	17.14	19.89	23.17	20.33	17.18	23.71	20.59	26.60	25.52	25.04	27.84	27.10	22.84

Table 11. **Quantitative results on the DTU [4] dataset with four input views. The three rows show LPIPS, SSIM and PSNR scores, respectively.**

Method \ Scene	Scan21	Scan31	Scan34	Scan38	Scan40	Scan41	Scan45	Scan55	Scan63	Scan82	Scan103	Scan114	Average
FreeNeRF [16]	0.18	0.14	0.13	0.24	0.14	0.12	0.09	0.06	0.04	0.03	0.08	0.07	0.11
	0.72	0.81	0.83	0.72	0.85	0.86	0.86	0.92	0.96	0.96	0.93	0.93	0.86
	18.72	21.29	25.97	19.43	22.88	25.59	22.39	28.63	27.35	31.51	27.30	28.65	24.98
ViP-NeRF [11]	0.33	0.19	0.21	0.31	0.35	0.24	0.23	0.24	0.08	0.08	0.10	0.12	0.21
	0.39	0.61	0.59	0.59	0.45	0.61	0.52	0.38	0.67	0.67	0.76	0.64	0.57
	14.24	17.22	19.44	18.19	15.76	18.84	15.57	16.62	17.19	16.45	22.67	19.50	17.64
SimpleNeRF [12]	0.27	0.28	0.23	0.25	0.32	0.27	0.25	0.21	0.27	0.27	0.18	0.29	0.26
	0.71	0.73	0.78	0.75	0.72	0.76	0.78	0.88	0.82	0.80	0.84	0.81	0.78
	11.81	12.95	14.72	12.71	10.42	11.67	14.12	18.84	14.05	14.43	16.87	14.23	13.90
VGOS [13]	0.27	0.35	0.31	0.28	0.27	0.27	0.37	0.16	0.43	0.42	0.28	0.18	0.30
	0.73	0.69	0.71	0.74	0.76	0.78	0.64	0.90	0.66	0.66	0.75	0.85	0.74
	11.09	9.53	10.57	11.15	9.12	10.00	8.10	19.53	6.55	7.14	12.69	15.65	10.93
SparseNeRF [14]	0.16	0.14	0.15	0.21	0.21	0.14	0.10	0.09	0.04	0.05	0.09	0.06	0.12
	0.72	0.80	0.85	0.74	0.80	0.86	0.86	0.88	0.95	0.95	0.93	0.93	0.86
	18.60	20.99	25.87	20.92	19.45	24.81	22.15	26.37	26.20	26.72	28.10	28.19	24.03
ZeroRF [10]	0.43	0.32	0.28	0.44	0.28	0.25	0.20	0.29	0.17	0.14	0.26	0.32	0.28
	0.36	0.62	0.66	0.47	0.68	0.73	0.73	0.67	0.87	0.87	0.72	0.62	0.67
	11.75	13.48	16.47	13.53	16.87	17.26	16.48	15.92	19.33	19.12	15.18	13.36	15.73
FrugalNeRF (Ours)	0.17	0.12	0.16	0.17	0.19	0.12	0.12	0.12	0.05	0.04	0.07	0.10	0.12
	0.73	0.81	0.81	0.79	0.81	0.85	0.85	0.89	0.95	0.95	0.93	0.92	0.86
	19.21	21.84	24.99	23.08	19.47	25.64	21.59	27.31	26.27	27.26	29.27	28.21	24.51
FrugalNeRF w/ mono. depth (Ours)	0.17	0.12	0.15	0.17	0.19	0.12	0.11	0.12	0.05	0.03	0.07	0.09	0.12
	0.73	0.81	0.82	0.80	0.82	0.86	0.86	0.90	0.96	0.95	0.93	0.92	0.86
	19.07	21.65	25.82	23.13	18.96	25.55	22.21	28.02	26.87	28.28	29.27	28.92	24.81

Table 12. **Quantitative results on the RealEstate-10K [18] dataset.** For SimpleNeRF [12] and ViP-NeRF [11], we calculate metrics using testing data provided in their respective clouds. As for other models, we rely on the scores provided in the SimpleNeRF paper.

Method	Venue	Learned priors	2-view			3-view			4-view			Training time ↓
			PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	
TensoRF [2]	ECCV 2022	-	24.81	0.78	0.17	24.86	0.78	0.17	24.84	0.78	0.17	11 mins
RegNeRF [7]	CVPR 2022	normalizing flow	16.87	0.59	0.45	17.73	0.61	0.44	18.25	0.62	0.44	2.35 hrs
DS-NeRF [3]	CVPR 2022	-	25.44	0.79	0.32	25.94	0.79	0.32	26.28	0.79	0.33	3.5 hrs
DDP-NeRF [9]	CVPR 2022	depth completion	26.15	0.85	0.15	25.92	0.85	0.16	26.48	0.86	0.16	3.5 hrs
FreeNeRF [16]	CVPR 2023	-	14.50	0.54	0.55	15.12	0.57	0.54	16.25	0.60	0.54	1.5 hrs
ViP-NeRF [11]	SIGGRAPH 2023	-	29.55	0.87	0.09	29.75	0.88	0.11	30.47	0.88	0.11	13.5 hrs
SimpleNeRF [12]	SIGGRAPH Asia 2023	-	30.30	0.88	0.07	31.40	0.89	0.08	31.73	0.89	0.09	9.5 hrs
FrugalNeRF (Ours)	-	-	30.12	0.87	0.07	31.04	0.89	0.06	31.78	0.90	0.06	20 mins

Table 13. **Quantitative results on the RealEstate-10K [18] dataset with two input views. The three rows show LPIPS, SSIM, and PSNR scores, respectively.**

Scene / Method	0	1	3	4	6	Average
RegNeRF [7]	0.35	0.32	0.49	0.54	0.54	0.45
	0.60	0.83	0.30	0.61	0.59	0.59
	16.51	21.04	13.88	17.13	15.79	16.87
DS-NeRF [3]	0.26	0.27	0.51	0.24	0.31	0.32
	0.81	0.91	0.50	0.88	0.83	0.79
	24.68	27.93	19.24	29.18	26.18	25.44
DDP-NeRF [9]	0.11	0.12	0.34	0.06	0.11	0.15
	0.89	0.95	0.56	0.94	0.92	0.85
	25.90	25.87	18.97	32.01	28.00	26.15
FreeNeRF [16]	0.45	0.50	0.64	0.67	0.48	0.55
	0.54	0.77	0.28	0.49	0.58	0.53
	15.00	17.00	12.15	12.84	15.50	14.50
ViP-NeRF [11]	0.05	0.05	0.22	0.04	0.08	0.09
	0.94	0.97	0.56	0.95	0.93	0.87
	30.41	32.03	18.96	34.74	31.61	29.55
SimpleNeRF [12]	0.04	0.04	0.21	0.03	0.05	0.07
	0.95	0.97	0.56	0.95	0.96	0.88
	31.89	33.8	18.65	34.93	32.24	30.30
FrugalNeRF (Ours)	0.04	0.04	0.20	0.04	0.05	0.07
	0.94	0.97	0.56	0.95	0.95	0.87
	30.13	34.69	18.35	35.00	32.45	30.12

Table 14. **Quantitative results on the RealEstate-10K [18] dataset with three input views. The three rows show LPIPS, SSIM, and PSNR scores, respectively.**

Method \ Scene	0	1	3	4	6	Average
RegNeRF [7]	0.40 0.60 15.99	0.32 0.82 20.89	0.53 0.29 13.87	0.56 0.62 17.60	0.37 0.71 20.28	0.44 0.61 17.73
DS-NeRF [3]	0.24 0.83 25.24	0.26 0.91 28.68	0.53 0.49 19.14	0.26 0.87 29.08	0.31 0.85 27.58	0.32 0.79 25.94
DDP-NeRF [9]	0.11 0.89 25.27	0.11 0.96 26.67	0.38 0.55 18.81	0.06 0.94 31.84	0.13 0.92 26.99	0.16 0.85 25.92
FreeNeRF [16]	0.54 0.53 13.79	0.51 0.75 15.59	0.64 0.29 12.45	0.59 0.61 15.72	0.42 0.66 18.05	0.54 0.57 15.12
ViP-NeRF [11]	0.06 0.94 30.66	0.10 0.95 29.89	0.26 0.60 19.59	0.04 0.95 35.17	0.08 0.95 33.43	0.11 0.88 29.75
SimpleNeRF [12]	0.04 0.95 32.23	0.04 0.98 36.44	0.23 0.61 19.65	0.03 0.95 35.85	0.08 0.95 32.81	0.08 0.89 31.40
FrugalNeRF (Ours)	0.04 0.95 31.11	0.03 0.98 35.39	0.18 0.61 18.85	0.03 0.95 35.78	0.04 0.96 34.07	0.06 0.89 31.04

Table 15. **Quantitative results on the RealEstate-10K [18] dataset with four input views. The three rows show LPIPS, SSIM, and PSNR scores, respectively.**

Method \ Scene	0	1	3	4	6	Average
RegNeRF [7]	0.43 0.59 16.09	0.35 0.83 20.98	0.59 0.29 13.91	0.56 0.65 18.48	0.27 0.75 21.78	0.44 0.62 18.25
DS-NeRF [3]	0.27 0.82 25.40	0.26 0.92 29.40	0.56 0.50 19.64	0.25 0.87 29.26	0.31 0.85 27.69	0.33 0.79 26.28
DDP-NeRF [9]	0.12 0.89 25.14	0.08 0.96 28.57	0.39 0.58 19.57	0.06 0.93 31.73	0.13 0.91 27.36	0.16 0.86 26.48
FreeNeRF [16]	0.56 0.53 13.84	0.48 0.80 17.93	0.65 0.31 12.69	0.58 0.66 17.29	0.39 0.69 19.48	0.53 0.60 16.25
ViP-NeRF [11]	0.06 0.94 31.64	0.08 0.96 32.24	0.27 0.62 20.35	0.05 0.94 34.84	0.09 0.95 33.28	0.11 0.88 30.47
SimpleNeRF [12]	0.04 0.96 32.95	0.05 0.97 36.44	0.24 0.64 20.52	0.03 0.95 35.97	0.09 0.94 32.77	0.09 0.89 31.73
FrugalNeRF (Ours)	0.04 0.96 32.29	0.03 0.98 36.06	0.17 0.64 19.81	0.03 0.95 36.54	0.05 0.96 34.22	0.06 0.90 31.78

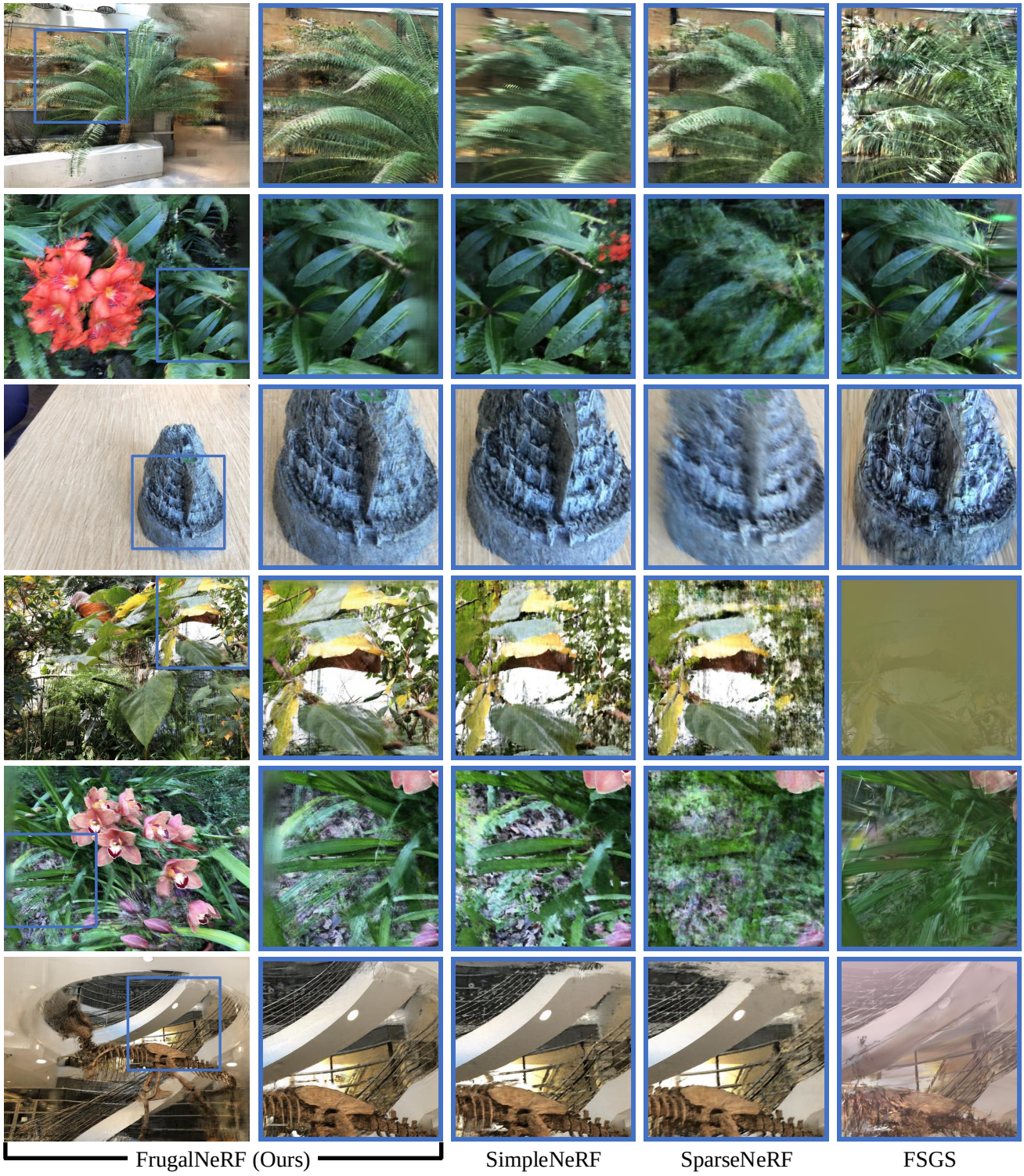


Figure 3. **More qualitative comparisons on the LLFF [6] dataset with two input views.** FrugalNeRF achieves better synthesis quality in different scenes.

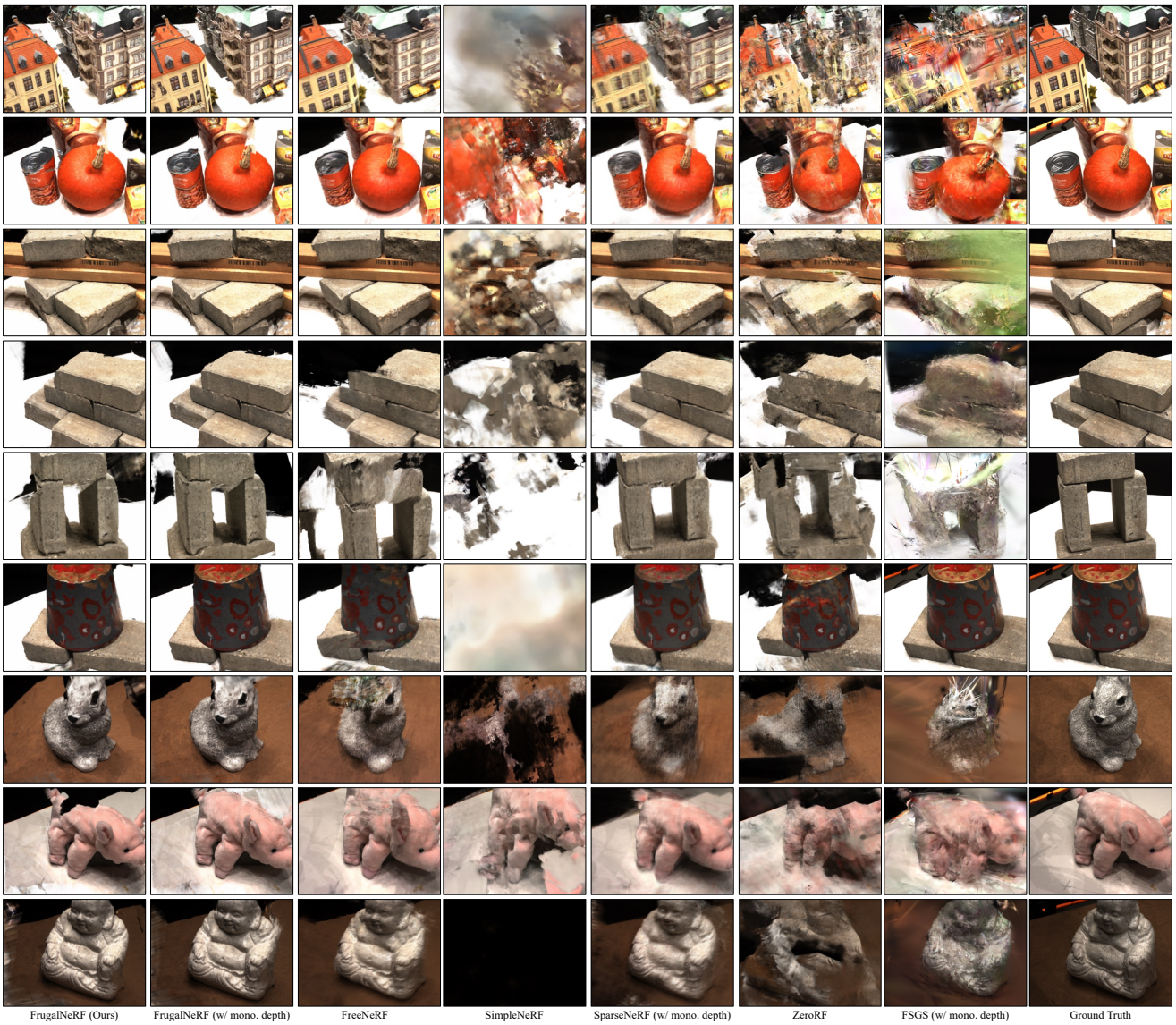


Figure 4. **More qualitative comparisons on the DTU [4] dataset with two input views.** FrugalNeRF achieves better synthesis quality in different scenes.

Figure 5. **Qualitative comparisons on the RealEstate-10K [18] dataset with two input views.** Compared to Vip-NeRF [11] and SimpleNeRF [12], our FrugalNeRF renders sharper details in the scene.

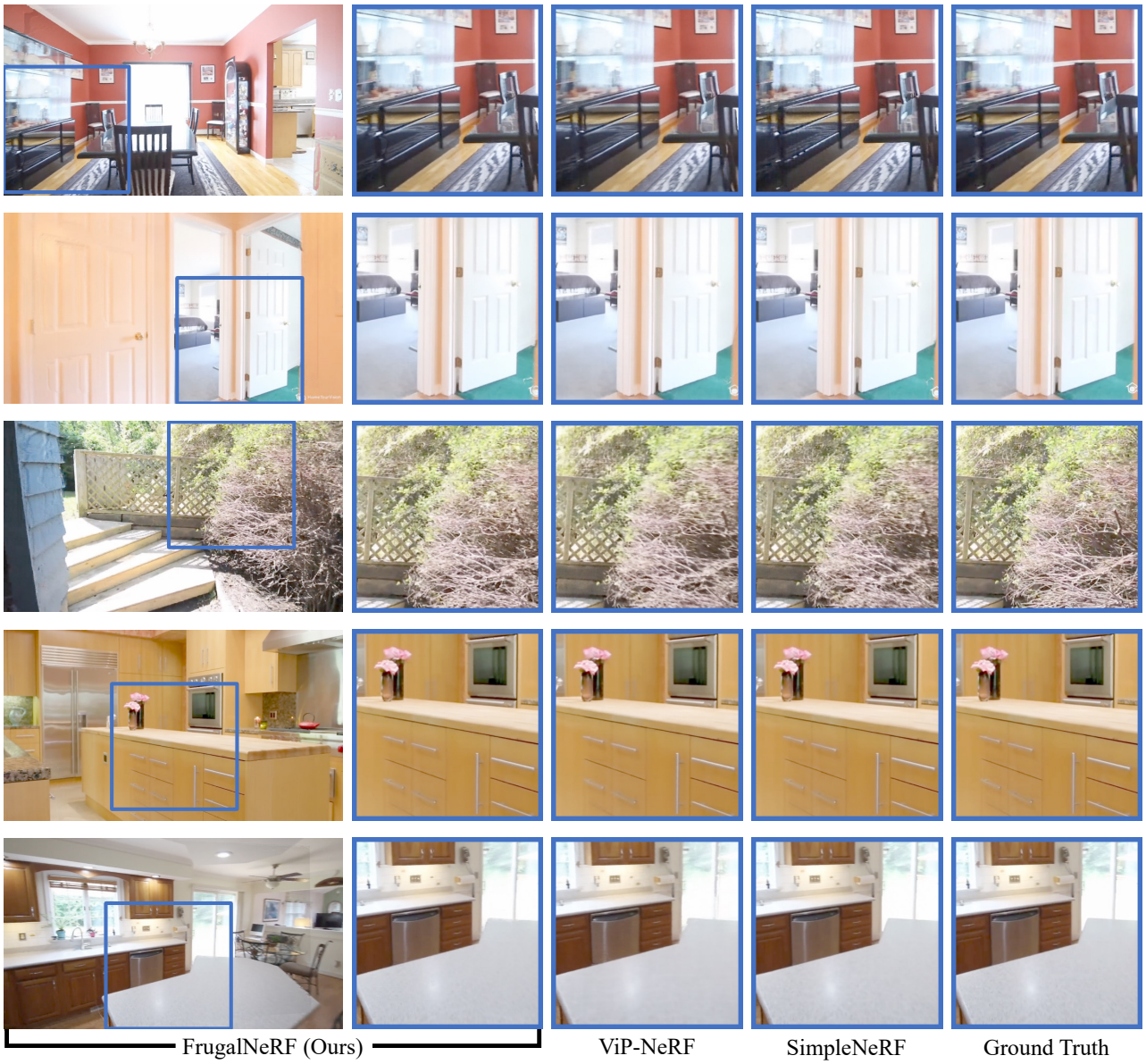


Figure 6. **More qualitative comparisons on the RealEstate-10K [18] dataset with two input views.** FrugalNeRF achieves synthesis quality comparable to the state-of-the-art methods.

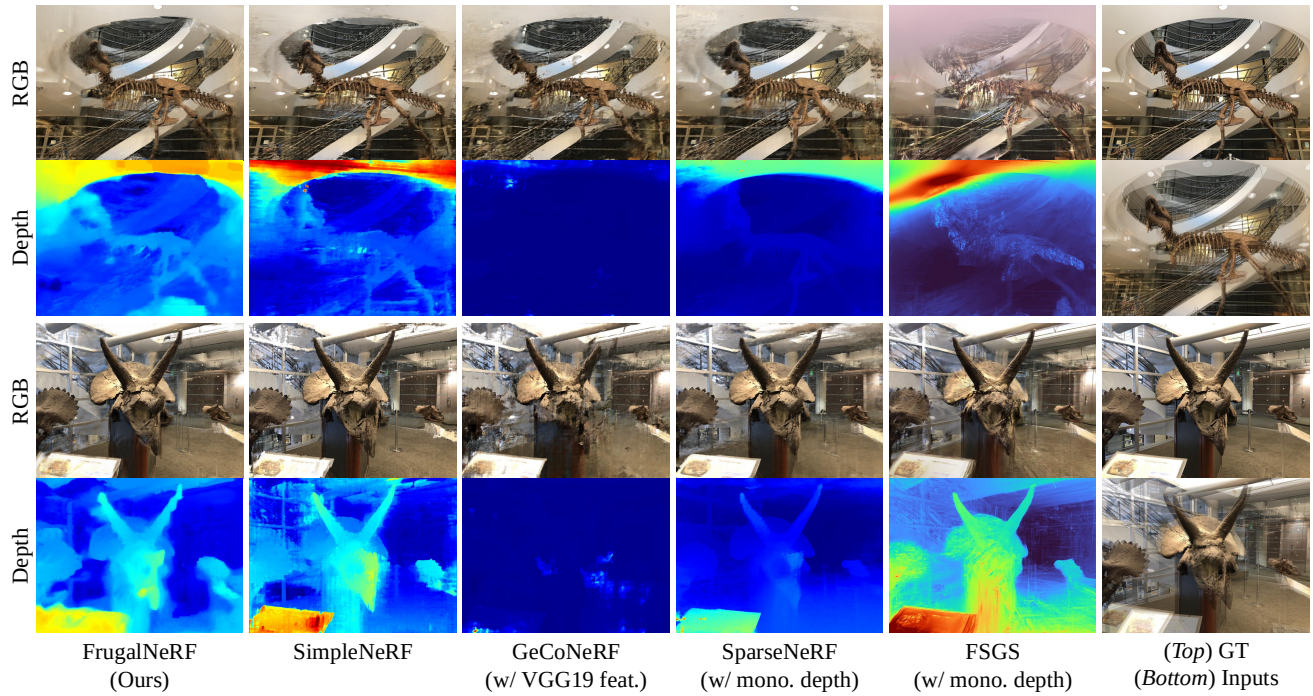


Figure 7. **Render depth map comparisons on the LLFF [6] dataset with two input views.** FrugalNeRF achieves better synthesis quality in different scenes.

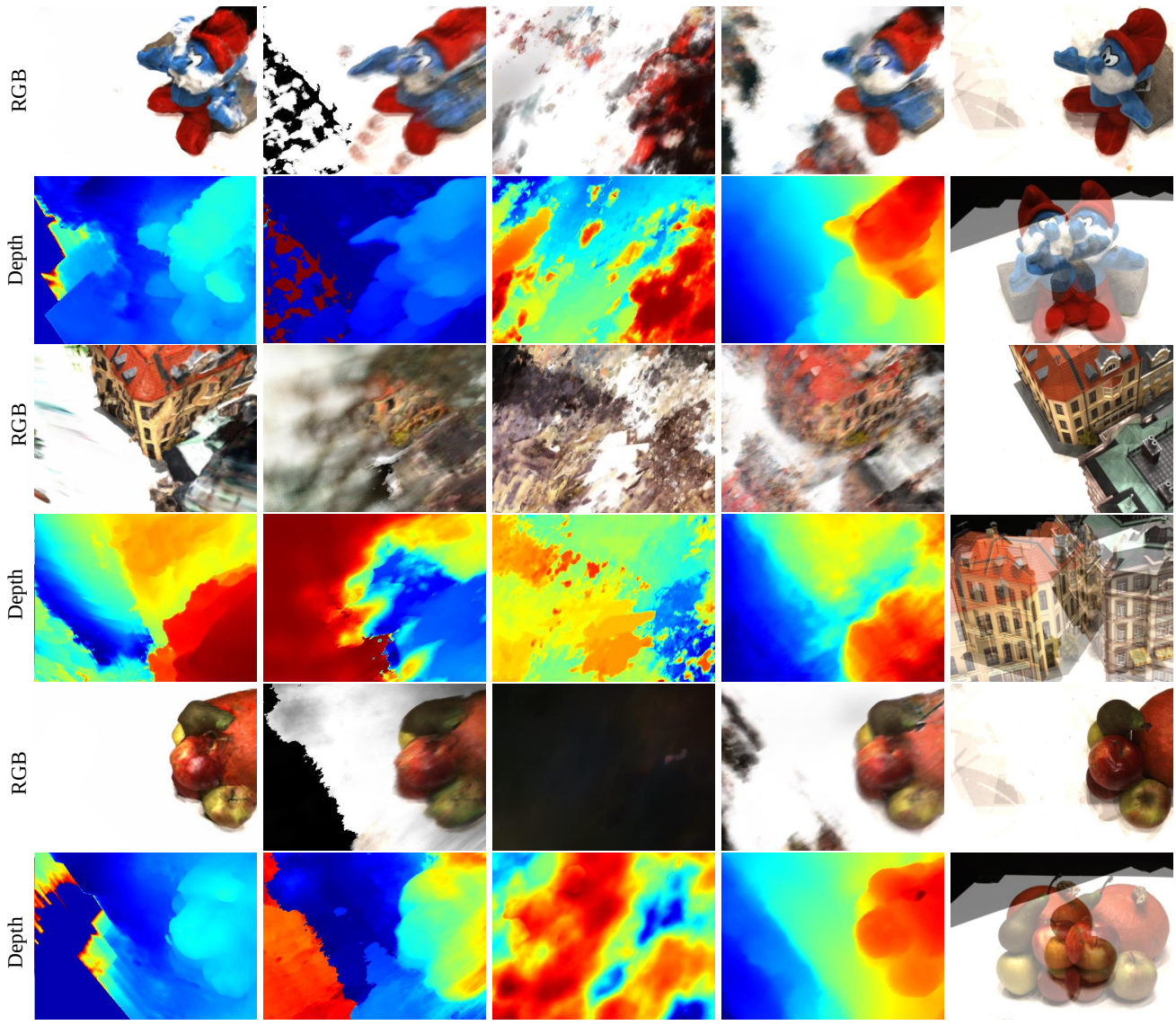


Figure 8. **Render depth map comparisons on the DTU [4] dataset with two input views.** FrugalNeRF achieves better synthesis quality in different scenes.