Geometry-aware Reconstruction and Fusion-refined Rendering for Generalizable Neural Radiance Fields (Supplementary Material)

Tianqi Liu Xinyi Ye Min Shi Zihao Huang Zhiyu Pan Zhan Peng Zhiguo Cao*
School of Artificial Intelligence and Automation,
Huazhong University of Science and Technology, Wuhan 430074, China
{tq_liu,xinyiye,min_shi,zihaohuang,zhiyupan,peng_zhan,zgcao}@hust.edu.cn

S1. Implementation and Network Details

Implementation Details. Our generalizable model is trained on four RTX 3090 GPUs using the Adam [9] optimizer, with an initial learning rate of $5e-4$. The learning rate is halved every 50k iterations. As shown in Fig. S1, we present the metrics of the DTU test set [1] varying with the number of iterations. The model tends to converge after approximately 150k iterations, taking about 25 hours. It is worth noting that, with only 8k iterations, our model can achieve metrics of $27.68/0.961/0.080$, surpassing the metrics of the SOTA method [10], which is $27.61/0.956/0.091$ (PSNR/SSIM [17]/LPIPS [20]). Following ENeRF [10], during training, we select 2, 3, and 4 source views as inputs with probabilities of 0.1, 0.8, and 0.1. To save computational costs, the Consistency-aware Fusion (CAF) is exclusively employed in the fine stage, while in the coarse stage, a blending approach is used to synthesize a low-resolution target view, supervised by the ground-truth target view. During training, the final target view is generated by fusing two intermediate views as $I_t = W_b I_b + W_r I_r$. However, during evaluation for other datasets [12, 13], which have a different domain compared with DTU, increasing the weights of I_b can lead to slightly better performance. Therefore, we obtain the final view as $I_t = (I_b + (W_b I_b + W_r I_r))/2$. Our evaluation setup is consistent with ENeRF [10] and MVSNeRF [3]. The results of the DTU test set are evaluated using segmentation masks. The segmentation mask is defined based on the availability of ground-truth depth at each pixel. Since the marginal region of images is typically invisible to input images on the Real Forward-facing dataset [12], we evaluate the 80% area in the center of the images. Incidentally, all the inference time presented in Fig. 1 of the main text is measured at an input image resolution of 512×640. The number of our model's parameters is 3.15M.

*Corresponding author

Input	Operation	Output
(C,D,H,W)	permute	(D,C,H,W)
(D,C,H,W)	cgr(C,C_h)	(D,C_h,H,W)
(D,C_h,H,W)	cgr(C_h,C_h)	(D,C_h,H,W)
(D,C_h,H,W)	conv2d(C_h,1)	(D,1,H,W)
(D,1,H,W)	sigmoid	(D,1,H,W)
(D,1,H,W)	permute	(1,D,H,W)

Table S1. **The network architecture of Adaptive Cost Aggregation (ACA).** The cgr represents a block composed of conv2d, groupnorm, and relu. In our implementation, $C = 16$ and $C_h = 4$.

Network Details. Here, we will introduce the network details of the pooling network (Sec. 4.2) and ACA (Sec. 4.1) mentioned in the main text.

Pooling network. [10, 15] apply a pooling network ρ to aggregate multi-view features to obtain the descriptor via $f_p = \rho(\{f_s^i\}_{i=1}^N)$. The implementation details are as follows: initially, the mean μ and variance v of $\{f_s^i\}_{i=1}^N$ are computed. Subsequently, μ and v are concatenated with each f_s^i and an MLP is applied to generate a weight. The f_p is blended via a soft-argmax operator using obtained weights and multi-view features $(\{f_s^i\}_{i=1}^N)$.

ACA. Per the Eq. (4) in the main text, $\alpha(.)$ represents the adaptive weight for each view, and the network architecture that learns these weights is shown in Table S1.

Inference Speed. For an image with 512×640 resolution, the inference time of our method is 143ms. We decompose the inference time in Table S2 and results demonstrate that the inference time of our modules is only 71ms, with the remaining 72ms spent saving results.

S2. Additional ablation experiments

Numbers of Views. As shown in Table S3, we evaluate the performance of our trained generalization model and

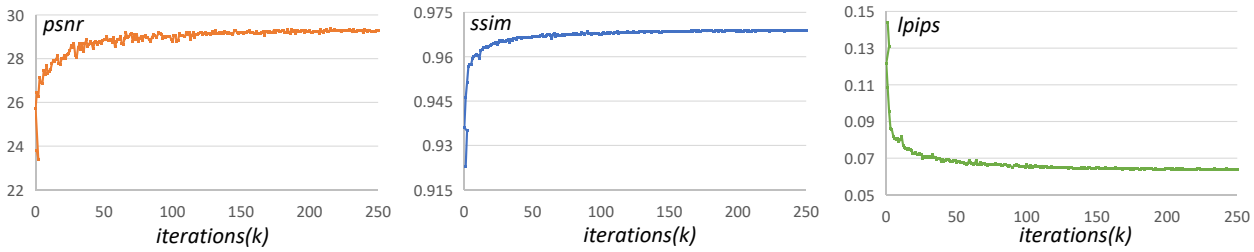

Figure S1. **The metrics of the DTU test set vary with the number of iterations.**

Modules		coarse stage	fine stage
feature extractor		1.32	
build rays		9.11	15.94
geometry	cost volume	7.48	8.24
	regularization	1.91	1.97
descriptor	pooling ρ	0.57	0.50
	SVA_{sm}	1.74	1.37
	SVA_{d}	0.73	1.01
view decoding		1.62	17.80
save in dictionary		13.27	58.31

Table S2. **Time overhead for each module (in milliseconds).** The term "save in dictionary" refers to storing tensor results in dictionary form for subsequent evaluation of various metrics.

ENeRF [10] with different numbers of input views on the DTU test set [1]. With an increase in the number of input views, the performance improves as the model can leverage more multi-view information. In terms of both overall performance and the magnitude of performance improvement, our method outperforms ENeRF, indicating its superior capability in leveraging multi-view information for reconstructing scene geometry and rendering novel views. Additionally, we also present the performance of our method on the Real Forward-facing [12] and NeRF Synthetic [13] datasets under different numbers of input views, as shown in Table S4. The results demonstrate the same trend, indicating the capability of our model in leveraging multi-view information is generalizable.

Features for Intermediate Views. In Sec. 4.3 of the main text, f_b and f_r are utilized as the feature representations for the two intermediate views I_b and I_r, respectively. Subsequently, their consistency with source views is individually computed to learn fusion weights. The choice of using f_b as the feature for I_b is based on their similar volume rendering generation manners, while the selection of f_r as the feature for I_r is driven by their direct projection relationship. Here, we will discuss different selection strategies for the features of intermediate views. An alternative approach for the features of I_b is to blend features from source views. Similar to Eq. (2) in the main text, the calculation of the features f_b for I_b is as follows:

$$f_b = \sum_{i=1}^{N} \frac{exp(w_i) f_s^i}{\sum_{j=1}^{N} exp(w_j)}, \tag{S1}$$

where $w_i = \text{MLP}(x, d, f_p, f_s^i)$, f_s^i is the feature of the source image I_s^i. x and d represent the coordinate and view direction, respectively. f_p is the descriptor for 3D point. Another more intuitive alternative is to use a feature extractor to extract features for both intermediate views, as:

$$F_{[b,r]} = \phi_e I_{[b,r]}, \tag{S2}$$

where ϕ_e represents a feature extractor, instantiated as a 2D U-Net. f_b and f_r are the pixel-wise features of F_b and F_r, respectively. As shown in Table S5, the strategy employed in the main text is slightly superior to the other two alternative strategies. For the first alternative Eq. (S1), f_b is obtained by blending features from source views. f_b lacks 3D context awareness, leading to some information loss in the subsequently accumulated pixel features. For the second alternative Eq. (S2), f_b and f_r are extracted from scratch at the RGB level. This practice is disadvantageous for the subsequent learning of 3D consistency weights, due to the lack of utilization of 3D information. Additionally, the introduction of a feature extractor also increases the burden on the model. However, the strategy in the main text maximally utilizes the obtained 3D-aware descriptors, while also having the smallest computational cost compared to the other two alternative approaches.

Intermediate View Supervision. In the main text, we only supervise the images fused through CAF. However, simultaneously supervising intermediate results is also a common practice, whose final result is $29.15/0.968/0.065$. This result is slightly inferior to supervising only the fused view ($29.36/0.969/0.064$). Because each of the two intermediate views has its own advantages, supervising only the fused view allows the network to focus on the fusion process, leveraging the strengths of both. However, simultaneously supervising the intermediate views burdens the network, diminishing its attention to the fusion process. In theory, if both intermediate views are entirely correct, the final

Views	PSNR↑	SSIM↑	LPIPS↓	Abs err↓	Acc(2)↑	Acc(10)↑
2	26.98 / 25.48	0.955 / 0.942	0.081 / 0.107	3.86 / 5.53	0.835 / 0.756	0.942 / 0.107
3	29.36 / 27.61	0.969 / 0.956	0.064 / 0.091	2.83 / 4.60	0.879 / 0.792	0.961 / 0.917
4	29.77 / 27.73	0.971 / 0.959	0.062 / 0.089	2.73 / 4.26	0.880 / 0.804	0.961 / 0.929
5	29.91 / 27.54	0.971 / 0.958	0.062 / 0.091	2.69 / 4.29	0.882 / 0.800	0.961 / 0.928

Table S3. **The performance of our method and ENeRF with different numbers of input views on the DTU test set.** Each item represents (Ours/ENeRF's). "Abs err" denotes the average absolute error and "Acc(X)" means the percentage of pixels with an error less than X mm.

Views	Real Forward-facing [12]			NeRF Synthetic [13]		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
2	23.39	0.839	0.176	25.30	0.939	0.082
3	24.28	0.863	0.162	26.99	0.952	0.070
4	24.91	0.876	0.157	27.31	0.953	0.069

Table S4. **The performance of our method with varying numbers of input views on the Real Forward-facing and NeRF Synthetic datasets.**

	PSNR↑	SSIM↑	LPIPS↓
No.1	29.36	0.969	0.064
No.2	29.24	0.969	0.065
No.3	29.08	0.968	0.066

Table S5. **Different strategies for the features of intermediate views.** No.1 represents the strategy in the main text. No.2 represents the strategy using Eq. (S1). No.3 represents the strategy using Eq. (S2).

Depth	PSNR↑	SSIM↑	LPIPS↓	Abs err↓	Acc(2)↑	Acc(10)↑
Self-supervision	29.21	0.968	0.064	3.21	0.873	0.957
Supervision	29.31	0.969	0.064	2.95	0.875	0.957
None	29.36	0.969	0.064	2.83	0.879	0.961

Table S6. **The comparison of different depth supervision signals.** The self-supervision represents using unsupervised depth loss and the supervision represents using ground-truth depth for supervision. The term "None" refers to training without any depth supervision signals.

fused view will be accurate regardless of the fusion process. The network prioritizes predicting two accurate intermediate views, which is a more challenging task.

Depth Supervision. A critical factor in the model's synthesis of high-quality views is its perception of the scene geometry. MVS-based generalizable NeRF methods [3, 10, 11], including our method, aim to improve the quality of view synthesis by enhancing the geometry prediction. By only supervising RGB images, excellent geometry predictions can be achieved (Sec. 5.4 in the main text). Here, we will discuss the impact of incorporating depth supervision signals on the model. We introduce supervision signals for depth in two ways: one through self-supervised and the other through supervision using ground-truth depth.

Following [2, 8], the unsupervised depth loss is:

$$L_d = \beta_1 L_{PC} + \beta_2 L_{SSIM} + \beta_3 L_{Smooth}\,, \tag{S3}$$

where L_{PC} represents the photometric consistency loss. L_{SSIM} and L_{Smooth} are the structured similarity loss and depth smoothness loss, respectively. β_1, β_2, and β_3 are set to 12, 6, and 0.18 in our implementation, respectively. Refer to [2, 8] for more details. L_d is used to supervise the final depth, *i.e.*, d_f (Sec. 4.3 in the main text). Since the DTU dataset provides ground-truth depth, another approach is to utilize the ground-truth depth for supervision. The depth loss is as follows:

$$L_d = \xi(d_f, d_{gt}) \tag{S4}$$

where d_f and d_{gt} represent the final predicted and ground-truth depth, respectively. ξ denotes a loss function. Following [6], ξ is instantiated as the Smooth L1 loss [5]. The quantitative results are presented in the Table S6. The performance of the three strategies in the table is comparable, indicating that supervising only the RGB images is sufficient, and there is no need for additional introduction of depth supervision signals.

More Comprehensive Depth Analysis. As shown in Fig. 3 in the main text, our pipeline first infers the geometry from the cost volume, re-samples 3D points around objects' surfaces, and finally encodes 3D descriptors for rendering. We can obtain two depths: one inferred from the cost volume and the other obtained through volume rendering, which is the final depth. Here, we report the depth obtained from the cost volume and the final depth as shown in Table S7. Compared to the baseline, our method performs better on both depth metrics. Thanks to Adaptive Cost Aggregation (ACA), the depth obtained from the cost volume has been significantly improved. Based on this, as the Spatial-View Aggregator (SVA) encodes 3D-aware descriptors, the final depth has also been further improved. In

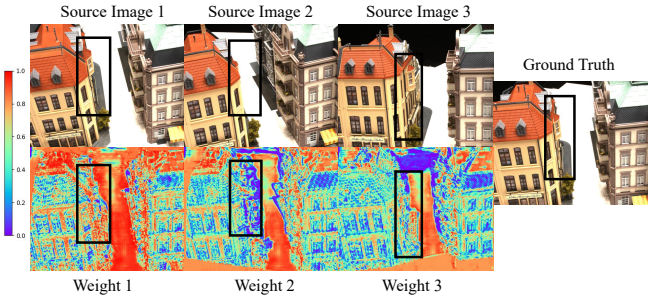

Figure S2. **Visualization of ACA.**

addition, the well-designed decoding approach, *i.e.*, CAF, greatly facilitates the depth prediction of the model (Sec. 5.5 in the main text).

Visualization of ACA. Previous approaches using variance struggle to encode efficient cost information for challenging areas, such as the occluded areas marked in the black box in Fig. S2. Our proposed ACA module learns an adaptive weight for different views to encode accurate geometric information. As illustrated in Fig. S2, the weights learned by ACA amplify the contribution of consistent pixel pairs, such as the visible areas in source image 1 and 3, while suppressing inconsistent ones, as shown in the occluded areas in the source image 2.

Different ACA networks. The primary challenge of applying ACA to the NVS task is the unavailability of the target view, which we addressed by adopting a coarse-to-fine framework. In the main text, the weight learning network utilized in ACA is illustrated in Table S1, following the MVS method, *i.e.*, AA-RMVSNet [18]. Moreover, other networks can also be embedded into our coarse-to-fine framework to learn inter-view weights. Here, we adopt another MVS method, *i.e.*, MVSTER [16], to learn adaptive weights. The result on the DTU test set is $29.31/0.969/0.064$ (PSNR/SSIM/LPIPS), which is comparable with the result obtained using [18]. In summary, our main contribution is to propose an approach for applying ACA to the NVS task, without specifying a particular network for learning weights.

Analysis of SVA. Previous approaches directly uses a pooling network to aggregate multi-view 2D features for encoding 3D descriptors, which are not spatially context-aware, leading to discontinuities in the decoded depth map and rendered view (see Fig. S3 (a)). To address this issue, convolutional networks can be used to impose spatial constraints on adjacent descriptors. However, due to the smooth nature of convolution, some high-frequency details may be lost. Since detailed information comes from the multi-view

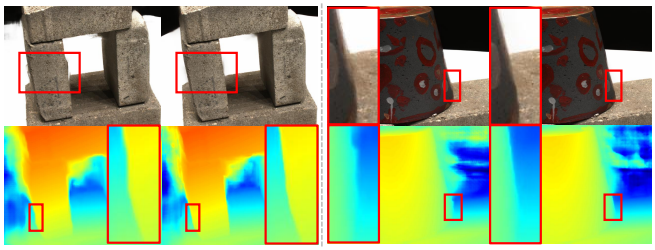

Figure S3. Visualization of SVA. SVA$_{sm}$ and SVA$_d$ represent ϕ_{sm} and ϕ_d of SVA, respectively.

Method	Reference view			Novel view		
	Abs err ↓	Acc(2)↑	Acc(10)↑	Abs err ↓	Acc(2)↑	Acc(10)↑
Baseline-mvs	3.71	0.815	0.942	4.49	0.778	0.928
Baseline-final	3.58	0.842	0.944	4.32	0.800	0.928
Ours-mvs	2.79	0.836	0.965	3.15	0.816	0.958
Ours-final	2.47	0.900	0.971	2.83	0.879	0.961

Table S7. **More comprehensive depth metrics.**"-mvs" represents the depth obtained from the cost volume and "-final" represents the final depth obtained through volume rendering.

Approach	PSNR↑	SSIM↑	LPIPS↓
Regression	27.32	0.945	0.119
Blending	28.40	0.962	0.091
Overall	29.36	0.969	0.064

Table S8. **Quantitative results for intermediate results.** Overall represents the fused views.

features, we employ a divide-and-conquer approach to aggregate descriptors. Firstly, we employ a 3D U-Net to aggregate spatial context and obtain smooth descriptors. Despite resolving the issue of discontinuities, an unsharpened object edge occurs (Fig. S3 (b)). Secondly, we propose using smoothed features as queries, with multi-view features serving as keys and values. Applying the attention mechanism allows us to gather high-frequency details adaptively. This practice results in continuities and sharp boundaries in both rendered views and depth maps.

S3. More Qualitative Results

Qualitative Results under the Generalization Setting. As shown in Fig. S4, S5, and S6, we present the qualitative comparison of rendering quality on the DTU [1], NeRF Synthetic [13], and Real Forward-facing [12] datasets, respectively. Our method can synthesize views with higher fidelity, especially in challenging areas. For example, in occluded regions and geometrically complex scenes, our method can reconstruct more details while exhibiting fewer artifacts at objects' edges and in reflective areas.

Qualitative Results under the Per-scene Optimization Setting. Benefiting from the strong initialization of our generalizable model, excellent performance can be achieved within just a short fine-tuning period, such as 15 minutes. As shown in Fig. S7, we present the results after fine-tuning. After per-scene optimization, the model demonstrates enhanced capabilities in handling scene details, resulting in views with higher fidelity.

Qualitative Comparison of Depth Maps. As shown in Figs. S8, S9, and S10, we present the qualitative comparison of depth maps on the DTU [1], NeRF Synthetic [13], and Real Forward-facing [12] datasets, respectively. The depth maps generated by our method can maintain sharper object edges and preserve more details of scenes, which verifies the strong geometry reasoning capability of our method.

Fusion Weight Visualization. As shown in Fig. S11, we present the fusion weights of the Consistency-aware Fusion (CAF) module. The blending approach generally demonstrates higher confidence in most areas, while the regression approach shows higher confidence in challenging regions such as object boundaries and reflections.

Error Map Visualization. As shown in Fig. S12, we present the error maps obtained by two decoding approaches, as well as the error maps of the fused views. The blending approach tends to exhibit lower errors in most areas, while the regression approach may have lower errors in some regions with reflections and edges. In addition, we also present quantitative results, as shown in Table S8. The views fused through Consistency-aware Fusion (CAF) integrate the advantages of both intermediate views, achieving a further improvement in quality.

S4. Per-scene Breakdown

As shown in Tables S9, S10, S11, and S12, we present the per-scene breakdown results of three datasets (DTU [1], NeRF Synthetic [13], and Real Forward-facing [12]). These results align with the averaged results in the main text.

S5. Limitations

Although our approach can achieve high performance for view synthesis, it still has the following limitations. 1) Like many other baselines [3, 15], our method is tailored specifically for static scenes and may not perform optimally when applied directly to dynamic scenes. 2) During per-scene optimization, the training speed and rendering speed of NeRF-based methods, including our method, are time-consuming. We will explore the potential of Gaussian Splatting [7] in generalizable NVS to address this issue in the future.

Scan	#1	#8	#21	#103	#114
Metric			PSNR ↑		
PixelNeRF [19]	21.64	23.70	16.04	16.76	18.40
IBRNet [15]	25.97	27.45	20.94	27.91	27.91
MVSNeRF [3]	26.96	27.43	21.55	29.25	27.99
NeuRay [11]	28.59	27.63	23.05	29.71	29.23
ENeRF [10]	28.85	29.05	22.53	30.51	28.86
GNT [14]	27.25	28.12	21.67	28.45	28.01
Ours	**30.72**	**30.87**	**23.96**	**31.78**	**29.84**
NeRF$_{10.2h}$ [13]	26.62	28.33	23.24	30.40	26.47
IBRNet$_{ft-1.0h}$ [15]	31.00	**32.46**	**27.88**	**34.40**	**31.00**
MVSNeRF$_{ft-15min}$ [3]	28.05	28.88	24.87	32.23	28.47
NeuRay$_{ft-1.0h}$ [11]	27.77	25.93	23.40	28.57	29.14
ENeRF$_{ft-1.0h}$ [10]	30.10	30.50	22.46	31.42	29.87
Ours$_{ft-15min}$	31.54	31.41	24.07	32.97	30.52
Ours$_{ft-1.0h}$	**31.58**	31.61	24.07	33.09	30.53
Metric			SSIM ↑		
PixelNeRF [19]	0.827	0.829	0.691	0.836	0.763
IBRNet [15]	0.918	0.903	0.873	0.950	0.943
MVSNeRF [3]	0.937	0.922	0.890	0.962	0.949
NeuRay [11]	0.872	0.826	0.830	0.920	0.901
ENeRF [10]	0.958	0.955	0.916	0.968	0.961
GNT [14]	0.922	0.931	0.881	0.942	0.960
Ours	**0.971**	**0.965**	**0.943**	**0.974**	**0.965**
NeRF$_{10.2h}$ [13]	0.902	0.876	0.874	0.944	0.913
IBRNet$_{ft-1.0h}$ [15]	0.955	0.945	**0.947**	**0.968**	0.964
MVSNeRF$_{ft-15min}$ [3]	0.934	0.900	0.922	0.964	0.945
NeuRay$_{ft-1.0h}$ [11]	0.872	0.751	0.845	0.868	0.900
ENeRF$_{ft-1.0h}$ [10]	0.966	0.959	0.924	0.971	0.965
Ours$_{ft-15min}$	**0.973**	**0.967**	0.945	0.976	**0.969**
Ours$_{ft-1.0h}$	**0.973**	**0.967**	0.945	0.976	**0.969**
Metric			LPIPS ↓		
PixelNeRF [19]	0.373	0.384	0.407	0.376	0.372
IBRNet [15]	0.190	0.252	0.179	0.195	0.136
MVSNeRF [3]	0.155	0.220	0.166	0.165	0.135
NeuRay [11]	0.157	0.201	0.156	0.140	0.128
ENeRF [10]	0.086	0.119	0.107	0.107	0.076
GNT [14]	0.143	0.210	0.171	0.149	0.139
Ours	**0.061**	**0.088**	**0.068**	**0.085**	**0.065**
NeRF$_{10.2h}$ [13]	0.265	0.321	0.246	0.256	0.225
IBRNet$_{ft-1.0h}$ [15]	0.129	0.170	0.104	0.156	0.099
MVSNeRF$_{ft-15min}$ [3]	0.171	0.261	0.142	0.170	0.153
NeuRay$_{ft-1.0h}$ [11]	0.155	0.272	0.142	0.177	0.125
ENeRF$_{ft-1.0h}$ [10]	0.071	0.106	0.097	0.102	0.074
Ours$_{ft-15min}$	0.057	**0.082**	0.067	0.080	0.061
Ours$_{ft-1.0h}$	**0.056**	**0.082**	**0.066**	**0.079**	**0.059**

Table S9. **Quantitative results of five sample scenes on the DTU test set.**

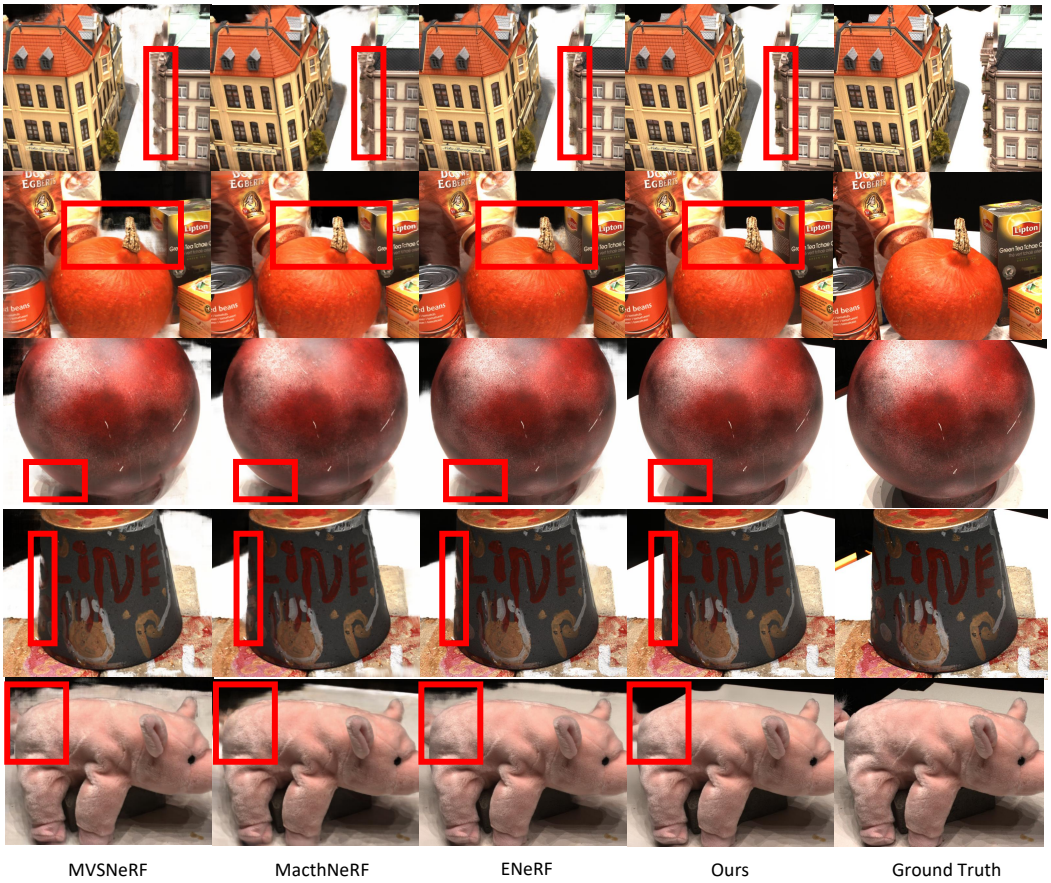

Figure S4. **Qualitative comparison of rendering quality with state-of-the-art methods [3, 4, 10] on the DTU dataset under generalization and three views settings.**

Scene	#30	#31	#34	#38	#40	#41	#45	#55	#63	#82	#110
Metric	PSNR ↑										
NeuRay [11]	21.10	23.35	24.46	26.01	24.16	27.17	23.75	27.66	23.36	23.84	**29.90**
ENeRF [10]	29.20	25.13	26.77	28.61	25.67	29.51	24.83	30.26	27.22	26.83	27.97
GNT [14]	27.13	23.54	25.10	27.67	24.48	28.10	24.54	28.86	26.36	26.09	26.93
Ours	**30.94**	**26.95**	**28.21**	**29.87**	**28.62**	**31.24**	**26.01**	**32.46**	**29.24**	**29.78**	29.30
Metric	SSIM ↑										
NeuRay [11]	0.916	0.851	0.767	0.800	0.812	0.872	0.878	0.870	0.927	0.919	0.927
ENeRF [10]	0.981	0.937	0.934	0.946	0.947	0.960	0.948	0.973	0.978	0.971	0.974
GNT [14]	0.954	0.907	0.880	0.921	0.893	0.908	0.918	0.934	0.938	0.949	0.930
Ours	**0.986**	**0.956**	**0.954**	**0.961**	**0.966**	**0.972**	**0.963**	**0.983**	**0.984**	**0.980**	**0.980**
Metric	LPIPS ↓										
NeuRay [11]	0.141	0.161	0.234	0.225	0.209	0.172	0.121	0.163	0.104	0.119	0.116
ENeRF [10]	0.052	0.108	0.117	0.118	0.120	0.091	0.077	0.069	0.048	0.066	0.069
GNT [14]	0.110	0.172	0.201	0.231	0.116	0.168	0.134	0.155	0.127	0.138	0.127
Ours	**0.039**	**0.075**	**0.085**	**0.082**	**0.082**	**0.065**	**0.051**	**0.045**	**0.032**	**0.044**	**0.052**

Table S10. **Quantitative results of other eleven scenes on the DTU test set.**

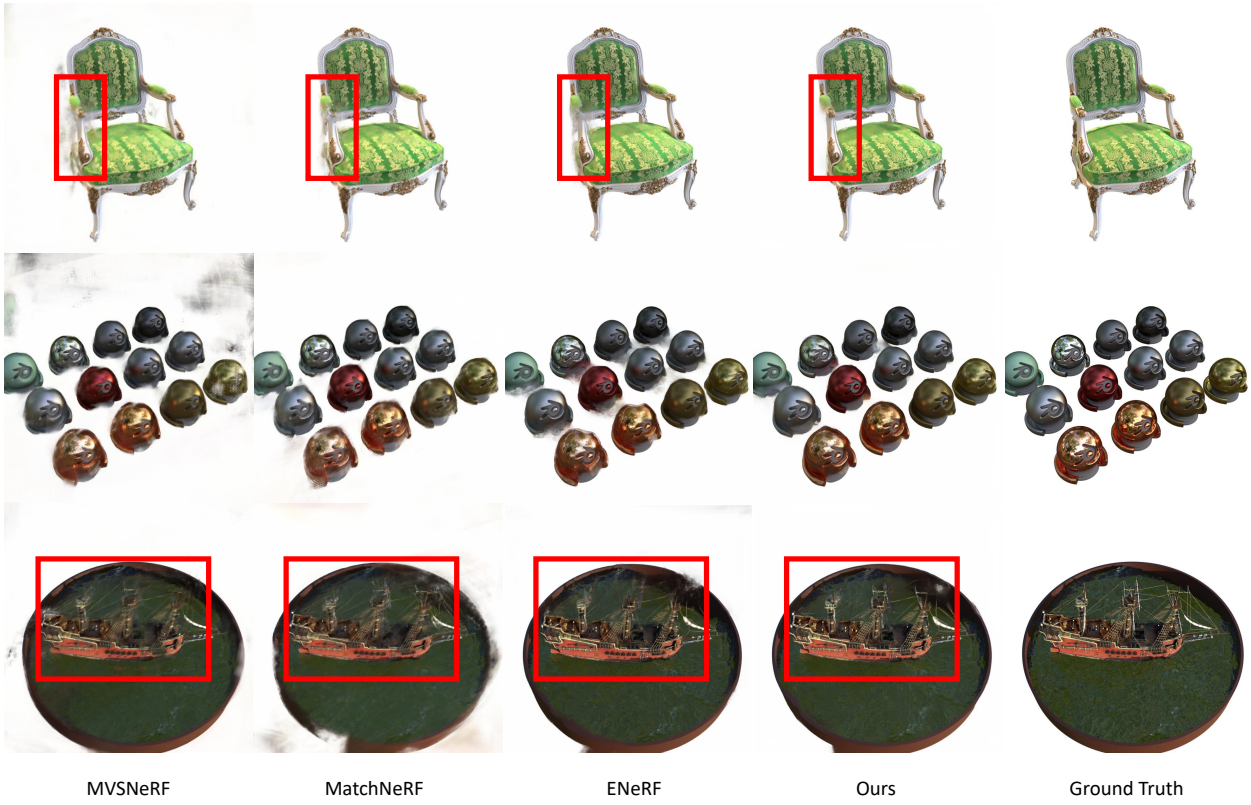

Figure S5. **Qualitative comparison of rendering quality with state-of-the-art methods [3, 4, 10] on the NeRF Synthetic dataset under generalization and three views settings.**

Figure S6. **Qualitative comparison of rendering quality with state-of-the-art methods [3, 4, 10] on the Real Forward-facing dataset under generalization and three views settings.**

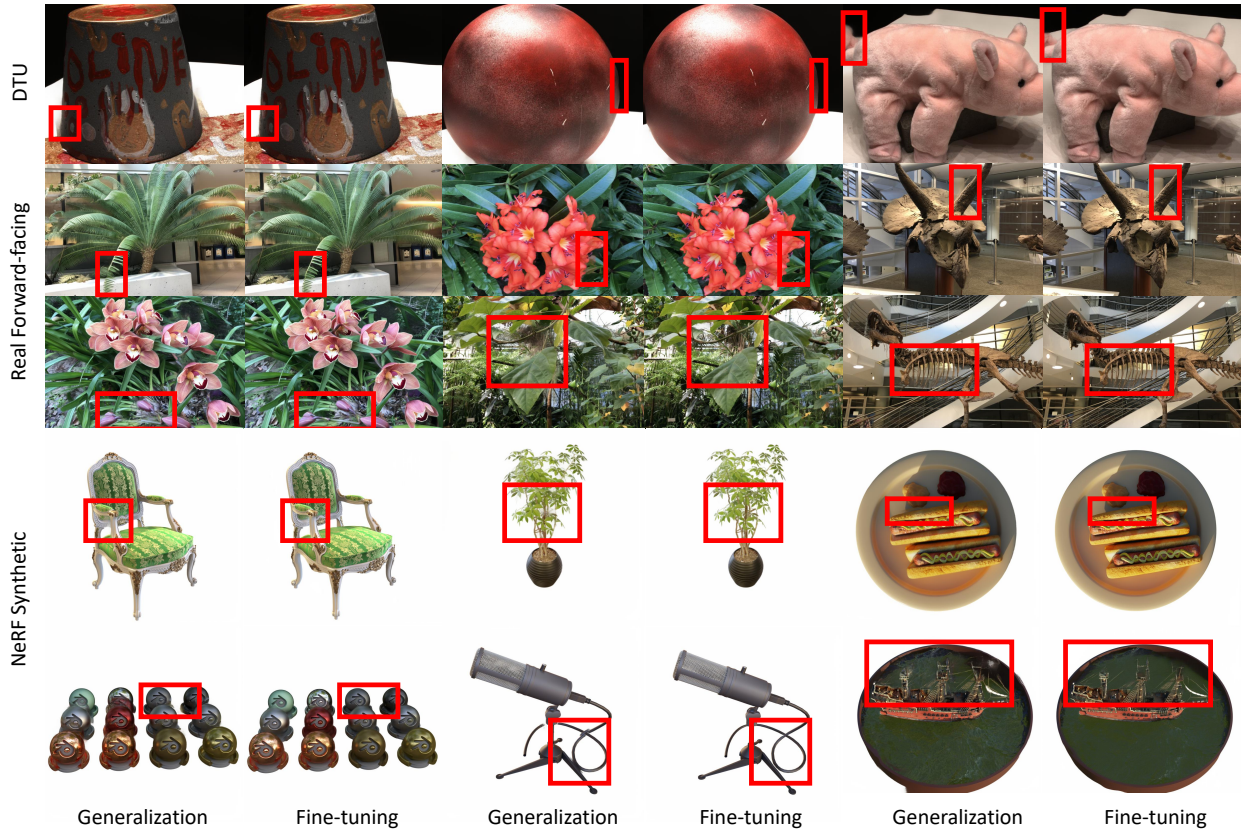

Figure S7. **Qualitative comparison of results before and after fine-tuning on the DTU [1], Real Forward-facing [12], and NeRF Synthetic [13] datasets.**

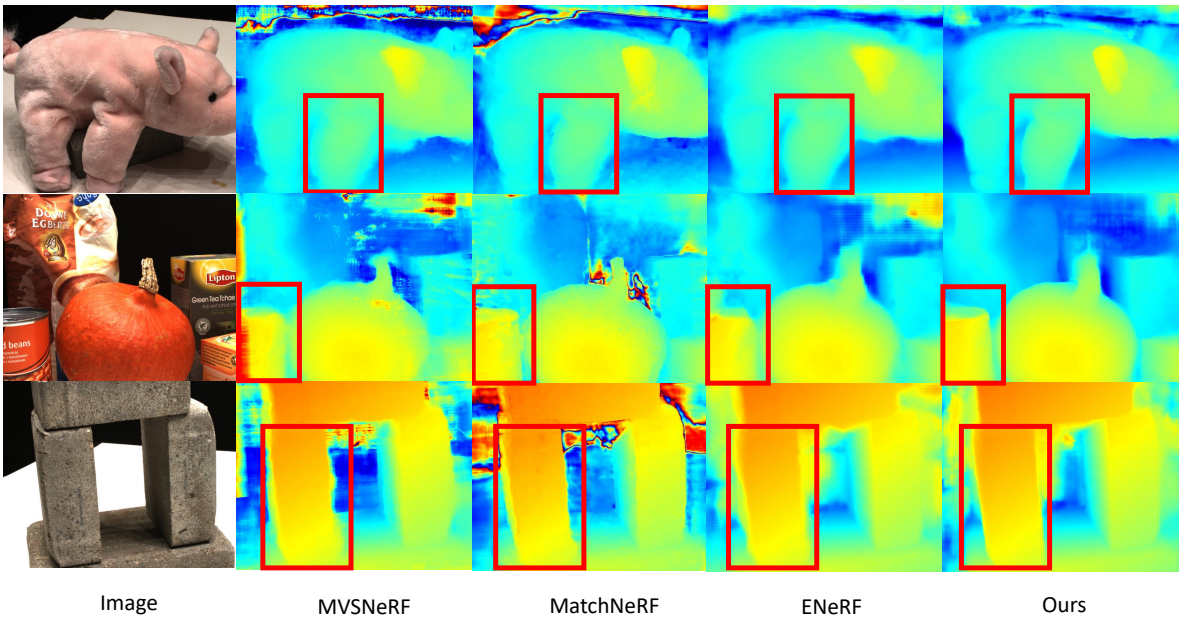

Figure S8. **Qualitative comparison of depth maps with state-of-the-art methods [3, 4, 10] on the DTU dataset.**

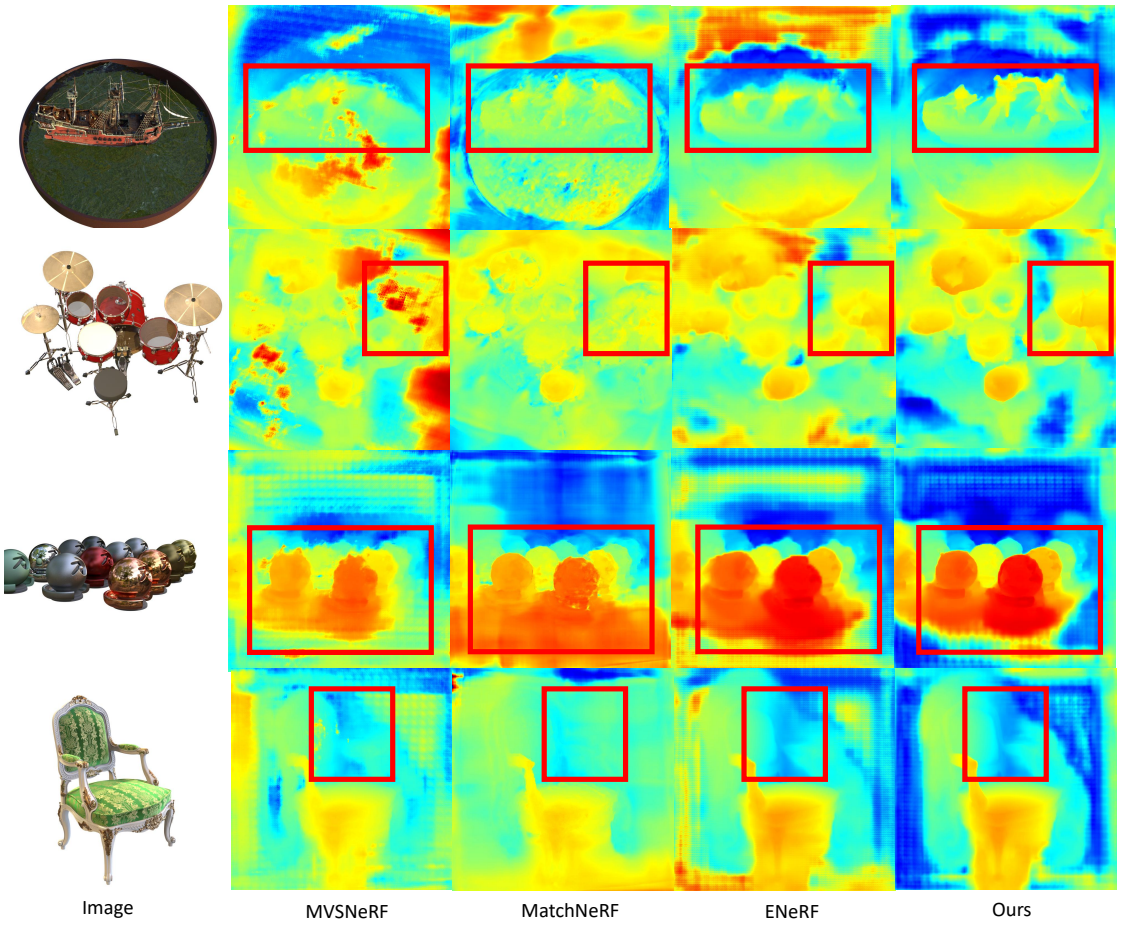

Figure S9. **Qualitative comparison of depth maps with state-of-the-art methods [3, 4, 10] on the NeRF Synthetic dataset.**

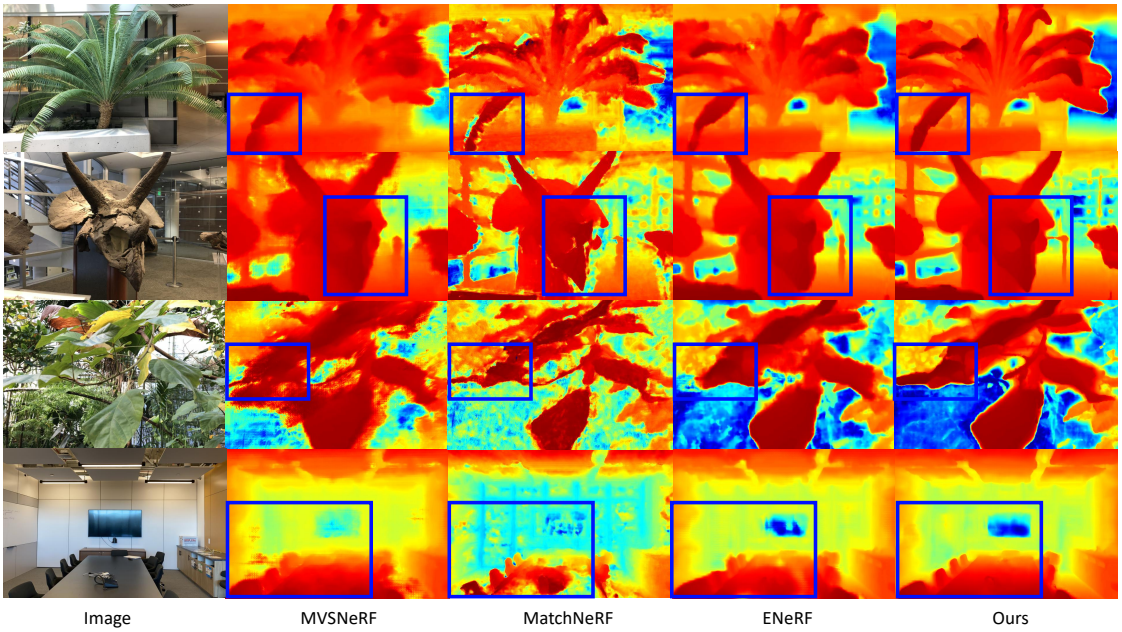

Figure S10. **Qualitative comparison of depth maps with state-of-the-art methods [3, 4, 10] on the Real Forward-facing dataset.**

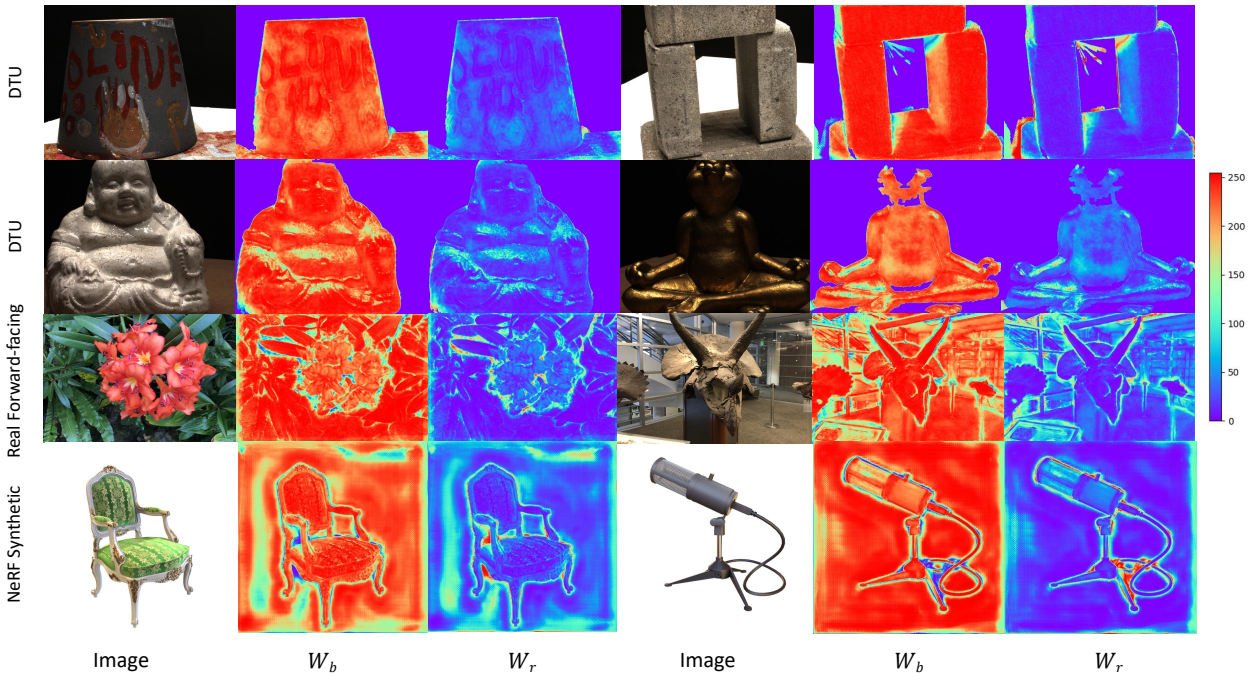

Figure S11. **Visualization of Fusion Weights.** W_b and W_r represent the weight maps of the blending approach and regression approach, respectively.

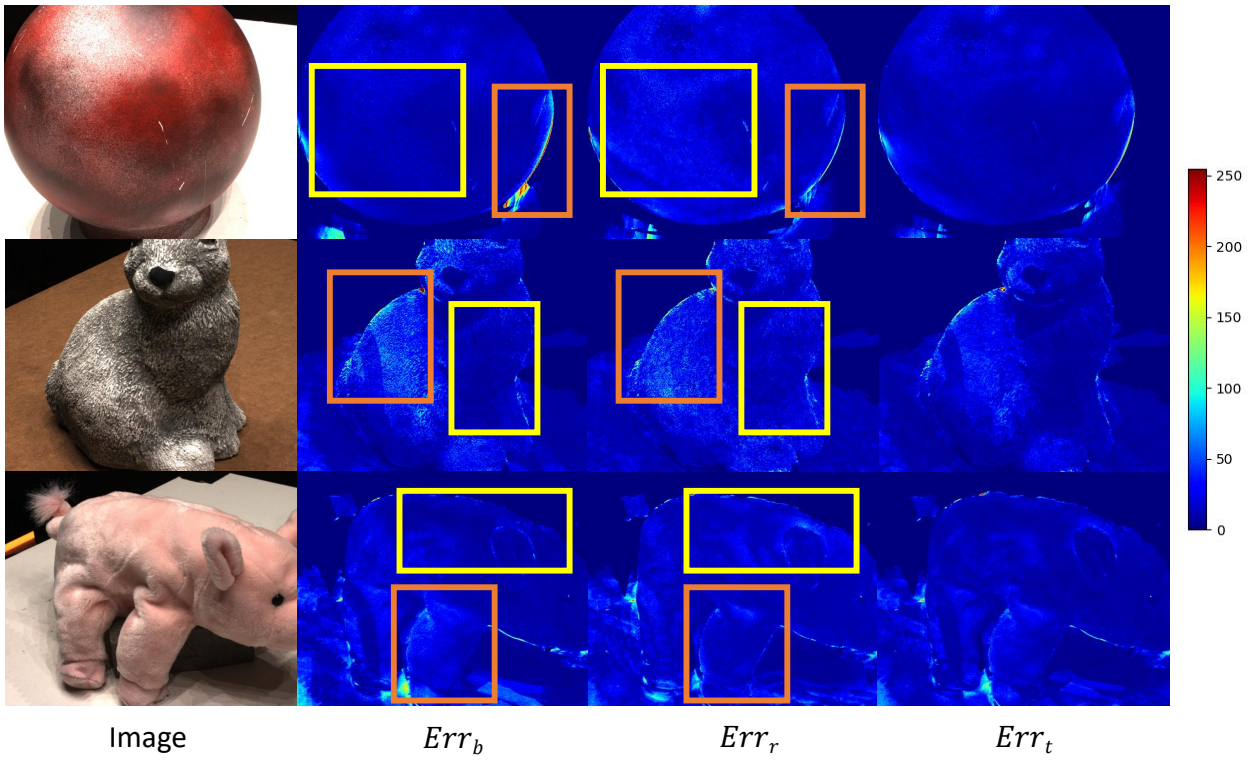

Figure S12. **Visualization of error maps.** Err_b and Err_r represent the error maps of the views obtained through the blending approach and the regression approach, respectively. Err_t is the error map of the final fused target view. The yellow boxes indicate that the blending approach outperforms the regression approach, while the orange boxes indicate regions where the regression approach outperforms the blending approach.

Scene	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
Metric	PSNR ↑							
PixelNeRF [19]	7.18	8.15	6.61	6.80	7.74	7.61	7.71	7.30
IBRNet [15]	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
NeuRay [11]	27.27	21.09	24.09	30.50	24.38	21.90	26.08	21.30
ENeRF [10]	28.29	21.71	23.83	34.20	**24.97**	24.01	26.62	25.73
GNT [14]	27.98	20.27	**26.86**	29.34	23.17	**30.75**	23.19	24.86
Ours	**28.87**	**22.33**	24.55	**34.96**	24.90	26.08	**27.98**	**26.22**
NeRF [13]	**31.07**	**25.46**	**29.73**	34.63	**32.66**	**30.22**	**31.81**	**29.49**
IBRNet$_{ft-1.0h}$ [15]	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
NeuRay$_{ft-1.0h}$ [11]	27.37	21.69	23.45	32.26	26.87	23.03	28.12	24.49
ENeRF$_{ft-1.0h}$ [10]	28.94	25.33	24.71	35.63	25.39	24.98	29.25	26.36
Ours$_{ft-15min}$	30.93	23.29	25.46	36.28	26.96	26.91	31.20	27.51
Ours$_{ft-1.0h}$	**31.07**	23.38	25.62	**36.73**	27.24	27.05	31.49	27.87
Metric	SSIM ↑							
PixelNeRF [19]	0.624	0.670	0.669	0.669	0.671	0.644	0.729	0.584
IBRNet [15]	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
NeuRay [11]	0.912	0.856	0.901	0.953	0.899	0.881	0.952	0.779
ENeRF [10]	0.965	0.918	0.932	0.981	0.948	0.937	0.969	0.891
GNT [14]	0.935	0.891	**0.941**	0.940	0.897	**0.974**	0.791	0.874
Ours	**0.971**	**0.931**	0.939	**0.983**	**0.956**	0.953	**0.980**	**0.899**
NeRF [13]	0.971	0.943	**0.969**	0.980	**0.975**	**0.968**	0.981	0.908
IBRNet$_{ft-1.0h}$ [15]	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
NeuRay$_{ft-1.0h}$ [11]	0.920	0.869	0.895	0.949	0.912	0.880	0.954	0.788
ENeRF$_{ft-1.0h}$ [10]	0.971	**0.960**	0.939	0.985	0.949	0.947	0.985	0.893
Ours$_{ft-15min}$	0.978	0.936	0.946	0.987	0.959	0.958	0.987	0.909
Ours$_{ft-1.0h}$	**0.979**	0.938	0.947	**0.988**	0.963	0.960	**0.989**	**0.912**
Metric	LPIPS ↓							
PixelNeRF [19]	0.386	0.421	0.335	0.433	0.427	0.432	0.329	0.526
IBRNet [15]	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
NeuRay [11]	0.146	0.211	0.184	0.113	0.126	0.165	0.104	0.256
ENeRF [10]	0.055	0.110	0.076	**0.059**	0.075	0.084	0.039	0.183
GNT [14]	0.065	0.116	**0.063**	0.095	0.112	**0.025**	0.243	**0.115**
Ours	**0.035**	**0.089**	0.064	0.060	**0.064**	0.054	**0.021**	0.175
NeRF [13]	0.055	0.101	**0.047**	0.089	0.054	0.105	0.033	0.263
IBRNet$_{ft-1.0h}$ [15]	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
NeuRay$_{ft-1.0h}$ [11]	0.074	0.136	0.105	0.072	0.091	0.137	0.072	0.230
ENeRF$_{ft-1.0h}$ [10]	0.030	**0.045**	0.071	0.028	0.070	0.059	0.017	0.183
Ours$_{ft-15min}$	0.024	0.080	0.059	0.028	0.052	0.044	0.015	0.181
Ours$_{ft-1.0h}$	**0.023**	0.076	0.058	**0.026**	**0.050**	**0.043**	**0.012**	**0.179**

Table S11. **Quantitative results on the NeRF Synthetic dataset.**

Scene	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex
Metric	PSNR ↑							
PixelNeRF [19]	12.40	10.00	14.07	11.07	9.85	9.62	11.75	10.55
IBRNet [15]	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**
NeuRay [11]	21.17	**26.29**	27.98	23.91	**19.51**	**18.81**	28.92	20.55
ENeRF [10]	21.92	24.28	30.43	24.49	19.01	17.94	29.75	21.21
GNT [14]	22.21	23.56	29.16	22.80	19.18	17.43	29.35	20.15
Ours	**22.53**	25.73	**30.54**	**25.41**	19.46	18.76	**29.79**	22.05
NeRF$_{ft-10.2h}$ [13]	**23.87**	26.84	31.37	25.96	21.21	19.81	33.54	25.19
IBRNet$_{ft-1.0h}$ [15]	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
NeuRay$_{ft-1.0h}$ [11]	22.57	25.98	29.17	25.40	20.74	20.36	27.06	23.43
ENeRF$_{ft-1.0h}$ [10]	22.08	27.74	29.58	25.50	21.26	19.50	30.07	23.39
Ours$_{ft-15min}$	23.67	27.89	**31.63**	27.47	22.41	20.63	33.69	25.53
Ours$_{ft-1.0h}$	23.82	**28.09**	**31.63**	**27.66**	**22.59**	**20.80**	**33.97**	**25.54**
Metric	SSIM ↑							
PixelNeRF [19]	0.531	0.433	0.674	0.516	0.268	0.317	0.691	0.458
IBRNet [15]	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**
NeuRay [11]	0.632	0.823	0.829	0.779	0.668	0.590	0.916	0.718
ENeRF [10]	0.774	0.893	**0.948**	0.905	0.744	0.681	0.971	0.826
GNT [14]	0.736	0.791	0.867	0.820	0.650	0.538	0.945	0.744
Ours	**0.798**	**0.912**	0.947	**0.924**	**0.773**	**0.725**	**0.975**	0.848
NeRF$_{ft-10.2h}$ [13]	0.828	0.897	0.945	0.900	0.792	0.721	0.978	0.899
IBRNet$_{ft-1.0h}$ [15]	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
NeuRay$_{ft-1.0h}$ [11]	0.687	0.807	0.854	0.822	0.714	0.657	0.909	0.799
ENeRF$_{ft-1.0h}$ [10]	0.770	0.923	0.940	0.904	0.827	0.725	0.965	0.869
Ours$_{ft-15min}$	0.825	0.930	**0.963**	0.948	0.869	0.785	0.986	**0.915**
Ours$_{ft-1.0h}$	**0.829**	**0.932**	**0.963**	**0.949**	**0.873**	**0.791**	**0.987**	**0.915**
Metric	LPIPS ↓							
PixelNeRF [19]	0.650	0.708	0.608	0.705	0.695	0.721	0.611	0.667
IBRNet [15]	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
NeuRay [11]	0.257	0.162	0.163	0.225	0.253	0.283	0.136	0.254
ENeRF [10]	0.224	0.164	**0.092**	0.161	0.216	0.289	0.120	0.192
GNT [14]	0.223	0.203	0.157	0.208	0.255	0.341	**0.103**	0.275
Ours	**0.185**	**0.126**	0.101	**0.130**	**0.188**	**0.243**	0.150	**0.176**
NeRF$_{ft-10.2h}$ [13]	0.291	0.176	0.147	0.247	0.301	0.321	0.157	0.245
IBRNet$_{ft-1.0h}$ [15]	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
NeuRay$_{ft-1.0h}$ [11]	0.229	0.173	0.162	0.209	0.243	0.257	0.160	0.208
ENeRF$_{ft-1.0h}$ [10]	0.197	0.121	0.101	0.155	0.168	0.247	0.113	0.169
Ours$_{ft-15min}$	0.156	0.090	0.069	0.093	0.123	0.192	0.052	0.106
Ours$_{ft-1.0h}$	**0.151**	**0.087**	**0.068**	**0.089**	**0.116**	**0.182**	**0.051**	**0.103**

Table S12. **Quantitative results on the Real Forward-facing dataset.**

References

[1] Henrik Aanaes, Rasmus Ramsbol Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *Int. J. Comput. Vis.*, 120:153–168, 2016. 1, 2, 4, 5, 8

[2] Di Chang, Aljaž Božič, Tong Zhang, Qingsong Yan, Yingcong Chen, Sabine Süsstrunk, and Matthias Nießner. Rc-mvsnet: Unsupervised multi-view stereo with neural rendering. In *Proc. Eur. Conf. Comput. Vis.*, 2022. 3

[3] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 14124–14133, 2021. 1, 3, 5, 6, 7, 8, 9, 11, 12

[4] Yuedong Chen, Haofei Xu, Qianyi Wu, Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Explicit correspondence matching for generalizable neural radiance fields. *arXiv preprint arXiv:2304.12294*, 2023. 6, 7, 8, 9

[5] Ross Girshick. Fast r-cnn. In *Proc. IEEE Int. Conf. Comput. Vis.*, 2015. 3

[6] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 2495–2504, 2020. 3

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 5

[8] Tejas Khot, Shubham Agrawal, Shubham Tulsiani, Christoph Mertz, Simon Lucey, and Martial Hebert. Learning unsupervised multi-view stereopsis via robust photometric consistency. *arXiv preprint arXiv:1905.02706*, 2019. 3

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[10] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia Conference Proceedings*, 2022. 1, 2, 3, 5, 6, 7, 8, 9, 11, 12

[11] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2022. 3, 5, 6, 11, 12

[12] 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 Trans. Graph.*, 38(4):1–14, 2019. 1, 2, 3, 4, 5, 8

[13] 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. In *Proc. Eur. Conf. Comput. Vis.*, 2020. 1, 2, 3, 4, 5, 8, 11, 12

[14] Mukund Varma T, Peihao Wang, Xuxi Chen, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. Is attention all that neRF needs? In *Proc. Int. Conf. Learn. Repr.*, 2023. 5, 6, 11, 12

[15] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2021. 1, 5, 11, 12

[16] Xiaofeng Wang, Zheng Zhu, Guan Huang, Fangbo Qin, Yun Ye, Yijia He, Xu Chi, and Xingang Wang. Mvster epipolar transformer for efficient multi-view stereo. In *Proc. Eur. Conf. Comput. Vis.*, pages 573–591. Springer, 2022. 4

[17] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 1

[18] Zizhuang Wei, Qingtian Zhu, Chen Min, Yisong Chen, and Guoping Wang. Aa-rmvsnet adaptive aggregation recurrent multi-view stereo network. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 6187–6196, 2021. 4

[19] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2021. 5, 11, 12

[20] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 586–595, 2018. 1