Supplementary Material for RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs

Michael Niemeyer[1,2,3*] Jonathan T. Barron[3] Ben Mildenhall[3]
Mehdi S. M. Sajjadi[3] Andreas Geiger[1,2] Noha Radwan[3]
[1]Max Planck Institute for Intelligent Systems, Tübingen [2]University of Tübingen
[3]Google Research
{firstname.lastname}@tue.mpg.de {barron, bmild, msajjadi, noharadwan}@google.com
https://m-niemeyer.github.io/regnerf

Abstract

In this supplementary material, we discuss potentital negative impact of our method on society in Sec. 1. In Sec. 2, we report implementation details. We report data processing steps in Sec. 3. Finally, we report additional qualitative as well as quantitative results in Sec. 4. The ***supplementary video*** *shows animated camera trajectories for our method and baselines.*

1. Potential Negative Impact on Society

In this section we discuss the potentital negative impact this work could have on society.

Unintended Uses and Security Considerations: Our proposed method enables the synthesis of novel views from a sparse set of input images (as few as 3 images). While highly beneficial for domains where access to densely captured data is challenging, it can also lead to negative consequences when applied to sensitive or private data without the consent of the involved parties. As such, we view it as crucial for potential users to always check the data license and contact the involved parties when in doubt. Similar to content creation methods, our approach could be used to create misleading content, e.g., by selecting input views in a way that the novel views either hide content or present it in a misleading manner. To this effect, work on deep fake detection methods would provide great value to aid in identifying generated content.

Fairness Considerations: Similar to most deep learning approaches, optimizing our method requires access to AI accelerators like GPUs or TPUs and the required optimization time reduces with larger infrastructure. As a result, the amount of infrastructure or monetary support available for an organization directly affects their ability to explore certain research topics. Research work investigating the use of efficient algorithms as well as active collaboration within the broader research community are key to addressing this challenge.

Environmental Impact: Deep learning-based optimization and inference are compute intensive processes leading to high energy usage. In an attempt to curb the unnecessary energy usage, we apply high learning rates to reduce the optimization time (see Sec. 3.4 of the main paper). We regard methods that investigate finding optimal weights within a few iterations as an interesting direction for future work that further tackles this topic.

2. Implementation Details

In this section, we discuss relevant implementation details regarding our method as well as baselines.

2.1. Hyperparameters

Neural Radiance Field: We adhere to the mip-NeRF [1] implementation and parameterize our neural radiance field as a fully-connected ReLU network with 8 layers and a hidden dimension of 256. We use 128 samples along the ray for both

*The work was primarily done during an internship at Google.

	PSNR ↑		SSIM ↑		LPIPS ↓		Average ↓	
	6-view	9-view	6-view	9-view	6-view	9-view	6-view	9-view
PixelNeRF [12] (original)	17.470	17.715	0.712	0.724	0.263	0.257	0.137	0.132
PixelNeRF [12] (retrained)	19.110	20.400	0.745	0.768	0.232	0.220	0.115	0.100

Table 1. **Improving PixelNeRF's Performance.** We report object-level evaluation for PixelNeRF in the original and the re-trained setting on the DTU dataset. While in PixelNeRF [12], Yu et al. train the model with 3 input views and evaluate it with different numbers of input views, we find that the performance can be improved when re-training the model with 6 and 9 input views, respectively, for the 6 and 9 input view scenarios.

	PSNR ↑			SSIM ↑			LPIPS ↓			Average ↓		
	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view
PixelNeRF [12] (original)	16.820	19.110	20.400	0.695	0.745	0.768	0.270	0.232	0.220	0.147	0.115	0.100
PixelNeRF [12] (ft long)	18.851	18.253	19.850	0.713	0.714	0.779	0.262	0.242	0.190	0.126	0.132	0.104
PixelNeRF [12] (ft optimized)	18.945	18.699	21.831	0.710	0.720	0.781	0.269	0.237	0.203	0.125	0.125	0.090

Table 2. **Improving PixelNeRF's Performance via Finetuning on DTU.**

	PSNR ↑			SSIM ↑			LPIPS ↓			Average ↓		
	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view
PixelNeRF [12] (original)	7.927	8.735	8.613	0.272	0.280	0.274	0.682	0.676	0.665	0.461	0.433	0.432
PixelNeRF [12] (ft long)	14.828	14.752	18.174	0.380	0.418	0.520	0.546	0.527	0.425	0.253	0.253	0.176
PixelNeRF [12] (ft optimized)	16.171	17.034	18.916	0.438	0.473	0.535	0.512	0.477	0.430	0.217	0.196	0.163

Table 3. **Improving PixelNeRF's Performance via Finetuning on LLFF.**

sample levels. Our model is implemented using the JAX framework [2] on top of the mip-NeRF code base.

Optimization Curriculum: We use the Adam optimizer [8] with an exponential learning rate decay from $2 \cdot 10^{-3}$ to $2 \cdot 10^{-5}$ and 512 warm up steps [1] with a delay multiplier of 0.01. We clip gradients by value at 0.1 and then by norm at 0.1. The model is always optimized for 500 pixel epochs with a pixel batch size of 4096.

Scene Space Annealing: The scene sampling space is annealed over the first iterations for the 3 and 6 input scenarios as discussed in the main paper. More specifically, we linearly anneal the sample space over the first $N_t = 256$ iterations starting at t_m as the mid point between t_n and t_f and an initial range of $p_s = 0.5$ (see Eq. 13 of the main publication). For the LLFF dataset where NDC ray parameterization is used [1, 10], we use 512 steps starting at the far plane $t_m = t_f$ and with a small initial range of $p_s = 0.0001$.

Loss Weights: The reconstruction loss $\mathcal{L}_{\text{MSE}}$ is weighted with a value of $\lambda_{\text{MSE}} = 1.0$, the depth smoothness loss $\mathcal{L}_{\text{DS}}$ with $\lambda_{\text{DS}} = 0.1$ and the negative log-likelihood loss $\mathcal{L}_{\text{NLL}}$ with $\lambda_{\text{NLL}} = 10^{-6}$. For training stability we linearly anneal the loss weight for the depth smoothness loss λ_{DS} from a high value (400.0) to the end value of 0.1 over the first 512 optimization steps.

2.2. Baseline Implementations

PixelNeRF: For PixelNeRF [12], We use the official implementation provided by the authors and use the published pre-trained model for the 3 input view scenario. While in their publication, the authors train only one model with 3 input views and then test with varying numbers of input views, we re-train one model for each scenario leading to improved performance (see Tab. 1). We further report PixelNeRF when additionally optimized per scene ("ft" for "fine-tuned"). We find that this additional fine-tuning improves performance for the DTU dataset (see Tab. 2) as well as the LLFF dataset (see Tab. 3). We observe that optimizing per scene for the same number of iterations as our model ("PixelNeRF ft (long)") leads to overfitting and we hence report PixelNeRF ft for the optimized iteration number of 20K ("PixelNeRF ft (optimized)") similar to [7].

MVSNeRF: For MVSNeRF [3], we similarly use the implementation provided by the authors and use the published pre-trained model. We observe that for sparse input scenarios the official fine-tuning curriculum, which is optimized for fine-tuning on many input views, can be improved by reducing the learning rate to 10^{-5} ("Improved Curriculum 1") as well as 10^{-6} ("Improved Curriculum 2") (see Tab. 4). To ensure a fair comparison, we report results obtained with the top-performing

	PSNR ↑			SSIM ↑			LPIPS ↓			Average ↓		
	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view
MVSNeRF [3] (Official Curriculum)	16.049	17.329	17.055	0.456	0.544	0.554	0.452	0.374	0.376	0.208	0.175	0.177
MVSNeRF [3] (Improved Curriculum 1)	17.652	19.694	20.089	0.589	0.654	0.667	0.329	0.280	0.272	0.160	0.128	0.120
MVSNeRF [3] (Improved Curriculum 2)	17.875	19.986	20.466	0.584	0.660	0.695	0.327	0.264	0.244	0.157	0.122	0.111

Table 4. **Improving MVSNeRF's Finetuning Curriculum on LLFF.**

	PSNR ↑	SSIM ↑	LPIPS ↓
DietNeRF [6]	23.147	0.866	0.109
DietNeRF LMSE ft [6]	23.591	0.874	0.097
DietNeRF (our implementation)	24.345	0.880	0.100

Table 5. **DietNeRF's Improved Performance on the Blender Dataset.**

curriculum in the main paper.

Stereo Radiance Fields: We use the official implementation of Stereo Radiance Fields [4] and train one model for each of the 3, 6, and 9 input view scenarios on the DTU dataset. For fine-tuning, we adhere to the published curriculum.

DietNeRF: As no official code is available for DietNeRF [6], we implement the proposed semantic consistency loss ourselves on top of the mip-NeRF code base. To validate our implementation, we compare results to the published table on the synthetic Blender dataset [10] and find that our reimplementation leads to an improvement in the performance compared to the published results (see Tab. 5).

2.3. Geometry Regularization

In Tab. (4) of the main publication, we report an ablation study of various geometry regularization techniques which we describe more detailed in the following.

Opacity Regularization: The opacity regularization [9] is defined as

$$\mathcal{L}_{\text{opacity}}(\theta, \mathcal{R}_r) = \sum_{\mathbf{r} \in \mathcal{R}_r} \log\left(\hat{\alpha}_\theta(\mathbf{r})\right) + \log\left(1 - \hat{\alpha}_\theta(\mathbf{r})\right) \tag{1}$$

where $\hat{\alpha}_\theta(\mathbf{r})$ indicates the accumulated alpha weight for ray $\mathbf{r}$.

Ray Density Entropy: The ray density entropy regularization we investigate is

$$\mathcal{L}_{\text{RDE}}(\theta, \mathcal{R}_r) = \sum_{\mathbf{r} \in \mathcal{R}_r} \sum_{i=1}^{N_s} -\frac{w_\theta^i}{\delta_i} \log\left(\frac{w_\theta^i}{\delta_i}\right) \qquad \text{where} \qquad w_\theta^i = T(i)(1 - \exp(-\sigma_\theta^i \delta_i)) \tag{2}$$

and $N_s = 128$ are the number of ray samples. We drop the dependency of $\mathbf{r}$ for w_θ^i and δ_i to avoid cluttered notation.

Normal Smoothness: The normal smoothness prior [11] is defined as

$$\mathcal{L}_{\text{NS}}(\theta, \mathcal{R}_r) = \sum_{\mathbf{r} \in \mathcal{R}_r} ||\mathbf{n}(\mathbf{x}_r) - \mathbf{n}(\mathbf{x}_r + \epsilon)||_2^2 \tag{3}$$

where $\mathbf{n}(\cdot)$ denotes the normal vector, $\mathbf{x}_r$ a sample point along ray $\mathbf{r}$ and ϵ a noise vector drawn from a Gaussian.

Density Surface Regularization: We enforce solid surfaces along the ray using

$$\mathcal{L}_{\text{SR}}(\theta, \mathcal{R}_r) = \sum_{\mathbf{r} \in \mathcal{R}_r} \sum_{i=1}^{N_s} \log(v_\theta^i) + \log(1 - v_\theta^i) \qquad \text{where} \qquad v_\theta^i = 1 - \exp(-\sigma_\theta^i \delta_i) \tag{4}$$

where $N_s = 128$ are the number of ray samples. Similar to before, we drop the dependency of v_θ^i on $\mathbf{r}$ for clarity.

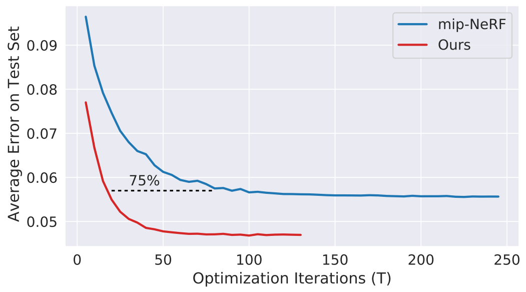


Figure 1. **Optimization Progress.** Our model requires 75% fewer iterations to match mip-NeRF's test error on DTU (9 input views).

	Setting	PSNR ↑			SSIM ↑			LPIPS ↓			Average ↓		
		3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view
SRF [4]	Trained on DTU	15.84	17.77	18.56	0.532	0.616	0.652	0.482	0.401	0.359	0.207	0.162	0.145
PixelNeRF [12]		18.74	21.02	22.23	0.618	0.684	0.714	0.401	0.340	0.323	0.154	0.119	0.105
MVSNeRF [3]		16.33	18.26	20.32	0.602	0.695	0.735	0.385	0.321	0.280	0.184	0.146	0.114
SRF ft [4]	Trained on DTU and Optimized per Scene	16.06	18.69	19.97	0.550	0.657	0.678	0.431	0.353	0.325	0.196	0.143	0.125
PixelNeRF ft [12]		17.38	21.52	21.67	0.548	0.670	0.680	0.456	0.351	0.338	0.185	0.121	0.117
MVSNeRF ft [3]		16.26	18.22	20.32	0.601	0.694	0.736	0.384	0.319	0.278	0.185	0.146	0.113
mip-NeRF [1]	Optimized per Scene	7.64	14.33	20.71	0.227	0.568	0.799	0.655	0.394	0.209	0.485	0.231	0.098
DietNeRF [6]		10.01	18.70	22.16	0.354	0.668	0.740	0.574	0.336	0.277	0.383	0.149	0.098
Ours		15.33	19.10	22.30	0.621	0.757	0.823	0.341	0.233	0.184	0.189	0.118	0.079

Table 6. **Quantitative Results on Full Images for DTU.**

Sparsity Regularization: The sparsity regularization [5] is defined as

$$\mathcal{L}_{\text{SR}}(\theta, \mathcal{R}_r) = \sum_{\mathbf{r}\in\mathcal{R}_r} \sum_{i=1}^{N_s} \log\left(1 + \frac{{\sigma_\theta^i}^2}{c}\right) \qquad \text{where} \qquad c = 0.5 \tag{5}$$

and, similar to before, $N_s = 128$ are the number of ray samples.

3. Data

We report the test set selection procedure for the DTU and LLFF dataset in this section.

DTU Dataset: We adhere to the evaluation protocol from [12] and use the following scan IDs as the test set: 8, 21, 30, 31, 34, 38, 40, 41, 45, 55, 63, 82, 103, 110, 114. The following image IDs (starting with "0"): 25, 22, 28, 40, 44, 48, 0, 8, 13 are used as input. For the 3 and 6 input scenarios, we use the first 3/6 image IDs, respectively. For evaluation, the remaining images are used with the exception of the following image IDs due to wrong exposure: 3, 4, 5, 6, 7, 16, 17, 18, 19, 20, 21, 36, 37, 38, 39. We use an image resolution of 300×400 pixels.

LLFF: For the LLFF dataset, we use every 8-th image as the holdout test set similar to previous works [10]. As input views, image IDs are selected evenly across the remaining views. We use an image resolution of 378×504.

Masked Evaluation: Due to the background evaluation bias, we propose to perform masked evaluation for DTU (see Fig. 4 of the main publication). More specifically, we use object masks and calculate PSNR only within the mask. For SSIM and LPIPS, we use the masks to composite the predicted object-of-interest onto white background before calculating the metrics.

4. Experiments

In this section, we report additional quantitative as well as qualitative results.

Optimization Convergence: In Fig. 1 we show average test set error for mip-NeRF and our method over the optimization progress on DTU (9 input views). Our model requires 75% fewer iterations to achieve mip-NeRF's top 5% performance.

Additional Quantitative Results: In Tab. 6 we report quantitative results for the full images on the DTU dataset. As discussed in the main publication, full image evaluation leads to a bias preferring correct background prediction over the

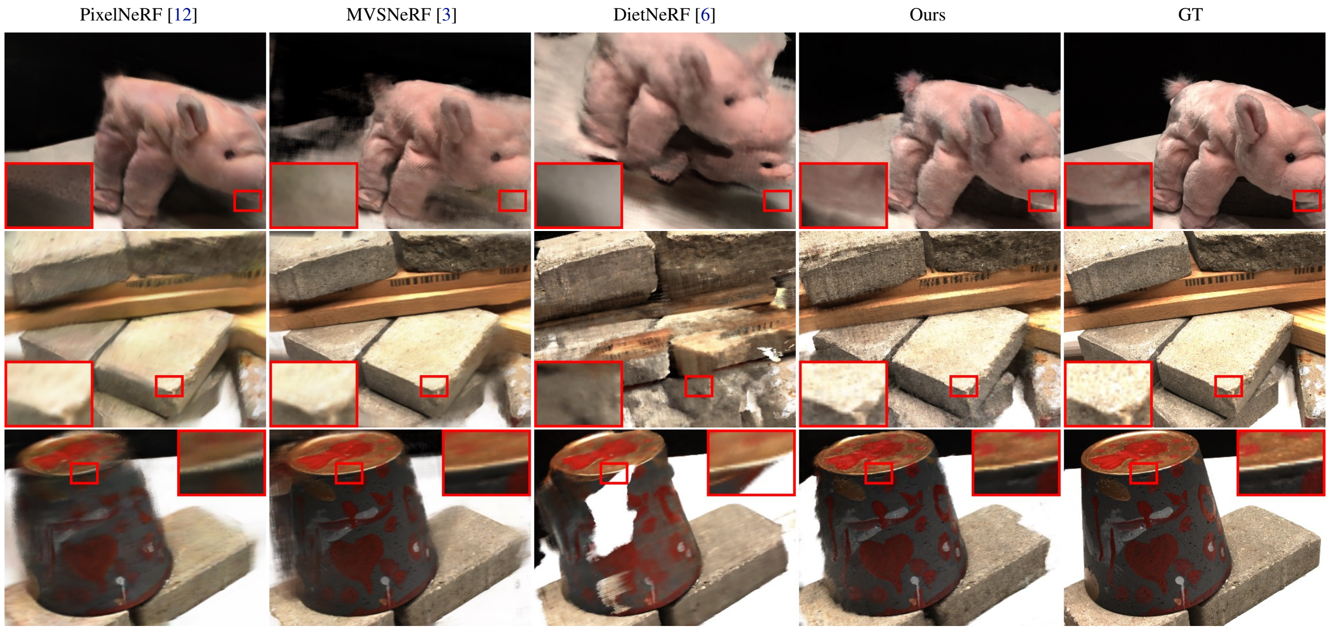


Figure 2. **Additional Qualitative Results on DTU for 3 Input Views.**

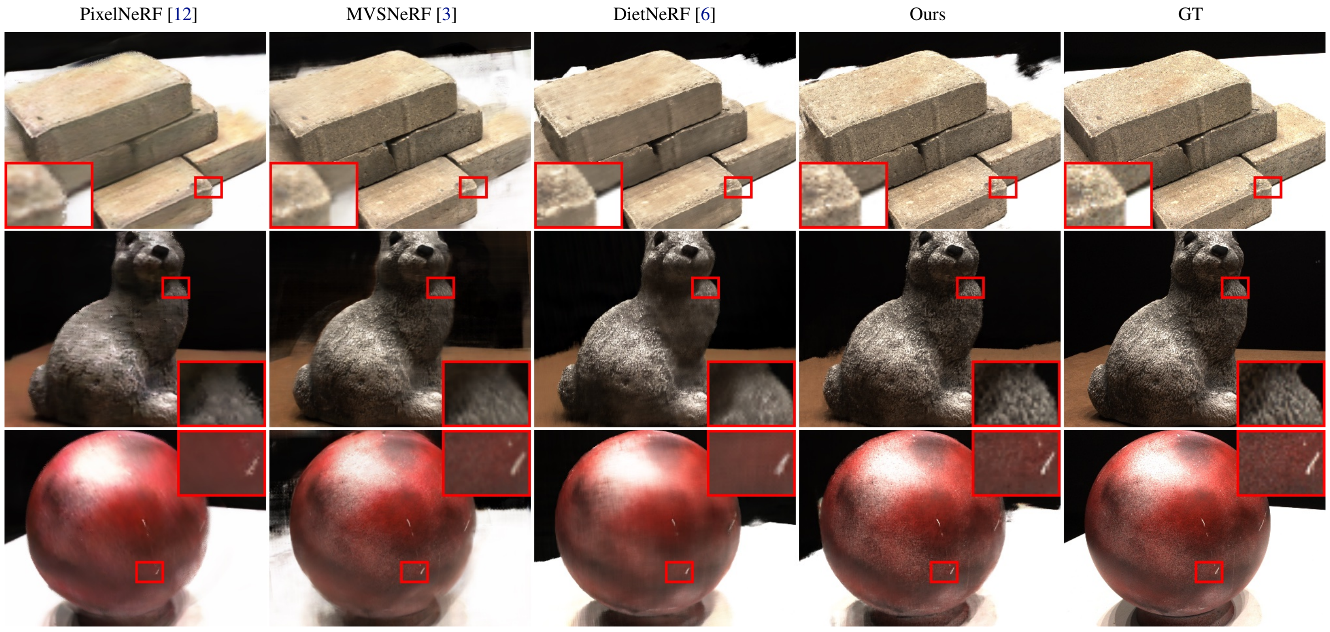


Figure 3. **Additional Qualitative Results on DTU for 6 Input Views.**

object-of-interest (see Fig. 4 of the main publication). In the main paper, we avoid this bias by applying object masks to the images before evaluation.

Additional Qualitative Results: We show additional qualitative comparisons in Fig. 2, Fig. 3, Fig. 5, and Fig. 6 for the DTU dataset and in Fig. 7, Fig. 8, Fig. 9, and Fig. 10 for the LLFF dataset. Further, we report multiple novel views for all scenes in Fig. 11, Fig. 12, and Fig. 13 for DTU and in Fig. 14, Fig. 15, and Fig. 16 for LLFF.

PixelNeRF [12] MVSNeRF [3] DietNeRF [6] Ours GT

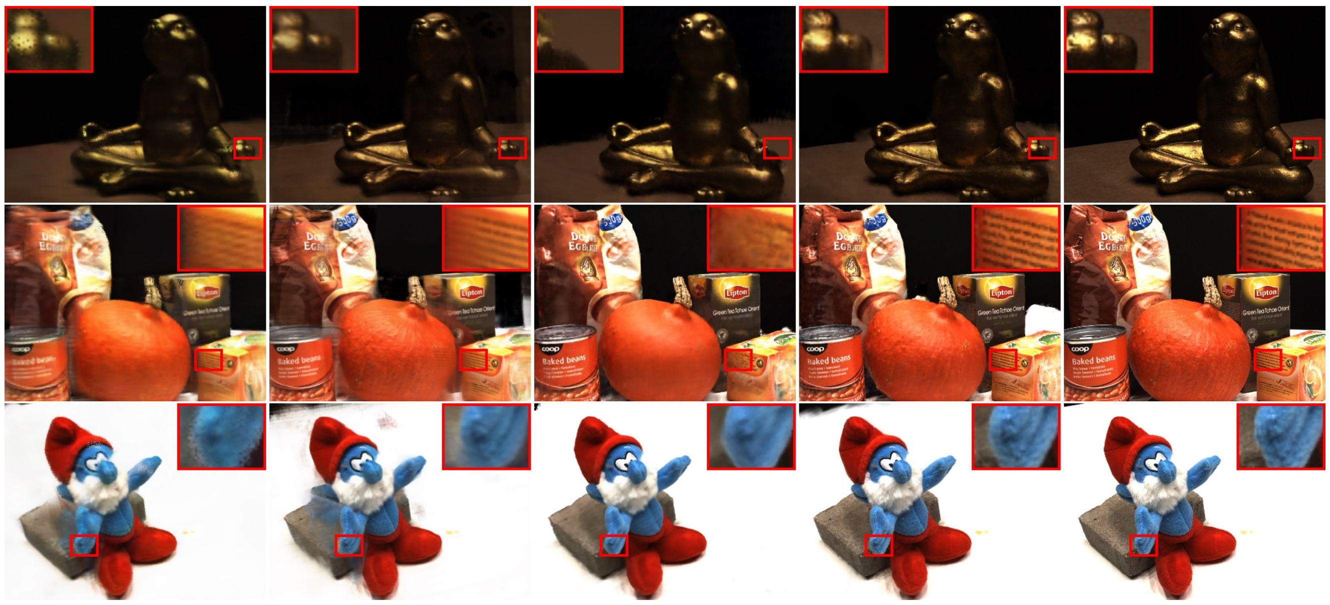
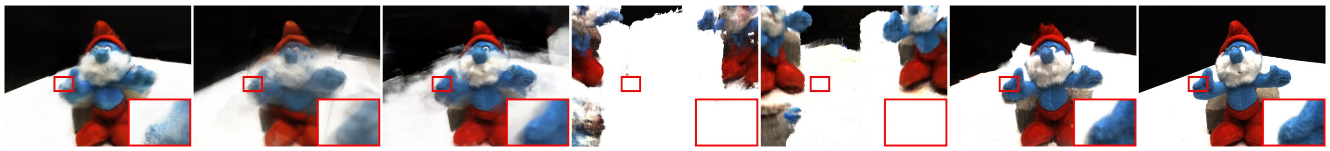


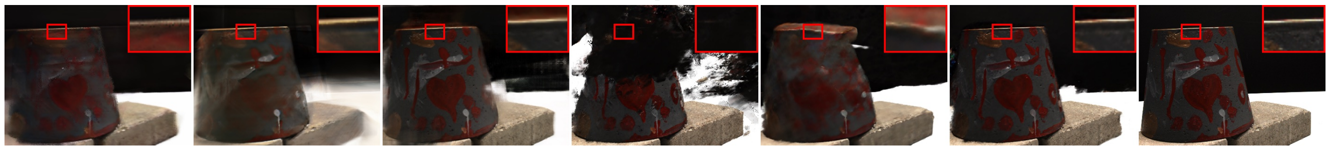
Figure 4. 9 Input Views

Figure 5. **Additional Qualitative Results on DTU for 9 Input Views.**

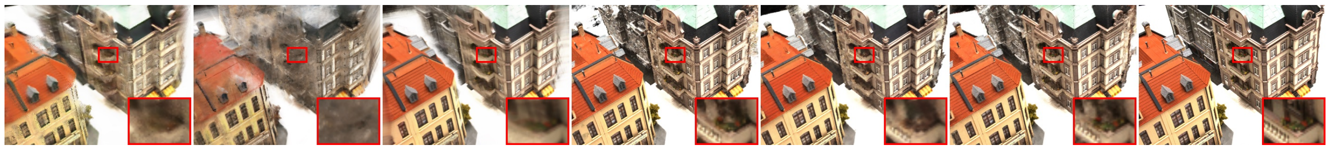
PixelNeRF [12] SRF [4] MVSNeRF [3] mip-NeRF [1] DietNeRF [6] Ours GT



(a) 3 Input Views



(b) 6 Input Views



(c) 9 Input Views

Figure 6. **Additional Qualitative Comparison on DTU.**

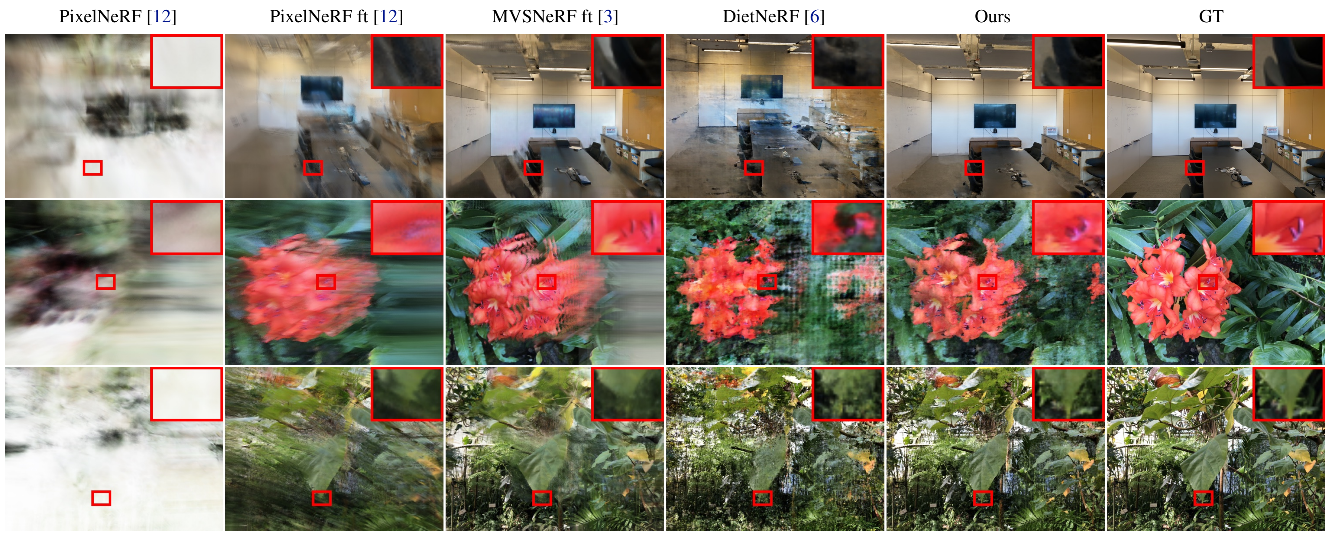


Figure 7. **Additional Qualitative Results on LLFF for 3 Input Views.**

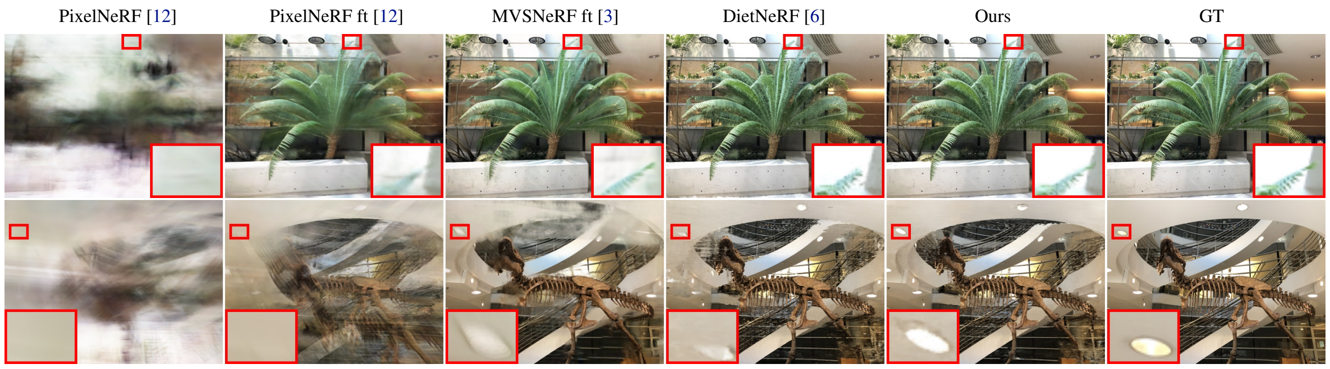


Figure 8. **Additional Qualitative Results on LLFF for 6 Input Views.**

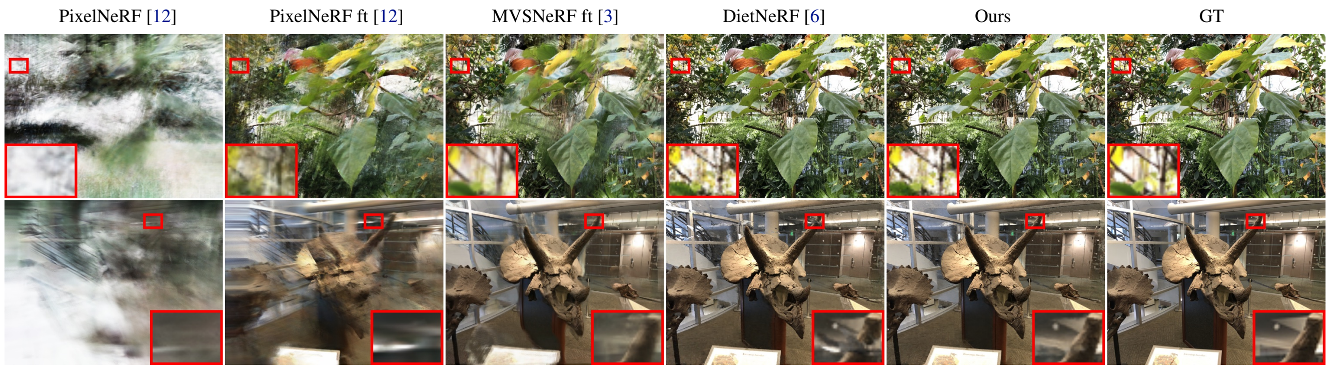
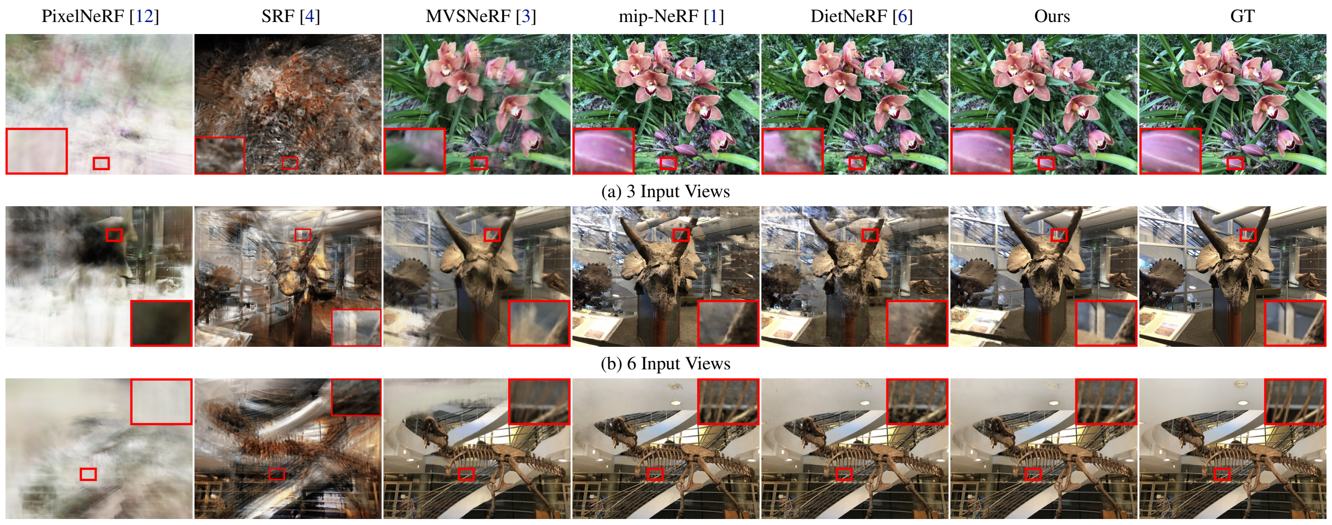


Figure 9. **Additional Qualitative Results on LLFF for 9 Input Views.**



(a) 3 Input Views

(b) 6 Input Views

(c) 9 Input Views

Figure 10. **Additional Qualitative Comparison on LLFF.**

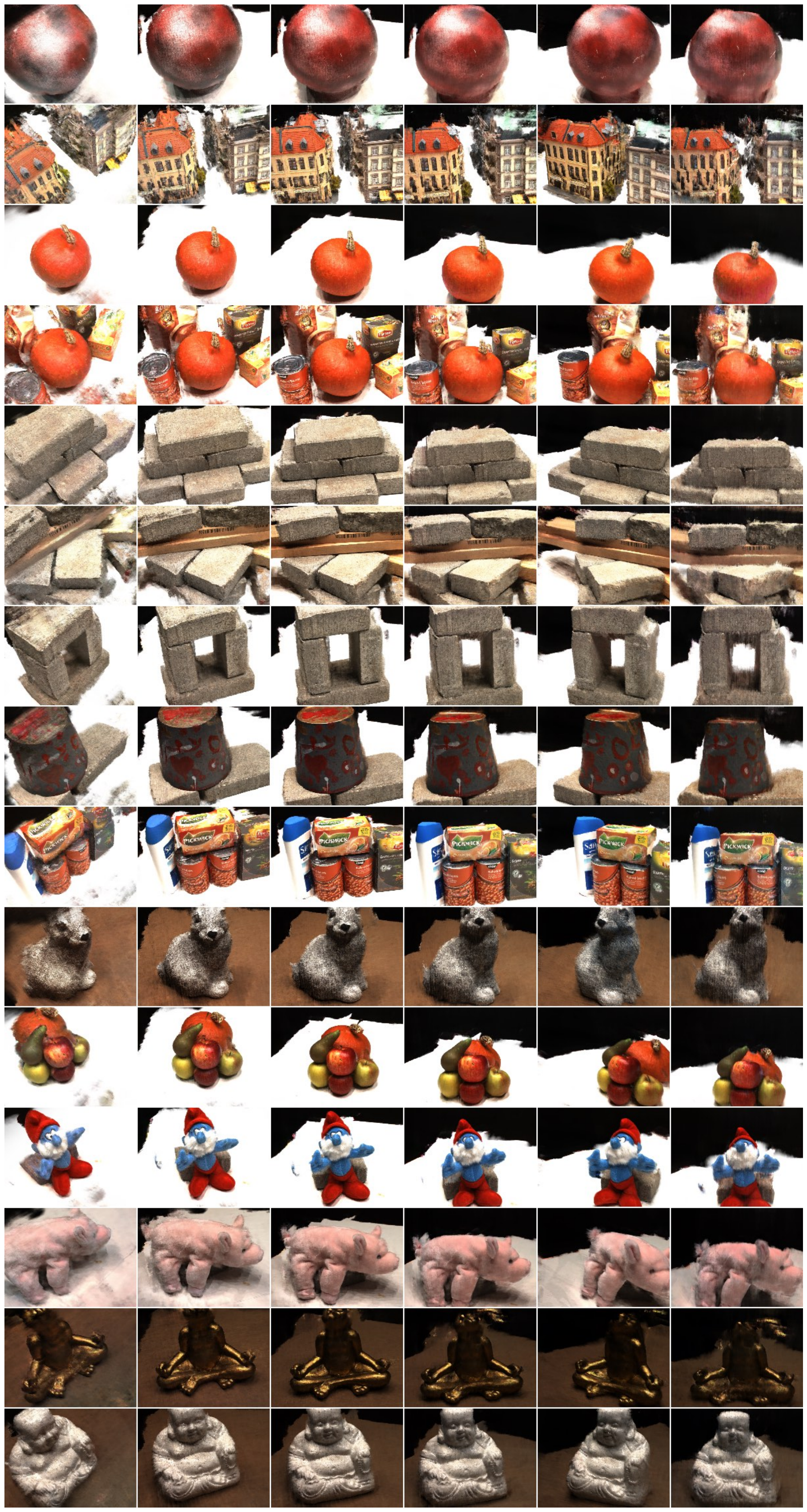


Figure 11. **Synthesized Novel Views for our Method with 3 Input Views on DTU.**

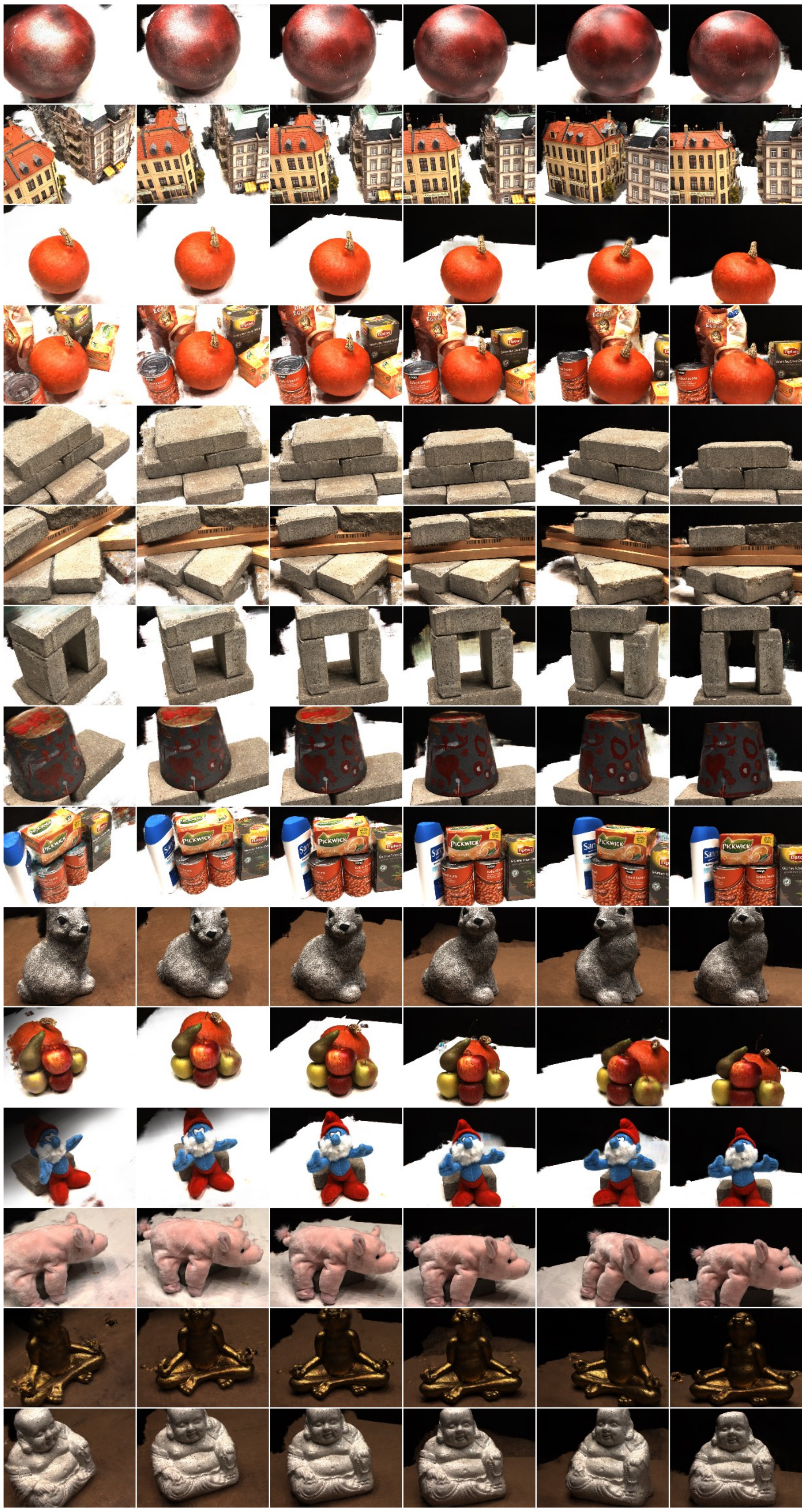


Figure 12. **Synthesized Novel Views for our Method with 6 Input Views on DTU.**

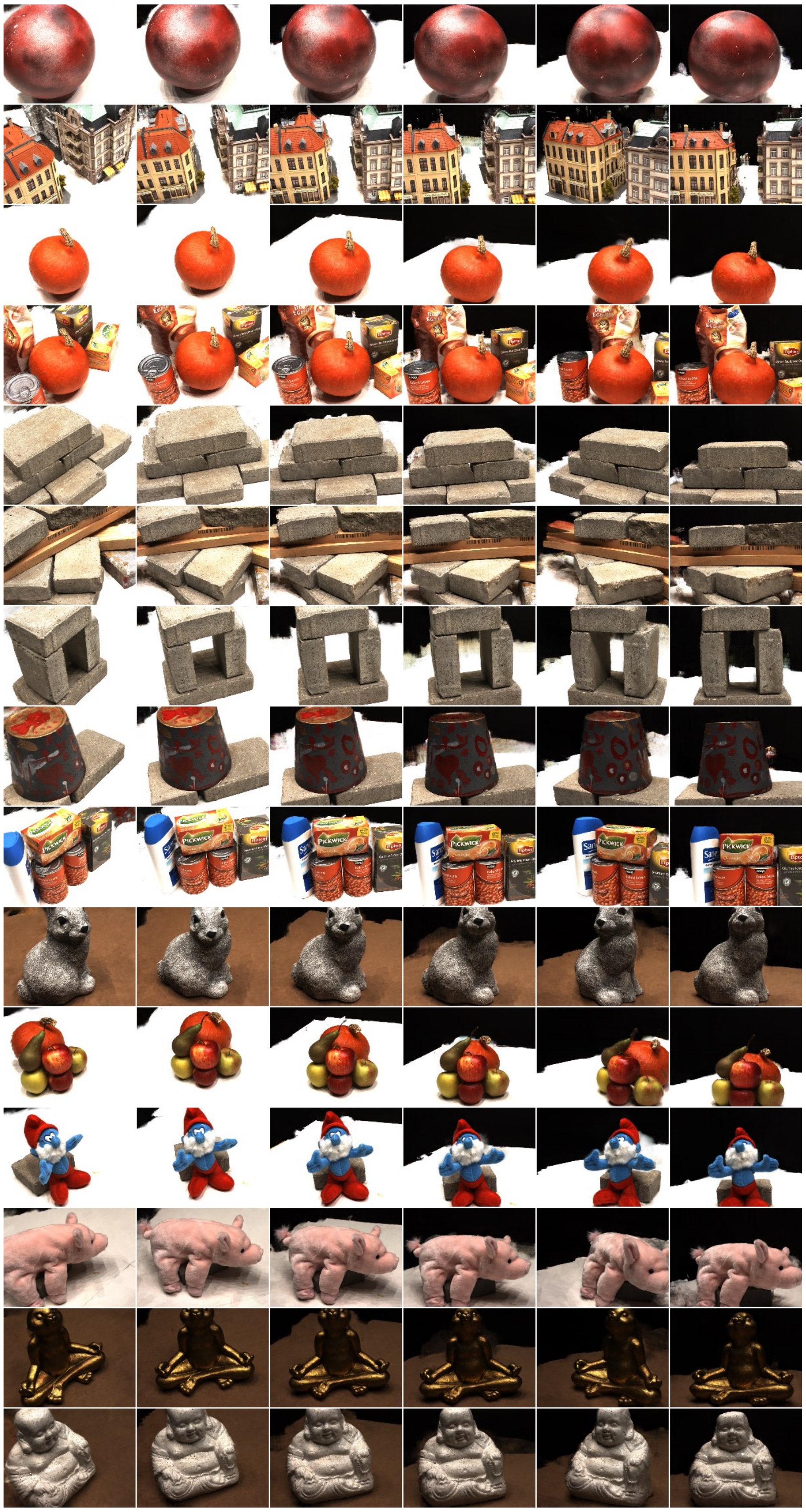


Figure 13. **Synthesized Novel Views for our Method with 9 Input Views on DTU.**

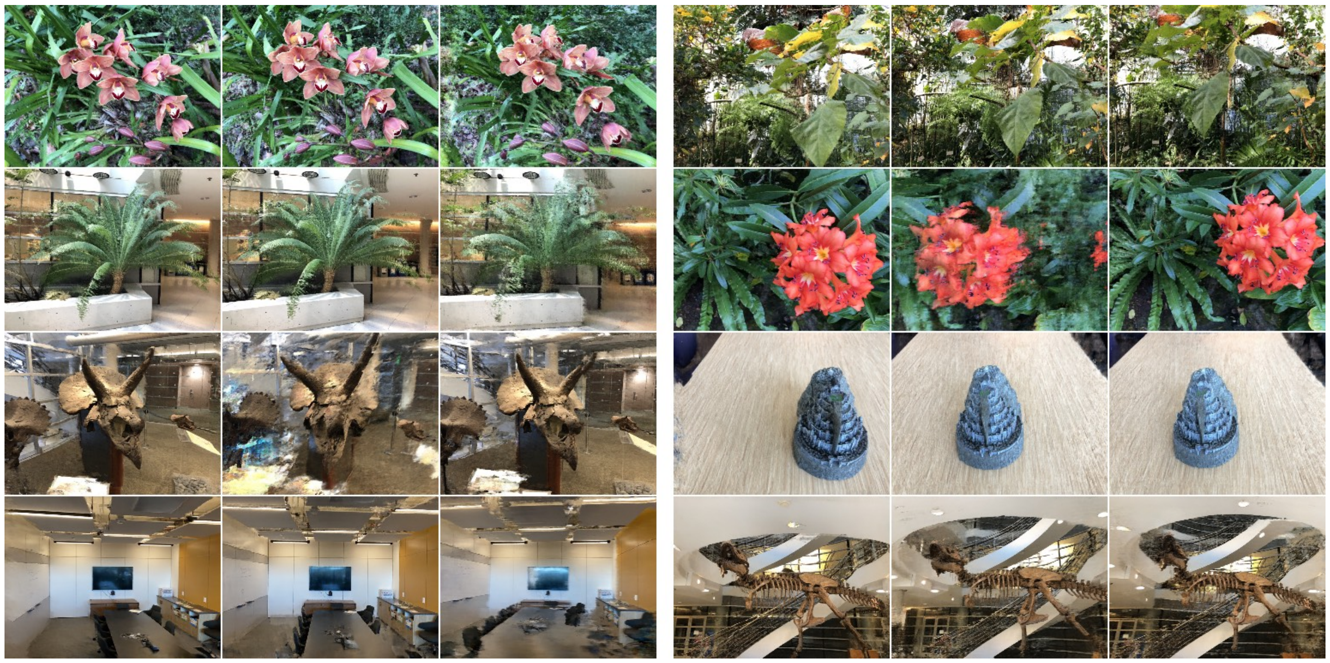


Figure 14. **Synthesized Novel Views for our Method with 3 Input Views on LLFF.**

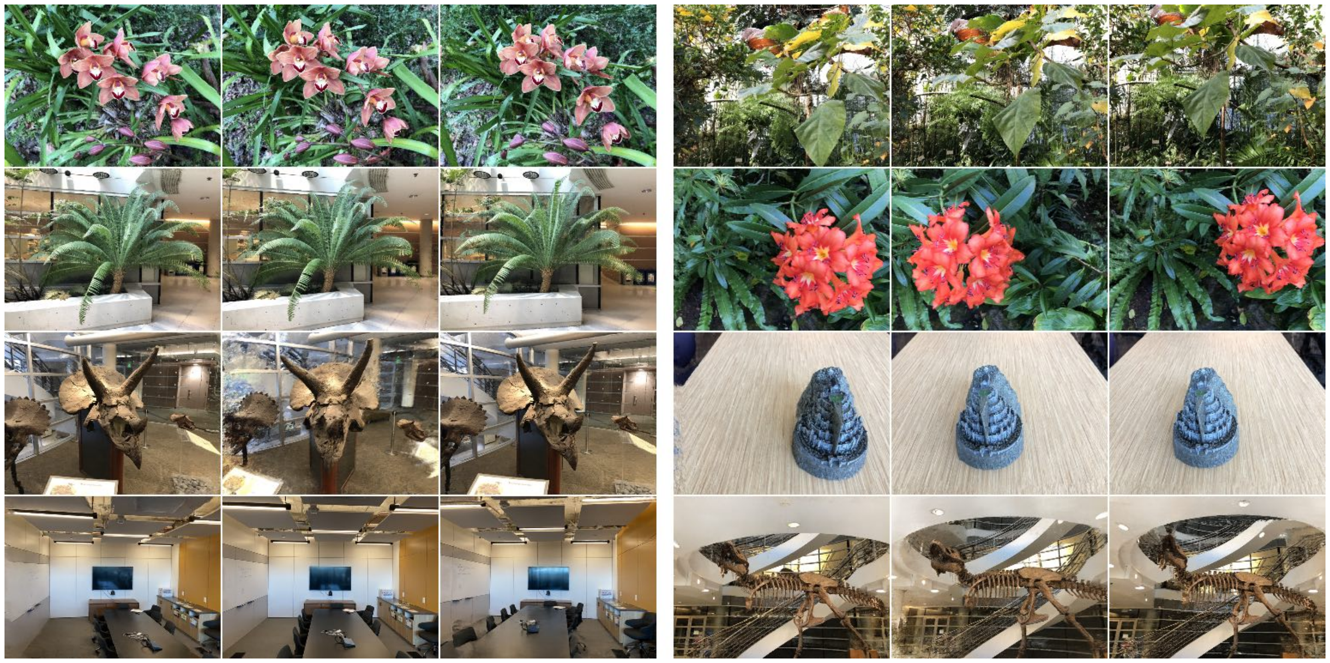


Figure 15. **Synthesized Novel Views for our Method with 6 Input Views on LLFF.**

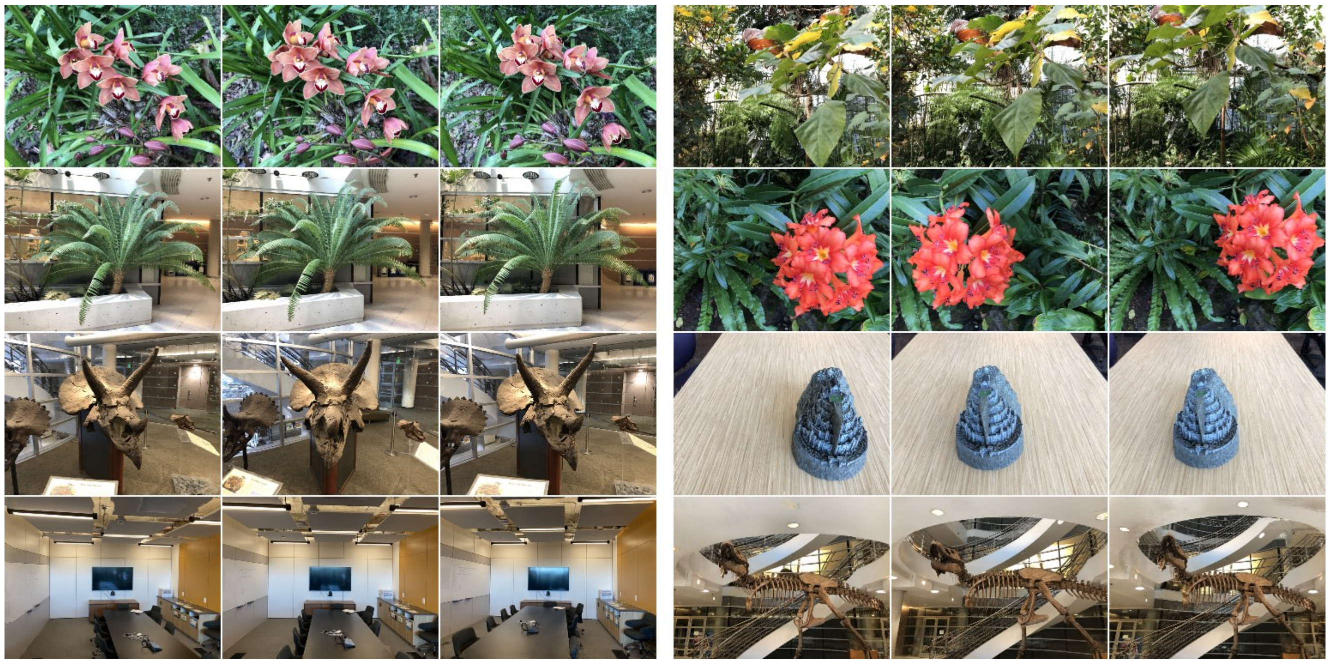


Figure 16. **Synthesized Novel Views for our Method with 9 Input Views on LLFF.**

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 *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 1, 2, 4, 6, 8

[2] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. 2

[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. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2, 3, 4, 5, 6, 7, 8

[4] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (SRF): learning view synthesis for sparse views of novel scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3, 4, 6, 8

[5] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 4

[6] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 3, 4, 5, 6, 7, 8

[7] Jun-Yan Zhu Kangle Deng, Andrew Liu and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv.org*, 2021. 2

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 2

[9] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. In *ACM Trans. on Graphics*, 2019. 3

[10] 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. of the European Conf. on Computer Vision (ECCV)*, 2020. 2, 3, 4

[11] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[12] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 4, 5, 6, 7, 8