DNGaussian: Optimizing Sparse-View 3D Gaussian Radiance Fields with Global-Local Depth Normalization

Supplementary Material

Overview

In the supplemental document, we first report additional studies in Sec. A of our proposed depth normalization, neural color renderer, and the performance of previous methods on fast grid-based backbones. Then, we describe the details of our implementation and dataset settings in our experiment in Sec. B. Finally, we discuss the limitations and future work of our method in Sec. C. More results can be found in our supplementary video.

A. Additional Results

A.1. Ablation Study on Depth Normalization

To better illustrate the roles of our Local and Global Depth Normalization, we conduct an additional ablation study and replace the L2 loss function with L1 to avoid its reduction of small losses. The quantitative visualization results are shown in Table 1 and Figure 1. In the comparison, we separately apply the Global and the Local one to illustrate the strengths and weaknesses of each: 1) Although the global one can also individually support the model to learn an overall scene, it is weak in optimizing minor errors, as we have discussed in Sec.3.3. 2) The local one can not stand alone due to the lack of absolute scale, but provides rich information on local depth changes. 3) By combining both techniques, our Global-Local Depth Normalization can simultaneously obtain the knowledge of both global scale and small local errors and achieve the best. Notably, since a different type of loss is used in this study, the scores vary from those reported in the main paper. Despite this, our method still performs well particularly in LPIPS and SSIM, which demonstrates the robustness of our depth normalization.

Setting	PSNR↑	LPIPS↓	SSIM↑	AVGE↓
Only Global	18.32	0.309	0.579	0.144
Only Lobal	17.17	0.338	0.523	0.167
All	**18.67**	**0.291**	**0.595**	**0.137**

Table 1. **Additional Ablation Study on Depth Normalization.** Combined with both two proposed depth normalizations, our Global-Local Depth Normalization achieves the best quality.

A.2. Ablation Study on Neural Color Renderer

In this work, we replace the spherical harmonic (SH) of 3D Gaussian Splatting with a neural color renderer to represent the direction-variant color. To better illustrate the function of this module, we compare it to the original SH function

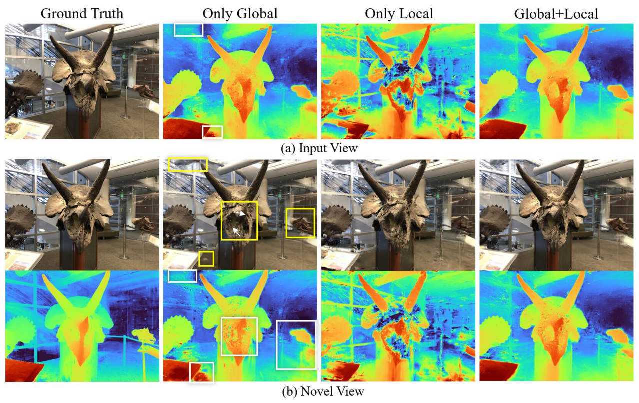

Figure 1. **Visualization of two proposed Depth Normalization.** The color and depth map of the input view and synthesized novel view are shown in (a) and (b). The global one provides a global view of the whole scene, however, is weak in handling small local errors (white box), which causes blurry and wrong appearances (yellow box). In contrast, the local one is more sensitive to local depth changes. By combining both of them, our method can learn a more accurate scene geometry. Zoom in for better visualization.

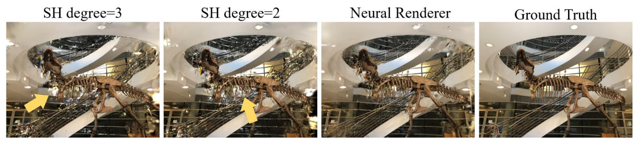

Figure 2. **Visualization of two proposed Depth Normalization.** In sparse-view situations, SH may produce inconsistent colors in unseen views (yellow arrow) due to overfitting, which can be relieved by a neural color renderer.

with different degrees in the LLFF dataset with 3 training views. The results are in Table 2 and Figure 2. The SH function is easy to overfit in the sparse-view situation and results in some strange colors during view changing. This may be caused by the independence of each primitive which leads to a lack of regional consistency. After introducing the neural color renderer, the problem has been relieved. By storing the intermediate result and only calculating the latest two MLP layers, we can maintain a fast rendering speed competitive to SH as well.

A.3. Transfer of Previous Strategies

In this section, we conduct an experiment to illustrate the necessity of our efficient DNGaussian. Indeed, there are some existing methods like FreeNeRF [21] and SparseNeRF [18] that are low in efficiency mainly due to their backbone rather than the strategy itself. However, they have only

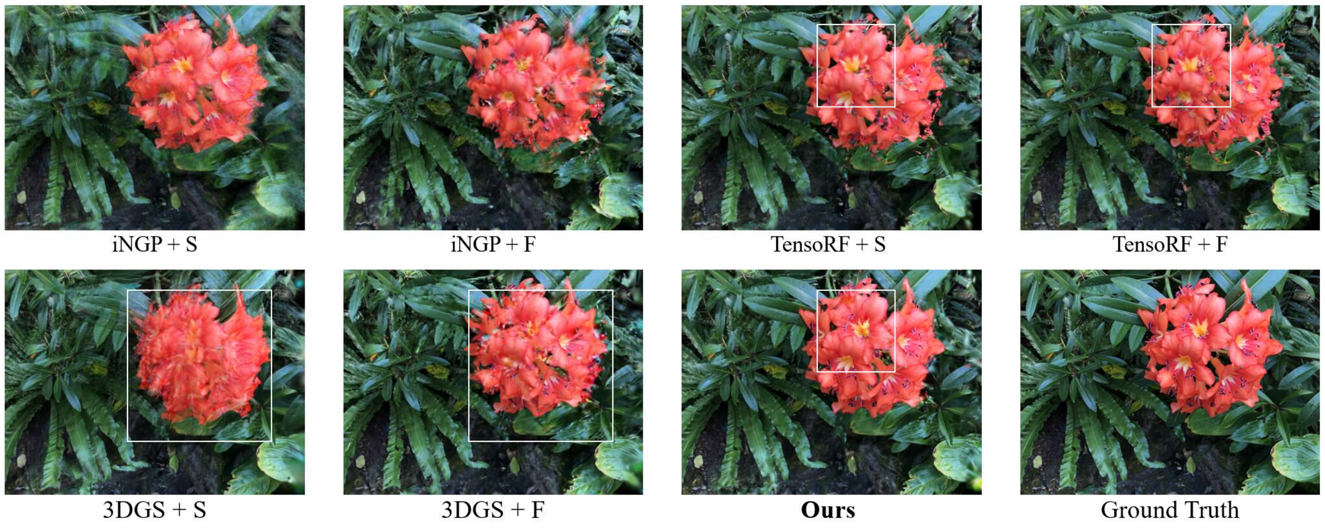

Figure 3. Visualization of the implicit-based SOTA methods SparseNeRF (S) [18] and FreeNeRF (F) [21] when transferred to fast backbones Instant-NGP (iNGP) [10], TensoRF [3], and 3D Gaussian Splatting (3DGS) [8]. The depth-based SparseNeRF causes blurry rendering on all fast backbones, while FreeNeRF has less effect on the improvement of quality. Instead, our DNGaussian achieves the best quality on the most efficient 3DGS backbone, which shows the necessity of our method.

Setting	PSNR↑	LPIPS↓	SSIM↑	AVGE↓	FPS
SH degree=2	17.06	0.333	0.549	0.167	**340**
SH degree=3	17.11	0.328	0.560	0.164	300
Neural Renderer	**19.12**	**0.294**	**0.591**	**0.132**	300

Table 2. **Ablation Study on Neural Color Renderer.** Our neural color renderer successfully improves the rendering quality while keeping an equally fast inference speed.

already been proven effective for some implicit backbones that are slow and costly. To verify whether they can directly transfer to faster backbones to achieve higher efficiency, we implement these two methods on two fast grid-based Instant-NGP [10] and TensoRF [3]. Also, we do this on our 3D Gaussian Splatting (3DGS) [8] backbone. Then, we test these implementations in the LLFF 3-view setting. The results are shown in Table 3 and Figure 3. Additionally, we report the training time (Time), GPU memory cost (VM Cost), and the inference FPS for each item.

Implementation details. We utilize a CUDA-implemented ray marching [1] for the two grid-based backbones to achieve faster speed and lower costs. The 3DGS backbone employs the same neural color renderer as our method. We follow the original implementation of SparseNeRF to produce monocular depth maps for all input views and transfer its Local Depth Ranking Distillation to these new backbones with the same hyperparameters. For FreeNeRF, since the three fast backbones do not contain a frequency-based positional encoding, we apply the Frequency Regularization to their grid-based positional encoding as an alternative.

Comparison on grid-based backbones. Although these two methods perform well on their original implicit Mip-NeRF, they are weak in both Instant-NGP and TensoRF. SparseNeRF distills the depth ranking from the monocular depth map for regularization, however, causes more blurs. This may be caused by the stronger spatial memory ability from the explicit grids that makes it easier to memorize noises. FreeNeRF performs even worse on both these two backbones, which may be due to the different representations of positional encoding. In TensoRF, all these two strategies fail to improve performance. One reason may lie in that TensoRF directly utilizes explicit grids without a neural decoder to store density value, which is more difficult to regularize.

Comparison on 3DGS. In the comparison, the 3DGS backbone achieves the best efficiency, with the fast FPS and lowest training cost. However, both SparseNeRF and FreeNeRF cannot effectively regularize this powerful and efficient backbone. Due to the lack of frequency positional encoding, FreeNeRF serves more like a coarse-to-fine strategy and leads to only a little improvement. From the visualization of SparseNeRF in Figure 3, it can be observed that it is insufficient in the 3D Gaussian radiance fields of 3DGS to only keep the depth ranking and wait for the color-supervised optimization process to refine the detailed geometry. Compared with these two methods, our DNGaussian achieves a much better quality with only a little increment of training time. With less noise in the learned geometry, our method also achieves a faster inference speed.

Conclusion. The experiments show that the previous methods for implicit backbones can hardly, at least in an easy way, transfer to current fast backbones. Also, they are not suitable for the 3D Gaussian radiance fields. In such a sit-

[1] https://github.com/ashawkey/torch-ngp

Backbone	Strategy	PSNR ↑	LPIPS ↓	SSIM ↑	Time ↓	VM Cost ↓	FPS ↑
Mip-NeRF [1]	None	14.62	0.495	0.351	2.2h	≥ 32 GB	0.09
	FreeNeRF	19.63	<u>0.308</u>	0.612	2.3h		
	SparseNeRF	**<u>19.86</u>**	0.328	**<u>0.624</u>**	1.5h		
Instant-NGP [10]	None	17.19	0.483	0.469	3.8min	3 GB	3
	FreeNeRF	15.30	0.516	0.369	4.2min		
	SparseNeRF	<u>17.19</u>	<u>0.478</u>	<u>0.476</u>	7.5min		
TensoRF [3]	None	<u>16.16</u>	<u>0.454</u>	<u>0.443</u>	4.1min	8 GB	5
	FreeNeRF	15.78	0.466	0.430	4.5min		
	SparseNeRF	16.11	0.465	0.443	8.9min		
3DGS [8]	None	16.46	0.401	0.440	**2.7min**	2 GB	280
	FreeNeRF	16.55	0.399	0.472	2.7min		
	SparseNeRF	16.80	0.374	0.504	2.9min		
3DGS [8]	Ours	<u>19.12</u>	**<u>0.294</u>**	<u>0.591</u>	3.5min	**2 GB**	**300**

Table 3. **Comparision of SOTA strategies FreeNeRF [21] and SparseNeRF [18] with different backbones**. The best results for all and for each backbone are marked with **bold** and <u>underline</u>. Although FreeNeRF and SparseNeRF perform well on the implicit Mip-NeRF [1], they can weakly improve the quality with current fast backbones Instant-NGP [10], TensoRF [3], and also the 3DGS [8] in our work.

uation, our DNGaussian shows significant value in providing an efficient way for high-quality and low-cost few-shot novel view synthesis.

A.4. Comparison with Grid-based Methods

There are some works [16, 17, 20] that utilize a grid-based backbone to improve the training efficiency. Since DaRF [16] is evaluated on another two datasets with at least 9 input views, while VGOS [17] and DiffusioNeRF [20] use different methods for the measurement of metrics, we do not take them as baselines in the main paper. Here we list the scores of VGOS and DiffusioNeRF in Table 4 in the LLFF 3-view setting for comparison. For VGOS, we only report scores for which the measurement method is definitely the same as in RegNeRF [11] and FreeNeRF [21]. The results of DiffusioNeRF are obtained from its updated paper on arXiv [2]. In the comparison, our method outperforms the other two with the highest scores in LPIPS, SSIM, and AVGE. In fact, our method also achieves the best in efficiency, with much lower cost and faster inference.

A.5. Additional Visualizations

We provide more rendering results in our experiments. The examples on DTU and LLFF with 3 training views are shown in Figure 5 and 6. We have also shown more quantitative comparison in the Blender 8-view setting with the SOTA method FreeNeRF [21] in Figure 4. More results can be found in our supplementary video.

[2]https://arxiv.org/abs/2302.12231

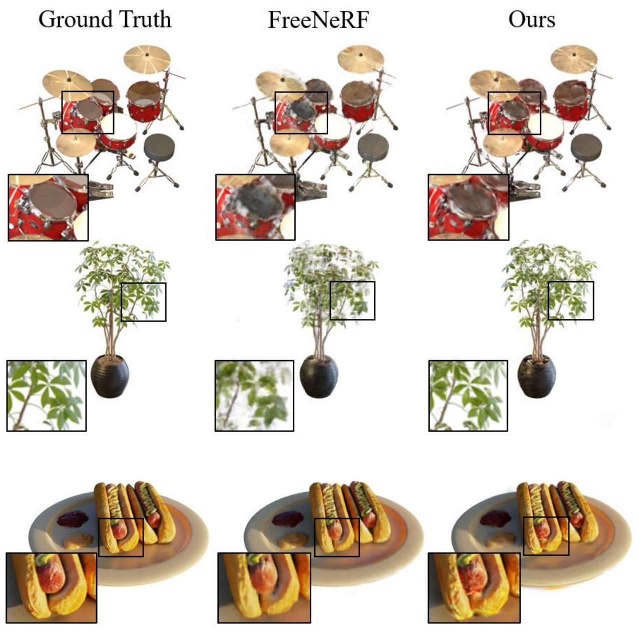

Figure 4. **Qualitative comparison on the Blender dataset with 8 input views.** FreeNeRF [21] learns the accurate geometry by masking high-frequency signals, however, suffers from blurry details as the trade-off. In contrast, our method does not explicitly constrain the learning of high-frequency content. Also attributed to the 3D Gaussian neural fields, our method performs better in the fine-grained details.

B. Details

B.1. Implementations

Pre-trained Depth Models. In this work, we use the pre-trained DPT [12, 13] estimator to predict the depth map, which has been widely used in many NeRF-based

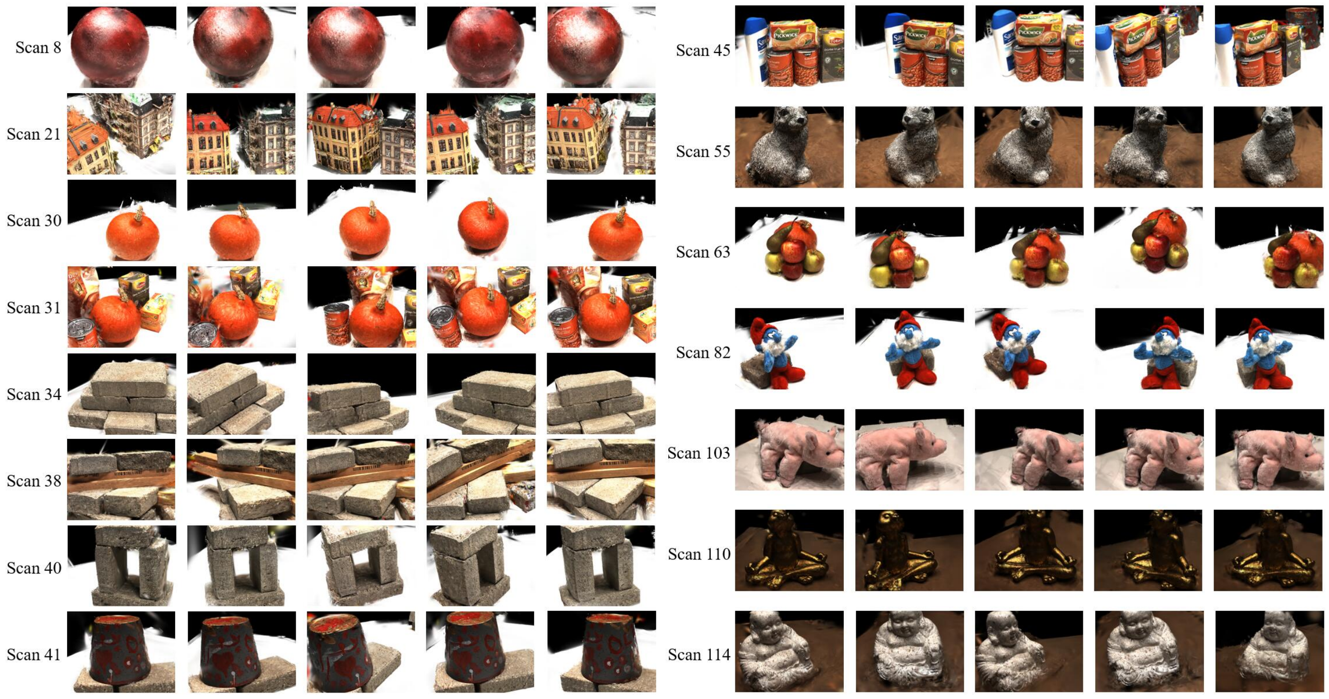

Figure 5. **Examples of the synthesized novel view results from DNGaussian with 3 input views on the DTU dataset.**

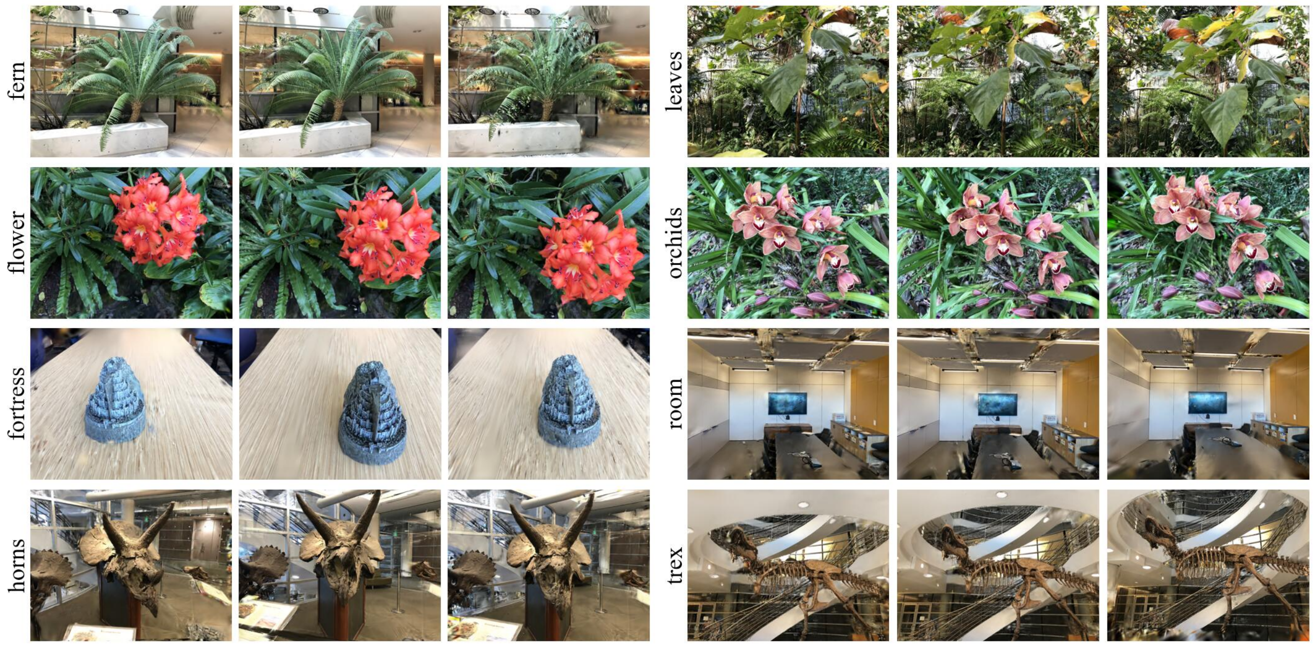

Figure 6. **Examples of the synthesized novel view results from DNGaussian with 3 input views on the LLFF dataset.**

Method	PSNR↑	LPIPS↓	SSIM↑	AVGE↓
VGOS [17]	19.35	0.432	-	-
DiffusioNeRF [20]	**19.79**	0.338	0.568	0.136
Ours	19.12	**0.294**	**0.591**	**0.132**

Table 4. **Comparison with grid-based few-shot NeRFs on LLFF with 3 training views.** Our method outperforms grid-based methods VGOS [17] and DiffusioNeRF [17].

Type	PSNR ↑	LPIPS ↓	SSIM ↑	AVGE ↓
LLFF				
dpt_hybrid_384*	19.12	0.294	0.591	0.132
dpt_large_384	19.03	0.297	0.590	0.135
DTU				
dpt_hybrid_384	18.86	0.179	0.784	0.106
dpt_large_384*	18.91	0.176	0.790	0.102

Table 5. **The influence of different pre-trained depth models.** We replace the pre-trained depth model with a different type in our LLFF and DTU settings while keeping the same hyperparameters. The results show that our method is robust to different monocular depth estimators. * denotes the default type of each dataset.

works [2, 4, 18, 19]. Particularly, we use the type of *dpt_hybrid_384* for the LLFF dataset, while *dpt_large_384* for DTU and Blender, which performs better for the pure white or black background. In fact, the performance gaps of our method when applying different types of depth models are slight, as shown in Table 5.

Patch Size. In our implementation, we randomly sample a patch size from a pre-defined range for our patch-based Global-Local Depth Normalization. This range is set of $[5, 17]$ for LLFF and Blender, and a larger $[17, 51]$ for DTU since the objects are smaller but occupy a large proportion of the image. Due to the flexibility of our normalization, we do not need to separately tune this value for each scene.

Metrics. Following previous methods [11, 21], we utilize the "structural_similarity" API from scikit-image [3] to compute the SSIM score, and use the implementation with a pre-trained VGG model to calculate the LPIPS score.

Camera Poses. Following existing works [6, 11, 18, 21], we assume all camera poses are already known. In practice, for LLFF and Blender, we use the given poses from the datasets. For DTU, we use COLMAP [14, 15] to calculate the camera poses according to all given views, and then sample the target sparse views from the results.

B.2. Datasets

LLFF. The LLFF dataset [9] contains 8 forward-facing scenes in total. Following [11, 18, 21], we take every 8-th image as the novel views for testing. The input views are evenly sampled across the remaining views. Images are downsampled $8\times$ to the resolution of 378×504. In practice, we ignore the distortion of the original images.

[3]https://scikit-image.org/docs/stable/api/skimage.metrics.html

DTU. The DTU dataset [7] consists of 124 object-centric scenes captured by a set of fixed cameras. We follow [11, 18, 21] to evaluate models directly on the 15 scenes with the scan IDs of 8, 21, 30, 31, 34, 38, 40, 41, 45, 55, 63, 82, 103, 110, and 114. In each scan, the images with the following IDs of 25, 22, and 28 are used as the input views in our 3-view setting. The test set consists of images with IDs of 1, 2, 9, 10, 11, 12, 14, 15, 23, 24, 26, 27, 29, 30, 31, 32, 33, 34, 35, 41, 42, 43, 45, 46 and 47 for evaluation. The images are downsampled $4\times$. In particular, we use the undistorted images from COLMAP to eliminate the negative impact of unerased lens distortion.

Blender. We follow the data split used in [6, 21] for the Blender dataset [5]. The 8 input views are selected from the training images, with IDs 26, 86, 2, 55, 75, 93, 16, 73, and 8. The 25 test views are sampled evenly from the testing images for evaluation. All images are downsampled $2\times$ to 400×400 during the experiment.

C. Discussions and Limitations

Our DNGaussian utilizes coarse monocular depth to regularize the scene geometry in situations with sparse input views, and achieves significant improvement in the appearance quality. However, our method still has limitations such as below. We hope these issues can be solved in future work.

More Input Views. Besides only 3 input views, we have also explored the performance when the number of input views increases to 6 and 9 on the LLFF dataset, as shown in Table 6. In the experiment, it can be observed that as the number of views increases, the performance of the baseline also improves. Our DNGaussian can still improve the quality of the synthesized novel view with 6 input views. However, it does not work well when the number of input views increases to 9, which is nearly enough to provide sufficient color constraints. This may be due to the errors in the depth map that negatively influence the optimization process. The next step of our work can lie in leveraging the uncertainty of the monocular depth to filter out unreliable supervision.

Views	Method	PSNR ↑	LPIPS ↓	SSIM ↑	AVGE ↓
3	3DGS	15.52	0.405	0.408	0.209
	3DGS†	16.46	0.401	0.440	0.192
	DNGaussian	**19.12**	**0.294**	**0.591**	**0.132**
6	3DGS	20.63	0.226	0.699	0.108
	3DGS†	21.09	0.229	0.699	0.103
	DNGaussian	**22.18**	**0.198**	**0.755**	**0.088**
9	3DGS	20.44	0.230	0.697	0.108
	3DGS†	**23.21**	**0.176**	0.785	**0.076**
	DNGaussian	23.17	0.180	**0.788**	0.077

Table 6. **Comparison with 3, 6, and 9 input views on LLFF dataset.** † denotes applied with the same hyperparameters and the neural color renderer as DNGaussian.

Solid Color Planes. The anisotropic shape of the Gaussian primitive makes it difficult to represent a solid color plane in a situation with sparse input views. First, the primitives are hard to constrain both by color and depth in the region of the plane, which may cause ray-like noises and hollows. Also, since they can freely move to other regions with similar colors, the densification operation can be activated more frequently and generate noises. This is hoped solved by additional geometry priors.

Specular Regions. Although our method can handle some specular regions by relying on depth supervision, especially from our Local Depth Normalization, the inconsistent appearances in these regions are still challenging for 3DGS. To completely solve this problem may still need more special designs.

Hollows and Cracks. The splatting technique of our Gaussian Splatting [8] backbone directly merges existing primitives to render the pixel-level color without interpolation. However, since not every pixel can be overlapped by the projected primitives, the empty space between two Gaussian primitives would cause hollows and cracks when the camera pose changes. For example, some hollows can be seen at Scan 40 in Figure 5. In this work, we try to solve this problem by paying more attention to high-frequency details and therefore encouraging the densifying of primitives to fill these empty areas. In the future, we believe this problem can be fundamentally solved by the improvement of the representation itself.

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. 3

[2] Wenjing Bian, Zirui Wang, Kejie Li, Jia-Wang Bian, and Victor Adrian Prisacariu. Nope-nerf: Optimising neural radiance field with no pose prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4160–4169, 2023. 5

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022. 2, 3

[4] 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. 5

[5] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022. 5

[6] 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. 5

[7] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014. 5

[8] 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. 2, 3, 6

[9] 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

[10] 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. 2, 3

[11] 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. 3, 5

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

[13] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022. 3

[14] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[15] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 5

[16] Jiuhn Song, Seonghoon Park, Honggyu An, Seokju Cho, Min-Seop Kwak, Sungjin Cho, and Seungryong Kim. Därf: Boosting radiance fields from sparse inputs with monocular depth adaptation, 2023. 3

[17] 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 *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 1414–1422. International Joint Conferences on Artificial Intelligence Organization, 2023. Main Track. 3, 5

[18] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *Proceedings of the IEEE/CVF In-*

ternational Conference on Computer Vision (ICCV), pages 9065–9076, 2023. 1, 2, 3, 5

[19] Zijin Wu, Xingyi Li, Juewen Peng, Hao Lu, Zhiguo Cao, and Weicai Zhong. Dof-nerf: Depth-of-field meets neural radiance fields. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1718–1729, 2022. 5

[20] Jamie Wynn and Daniyar Turmukhambetov. Diffusionerf: Regularizing neural radiance fields with denoising diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4180–4189, 2023. 3, 5

[21] 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, 2, 3, 5