MVSGaussian: Fast Generalizable Gaussian Splatting Reconstruction from Multi-View Stereo

Tianqi Liu[1], Guangcong Wang[2,3], Shoukang Hu[2], Liao Shen[1], Xinyi Ye[1], Yuhang Zang[4], Zhiguo Cao[1*], Wei Li[2†], and Ziwei Liu[2]

[1] School of AIA, Huazhong University of Science and Technology
[2] S-Lab, Nanyang Technological University
[3] Great Bay University
[4] Shanghai AI Laboratory
{tq_liu,zgcao}@hust.edu.cn
https://mvsgaussian.github.io/

1 Implementation and Network Details

Implementation Details. Following ENeRF [8], we partition the DTU [1] dataset into 88 training scenes and 16 test scenes. We train the generalizable model on four RTX 3090 GPUs using the Adam [6] optimizer, with an initial learning rate set to $5e-4$. The learning rate is halved every 50k iterations. During the training process, we select 2, 3, and 4 source views as inputs with respective probabilities of 0.1, 0.8, and 0.1. For evaluation, we follow the criteria established in prior works such as ENeRF [8] and MVSNeRF [3]. Specifically, for the DTU test set, segmentation masks are employed to evaluate performance, defined based on the availability of ground-truth depth at each pixel. For Real Forward-facing dataset [10], where the marginal region of images is typically invisible to input images, we evaluate the 80% area in the center of the images. This evaluation methodology is also applied to the Tanks and Temples dataset [7]. The image resolutions of the DTU, the Real Forward-facing, the NeRF Synthetic [11], and the Tanks and Temples datasets are 512×640, 640×960, 800×800, and 640×960 respectively. As discussed in Sec. 4.3 of the main text, we employ a consistency check to filter out noisy points for high-quality initialization. Specifically, we apply a dynamic consistency checking algorithm [9, 15], the details of which are provided in Algorithm 1. The predefined thresholds $\{\theta_p(n)\}_{n=1}^{N_\theta}$ are set to $\{\frac{n}{8}\}_{n=1}^{N_\theta}$, and $\{\theta_d(n)\}_{n=1}^{N_\theta}$ are set to $\{\frac{n}{10}\}_{n=1}^{N_\theta}$. For 3D-GS [5], following [20], we use COLMAP [12] to reconstruct the point cloud from the working set (training views) as initialization. Specifically, we employ COLMAP's automatic reconstruction to achieve the reconstruction of sparse point clouds. Some examples are shown in Fig. 1. As mentioned in Sec.5.1 of the main text, our optimization strategy and hyperparameters settings remain consistent with the vanilla 3D-GS, except for the number of iterations. The iterations of our method on Real

[*] Corresponding author
[†] Project lead

Algorithm 1: Dynamic Consistency Checking

```
Input: Camera parameters, Depth maps $D_0$ and $\{D_i\}_{i=1}^{N}$, predefined
       thresholds $\{\theta_p(n)\}_{n=1}^{N_\theta}$ and $\{\theta_d(n)\}_{n=1}^{N_\theta}$
Output: $Mask$
Initialization: $Mask \leftarrow 0$
for $i$ in $(1, ..., N)$ do
    $Err_p^i \leftarrow zeros(H, W), Err_d^i \leftarrow zeros(H, W)$
    for $p$ in $(0, 0)$ to $(H-1, W-1)$ do
        $\xi_p^i \leftarrow \|p - p'\|_2$,   ▷ calculate the reprojetcion error between $D_0$ and $D_i$
        $\xi_d^i \leftarrow \|D_0(p) - d'\|_1 / D_0(p)$
        $Err_p^i(p) \leftarrow \xi_p^i$
        $Err_d^i(p) \leftarrow \xi_d^i$
    end
    for $n$ in $(1, ..., N_\theta)$ do
        $Mask_n^i \leftarrow (Err_p^i < \theta_p(n)) \& (Err_d^i < \theta_d(n))$
    end
end
for $n$ in $(1, ..., N_\theta)$ do
    $Mask_n \leftarrow 0$
    for $i$ in $(1, ..., N)$ do
        $Mask_n \leftarrow Mask_n + Mask_n^i$
    end
    $Mask_n \leftarrow (Mask_n > n)$
    $Mask \leftarrow Mask \cup Mask_n$
end
```

Forward-facing, NeRF Synthetic and Tanks and Temples datasets are 2.5k, 5k and 5k, respectively.

Network Details. As mentioned in Sec. 4.2 of the main text, we apply a pooling network ρ to aggregate multi-view features to obtain the aggregated features via $f_v = \rho(\{f_i\}_{i=1}^{N})$. The implementation details are consistent with [8]: initially, the mean μ and variance v of $\{f_i\}_{i=1}^{N}$ are computed. Subsequently, μ and v are concatenated with each f_i and an MLP is applied to generate weights. The f_v is then blended using a soft-argmax operator, combining the obtained weights and multi-view features ($\{f_i\}_{i=1}^{N}$).

2 Additional Ablation Experiments

Numbers of Views. Existing generalizable Gaussian methods, such as PixelSplat [2] and GPS-Gaussian [19], focus on image pairs as input, while Splatter Image [13] prioritize single-view reconstruction. Our method is view-agnostic, capable of supporting varying numbers of views as input. We report the performance with varying numbers of input views in Table 1. As the number of views increases, the model can leverage more scene information, leading to improved

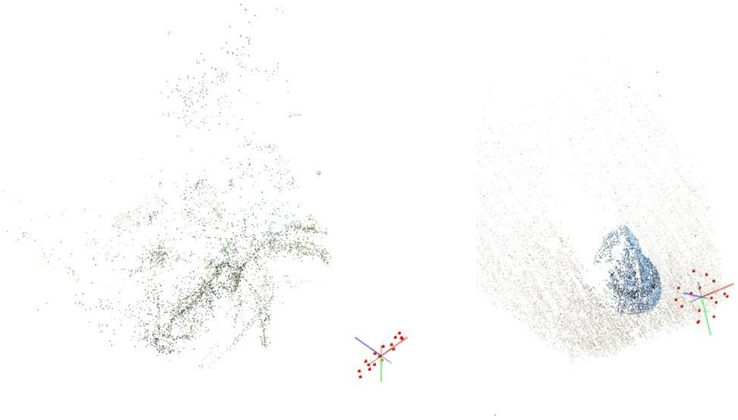

Fig. 1: Visualization of camera calibration and point cloud reconstruction by COLMAP.

Table 1: The performance of our method with varying numbers of input views on the DTU, Real Forward-facing, and NeRF Synthetic datasets. "Mem" and "FPS" are measured under the image resolution of 512 × 640.

Views	DTU [1]			Real Forward-facing [10]			NeRF Synthetic [11]			Mem(GB)↓	FPS↑
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓		
2	25.78	0.947	0.095	23.11	0.834	0.175	25.06	0.937	0.079	0.866	24.5
3	28.21	0.963	0.076	24.07	0.857	0.164	26.46	0.948	0.071	0.876	21.5
4	28.43	0.965	0.075	24.46	0.870	0.164	26.50	0.949	0.071	1.106	19.1

performance. Meanwhile, increasing the number of views only introduces a slight increase in computational cost and memory consumption.

Numbers of Sampled Points. In the main text, we apply a pixel-align Gaussian representation, where each pixel is unprojected into 3D space based on the estimated depth, corresponding to a 3D Gaussian. An alternative approach is to sample M depths centered at the estimated depth map, resulting in each pixel being unprojected into M Gaussians. As shown in Table 2, increasing the number of 3D sampled points improves performance but raises computational costs. To strike a balance between cost and performance, we set $M = 1$.

Density for Volume Rendering. Since only one point per ray is sampled, our model predicts single radiance r and density σ. In this case, the pixel's color c obtained through volume rendering is given by $c = (1 - \exp(-\sigma))r$. This resembles pixel-aligned splatting, where one Gaussian contributes one pixel, sharing the alpha-based rendering principles but offering a simpler implementation. Therefore, predicting density is necessary as it indicates the point's opacity, as validated by ablation results in Table 3.

Initialization Comparison. Our generalization model can provide a denser point cloud for 3D-GS [5] than Structure-from-Motion (SfM) as initialization.

Table 2: The performance of different numbers of sampled points on the DTU, Real Forward-facing, NeRF Synthetic, and Tanks and Temples datasets. Here, “Samples" represents the number of 3D points sampled along the ray for each pixel.

Samples	DTU [1]			Real Forward-facing [10]			NeRF Synthetic [11]			Tanks and Temples [7]			Mem(GB)↓	FPS↑
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓		
1	28.21	0.963	0.076	24.07	0.857	0.164	26.46	0.948	0.071	23.28	0.877	0.139	0.876	21.5
2	28.26	0.963	0.075	24.20	0.861	0.163	26.64	0.949	0.070	23.20	0.879	0.151	1.508	19.0

Table 3: The ablation study on the density prediction.

Settings	DTU			Real Forward-facing			NeRF Synthetic		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
w/o density	28.03	0.963	0.076	23.96	0.854	0.165	26.22	0.947	0.071
w density	28.21	0.963	0.076	24.07	0.857	0.164	26.46	0.948	0.071

Considering that the MVS method can also obtain a denser point cloud, we conduct the comparison in Table 4. MVS methods are typically trained on DTU [1] and BlendedMVS [17], then tested on Tanks and Temples dataset [7]. Thus, we select the latest ET-MVSNet [9] and compare it on Tanks and Temples dataset. While ET-MVSNet [9] surpasses SfM, it’s still limited. Because it focuses solely on accurate depth, while our method generates point clouds tailored for view synthesis. Depth and view quality aren’t directly proportional, as mentioned in previous works such as ENeRF [8].

Depth Analysis. Benefiting from the explicit geometry reasoning of MVS, our method can produce reasonable depth maps, as illustrated in Fig. 2. The quantitative results are shown in Table 5. Compared with previous generalizable NeRF methods, our method can achieve the most accurate depth estimation.

Point Cloud Analysis. As discussed in Sec. 5.4 of the main text, different point cloud aggregation strategies can provide varying-quality initialization for subsequent per-scene optimization. Here, we report the initial and final numbers of point clouds in Table 6 and provide the visual comparison in Fig. 4. The direct concatenation approach leads to excessively large initialization point clouds, which slow down optimization and rendering speeds. The down-sampling approach can reduce the total number of points and mitigate noisy points, but it also leads to a reduction in effective points. Our applied consistency check strategy can filter out noisy points while retaining effective ones.

Inference Speed Analysis. As shown in Table 1 of the main text, the inference speed (FPS) of our generalizable model is 21.5. Here, we present the inference time breakdown result in Table 7. The primary time overhead comes from the neural network, while the subsequent rendering process incurs minimal time overhead. Therefore, we discard the neural network component during the per-scene optimization stage, resulting in a significant increase in speed.

Table 4: Initialization Comparison.

Initialization	PSNR	SSIM	LPIPS	Time$_{ft}$	FPS
ET-MVSNet [9]	22.66	0.861	0.204	90s	300+
Ours	24.58	0.903	0.137	90s	300+

Table 5: Quantitative comparison of depth reconstruction on the DTU test set. MVSNet is trained using depth supervision, while other methods are trained with only RGB image supervision. "Abs err" refers to the average absolute error, and "Acc(X)" denotes the percentage of pixels with an error less than X mm.

Method	Reference view			Novel view		
	Abs err ↓	Acc(2)↑	Acc(10)↑	Abs err ↓	Acc(2)↑	Acc(10)↑
MVSNet [16]	3.60	0.603	0.955	-	-	-
PixelNeRF [18]	49	0.037	0.176	47.8	0.039	0.187
IBRNet [14]	338	0.000	0.913	324	0.000	0.866
MVSNeRF [3]	4.60	0.746	0.913	7.00	0.717	0.866
ENeRF-MVS [8]	3.80	0.823	0.937	4.80	0.778	0.915
ENeRF-NeRF [8]	3.80	0.837	0.939	4.60	0.792	0.917
Ours	**3.11**	**0.866**	**0.956**	**3.66**	**0.838**	**0.945**

3 More Qualitative Results

Qualitative Results under the Generalization Setting. As shown in Fig. 3, we present qualitative comparisons of the generalization results obtained by different methods. Our method is capable of producing higher-fidelity views, particularly in some challenging areas. For instance, in geometrically complex scenes, around objects' edges, and in reflective areas, our method can reconstruct more details while exhibiting fewer artifacts.

Qualitative Results under the Per-scene Optimization Setting. As shown in Fig. 5, we present the visual comparison after fine-tuning. Benefiting from the strong initialization provided by our generalizable model, excellent performance can be achieved with just a short fine-tuning period. The views synthesized by our method preserve more scene details and exhibit fewer artifacts.

Table 6: Comparison of point cloud quantities under different aggregation strategies on the Real Forward-facing dataset. For downsampling, we employ widely-used voxel downsampling, with a voxel size set to 2. The iteration number for all strategies is set to 2.5k.

Strategy	initial points(k)	final points(k)
direct concatenation	2458	2176
downsampling	836	839
consistency check	860	913

Table 7: Time overhead for each module (in milliseconds).

Modules	coarse stage	fine stage
Feature extractor	1.3	
Depth estimation	8.1	7.9
Gaussian representation	-	24.0
Gaussian rendering	-	4.4

4 Per-scene Breakdown

As shown in Tables 8, 11, 10, and 9, we present the per-scene breakdown results of DTU [1], NeRF Synthetic [11], Real Forward-facing [10], and Tanks and Temples [7] datasets. These results align with the averaged results presented in the main text.

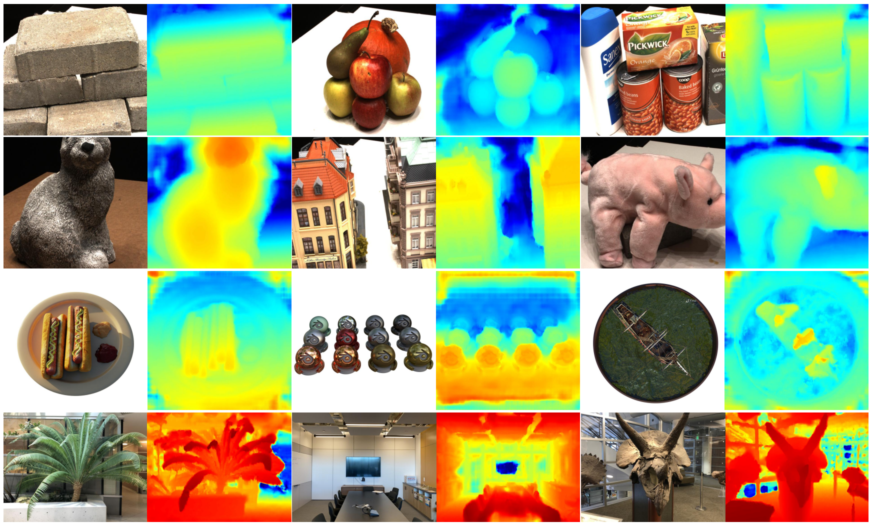

Fig. 2: Depth maps visualization. We visualize the depth maps predicted by our method on different datasets [1, 10, 11].

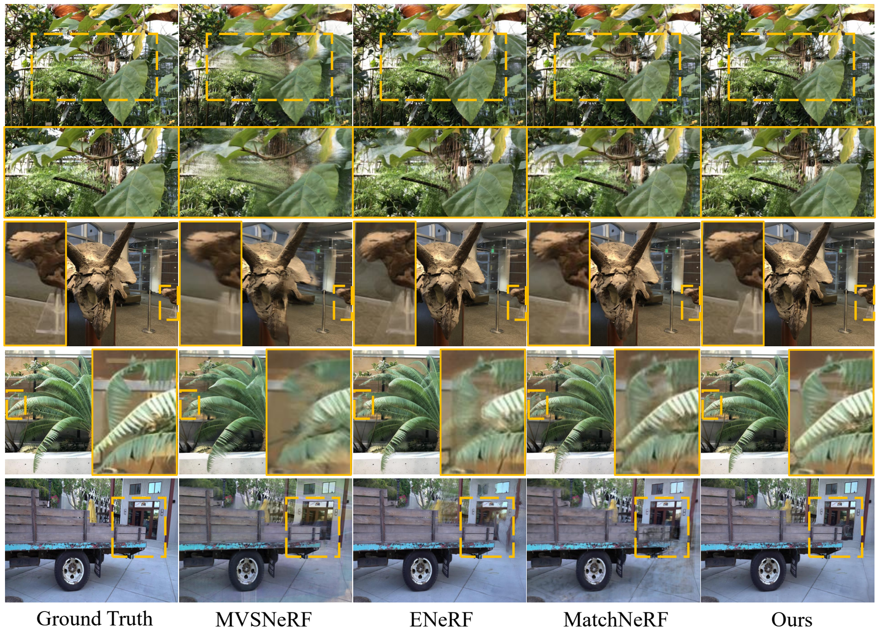

Fig. 3: Qualitative comparison of rendering quality with state-of-the-art methods [3, 4, 8] under generalization and three views settings.

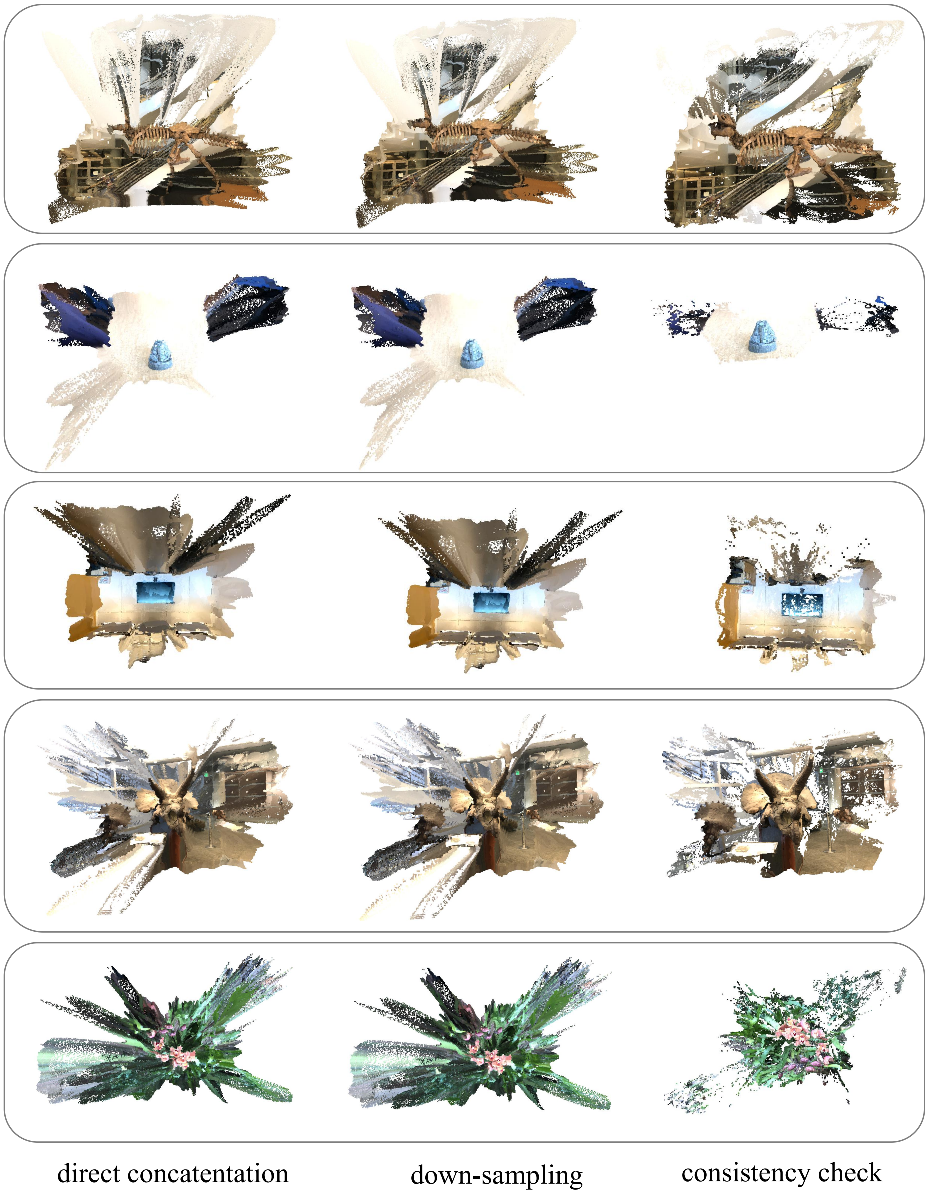

Fig. 4: Point cloud visualization under different aggregation strategies.

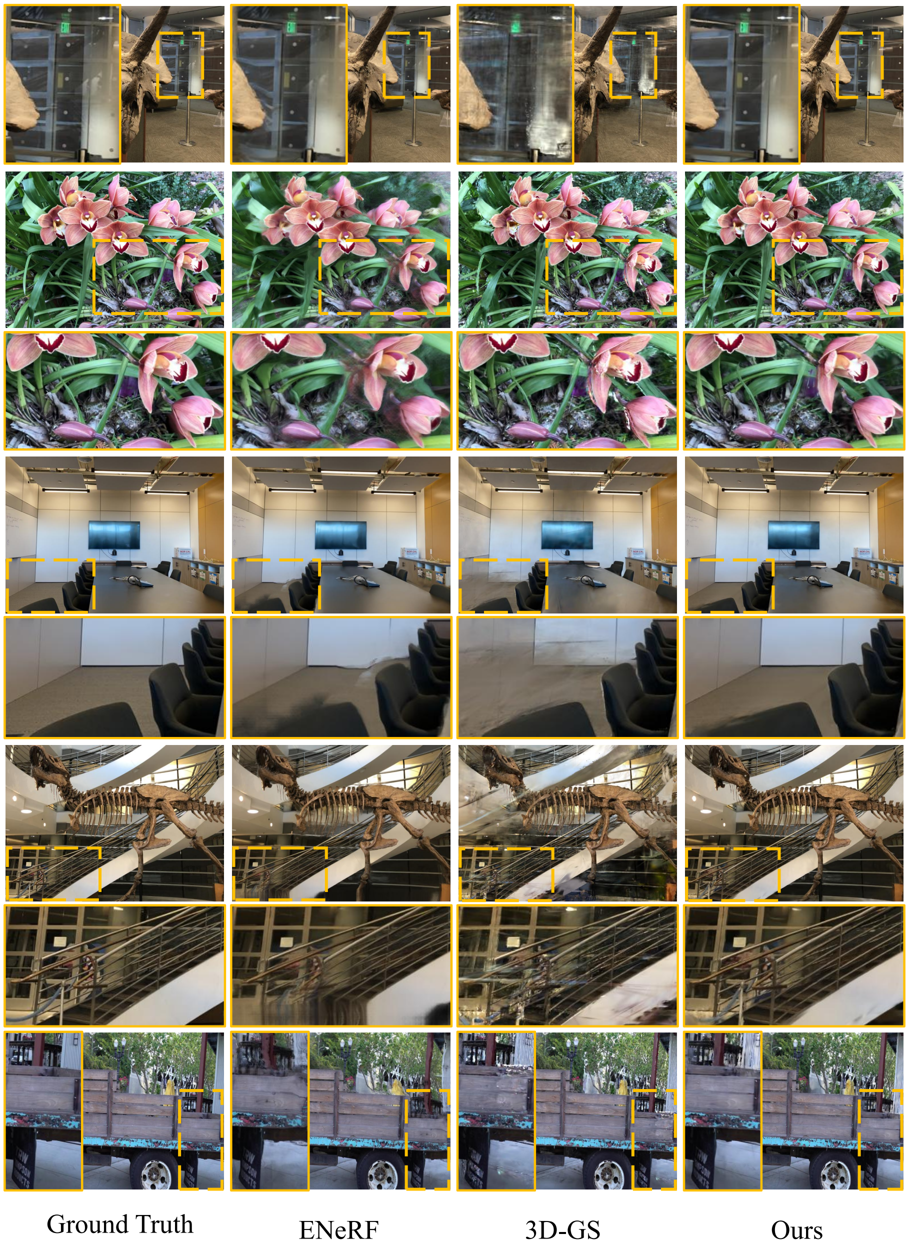

Fig. 5: Qualitative comparison of rendering quality with state-of-the-art methods [5, 8] after per-scene optimization.

Table 8: Quantitative per-scene breakdown results on the DTU test set. PixelSplat* and Ours* represent the results obtained with a 2-view input, while the others are the results obtained with a 3-view input.

Scan	#1	#8	#21	#103	#114	#30	#31	#34	#38	#40	#41	#45	#55	#63	#82	#110
Metric	PSNR ↑															
PixelNeRF [18]	21.64	23.70	16.04	16.76	18.40	-	-	-	-	-	-	-	-	-	-	-
IBRNet [14]	25.97	27.45	20.94	27.91	27.91	-	-	-	-	-	-	-	-	-	-	-
MVSNeRF [3]	26.96	27.43	21.55	29.25	27.99	-	-	-	-	-	-	-	-	-	-	-
ENeRF [8]	28.85	29.05	22.53	30.51	28.86	29.20	25.13	26.77	28.61	25.67	29.51	**24.83**	30.26	27.22	26.83	27.97
MatchNeRF [4]	27.69	27.76	22.75	29.35	28.16	29.16	24.26	25.66	27.52	25.16	28.27	23.94	26.64	**29.40**	27.65	27.15
Ours	**29.67**	**29.65**	**23.24**	**30.60**	**29.26**	**30.10**	**25.94**	**26.82**	**29.27**	**26.13**	**30.33**	24.55	**31.40**	28.46	**27.82**	**28.15**
PixelSplat* [2]	14.65	14.72	10.69	16.88	15.31	10.93	13.28	14.70	14.81	13.26	16.09	12.62	15.76	12.18	12.11	16.18
Ours*	27.22	26.88	20.49	28.25	27.89	27.55	22.96	25.32	27.13	22.89	27.71	21.78	28.85	27.01	24.64	25.92
Metric	SSIM ↑															
PixelNeRF [18]	0.827	0.829	0.691	0.836	0.763	-	-	-	-	-	-	-	-	-	-	-
IBRNet [14]	0.918	0.903	0.873	0.950	0.943	-	-	-	-	-	-	-	-	-	-	-
MVSNeRF [3]	0.937	0.922	0.890	0.962	0.949	-	-	-	-	-	-	-	-	-	-	-
ENeRF [8]	0.958	0.955	0.916	0.968	0.961	0.981	0.937	0.934	0.946	0.947	0.960	0.948	0.973	0.978	0.971	0.974
MatchNeRF [4]	0.936	0.918	0.901	0.961	0.948	0.974	0.921	0.874	0.902	0.903	0.936	0.934	0.929	0.976	0.966	0.962
Ours	**0.966**	**0.961**	**0.930**	**0.970**	**0.963**	**0.983**	**0.946**	**0.947**	**0.954**	**0.957**	**0.967**	**0.954**	**0.979**	**0.980**	**0.974**	**0.976**
PixelSplat* [2]	0.690	0.706	0.492	0.778	0.651	0.782	0.624	0.534	0.513	0.571	0.714	0.541	0.624	0.807	0.769	0.802
Ours*	0.950	0.948	0.895	0.963	0.954	0.977	0.919	0.925	0.933	0.928	0.951	0.933	0.967	0.974	0.965	0.966
Metric	LPIPS ↓															
PixelNeRF [18]	0.373	0.384	0.407	0.376	0.372	-	-	-	-	-	-	-	-	-	-	-
IBRNet [14]	0.190	0.252	0.179	0.195	0.136	-	-	-	-	-	-	-	-	-	-	-
MVSNeRF [3]	0.155	0.220	0.166	0.165	0.135	-	-	-	-	-	-	-	-	-	-	-
ENeRF [8]	0.086	0.119	0.107	0.107	0.076	0.052	0.108	0.117	0.118	0.120	0.091	0.077	0.069	0.048	0.066	0.069
MatchNeRF [4]	0.157	0.227	0.149	0.179	0.132	0.085	0.169	0.234	0.220	0.216	0.174	0.127	0.164	0.077	0.093	0.141
Ours	**0.069**	**0.102**	**0.088**	**0.098**	**0.070**	**0.048**	**0.093**	**0.097**	**0.098**	**0.101**	**0.075**	**0.067**	**0.055**	**0.041**	**0.057**	**0.057**
PixelSplat* [2]	0.423	0.366	0.471	0.357	0.366	0.329	0.429	0.435	0.493	0.427	0.438	0.488	0.343	0.278	0.326	0.254
Ours*	0.087	0.118	0.121	0.114	0.079	0.057	0.126	0.118	0.126	0.132	0.093	0.090	0.074	0.049	0.067	0.079

Table 9: Quantitative per-scene breakdown results on the Tanks and Temples dataset. PixelSplat* and Ours* represent the results obtained with a 2-view input and low-resolution images, while the other generalizable results are obtained with a 3-view input.

Scene	Train			Truck		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
IBRNet [14]	22.35	0.763	0.285	19.13	0.755	0.280
MVSNeRF [3]	20.58	0.816	0.278	21.16	0.830	0.242
ENeRF [8]	22.54	0.851	0.204	22.53	0.856	0.163
MatchNeRF [4]	20.44	0.789	0.332	21.16	0.796	0.269
Ours	**23.00**	**0.872**	**0.154**	**23.55**	**0.883**	**0.124**
PixelSplat* [2]	18.21	0.638	0.252	20.58	0.741	0.195
Ours*	23.67	0.864	0.132	22.68	0.834	0.127
NeRF [11]	21.02	0.707	0.538	21.82	0.696	0.577
IBRNet$_{ft-1.0h}$ [14]	23.92	0.816	0.229	20.51	0.810	0.212
MVSNeRF$_{ft-15min}$ [3]	21.34	0.831	0.253	22.32	0.850	0.217
ENeRF$_{ft-1.0h}$ [8]	24.35	0.884	0.148	24.01	0.885	0.141
3D-GS$_{ft-2min30s}$ [5]	21.07	0.825	0.255	19.19	0.731	0.384
3D-GS$_{ft-15min}$ [5]	**24.89**	0.897	0.152	22.42	0.838	0.215
Ours$_{ft-90s}$	24.73	**0.910**	**0.133**	**24.43**	**0.896**	**0.140**

Table 10: Quantitative per-scene breakdown results on the Real Forward-facing dataset. PixelSplat* and Ours* represent the results obtained with a 2-view input and low-resolution images, while the other generalizable results are obtained with a 3-view input.

Scene	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex
Metric	PSNR ↑							
PixelNeRF [18]	12.40	10.00	14.07	11.07	9.85	9.62	11.75	10.55
IBRNet [14]	20.83	22.38	27.67	22.06	18.75	15.29	27.26	20.06
MVSNeRF [3]	21.15	24.74	26.03	23.57	17.51	17.85	26.95	**23.20**
ENeRF [8]	21.92	24.28	30.43	24.49	19.01	17.94	29.75	21.21
MatchNeRF [4]	20.98	23.97	27.44	23.14	18.62	**18.07**	26.77	20.47
Ours	**22.45**	**25.66**	**30.46**	**24.70**	**19.81**	17.86	**29.86**	21.75
PixelSplat* [2]	22.41	24.48	27.00	25.02	19.80	18.39	27.56	19.28
Ours*	22.47	23.96	30.00	23.97	19.42	17.06	28.59	20.95
NeRF$_{ft-10.2h}$ [11]	23.87	26.84	**31.37**	25.96	21.21	19.81	**33.54**	**25.19**
IBRNet$_{ft-1.0h}$ [14]	22.64	26.55	30.34	25.01	22.07	19.01	31.05	22.34
MVSNeRF$_{ft-15min}$ [3]	23.10	27.23	30.43	26.35	21.54	20.51	30.12	24.32
ENeRF$_{ft-1.0h}$ [8]	22.08	**27.74**	29.58	25.50	21.26	19.50	30.07	23.39
3D-GS$_{ft-2min}$ [5]	**24.62**	23.23	28.94	20.49	15.81	**22.76**	22.17	19.19
3D-GS$_{ft-10min}$ [5]	24.58	24.90	29.27	21.90	15.77	22.42	31.45	21.09
Ours$_{ft-45s}$	24.32	27.66	31.05	**30.30**	**22.53**	22.38	33.11	24.51
Metric	SSIM ↑							
PixelNeRF [18]	0.531	0.433	0.674	0.516	0.268	0.317	0.691	0.458
IBRNet [14]	0.710	0.854	0.894	0.840	0.705	0.571	0.950	0.768
MVSNeRF [3]	0.638	0.888	0.872	0.868	0.667	0.657	0.951	**0.868**
ENeRF [8]	0.774	0.893	**0.948**	0.905	0.744	0.681	0.971	0.826
MatchNeRF [4]	0.726	0.861	0.906	0.870	0.690	0.675	0.949	0.767
Ours	**0.792**	**0.908**	**0.948**	**0.913**	**0.784**	**0.701**	**0.973**	0.841
PixelSplat* [2]	0.754	0.868	0.891	0.884	0.747	0.673	0.952	0.712
Ours*	0.787	0.877	0.937	0.896	0.772	0.649	0.962	0.798
NeRF$_{ft-10.2h}$ [11]	0.828	0.897	0.945	0.900	0.792	0.721	0.978	0.899
IBRNet$_{ft-1.0h}$ [14]	0.774	0.909	0.937	0.904	0.843	0.705	0.972	0.842
MVSNeRF$_{ft-15min}$ [3]	0.795	0.912	0.943	0.917	0.826	0.732	0.966	0.895
ENeRF$_{ft-1.0h}$ [8]	0.770	0.923	0.940	0.904	0.827	0.725	0.965	0.869
3D-GS$_{ft-2min}$ [5]	**0.845**	0.850	0.918	0.813	0.495	**0.850**	0.930	0.759
3D-GS$_{ft-10min}$ [5]	0.841	0.870	0.934	0.820	0.490	0.843	0.975	0.807
Ours$_{ft-45s}$	0.835	**0.937**	**0.963**	**0.962**	**0.871**	0.844	**0.986**	**0.911**
Metric	LPIPS ↓							
PixelNeRF [18]	0.650	0.708	0.608	0.705	0.695	0.721	0.611	0.667
IBRNet [14]	0.349	0.224	0.196	0.285	0.292	0.413	0.161	0.314
MVSNeRF [3]	0.238	0.196	0.208	0.237	0.313	**0.274**	0.172	0.184
ENeRF [8]	0.224	0.164	**0.092**	0.161	0.216	0.289	0.120	0.192
MatchNeRF [4]	0.285	0.202	0.169	0.234	0.277	0.325	0.167	0.294
Ours	**0.193**	**0.133**	0.096	**0.148**	**0.189**	0.275	**0.104**	**0.177**
PixelSplat* [2]	0.181	0.158	0.149	0.160	0.214	0.275	0.128	0.258
Ours*	0.173	0.124	0.082	0.142	0.182	0.261	0.083	0.167
NeRF$_{ft-10.2h}$ [11]	0.291	0.176	0.147	0.247	0.301	0.321	0.157	0.245
IBRNet$_{ft-1.0h}$ [14]	0.266	0.146	0.133	0.190	0.180	0.286	0.089	0.222
MVSNeRF$_{ft-15min}$ [3]	0.253	0.143	0.134	0.188	0.222	0.258	0.149	0.187
ENeRF$_{ft-1.0h}$ [8]	0.197	0.121	0.101	0.155	0.168	0.247	0.113	0.169
3D-GS$_{ft-2min}$ [5]	0.154	0.204	0.146	0.338	0.425	**0.142**	0.222	0.309
3D-GS$_{ft-10min}$ [5]	**0.147**	0.183	0.121	0.289	0.421	0.145	0.123	0.276
Ours$_{ft-45s}$	0.161	**0.097**	**0.077**	**0.091**	**0.143**	0.145	**0.079**	**0.113**

Table 11: Quantitative per-scene breakdown results on the NeRF Synthetic dataset. PixelSplat* and Ours* represent the results obtained with a 2-view and low-resolution input, while the other generalizable results are obtained with a 3-view input.

Scene	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
Metric	PSNR ↑							
PixelNeRF [18]	7.18	8.15	6.61	6.80	7.74	7.61	7.71	7.30
IBRNet [14]	24.20	18.63	21.59	27.70	22.01	20.91	22.10	22.36
MVSNeRF [3]	23.35	20.71	21.98	28.44	23.18	20.05	22.62	23.35
ENeRF [8]	28.29	21.71	**23.83**	34.20	**24.97**	24.01	26.62	25.73
MatchNeRF [4]	25.23	19.97	22.72	24.19	23.77	23.12	24.46	22.11
Ours	**28.93**	**22.20**	23.55	**35.01**	**24.97**	**24.49**	**26.80**	**25.75**
PixelSplat* [2]	16.45	15.40	17.47	13.25	16.86	15.88	16.83	14.06
Ours*	27.95	21.20	23.22	33.79	24.23	24.55	24.22	23.54
NeRF [11]	31.07	25.46	29.73	34.63	32.66	**30.22**	31.81	29.49
IBRNet$_{ft-1.0h}$ [14]	28.18	21.93	25.01	31.48	25.34	24.27	27.29	21.48
MVSNeRF$_{ft-15min}$ [3]	26.80	22.48	26.24	32.65	26.62	25.28	29.78	26.73
ENeRF$_{ft-1.0h}$ [8]	28.94	25.33	24.71	35.63	25.39	24.98	29.25	26.36
3D-GS$_{ft-1min15s}$ [5]	31.90	**26.56**	**34.21**	34.21	**36.28**	29.80	**34.56**	29.70
3D-GS$_{ft-7min}$ [5]	31.20	26.26	33.93	34.30	36.10	29.53	34.39	28.90
Ours$_{ft-50s}$	**32.80**	25.91	31.54	**36.85**	35.68	29.83	33.92	**31.09**
Metric	SSIM ↑							
PixelNeRF [18]	0.624	0.670	0.669	0.669	0.671	0.644	0.729	0.584
IBRNet [14]	0.888	0.836	0.881	0.923	0.874	0.872	0.927	0.794
MVSNeRF [3]	0.876	0.886	0.898	0.962	0.902	0.893	0.923	0.886
ENeRF [8]	0.965	0.918	0.932	0.981	0.948	0.937	0.969	0.891
MatchNeRF [4]	0.908	0.868	0.897	0.943	0.903	0.908	0.947	0.806
Ours	**0.969**	**0.927**	**0.935**	**0.984**	**0.953**	**0.946**	**0.974**	**0.895**
PixelSplat* [2]	0.816	0.787	0.857	0.644	0.799	0.764	0.861	0.508
Ours*	0.962	0.909	0.920	0.978	0.940	0.940	0.957	0.873
NeRF [11]	0.971	0.943	0.969	0.980	0.975	0.968	0.981	0.908
IBRNet$_{ft-1.0h}$ [14]	0.955	0.913	0.940	0.978	0.940	0.937	0.974	0.877
MVSNeRF$_{ft-15min}$ [3]	0.934	0.898	0.944	0.971	0.924	0.927	0.970	0.879
ENeRF$_{ft-1.0h}$ [8]	0.971	**0.960**	0.939	0.985	0.949	0.947	0.985	0.893
3D-GS$_{ft-1min15s}$ [5]	0.981	0.956	**0.986**	0.983	0.987	**0.970**	0.991	0.918
3D-GS$_{ft-7min}$ [5]	0.977	0.951	0.985	0.981	0.987	0.968	**0.992**	0.909
Ours$_{ft-50s}$	**0.983**	0.952	0.981	**0.987**	**0.988**	**0.970**	**0.992**	**0.921**
Metric	LPIPS ↓							
PixelNeRF [18]	0.386	0.421	0.335	0.433	0.427	0.432	0.329	0.526
IBRNet [14]	0.144	0.241	0.159	0.175	0.202	0.164	0.103	0.369
MVSNeRF [3]	0.282	0.187	0.211	0.173	0.204	0.216	0.177	0.244
ENeRF [8]	0.055	0.110	0.076	0.059	0.075	0.084	0.039	0.183
MatchNeRF [4]	0.107	0.185	0.117	0.162	0.160	0.119	0.060	0.398
Ours	**0.036**	**0.091**	**0.069**	**0.040**	**0.066**	**0.063**	**0.027**	**0.179**
PixelSplat* [2]	0.260	0.287	0.282	0.365	0.273	0.309	0.241	0.493
Ours*	0.039	0.098	0.066	0.038	0.071	0.050	0.038	0.170
NeRF [11]	0.055	0.101	0.047	0.089	0.054	0.105	0.033	0.263
IBRNet$_{ft-1.0h}$ [14]	0.079	0.133	0.082	0.093	0.105	0.093	0.040	0.257
MVSNeRF$_{ft-15min}$ [3]	0.129	0.197	0.171	0.094	0.176	0.167	0.117	0.294
ENeRF$_{ft-1.0h}$ [8]	0.030	**0.045**	0.071	**0.028**	0.070	0.059	0.017	0.183
3D-GS$_{ft-1min15s}$ [5]	0.022	0.059	**0.016**	0.042	**0.021**	0.041	0.010	0.180
3D-GS$_{ft-7min}$ [5]	0.026	0.062	0.018	0.044	**0.021**	0.043	**0.009**	0.172
Ours$_{ft-50s}$	**0.021**	0.059	0.022	0.032	**0.021**	**0.038**	0.010	**0.138**

References

1. Aanaes, H., Jensen, R.R., Vogiatzis, G., Tola, E., Dahl, A.B.: Large-scale data for multiple-view stereopsis. IJCV **120**, 153–168 (2016)
2. Charatan, D., Li, S., Tagliasacchi, A., Sitzmann, V.: pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In: arXiv (2023)
3. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: ICCV. pp. 14124–14133 (2021)
4. Chen, Y., Xu, H., Wu, Q., Zheng, C., Cham, T.J., Cai, J.: Explicit correspondence matching for generalizable neural radiance fields. arXiv preprint arXiv:2304.12294 (2023)
5. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples benchmarking large-scale scene reconstruction. ACM Trans. Graph. **36**(4), 1–13 (2017)
8. Lin, H., Peng, S., Xu, Z., Yan, Y., Shuai, Q., Bao, H., Zhou, X.: Efficient neural radiance fields for interactive free-viewpoint video. In: SIGGRAPH Asia Conference Proceedings (2022)
9. Liu, T., Ye, X., Zhao, W., Pan, Z., Shi, M., Cao, Z.: When epipolar constraint meets non-local operators in multi-view stereo. In: ICCV. pp. 18088–18097 (2023)
10. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Trans. Graph. **38**(4), 1–14 (2019)
11. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
12. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR. pp. 4104–4113 (2016)
13. Szymanowicz, S., Rupprecht, C., Vedaldi, A.: Splatter image: Ultra-fast single-view 3d reconstruction. In: arXiv (2023)
14. Wang, Q., Wang, Z., Genova, K., Srinivasan, P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: CVPR (2021)
15. Yan, J., Wei, Z., Yi, H., Ding, M., Zhang, R., Chen, Y., Wang, G., Tai, Y.W.: Dense hybrid recurrent multi-view stereo net with dynamic consistency checking. In: ECCV. pp. 674–689. Springer (2020)
16. Yao, Y., Luo, Z., Li, S., Fang, T., Quan, L.: Mvsnet depth inference for unstructured multi-view stereo. In: ECCV. pp. 767–783 (2018)
17. Yao, Y., Luo, Z., Li, S., Zhang, J., Ren, Y., Zhou, L., Fang, T., Quan, L.: Blendedmvs a large-scale dataset for generalized multi-view stereo networks. In: CVPR. pp. 1790–1799 (2020)
18. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelNeRF: Neural radiance fields from one or few images. In: CVPR (2021)
19. Zheng, S., Zhou, B., Shao, R., Liu, B., Zhang, S., Nie, L., Liu, Y.: Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. arXiv (2023)
20. Zhu, Z., Fan, Z., Jiang, Y., Wang, Z.: Fsgs: Real-time few-shot view synthesis using gaussian splatting (2023)