OmniLocalRF: Omnidirectional Local Radiance Fields from Dynamic Videos

Supplementary Material

1. Implementation Details

We schedule learning rate and upsampling based on frames per RF block to balance iteration over blocks, following LocalRF [7].

1.1. Regularizers for Motion Mask

We employ a total variation loss $\mathcal{L}_{\text{TV}}^{\text{mask}}$ and a binary entropy [11] like function $\mathcal{L}_{\text{bin}}^{\text{mask}}$ to regularize the motion mask. Total variation is commonly utilized for smoothing a target and we adapt it as follows:

$$\mathcal{L}_{\text{TV}}^{\text{mask}} = \Delta^2 M_x + \Delta^2 M_y + 0.1 \cdot \Delta^2 M_t, \tag{1}$$

where M_x, M_y and M_t denote the predicted mask value along the x, y and temporal axes. Here, Δ^2 represents a squared difference between neighboring values along the axes. Considering object movement over time, we impose a relatively small weight along the temporal axis. Additionally, we enforce the mask to converge into a binary value using

$$\mathcal{L}_{\text{bin}}^{\text{mask}} = M(r)^2 (1 - M(r))^2. \tag{2}$$

The function exhibits a similar shape to the binary entropy regularizer, used as a slightly modified form in D^{2}NeRF [10] for the same purpose.

1.2. Optimization Details

In our method, we insert frames every 100 iterations. The insertion process stops when the distance between the inserted frames and the center of the current radiance block exceeds 1. This distance is equivalent to the radius of the contraction fields. After that, we optimize the blocks until the iteration reaches the number of frames in the current block multiplied by 600. The resolution of RF blocks increases exponentially from 64^3 to 640^3 at iterations $[100, 150, 200, 250, 300]$ times the number of frames in the block. Once we optimize the block, a new radiance block is created to overlap frames with the previous block for further optimization.

For pose estimation, we follow the camera parameterization of LocalRF. The rotation matrix comprises two perpendicular vectors with their cross-product, and the translation matrix is defined in Euclidean space. During training, the camera pose is updated by photometric error while training RF blocks. Test views are solely used to optimize the pose without contributing to the training of RF blocks. We train our model for each scene using 155K iterations. Our method utilizes adaptive sampling, as proposed in TensoRF [2]. The number of samples exponentially increases from 219 to 2,214 at the upsampling iteration. We initiate a backward step after completing the upsampling of the current radiance block. We do not perform a backward step during frame insertion as it could contaminate the previous radiance fields due to the inaccurate pose of the current frames.

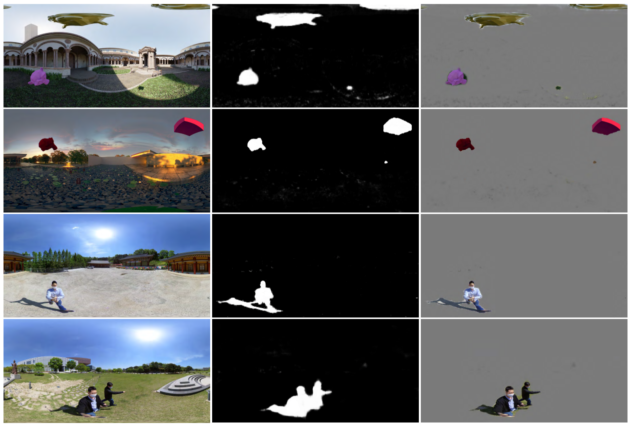


Figure 1. Predicted motion mask results. The second column is estimation results from our mask module, and the last column shows blended results with the input images.

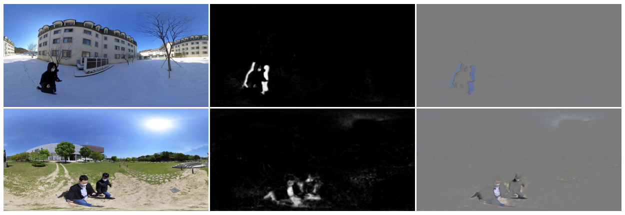


Figure 2. Predicted mask results without photometric supervision of the ground truth. Unresolving factorization ambiguity often leads to an intermediate alpha value of the motion mask, leaving dynamic artifacts on the static geometry.

1.3. Hyperparameters

We set the volumetric density channel to 8 for the RF block's volumetric density channel and 24 for its color channel. For both the MLP used to decode color from radiance blocks and the MLP used to estimate dynamic components, we use 2 layers with 128 hidden nodes. We adopt the Adam optimizer [4] with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ to train all parameters. To account for the distinct convergence speeds of each parameter, we use different learning rate schedules. The learning rate for radiance blocks and

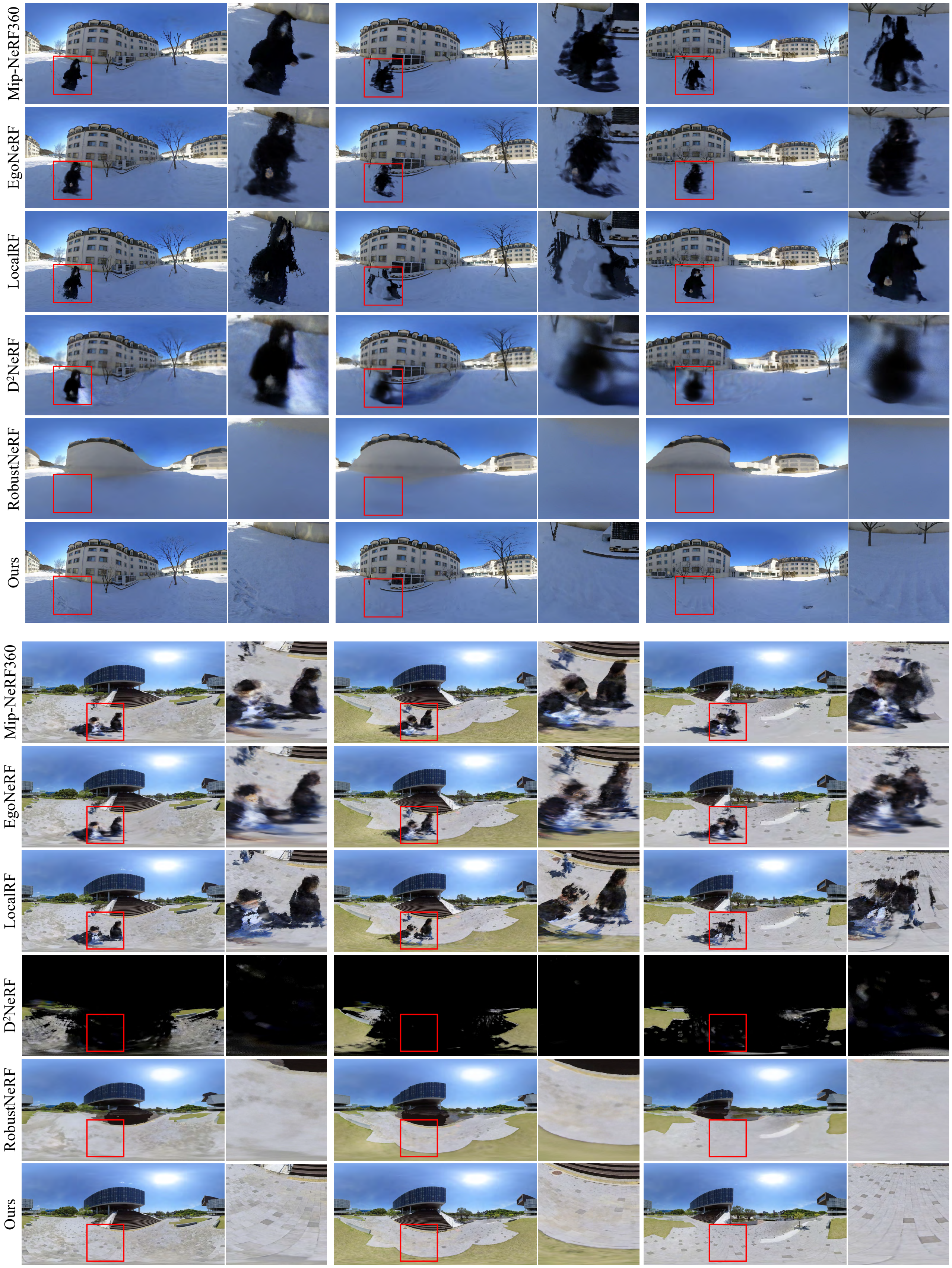


Figure 3. Qualitative comparisons of the *Dormitory* and *Library* scenes in the real dataset

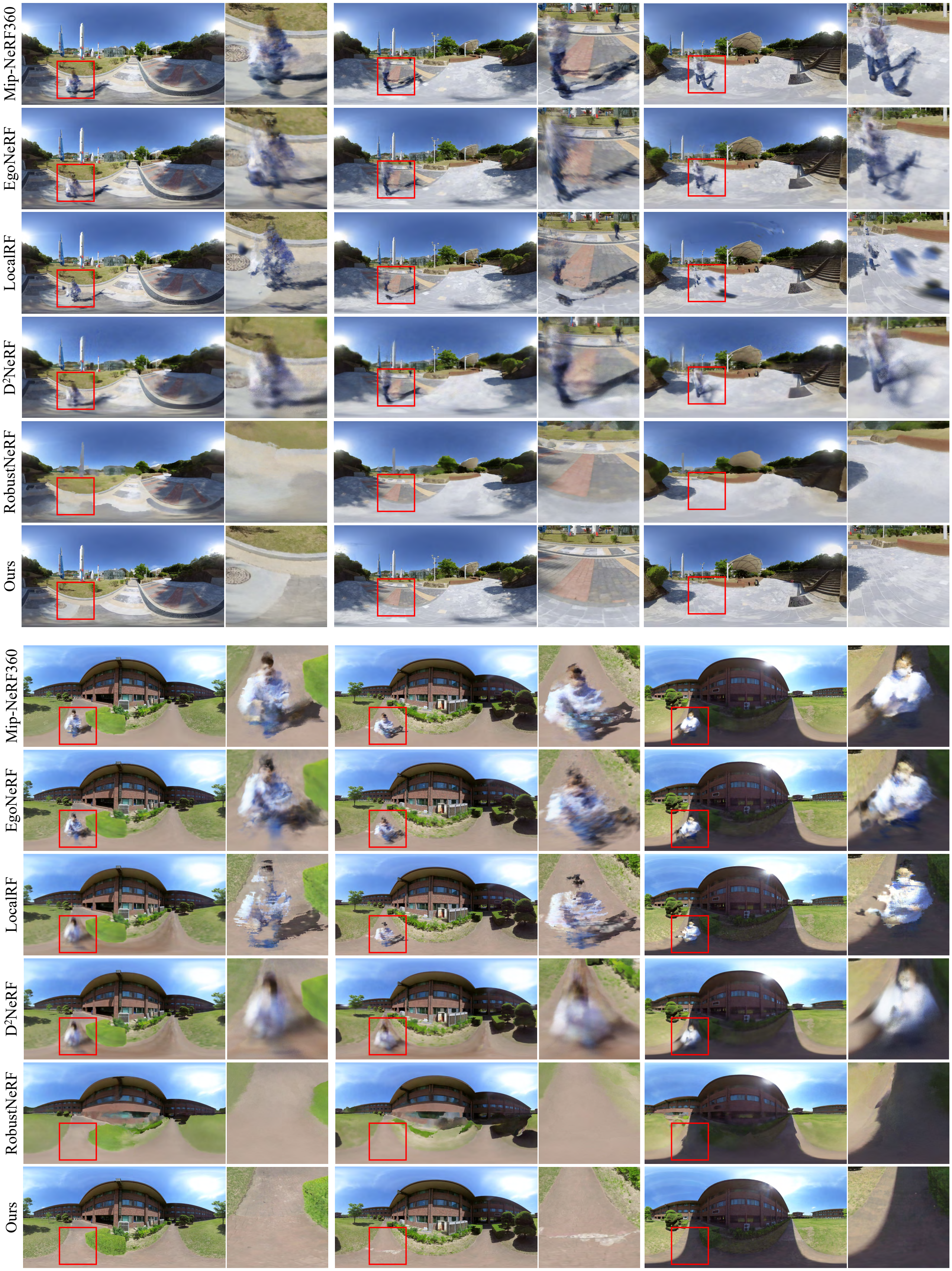


Figure 4. Qualitative comparisons of the *Rocket* and *Red Building* scenes in the real dataset

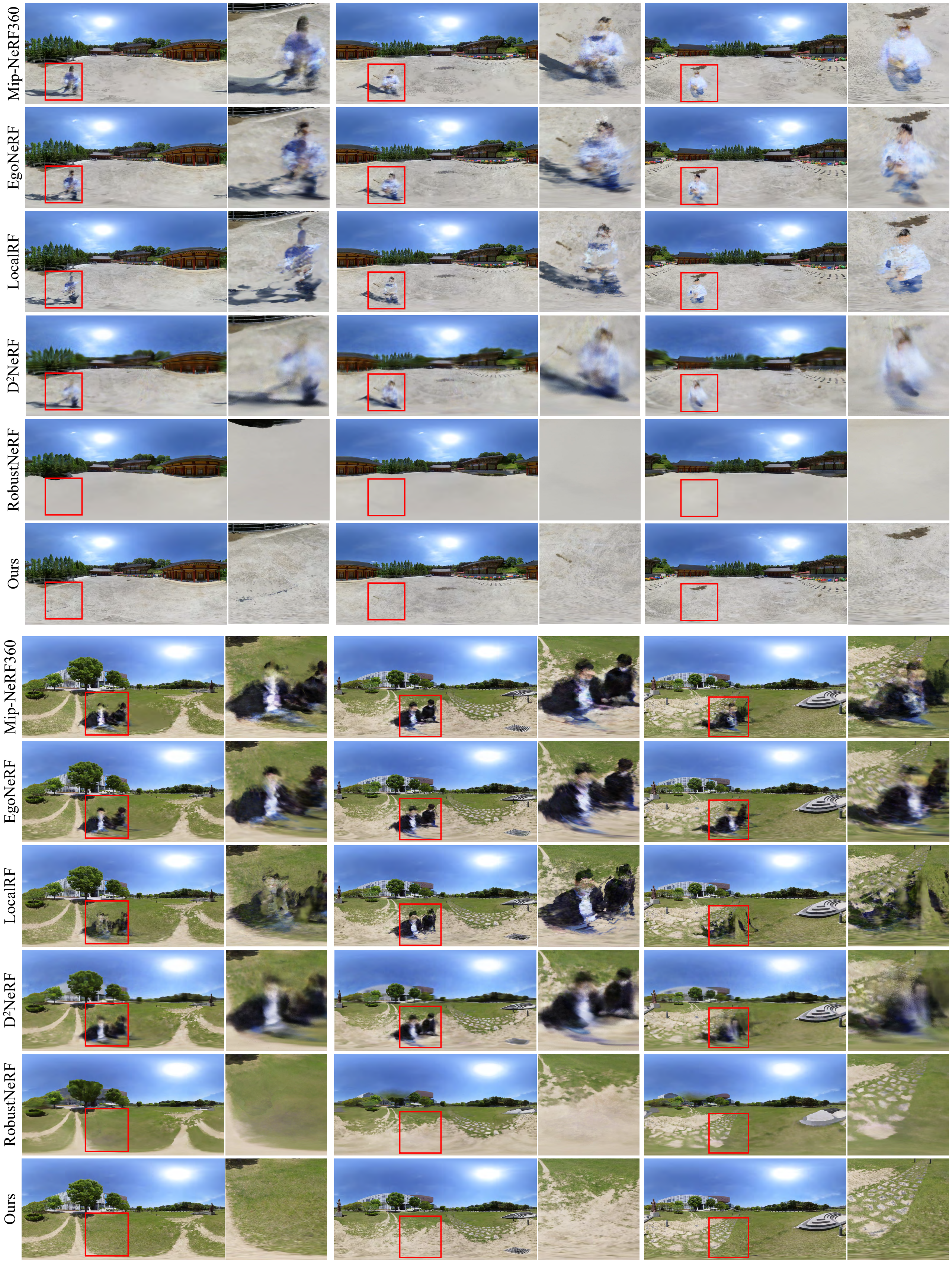


Figure 5. Qualitative comparisons of the *Temple* and *Yongsil* scenes in the real dataset

Table 1. Quantitative comparisons of the real dataset. Refer to Figures 3, 4 and 5 for qualitative comparisons.

			Dormitory					*Library*					*Red building*		
	PSNR	PSNR^{WS}	SSIM	SSIM^{WS}	LPIPS	PSNR	PSNR^{WS}	SSIM	SSIM^{WS}	LPIPS	PSNR	PSNR^{WS}	SSIM	SSIM^{WS}	LPIPS
Mip-NeRF360 [1]	25.95	24.70	0.8564	0.8220	0.3837	28.02	27.75	0.8485	0.8489	0.3280	26.60	25.77	0.7998	0.7672	0.3719
EgoNeRF [3]	26.11	24.80	0.8404	0.7988	0.4190	27.05	26.49	0.7977	0.7884	0.4141	25.55	24.64	0.7557	0.7078	0.4568
LocalRF [7] w/ pose	27.20	26.12	0.8826	0.8589	0.3180	27.72	27.47	0.8547	0.8540	0.2930	25.68	24.90	0.7860	0.7492	0.3743
D^{2}NeRF [10]	24.96	21.72	0.7887	0.731	0.4687	4.44	4.90	0.0733	0.0779	0.7084	25.55	25.16	0.7487	0.7056	0.4441
RobustNeRF [9]	19.44	17.98	0.794	0.7330	0.4850	20.78	19.23	0.7338	0.6908	0.4738	19.91	18.49	0.7098	0.6409	0.4974
Ours w/ pose	29.10	27.86	0.8956	0.8748	0.2992	29.61	28.81	0.8544	0.8535	0.2970	26.03	25.18	0.8010	0.7641	0.3534
LocalRF [7] wo/ pose	26.59	25.48	0.8762	0.8493	0.3227	27.70	27.35	0.8399	0.8408	0.3096	26.56	25.75	0.8040	0.7728	0.3452
Ours wo/ pose	29.23	27.99	0.8971	0.8767	0.2977	29.69	28.90	0.8540	0.8538	0.2984	26.00	25.24	0.8054	0.7713	0.3480

			Rocket					*Temple*					*Yongsil*		
	PSNR	PSNR^{WS}	SSIM	SSIM^{WS}	LPIPS	PSNR	PSNR^{WS}	SSIM	SSIM^{WS}	LPIPS	PSNR	PSNR^{WS}	SSIM	SSIM^{WS}	LPIPS
Mip-NeRF360 [1]	27.92	27.47	0.8531	0.8407	0.3457	27.19	27.41	0.7458	0.7554	0.3825	25.57	25.55	0.7531	0.7522	0.3393
EgoNeRF [3]	26.82	26.07	0.8085	0.7857	0.4393	26.05	26.04	0.6930	0.6961	0.4699	24.14	24.21	0.6704	0.6781	0.4310
LocalRF [7] w/ pose	27.14	26.61	0.8301	0.8171	0.3959	27.08	27.45	0.7559	0.7724	0.3537	24.56	24.76	0.7159	0.7279	0.3475
D^{2}NeRF [10]	25.17	24.22	0.7576	0.7225	0.4560	23.78	22.63	0.6398	0.6108	0.5370	23.81	23.39	0.6548	0.6495	0.4456
RobustNeRF [9]	22.05	20.89	0.7633	0.7214	0.4741	22.38	21.52	0.6456	0.6259	0.5113	20.15	19.25	0.6091	0.5953	0.4768
Ours w/ pose	28.35	27.38	0.8534	0.8359	0.3360	27.66	27.63	0.7549	0.7663	0.3603	25.56	25.70	0.7430	0.7566	0.3336
LocalRF [7] wo/ pose	27.22	26.88	0.8534	0.8396	0.3320	26.55	26.82	0.7286	0.7462	0.3937	24.72	25.11	0.7185	0.7418	0.3433
Ours wo/ pose	28.38	27.42	0.8544	0.8369	0.3328	27.57	27.52	0.7494	0.7600	0.3653	25.53	25.72	0.7387	0.7540	0.3359

Table 2. Quantitative comparisons of the synthetic video dataset where dynamic objects are inserted. Refer to Figure 6 for qualitative comparisons.

			Sponza					*Pavillion*					*Lone monk*		
	PSNR	PSNR^{WS}	SSIM	SSIM^{WS}	LPIPS	PSNR	PSNR^{WS}	SSIM	SSIM^{WS}	LPIPS	PSNR	PSNR^{WS}	SSIM	SSIM^{WS}	LPIPS
Mip-NeRF360 [1], *static only*	30.91	32.90	0.9182	0.9183	0.2252	31.01	30.12	0.8810	0.8736	0.2218	26.52	26.52	0.7884	0.7966	0.3628
Ours wo/ pose, *static only*	34.22	34.28	0.9339	0.8681	0.2561	29.76	28.58	0.8532	0.8360	0.2771	26.90	27.85	0.8167	0.8397	0.3108
Mip-NeRF360 [1]	22.01	22.71	0.8767	0.8464	0.2973	30.35	29.32	0.8798	0.8715	0.2232	24.40	23.94	0.7799	0.7858	0.3751
EgoNeRF [3]	22.14	22.62	0.8452	0.8379	0.3775	26.26	25.48	0.8146	0.7983	0.3736	23.57	22.90	0.7534	0.7491	0.4336
LocalRF [7] w/ pose	22.39	22.87	0.8753	0.8665	0.2725	27.19	26.35	0.8460	0.8264	0.3115	25.67	25.66	0.8074	0.8282	0.3213
D^{2}NeRF [10]	18.05	18.72	0.5792	0.5677	0.6481	23.38	22.51	0.6927	0.6646	0.5912	18.31	17.05	0.5917	0.5463	0.6501
RobustNeRF [9]	18.88	18.86	0.7242	0.7029	0.5489	24.17	23.07	0.7706	0.7484	0.4123	18.73	17.43	0.7031	0.6774	0.4589
Ours w/ pose	33.00	32.91	0.9232	0.9169	0.2008	29.37	28.21	0.8444	0.8264	0.2845	26.29	27.01	0.8093	0.8292	0.3194
LocalRF [7] wo/ pose	22.33	22.76	0.8710	0.8603	0.2781	28.02	27.07	0.8447	0.8266	0.2758	25.31	25.23	0.8011	0.8193	0.3309
Ours wo/ pose	33.35	33.32	0.9270	0.9210	0.1925	29.71	28.55	0.8525	0.8357	0.2767	26.74	27.68	0.8149	0.8377	0.3140

neural feature planes starts at 2×10^{-2} and exponentially decays to 2×10^{-3}. The learning rate for the camera matrix starts at 5×10^{-3} and exponentially decays to 5×10^{-4}. To prevent excessive change in the mask MLP, which is used to estimate motion masks of entire frames, we set the learning rate to start at 5×10^{-4} and exponentially decay to 5×10^{-6}.

2. Motion Mask Estimation

In recent works such as [5, 6], dynamic objects, which are temporally dependent transient data that are independent of global geometry, are excluded under the Bayesian learning framework. This is because dynamic components introduce high epistemic uncertainty, which slows down the learning process and results in high errors when performing tasks such as volume-based view synthesis that require 3D reconstruction. To implicitly address dynamic components, RobustNeRF [9] down-weights the loss of regions with high uncertainty. However, in our approach, we explicitly separate dynamic components from stationary geometry. In the task of predicting dynamic color based on pixel position input imposed on the mask module, the uncertainty of transient data significantly decreases because of a lack of geometrical constraint. Therefore, by performing both tasks simultaneously, the mask module converges faster to predict dynamic color values and alpha values compared to radiance fields. The segmentation of dynamic components allows radiance fields to ignore them during the training process, focusing solely on static elements. Figure 1 shows examples of our mask results.

Table 3. Our method performs better than existing techniques when training on the real dataset with slightly increased computing time.

	Iteration	Time ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Mip-NeRF360 [1]	250K	20 hrs	26.88	0.8094	0.3583
EgoNeRF [3]	100K	5.2 hrs	25.95	0.7609	0.4383
LocalRF [7]	155K	8.5 hrs	26.56	0.8034	0.3410
D^{2}NeRF [10]	100K	6 hrs	20.95	0.6105	0.5100
RobustNeRF [9]	500K	40 hrs	20.78	0.7093	0.4864
Ours	120K	8.7 hrs	27.61	0.8135	0.3372
Ours	155K	12.5 hrs	27.73	0.8165	0.3297

When training the mask module to estimate motion masks, there is a problem with learning dynamic alpha values because of factorization ambiguity. To solve this issue, we supervise the estimated dynamic color by ground truth with added Gaussian noise. This helps to narrow down the possibilities for alpha values, making it easier to obtain a mask closer to a binary value. Without this supervision, the motion mask prediction struggles to reach a binary value, which can result in dynamic artifacts on the static geometry. Refer to Figure 2 to see an illustration of this issue.

3. Comparison Details

To compare our results, we use the official implementations of D^{2}NeRF [10], EgoNeRF [3], and LocalRF [7]. We employ the Multi-NeRF [8] code for Mip-NeRF360 [1] and RobustNeRF [9]. All experiments are conducted on a machine with a single NVIDIA A6000 GPU and an Intel Xeon

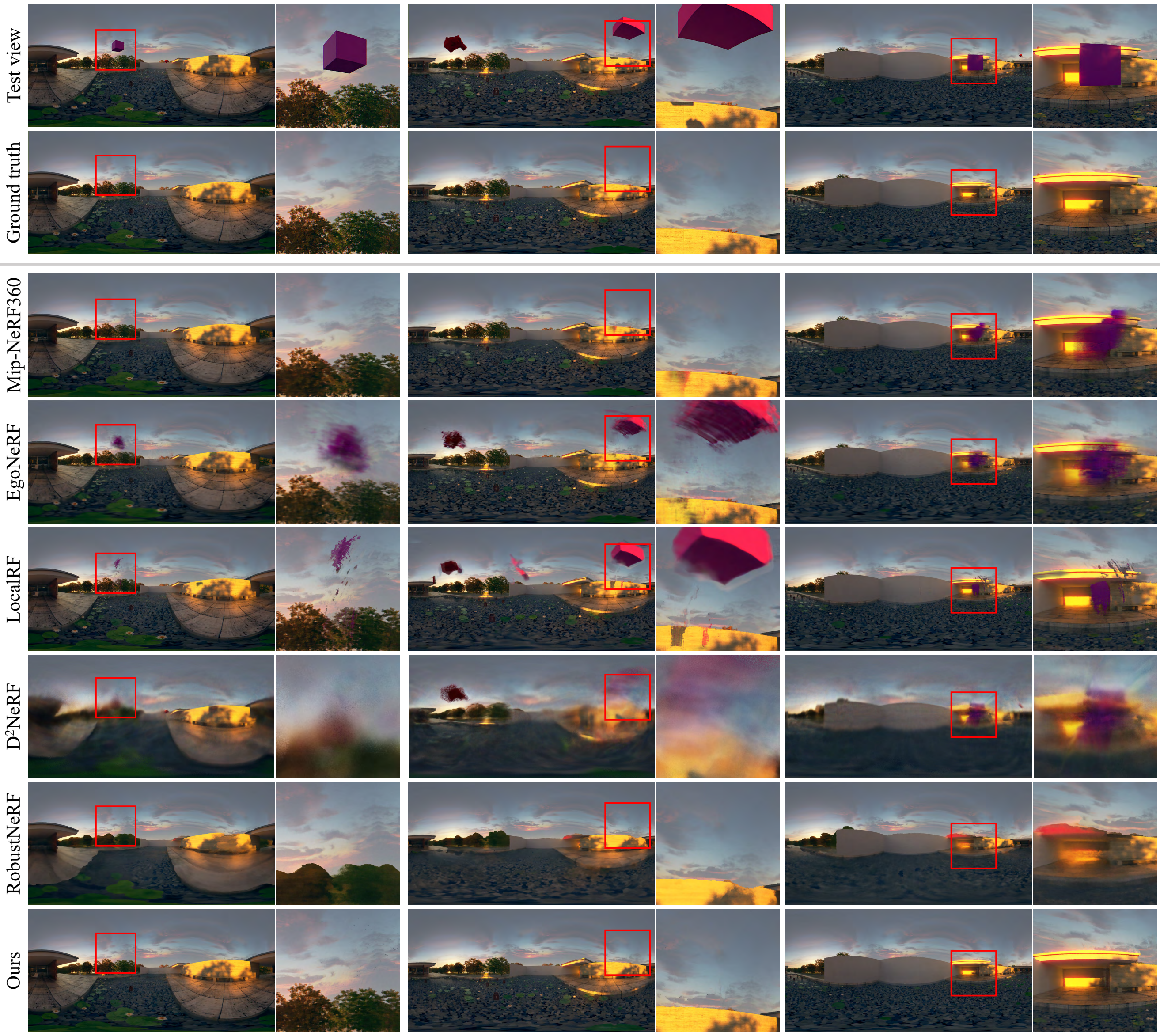


Figure 6. Qualitative comparisons of the *Pavillion* scene in the synthetic video dataset where synthetic moving objects are inserted.

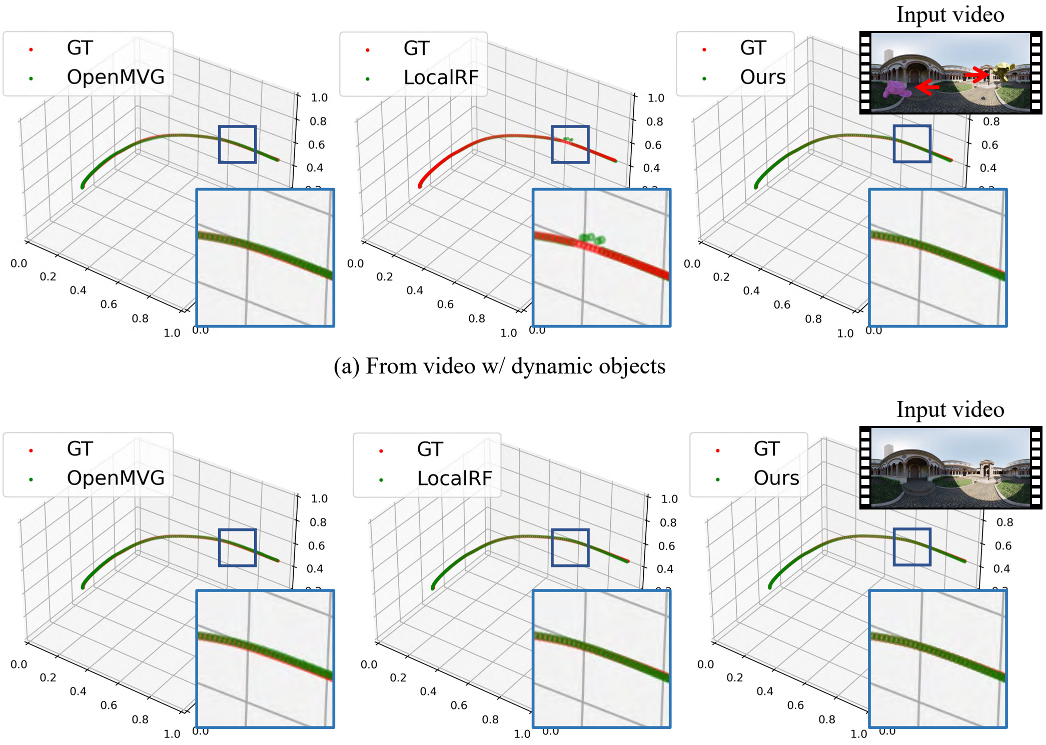
Silver 4214R 2.40 GHz CPU with 256 GB RAM. The baseline methods use the default hyperparameters, including the number of iterations, except for RobustNeRF, for which we use 500K iterations due to its slow convergence by the IRLS-based approach. We present the training time and metrics in Table 3. Additionally, we present the results of our method with 120K iterations, which show comparable training time to LocalRF. We also use the results of our method with 155K iterations, chosen for its ability to capture intricate details.

4. Additional Results

We present additional results for the real dataset. Figures 3, 4, and 5 display the results, and the corresponding metrics are available in Table 1. Additionally, we include the results from the *Pavillion* scene in the synthetic dataset, which is presented in Figure 6. The respective metrics are available in Table 2. We also include the pose estimation results in Figure 7 and Table 4.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded



(a) From video w/ dynamic objects

(b) From video w/o dynamic objects

Figure 7. In the *Lone Monk* scene, we compare the estimated pose with the ground-truth pose both with and without dynamic objects. The results with dynamic objects are shown in (a), and without dynamic objects in (b). Our method is capable of robustly estimating frame poses, regardless of the presence of dynamic objects.

Table 4. Quantitative comparisons of the pose accuracy on the synthetic dataset. Our method shows superior performance, especially in scenarios where dynamic objects are present. Refer to Figure 7 for qualitative comparisons.

	Sponza			*Pavillion*			*Lone monk*		
	RPE$_r$ ↓	RPE$_t$ ↓	ATE ↓	RPE$_r$ ↓	RPE$_t$ ↓	ATE ↓	RPE$_r$ ↓	RPE$_t$ ↓	ATE ↓
OpenMVG	0.09310	0.00045	0.00032	0.17250	0.01962	0.00347	0.05726	0.03388	0.00263
LocalRF	0.09309	0.00074	0.00336	0.17133	0.00065	0.00417	0.04770	0.00145	0.00374
Ours	0.09309	0.00020	0.00207	0.17133	0.00056	0.00116	0.04753	0.00143	0.00236
OpenMVG, *static only*	0.09309	0.00047	0.00036	0.17255	0.01957	0.00332	0.05552	0.03384	0.00195
LocalRF, *static only*	0.09309	0.00018	0.00173	0.17133	0.00052	0.00222	0.04753	0.00143	0.00230
Ours, *static only*	0.09309	0.00020	0.00207	0.17133	0.00058	0.00135	0.04753	0.00146	0.00153

Anti-Aliased Neural Radiance Fields. In *CVPR*, pages 5470–5479, 2022. 5

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial Radiance Fields. In *ECCV*, pages 333–350. Springer, 2022. 1

[3] Changwoon Choi, Sang Min Kim, and Young Min Kim. Balanced Spherical Grid for Egocentric View Synthesis. In *CVPR*, pages 16590–16599, 2023. 5

[4] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 1

[5] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. DynIBaR: Neural Dynamic Image-Based Rendering. In *CVPR*, 2023. 5

[6] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, pages 7210–7219, 2021. 5

[7] Andreas Meuleman, Yu-Lun Liu, Chen Gao, Jia-Bin Huang, Changil Kim, Min H Kim, and Johannes Kopf. Progressively Optimized Local Radiance Fields for Robust View Synthesis. In *CVPR*, pages 16539–16548, 2023. 1, 5

[8] Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, Peter Hedman, Ricardo Martin-Brualla, and Jonathan T. Barron. MultiNeRF: A Code Release for Mip-NeRF 360, Ref-NeRF, and RawNeRF, 2022. 5

[9] Sara Sabour, Suhani Vora, Daniel Duckworth, Ivan Krasin, David J Fleet, and Andrea Tagliasacchi. RobustNeRF: Ignoring Distractors with Robust Losses. In *CVPR*, pages 20626–20636, 2023. 5

[10] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. D2NeRF: Self-Supervised Decoupling of Dynamic and Static Objects from a Monocular Video. *NIPS*, 35:32653–32666, 2022. 1, 5

[11] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. STaR: Self-supervised Tracking and Reconstruction of Rigid Objects in Motion with Neural Rendering. In *CVPR*, pages 13144–13152, 2021. 1