NexusGS: Sparse View Synthesis with Epipolar Depth Priors in 3D Gaussian Splatting – Supplementary Materials –

Yulong Zheng[1], Zicheng Jiang[1], Shengfeng He[2], Yandu Sun[1], Junyu Dong[1], Huaidong Zhang[3], Yong Du[1*]
[1] Ocean University of China,
[2] Singapore Management University, [3] South China University of Technology

1. Derivations

1.1. Epipolar Depth Nexus

Once the coordinates of the perpendicular foot, denoted as $\bar{p}_j$, are determined in the target view's camera coordinate system (as shown in Eq. (7) in the main paper), the depth of point p_i, denoted as $D^{i\to j}(p_i, \bar{p}_j)$, can be computed using Eq. (8). Below, we provide a detailed derivation of Eq. (8) as presented in the main paper.

Since both p_i and $\bar{p}_j$ are expressed in their respective camera coordinate systems, we first convert them into normalized image coordinates to facilitate the depth calculation. These coordinates, situated in their respective 3D domains, are referenced with the source view camera O_i and the target view camera O_j as origins. The transformation is formulated as follows:

$$\tilde{p}_i = K_i^{-1}(x_i, y_i, 1)^\top, \quad \tilde{p}_j = K_j^{-1}(\bar{x}_j, \bar{y}_j, 1)^\top, \tag{1}$$

where K_i and K_j represent the intrinsic parameters of the source and target cameras, while $(x_i, y_i), (\bar{x}_j, \bar{y}_j)$ are the image coordinates of points p_i and $\bar{p}_j$, respectively.

Next, we refer to Fig. 1 to elucidate the geometric relationships employed in our method. Notably, the triangles $\triangle O_i A \bar{P}_j$ and $\triangle O_i B \tilde{p}_i$ are similar, allowing us to establish the following relationship:

$$\frac{|\overrightarrow{O_i\bar{P}_j}|}{|\overrightarrow{O_i\tilde{p}_i}|} = \frac{|O_iA|}{|O_iB|} = \frac{D^{i\to j}(p_i)}{1}. \tag{2}$$

Here, $|\overrightarrow{O_i\bar{P}_j}|$ and $|\overrightarrow{O_i\tilde{p}_i}|$ represent the magnitudes of the respective vectors. Similarly, $|O_iA|$ and $|O_iB|$ represent the respective distances from O_i to the planes containing $\bar{P}_j$ and $\tilde{p}_i$ along the optical axis.

To further analyze the geometry, we introduce an auxiliary point $\tilde{p}_i^{'}$ such that $\overrightarrow{O_j\tilde{p}_i^{'}}$ has the same length and direction as $\overrightarrow{O_i\tilde{p}_i}$. Projecting O_i perpendicularly onto the line $O_j\bar{P}_j$ yields point Q, and projecting $\tilde{p}_i^{'}$ onto the same line yields point U. The alternate interior angles $\angle O_i\bar{P}_jQ$ and $\angle\tilde{p}_i^{'}O_jU$ are equal, leading to similar triangles $\triangle O_i\bar{P}_jQ$ and $\triangle\tilde{p}_i^{'}O_jU$. Notice that triangles $\triangle O_i\tilde{p}_jO_j$ and $\triangle\tilde{p}_i^{'}O_j\tilde{p}_j$ share the same base, allowing us to further derive Eq. (2) as follows:

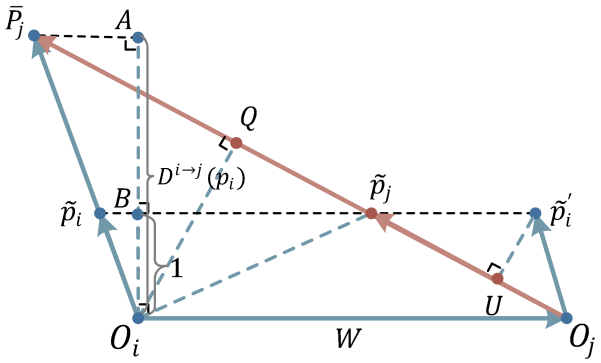

Figure 1. Illustration of the geometry relationships used in Epipolar Depth Nexus step.

$$D^{i\to j}(p_i) = \frac{|\overrightarrow{O_i\bar{P}_j}|}{|\overrightarrow{O_i\tilde{p}_i}|} = \frac{|O_iQ|}{|\tilde{p}_i^{'}U|} = \frac{\text{Area}(\triangle O_i\tilde{p}_jO_j)}{\text{Area}(\triangle\tilde{p}_i^{'}O_j\tilde{p}_j)}. \tag{3}$$

Using the formula for the area of a triangle, the above equation simplifies to:

$$D^{i\to j}(p_i) = \frac{|\overrightarrow{O_j\tilde{p}_j} \times \overrightarrow{O_jO_i}|}{|\overrightarrow{O_j\tilde{p}_i} \times \overrightarrow{O_j\tilde{p}_j}|}. \tag{4}$$

The 3D coordinates of vectors $\overrightarrow{O_j\tilde{p}_j}$, $\overrightarrow{O_jO_i}$, and $\overrightarrow{O_j\tilde{p}_i^{'}}$ can be determined using the extrinsic parameter transformation formulas:

$$\begin{aligned}\overrightarrow{O_j\tilde{p}_j} &= \overrightarrow{O_i\tilde{p}_j} - \overrightarrow{O_iO_j} \\ &= (R_jR_i^{-1})^{-1}\tilde{p}_j + W - W \\ &= (R_jR_i^{-1})^{-1}K_j^{-1}(\bar{x}_j, \bar{y}_j, 1)^\top,\end{aligned} \tag{5}$$

*Corresponding author (csyongdu@ouc.edu.cn).

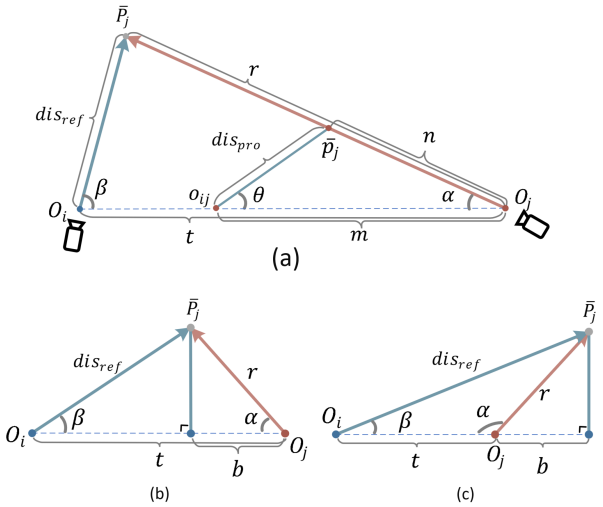

Figure 2. Illustration of the geometric definitions used.

$$\overrightarrow{O_jO_i} = -W = -(R_iR_j^{-1}T_j - T_i), \tag{6}$$

$$\overrightarrow{O_j\tilde{p}_i} = \overrightarrow{O_i\tilde{p}_i} = \tilde{p}_i, \tag{7}$$

where R_i, T_i and R_j, T_j are the extrinsic parameters of the source and target cameras. Finally, we plug Eqs. (5), (6) and (7) into Eq. (4) to obtain the analytical form of the depth of point p_i as follows:

$$D^{i\to j}(p_i) = \frac{|H \times -(R_iR_j^{-1}T_j - T_i)|}{|(K_i^{-1}(x_i, y_i, 1)^\top) \times H|}, \tag{8}$$
$$\text{where } H = (R_jR_i^{-1})^{-1}K_j^{-1}(\bar{x}_j, \bar{y}_j, 1)^\top.$$

1.2. Flow-Resilient Depth Blending

Here, we derive Eq. (9) from the main paper, used in our Flow-Resilient Depth Blending technique. Fig. 2 illustrates the geometric definitions of all relevant symbols. Our objective is to obtain the analytical form of the derivative of dis_{ref} with respect to dis_{pro}, denoted as $dis'_{ref}(dis_{pro}) = \frac{\mathrm{d}dis_{ref}}{\mathrm{d}dis_{pro}}$. This derivative is decomposed into two factors, $\frac{\mathrm{d}dis_{ref}}{\mathrm{d}\alpha}$ and $\frac{\mathrm{d}\alpha}{\mathrm{d}dis_{pro}}$, connected by the chain rule.

Derivation of $\frac{\mathrm{d}dis_{ref}}{\mathrm{d}\alpha}$. We begin by determining the expression of $\frac{\mathrm{d}dis_{ref}}{\mathrm{d}\alpha}$. Using the law of cosines and the law of sines, and referring to Fig. 2 (a) (similar to Fig. 3 (b) in the main paper), we derive the following equations:

$$\cos\beta = \frac{dis_{ref}^2 + t^2 - r^2}{2dis_{ref}t}, \tag{9}$$

$$r = \frac{dis_{ref}\sin\beta}{\sin\alpha}. \tag{10}$$

Substituting Eq. (10) into Eq. (9) and simplifying yields:

$$\begin{aligned}\frac{dis_{ref}^2\sin^2\beta}{\sin^2\alpha} &= dis_{ref}^2 + t^2 - 2dis_{ref}t\cos\beta \\ &= dis_{ref}^2\sin^2\beta + dis_{ref}^2\cos^2\beta \\ &\quad + t^2 - 2dis_{ref}t\cos\beta \\ &= dis_{ref}^2\sin^2\beta + (dis_{ref}\cos\beta - t)^2.\end{aligned} \tag{11}$$

Furthermore, we rewrite Eq. (11) in the following form:

$$\begin{aligned}(dis_{ref}\cos\beta - t)^2 &= dis_{ref}^2\sin^2\beta(\frac{1}{\sin^2\alpha} - 1) \\ &= \frac{dis_{ref}^2\sin^2\beta\cos^2\alpha}{\sin^2\alpha}.\end{aligned} \tag{12}$$

Next, we isolate $\frac{\cos^2\alpha}{\sin^2\alpha}$ from the right-hand side of Eq. (12) to get:

$$\frac{\cos^2\alpha}{\sin^2\alpha} = (\frac{\cos\beta}{\sin\beta} - \frac{t}{dis_{ref}\sin\beta})^2. \tag{13}$$

To take the square root of both sides of the equation, we need to ascertain the positive or negative nature of each side under the square root. Specifically, the formula after taking the square root can be determined based on geometric relationships. When $\frac{\cos\alpha}{\sin\alpha} > 0$, indicating that α is an acute angle, as illustrated in Fig. 2 (b), the following formulations hold:

$$\frac{\cos\beta}{\sin\beta} = \frac{t-b}{dis_{ref}\sin\beta}, \tag{14}$$

$$\frac{\cos\beta}{\sin\beta} - \frac{t}{dis_{ref}\sin\beta} = -\frac{b}{dis_{ref}\sin\beta} < 0, \text{ s.t. } \frac{\cos\alpha}{\sin\alpha} > 0. \tag{15}$$

Conversely, when $\frac{\cos\alpha}{\sin\alpha} < 0$, indicating that α is an obtuse angle, as illstrated in Fig. 2 (c), the following relationships hold:

$$\frac{\cos\beta}{\sin\beta} = \frac{t+b}{dis_{ref}\sin\beta}, \tag{16}$$

$$\frac{\cos\beta}{\sin\beta} - \frac{t}{dis_{ref}\sin\beta} = \frac{b}{dis_{ref}\sin\beta} > 0, \text{ s.t. } \frac{\cos\alpha}{\sin\alpha} < 0. \tag{17}$$

Combining both cases, we derive the following equation:

$$\frac{\cos\alpha}{\sin\alpha} = \frac{t}{dis_{ref}\sin\beta} - \frac{\cos\beta}{\sin\beta}, \tag{18}$$

and we can rewritten the above equation as follows:

$$dis_{ref} = \frac{t\sin\alpha}{\sin(\alpha+\beta)}. \tag{19}$$

Thus, we obtain the gradient expression of dis_{ref} with respect to α, which is formulated as follows:

$$\frac{\mathrm{d}dis_{ref}}{\mathrm{d}\alpha} = \frac{t\sin\beta}{\sin^2(\alpha+\beta)}. \tag{20}$$

Derivation of $\frac{\mathrm{d}\alpha}{\mathrm{d}dis_{pro}}$. Using a similar approach as for Eq. (19), we can derive the relationship between α and dis_{pro}:

$$dis_{pro} = \frac{m \sin \alpha}{\sin(\alpha + \theta)}. \tag{21}$$

Differentiating both sides with respect to dis_{pro}, we obtain $\frac{\mathrm{d}\alpha}{\mathrm{d}dis_{pro}}$, which are formulated as follows:

$$1 = \frac{m \cos \alpha \sin(\alpha + \theta) \frac{\mathrm{d}\alpha}{\mathrm{d}dis_{pro}} - m \sin \alpha \cos(\alpha + \theta) \frac{\mathrm{d}\alpha}{\mathrm{d}dis_{pro}}}{\sin^2(\alpha + \theta)}, \tag{22}$$

$$\frac{\mathrm{d}\alpha}{\mathrm{d}dis_{pro}} = \frac{\sin^2(\alpha + \theta)}{m \sin \theta}. \tag{23}$$

Final Expression for $dis'_{ref}(dis_{pro})$. Using Eqs. (20) and (23), we obtain the analytical form of $dis'_{ref}(dis_{pro})$, that is

$$\begin{aligned} dis'_{ref}(dis_{pro}) &= \frac{\mathrm{d}dis_{ref}}{\mathrm{d}dis_{pro}} \\ &= \frac{\mathrm{d}dis_{ref}}{\mathrm{d}\alpha} \frac{\mathrm{d}\alpha}{\mathrm{d}dis_{pro}} \\ &= \frac{t \sin \beta \sin^2(\alpha + \theta)}{m \sin \theta \sin^2(\alpha + \beta)}. \end{aligned} \tag{24}$$

Note that the side lengths in Fig. 2 (a) (*e.g.*, t, m) can be derived from the known point coordinates and camera poses. The angles α, β, and θ can then be calculated using the law of cosines.

2. Implementation Details

We implement our method using PyTorch 2.0.0 on an RTX 3090. We conduct 30k iterations for training on the LLFF dataset, 10k iterations for DTU and MipNeRF-360 Datasets, and 4k iterations for Blender dataset. During training, the learning rate for scale was set to 0.03 across the LLFF, MipNeRF-360, and Blender datasets, while the other parameters remain consistent with those used in 3DGS.

2.1. Dataset Split

LLFF&MipNeRF-360. Following previous methods [8, 13], we sample images at intervals of 8 from the LLFF [6] and MipNeRF-360 [1] datasets to create the test set, while the remaining images are used as the training set. For the sparse-view synthesis task, we perform uniform sampling within the training set to select the training views. Consistent with previous work, we downsample all images by a factor of 8.

DTU. Following previous methods [12], we select 15 scenes from the DTU [3] dataset out of 124, specifically scene IDs 8, 21, 30, 31, 34, 38, 40, 41, 45, 55, 63, 82, 103, 110, and 114. For each scene, views 25, 22, and 28 are used as the 3-view training set, while views 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 are designated as the test set. All images are downsampled by a factor of 4.

Blender. In the Blender [7] dataset, following previous methods [2], we select views 26, 86, 2, 55, 75, 93, 16, 73, and 8 for training. For evaluation, we uniformly sample 25 images from the test set. All images are downsampled by a factor of 2.

	Setting	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS		17.65	0.816	0.146
Ours	**A**. Average	18.92	0.836	0.128
	B. Nearest	19.57	0.860	0.108
	C. Weighted	19.05	0.850	0.119
	D. FRDB	19.76	0.864	0.109
	E. Average + FFDP	18.72	0.830	0.135
	F. Nearest + FFDP	19.74	0.860	0.108
	G. Weighted + FFDP	19.05	0.853	0.116
	H. FRDB + FFDP	**20.21**	**0.869**	**0.102**

Table 1. Ablation study on DTU with 3 input views.

2.2. Additional Training Details

During training, we maintain most parameters consistent with those used in 3DGS. Here, we provide additional details beyond those in the main paper. Specifically, the threshold ϵ_d used in Flow-Filtered Depth Pruning is set to 1.0 for LLFF and DTU, 0.1 for MipNeRF-360, and 0.01 for Blender. The hyperparameter λ_c in the objective function was fixed at 0.2. Camera poses are estimated using COLMAP [9], following the methodology of existing sparse-view synthesis studies [5, 11, 13]. We utilize FlowFormer++ [10] as the optical flow estimator.

3. Extended Ablation Analysis

3.1. Quantitative Ablation Study on DTU Dataset

To complement the ablation results presented in the main paper for the LLFF real-world benchmark, we conduct additional experiments on the object-centric DTU dataset, with the results summarized in Tab. 1. Our Flow-Resilient Depth Blending (FRDB) method significantly improves sparse view synthesis performance over variants with alternative blending strategies. Furthermore, when combined with Flow-Filtered Depth Pruning (FFDP), our approach generally outperforms most configurations. This improvement is driven by cleaner, more accurate, and more comprehensive point clouds generated using epipolar depth priors, which lead to enhanced geometric precision and higher-quality details. However, under the Average settings (**A.** vs. **E.**), where depth estimates are extremely inaccurate, excessive splitting and replication of erroneous points during refinement lead to performance degradation. By contrast, in scenarios with

Method	Point Number	PSNR ↑	SSIM ↑	LPIPS ↓
DNGaussian	43K	18.86	0.598	0.297
DNGaussian*	77K	19.96	0.684	0.232
CoR-GS	80K	20.29	0.705	0.201
FSGS	299K	20.34	0.695	0.207
Threshold	Point Number	PSNR ↑	SSIM ↑	LPIPS ↓
0.01	110K	20.21	0.678	0.218
0.1	191K	20.70	0.716	0.193
1.0	427K	21.07	0.738	0.177
2.0	456K	20.97	0.733	0.179
3.0	465K	20.89	0.733	0.180
4.0	469K	20.86	0.733	0.180

Table 2. Influence of distance threshold choices and point cloud comparison with state-of-the-art 3DGS-based competitors on the LLFF dataset with 3 input views. $*$ denotes fused stereo initial point clouds.

relatively accurate depth priors, FFDP effectively enhances reconstruction quality by refining point clouds and preserving finer details. Theses consistent performance gains across different datasets demonstrate the robustness and effectiveness of our proposed method.

3.2. Influence of Distance Threshold

We further investigate the impact of varying threshold ϵ_d in FFDP, as detailed in Tab. 2. The table presents both quantitative metrics and the number of points in the Gaussian representation point cloud. Unlike Tab. 1 in the main paper, where competing methods' results are taken from their original publications, we obtain these metrics by training their official implementations with default settings, as the point counts were not reported.

Examining the lower half of Tab. 2, we observe that when ϵ_d is set to low values (*e.g.*, 0.1 or 0.01), the performance of NexusGS declines due to the insufficient number of points in the initial point clouds. This reduction in point density leads to excessive splitting in 3DGS, which introduces randomness in point placement. Moreover, the lack of supervision from sparse views prevents the generation of a dense, comprehensive point cloud, ultimately degrading performance.

The optimal results are achieved with a threshold of 1.0. As the threshold increases, more comprehensive and still relatively accurate initial points are obtained, significantly improving the quality of the generated point cloud and the final reconstruction. However, further increases may introduce additional inaccurate points, negatively affecting performance. Despite this, we find that as the threshold grows beyond 1.0, the impact on PSNR becomes more noticeable, while SSIM and LPIPS—metrics that align better with human visual perception—remain less affected. This suggests that our method exhibits tolerance for erroneous initial points, maintaining stable performance while revealing rich high-frequency details in the output.

Model Type	chairs	kitti	sintel	things_288960	things
PSNR ↑	21.049	21.060	21.045	21.068	21.075
SSIM ↑	0.738	0.738	0.738	0.739	0.738
LPIPS ↓	0.178	0.178	0.178	0.177	0.177

Table 3. The influence of different pre-trained flow estimation models on the LLFF dataset with 3 input views.

We also quantitatively analyze the quality of the point clouds generated by state-of-the-art methods such as DNGaussian, FSGS, and CoR-GS, as shown in the upper half of Tab. 2. Regardless of whether random or fused stereo initial point clouds (indicated by an asterisk) are used, DNGaussian, as a lightweight design-focused method, consistently generates fewer points and provides less comprehensive coverage than our approach, even at $\epsilon_d = 0.01$, resulting in inferior performance. Although FSGS generates more points, its limited point addition strategy results in lower accuracy and coverage. Notably, even with a reduced number of points (*e.g.*, $\epsilon_d = 0.1$), our method still outperforms FSGS. As for CoR-GS, while it generates relatively comprehensive coverage, it lacks the ability to produce a dense point cloud. This limitation is reflected in the point count, ultimately restricting the reconstruction quality, especially in high-frequency details. In contrast, NexusGS, with epipolar depth priors, generates a more accurate, dense, and comprehensive point cloud, leading to superior reconstruction performance.

3.3. Robustness Across Various Flow Estimators

Inspired by existing approaches [5, 13], which utilize monocular depth estimators to provide depth priors, we hypothesize that similar variability in performance might occur when using different optical flow estimators with varying network parameters. To explore this possibility, we conduct experiments on the LLFF dataset, comparing the performance of our method using optical flow estimators trained on five different datasets. Specifically, we evaluate the *chairs*, *kitti*, *sintel*, *things_288960*, and *things* models of FlowFormer++. The quantitative results are summarized in Tab. 3. As illustrated, despite utilizing different pretrained flow estimation models, our method consistently shows minimal variations in PSNR, with SSIM and LPIPS scores remaining nearly identical across the different models. These results highlight the robustness of our approach, demonstrating its effectiveness regardless of the specific optical flow estimator employed.

3.4. Robustness on Varying View Counts

To validate the robustness of our method under varying numbers of training views, we conduct experiments on the LLFF dataset, with results presented in Tab. 4. Our method consistently outperforms all competitors when using 2, 3, and 4 views. Notably, with only 2 training views, COLMAP fails to generate a fused stereo point cloud, leading to poor perfor-

Method	2 Views			3 Views			4 Views		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
RegNeRF	16.16	0.396	0.455	19.08	0.587	0.336	20.95	0.689	0.272
FreeNeRF	17.12	0.490	0.364	19.63	0.612	0.308	21.63	0.709	0.253
SparseNeRF	17.51	0.450	0.423	19.86	0.624	0.328	21.09	0.681	0.295
3DGS	12.21	0.282	0.501	18.54	0.588	0.272	16.98	0.563	0.313
DNGaussian	15.92	0.454	0.391	19.12	0.591	0.294	20.58	0.688	0.253
FSGS	16.09	0.438	0.384	20.43	0.682	0.248	21.93	0.760	0.167
CoR-GS	14.63	0.417	0.423	20.45	0.712	0.196	21.62	0.761	0.163
Ours	19.28	0.659	0.220	21.07	0.738	0.177	22.12	0.774	0.158

Table 4. Quantitative evaluation of the impact of training views on the LLFF dataset.

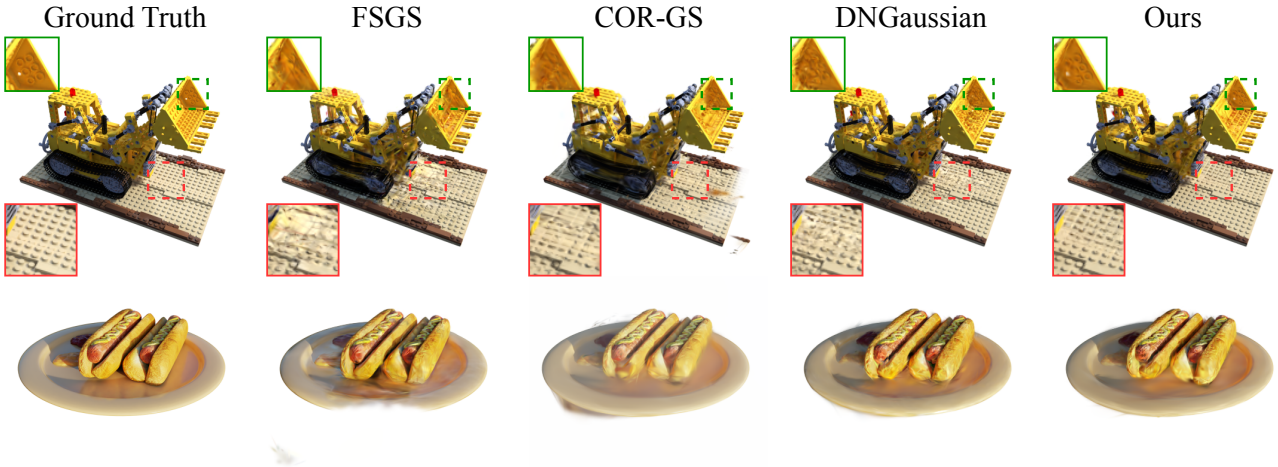

Figure 3. Visual results on the Blender dataset with 8 input views.

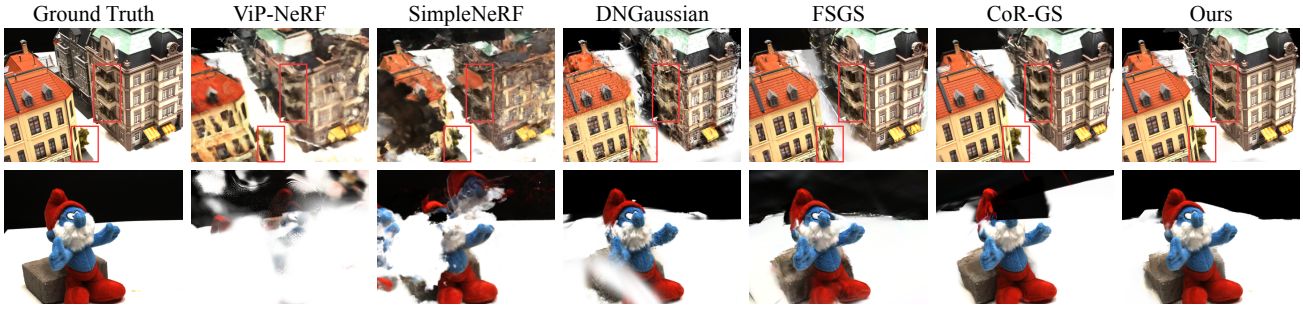

Figure 4. Additional visual comparisons on the DTU dataset with 3 input views.

mance by DNGaussian, FSGS, and CoR-GS, which perform even worse than NeRF-based methods. In contrast, our method does not suffer from this limitation. By leveraging the point cloud generated through our approach, we achieve superior results with just 2 views, effectively overcoming the constraints of previous 3DGS methods and surpassing NeRF-based methods across all evaluated metrics.

4. Additional Visual Results

Additional visual results are provided in the supplementary materials. Specifically, Fig. 3 showcases the results of our method on the Blender dataset. As shown, our method achieves a more complete and detailed reconstruction compared to previous approaches. This improvement is attributed to the accurate geometry provided by the precise point cloud,

Figure 5. Per-scene visual results on the LLFF dataset with 3 input views.

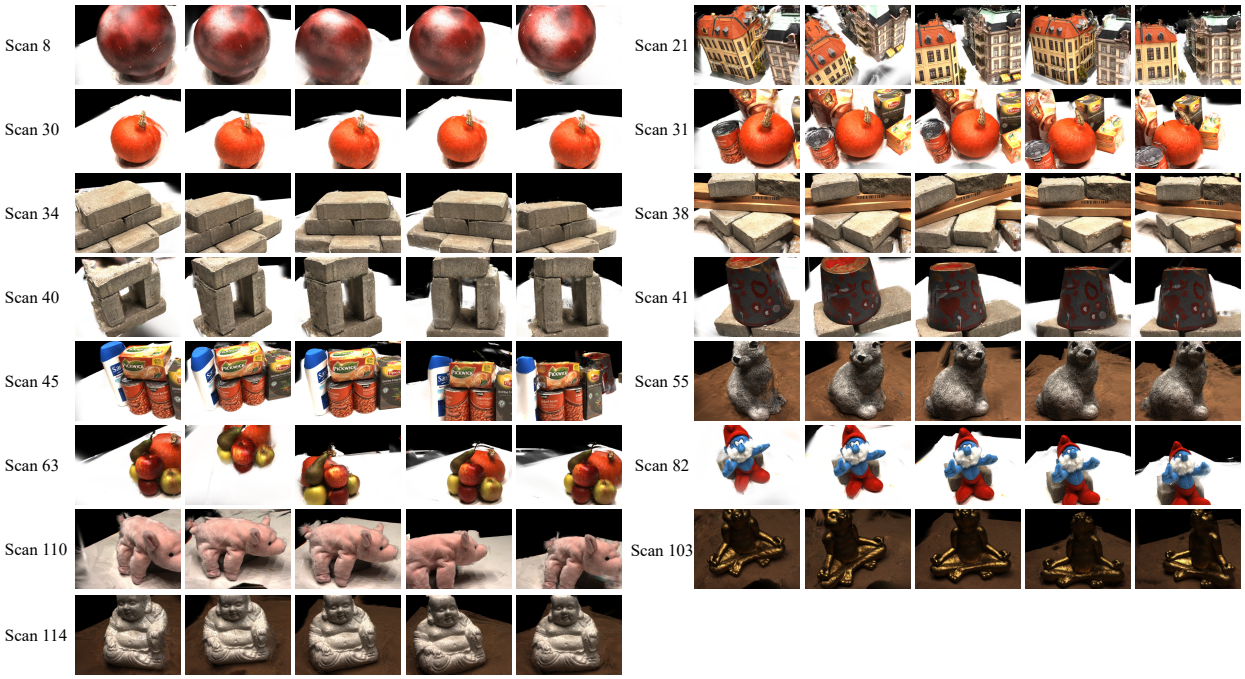

Figure 6. Per-scene visual results on the DTU dataset with 3 input views.

which enables the faithful reconstruction of finer textures.

We also present additional visual results on the DTU dataset in Fig. 4. It is evident that the results from NeRF-based methods, constrained by depth priors, are adversely affected by floaters, leading to degraded reconstruction quality. This suggests that their depth priors are not well-suited for object-centered datasets. In contrast, subsequent 3DGS-based methods that utilize depth priors mitigate the presence of floaters. However, due to the lack of dense and accurate point clouds, their final reconstructions appear visually smoother, lacking finer detail.

Furthermore, Figs. 5 and 6 present the results for each test scene on the LLFF and DTU datasets, respectively. Fig. 7 compares the depth maps of our method and the competitors.

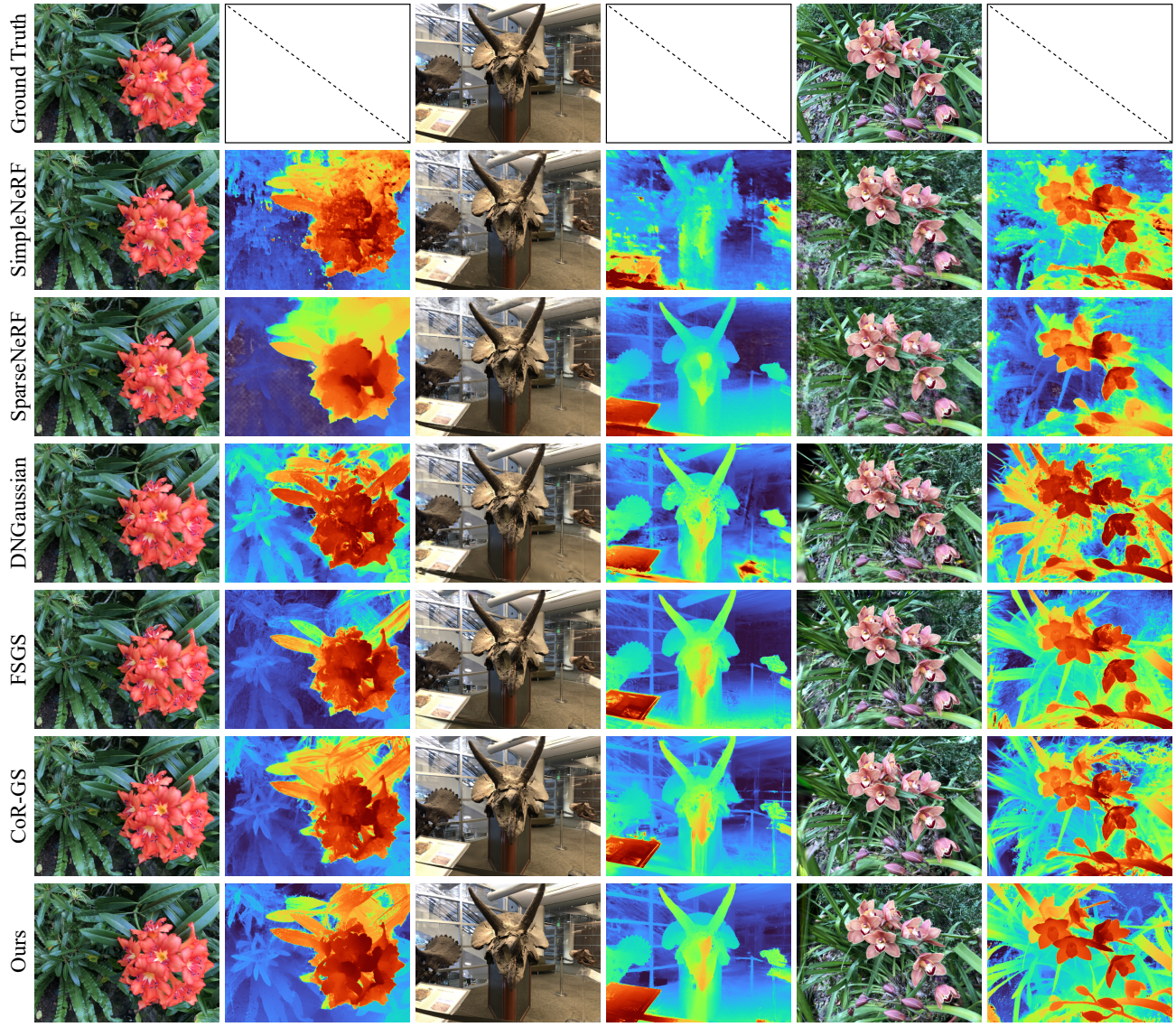

Figure 7. Visual comparisons of depth maps on the LLFF dataset with 3 input views.

5. Discussion, Limitations, and Future Work

Despite NexusGS's remarkable performance and generalizability, it, like most sparse-view synthesis methods, relies on known camera poses to enforce epipolar constraints. To evaluate the robustness of our method, we conduct a sensitivity analysis of camera calibration errors, using the SSIM score as the primary metric, as summarized in Tab. 5. Perturbations are introduced to the camera pose along all axes, resulting in performance degradation across all competitors, all of which depend on accurate pose information. Severe overfitting is observed when perturbations reach 0.1. In contrast, our method demonstrates superior robustness, consistently outperforming others despite pose errors, thanks to our effective error-handling strategies, FRDB and FFDP.

Perturbation	0	0.02	0.04	0.06	0.08	0.1
DNGaussian	0.591	0.586	0.565	0.556	0.540	0.476
FSGS	0.682	0.688	0.664	0.633	0.606	0.579
CoR-GS	0.712	0.693	0.664	0.639	0.609	0.586
Ours	**0.738**	**0.724**	**0.690**	**0.653**	**0.617**	**0.588**

Table 5. Sensitivity analysis of calibration errors on the LLFF dataset with 3 input views using SSIM.

Although recent pose-free methods, such as COGS [4], bypass the need for camera poses, the trade-off between flexibility and rendering accuracy leaves room for improvement, offering a promising avenue for future work.

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*, pages 5470–5479, 2022. 3

[2] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *ICCV*, pages 5885–5894, 2021. 3

[3] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, pages 406–413, 2014. 3

[4] Kaiwen Jiang, Yang Fu, Yash Belhe, Xiaolong Wang, Hao Su, Ravi Ramamoorthi, et al. A construct-optimize approach to sparse view synthesis without camera pose. In *ACM SIGGRAPH*, 2024. 7

[5] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *CVPR*, 2024. 3, 4

[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 TOG*, 38(4):1–14, 2019. 3

[7] 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. 3

[8] 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*, pages 5480–5490, 2022. 3

[9] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016. 3

[10] Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. In *CVPR*, pages 1599–1610, 2023. 3

[11] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *ICCV*, pages 9065–9076, 2023. 3

[12] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *CVPR*, pages 8254–8263, 2023. 3

[13] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *ECCV*, 2024. 3, 4