A. 3D View Synthesis

A.1. Sparse and Unconstrained Multi-Views

We use 24 time instants from multi-view temporal sequences from the Open4D dataset [3]. The dynamic scenes are captured by a varying number of cameras in these sequences. The number of views vary from 7 to 11. We use one held-out view (or camera) for evaluation. Figure 9 contrasts our results on these sequences with NeRF and DS-NeRF. Following is the setup for this analysis:

Sequences

`WFD-01`: 6 time-stamps - {`2000`, `2500`, `3000`, `3500`, `4000`, `4500`}. Test CAM-ID: {`2, 9, 2, 2, 6, 4`}.
`WFD-02`: 5 time-stamps - {`1900`, `3000`, `3500`, `4000`, `4500`}. Test CAM-ID: {`3, 6, 4, 2, 3`}.
`JiuJitsu`: 7 time-stamps - {`3000`, `3500`, `4000`, `4500`, `5000`, `5500`, `6000`}. Test CAM-ID: {`5, 4, 9, 5, 7, 11, 1`}.
`Gangnam`: 3 time-stamps - {`0200`, `0300`, `0900`}. Test CAM-ID: {`4, 4, 4`}.
`Jumping`: 3 time-stamps - {`0200`, `0300`, `0400`}. Test CAM-ID: {`0, 0, 0`}.

A.2. Hi-Resolution View Synthesis

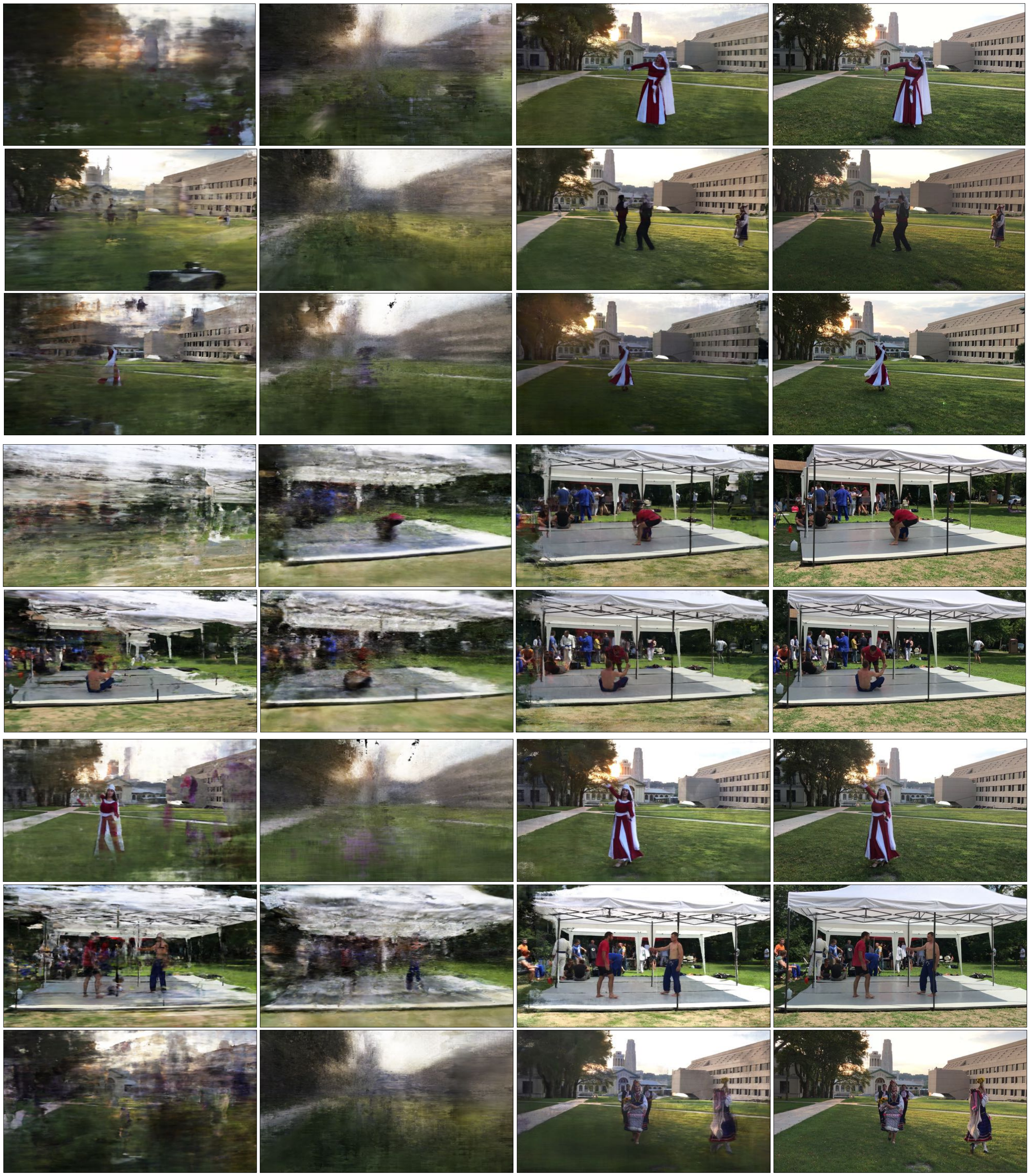
Shiny Dataset: We use 8 multi-view sequences from the Shiny Dataset [45] that consists of multi-views captured for specular surfaces. The resolution of 6 sequences (less than 60 samples in each) in this dataset is 4032×3024, and the remaining two (`cd` and `labs` have more than 300 samples) have resolution 1920×1080. We train NeRF on the original resolution of these sequences for $2M$ iterations (64 hours per GPU). We contrast the performance with our approach that is trained for 10 epochs and 50 epochs. Table 7 shows the performance of different methods. We follow the evaluation criteria (average of per-sequence PSNR, multi-channel SSIM, LPIPS[1]) from NeX [45]. We also add the results generated by NeX [45] that synthesizes on one-fourth resolution for these sequences. We do a simple $4\times$-upsampling of their results to target resolution for an apples-to-apples comparison. Our model trained for 10 minutes achieves results close to the best performance.

LLFF-12: We use twelve high-resolution (4032×3024) multi-view sequences from the LLFF dataset [26] that contain challenging specular surfaces. In this setting, we train NeRF [27] on these sequences for $2,000,000$ iterations which take approximately 64 hours on a single NVIDIA V100 GPU ($10,000$ iterations take 20 minutes). Performance saturates at $1M$ iterations after 32 hours of training. We also show the performance for `vanilla` NeRF that is trained for $200,000$ iterations and takes $400 - 420$ minutes to train. We train our model for 10 epochs, which takes around 10 minutes on a single GPU and only 1GB GPU of memory. We estimate disparity [47] for multiple stereo pairs at one-fourth resolution for these sequences. Disparity estimation using the off-the-shelf model takes less than 5 minutes per sequence on a single GPU. Table 8 contrasts the performance of NeRF models at different intervals of training using PSNR, SSIM, and LPIPS (AlexNet). We compute the average of per-frame statistics as the number of samples in the test set for these 12 sequences are roughly the same. We once again observe that it is crucial to include all three evaluation criteria. Figure 10 shows the results of NeRF at different intervals of time. We observe that the NeRF model improves over time and captures sharp results as suggested by LPIPS. Our method enables sharper outputs as compared to NeRF. Interestingly, NeRF does not capture details even for training samples when trained sufficiently long (64 hours) which suggests that it is non-trivial to capture details using NeRF on held-out samples. The qualitative and quantitative analysis suggest that we can efficiently generate results on 12MP images without drastically increasing the computational resources. We also show the performance of naive composition to generate the final outputs. We observe that MLPs allow us to obtain better results. We also vary the number of stereo pairs (K) to synthesize the target view. We observe that we can get better results with a few stereo pairs than using all pairs. Synthesizing a new view for a dense multi-view sequence can be

8 sequences	**PSNR↑**	**MCSSIM↑**	**LPIPS ↓**
`4032×3024`			
NeRF			
`vanilla`	21.141 ± 3.528	0.735 ± 0.155	0.528 ± 0.157
2M iterations	21.457 ± 3.657	0.751 ± 0.155	0.498 ± 0.153
Naive Composition	16.624 ± 2.906	0.648 ± 0.197	0.342 ± 0.096
Naive Composition++	17.535 ± 2.698	0.688 ± 0.184	0.317 ± 0.107
Ours (10 minutes)			
$K = 50, N = 50$	22.430 ± 4.748	0.795 ± 0.142	0.256 ± 0.108
$\boxed{K = 100, N = 100}$	22.868 ± 4.588	0.802 ± 0.140	0.269 ± 0.120
$K = 200, N = 200$	23.016 ± 4.698	0.803 ± 0.144	0.285 ± 0.132
$K = ALL, N = 200$*	22.090 ± 4.263	0.786 ± 0.154	0.332 ± 0.145
Ours (50 minutes)			
$K = 50, N = 50$	22.261 ± 4.812	0.791 ± 0.144	$\mathbf{0.252 \pm 0.105}$
$K = 100, N = 100$	22.739 ± 4.637	0.801 ± 0.142	0.258 ± 0.113
$K = 200, N = 200$	$\mathbf{23.020 \pm 4.690}$	$\mathbf{0.805 \pm 0.143}$	0.271 ± 0.123
$K = ALL, N = 200$*	21.788 ± 4.243	0.780 ± 0.154	0.317 ± 0.137
`resized`			
NeRF [27]	22.009 ± 3.148	0.757 ± 0.156	0.487 ± 0.180
NeX [45]	22.292 ± 3.137	0.774 ± 0.152	0.423 ± 0.156

Table 7. **Shiny dataset:** We study our approach on the 8 real sequences from the Shiny dataset [45]. NeRF is trained for $2M$ iterations taking approx 64 hours. We also add the results of $4\times$ bi-linearly upsampled results from NeX [45] on these sequences. Our approach gets competitive performance in only a few minutes.

[1]We, however, use LPIPS via AlexNet (alex) instead of VGG-Net (vgg) to fit 12MP images on a single GPU.



NeRF DS-NeRF Ours Ground Truth

Figure 9. **View synthesis given sparse and spread-out multi-views:** Our approach allows us to operate on sparse multi-views of unbounded scenes [3]. We show novel view points for a fixed time instant for three unbounded scenes. Prior approaches such as NeRF [27] and DS-NeRF [6] lead to degenerate outputs on these sequences.

12 sequences	PSNR↑	SSIM↑	LPIPS ↓
NeRF [27]			
2 hours	21.151 ± 2.783	0.577 ± 0.157	0.662 ± 0.099
4 hours	21.469 ± 2.881	0.588 ± 0.153	0.628 ± 0.096
`vanilla`	21.625 ± 2.933	0.596 ± 0.150	0.605 ± 0.092
8 hours	21.674 ± 2.958	0.598 ± 0.149	0.599 ± 0.091
16 hours	21.734 ± 2.981	0.602 ± 0.148	0.586 ± 0.088
32 hours	21.741 ± 2.985	0.602 ± 0.147	0.584 ± 0.087
64 hours	**21.741 ± 2.985**	**0.602 ± 0.147**	0.584 ± 0.087
Naive Composition	16.008 ± 2.315	0.415 ± 0.142	0.427 ± 0.068
Naive Composition++	17.022 ± 2.483	0.460 ± 0.144	**0.406 ± 0.066**
Ours (10 minutes)			
$K = 50, N = 50$	20.834 ± 2.784	0.594 ± 0.136	0.426 ± 0.075
$K = 100, N = 100$	20.953 ± 2.805	0.598 ± 0.136	0.460 ± 0.078
$K = 200, N = 200$	20.783 ± 2.749	0.593 ± 0.135	0.494 ± 0.081
$K = ALL, N = 200$*	20.712 ± 2.656	0.591 ± 0.134	0.497 ± 0.077
Ours (50 minutes)			
$K = 50, N = 50$	20.777 ± 2.809	0.591 ± 0.137	0.416 ± 0.075
$K = 100, N = 100$	21.006 ± 2.869	0.597 ± 0.136	0.448 ± 0.080
$K = 200, N = 200$	20.924 ± 2.847	0.592 ± 0.134	0.477 ± 0.082
$K = ALL, N = 200$*	20.825 ± 2.708	0.589 ± 0.132	0.480 ± 0.079
Ours (250 minutes)			
$K = 50, N = 50$	20.582 ± 2.751	0.585 ± 0.135	0.409 ± 0.076
$K = 100, N = 100$	20.916 ± 2.874	0.593 ± 0.135	0.433 ± 0.078
$K = 200, N = 200$	20.640 ± 3.234	0.582 ± 0.139	0.474 ± 0.095
$K = ALL, N = 200$*	20.548 ± 3.104	0.580 ± 0.137	0.478 ± 0.092

Table 8. **Hi-Res (12MP) View Synthesis:** We evaluate on 12 sequences from LLFF containing specular surfaces on original 4032×3024 resolution. The details of these sequences are available in Appendix A.2. We contrast the performance of our approach with different intervals of training a NeRF model. Performance saturates at $1M$ iterations after 32 hours of training. Our composition model converges quickly in a few minutes. Here, we show the results of our composition model trained for 10 epochs that takes around 10 minutes, 50 epochs that takes less than 1 hour. Training our model require 1 GB of GPU memory for training. We also show the results when the model is trained for 250 epochs. For each setting, we vary the number of stereo pairs (K) and number of 3D points (N). We observe that using a few stereo-pairs gives competitive and better results than using all the pairs. We posit that noise introduced by using more stereo pairs might be responsible for the lower performance. Finally, we study the benefit of using an MLP for composing per-pixel color and depth information. The MLP allows us to obtain better results than a naive composition (Fig. 4 in main paper). We refer the reader to Figure 10 for visual comparisons. We observe that details become better for NeRF when trained for long. However, our approach captures more details in a few minutes as compared to 32 hours of training of a NeRF model. Consistent with the observation of Zhang et al. [50], PSNR may favor averaged/blurry results while LPIPS favors sharp results.

achieved by looking at the local neighborhood of the target location instead of using all the views. Local neighborhood is determined based on position in world space, i.e., we use stereo-pairs corresponding to the closest camera and then next and so on, unless we have K samples. This allows us to speed-up training and testing.

Following are the details of 12 sequences from LLFF dataset [26] for this analysis:

Sequences: `airplants, data2_apeskeleton, data2_benchflower, data2_bridgecar, data2_chesstable, data2_colorfountain, data2_colorspout, data2_redtoyota, data3_ninjabike, data4_colinepiano, data5_piano, pond`.
Test IDs: For each sequence, we held-out every 8^{th} frame for evaluation.

Standard LLFF Sequences: We do not make any change in our settings and quantitatively evaluate our approach on 8 forward-facing real-world multi-view sequences [27] in Table 9. We use the original hi-res (4032×3024) undistorted images provided by Wizadwongsa et al. [45]. We once again train NeRF models for these hi-res sequences for $2M$ iterations (64 hours per GPU). Training the model for long allows us to get better performing NeRF models for these sequences. We follow the evaluation criteria (average of per-sequence PSNR, multi-channel SSIM, LPIPS) from NeX [45]. We also add the results reported by NeX [45]. These results were generated on one-fourth resolution. We upsample them to the desired resolution. We report the performance of our approach (without any modification for these sequences) trained for 10 and 50 epochs. While other methods were specifically tuned for these sequences, we use our approach as-is. Our approach underperform both PSNR and SSIM but achieves a competitive LPIPS score. However, we can generate novel hi-res views (12MP) in a few minutes with limited computational resources.

A.3. Unbounded Views and Number of Views

We show the best performing result of NeRF on a held-out view from one of these sequences in Figure 11. We observe that our approach captures details better than NeRF. We also study the influence of camera parameters using synthetic multi-view sequences. We make two settings: (1) camera parameters are estimated using Agisoft Metashape; and (2) camera parameters provided with multi-view sequences. Table 10 contrasts the performance of our approach in these two settings. We vary the number of views between $\{10, 20, 30, 40, 50\}$. We observe that performance improves as we get better camera parameters. We also show the improvement in performance when using more views in Figure 12. Consistent with the quantitative analysis, we see better results visually when increasing the number of views.

We use the following 13 synthetic multi-view sequences for this analysis from the MVS-Synth dataset [13]:

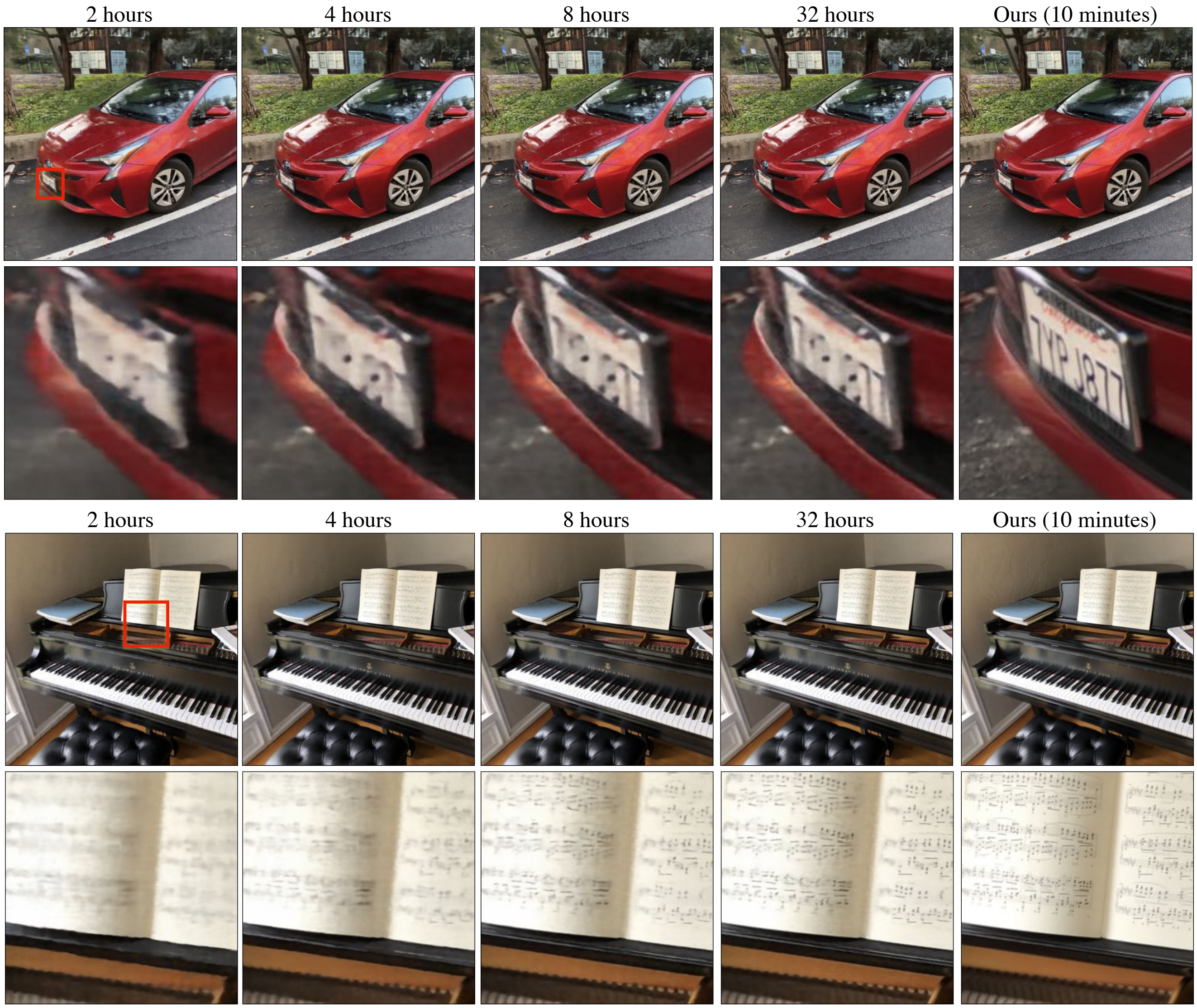


Figure 10. **Improvement in NeRF over time:** We show the progression (first 32 hours) of improvement for the NeRF model. We observe that results improve over time as details become clearer over time. We contrast this with our approach that can generate sharp results in only 10 minutes. Best viewed in electronic format.

Sequence IDs: {0000, 0001, 0002, 0003, 0004, 0005, 0006, 0007, 0008, 0009, 0010, 0011, 0012}.

For each sequence, we held-out every other frame for evaluation:

Test IDs: {000:002:098}.

Train IDs:

10 views: {003, 013, 023, 033, 043, 053, 063, 073, 083, 093}.

20 views: {003, 009, 013, 019, 023, 029, 033, 039, 043, 049, 053, 059, 063, 069, 073, 079, 083, 089, 093, 099}.

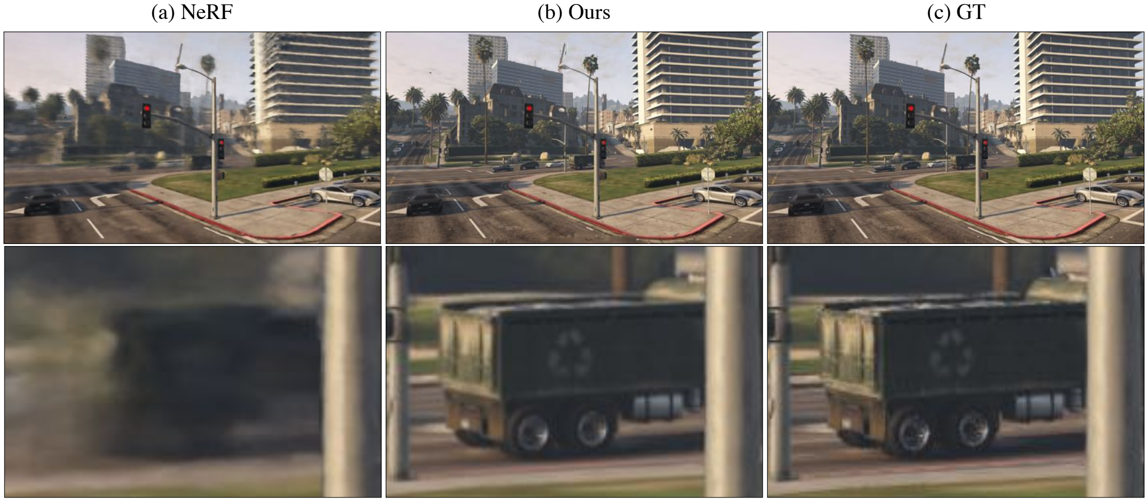
30 views: {003, 007, 009, 013, 017, 019, 023, 027, 029, 033, 037, 039, 043, 047, 049, 053, 057, 059, 063, 067, 069, 073, 077, 079, 083, 087, 089, 093, 097, 099}.

40 views: {001, 003, 007, 009, 011, 013, 017, 019, 021, 023, 027, 029, 031, 033, 037, 039, 041, 043, 047, 049, 051, 053, 057, 059, 061, 063, 067, 069, 071, 073, 077, 079, 081, 083, 087, 089, 091, 093, 097, 099}.

50 views: {001:002:099}.

(a) NeRF (b) Ours (c) GT

(a) NeRF (b) Ours (c) GT



(a) NeRF (b) Ours (c) GT

Figure 11. **Best performing NeRF output on a synthetic sequence, Num Views = 50: (a)** We cherry-pick the best performing synthesized result on a held-out view synthesized using NeRF trained on a sequence with unbounded depth. **(b)** We then show results using our approach. We zoom in to the billboard in the center of image (top-example), towards the bottom-left in second example, and on the truck in the middle in bottom-example. Our approach captures details better than NeRF. **(c)** The ground truth is shown for reference.

Figure 12. **Varying number of views:** We vary the number of views for training our model. We show three examples here: (1) **left** column shows drastic performance improvement as we increase the number of views; (2) **middle** column shows improvement when increasing from 10 to 30 and then saturating; and (3) **right** column where the improvement is little as we increase the number of views. In general, we observe that performance improves as we increase the number of views.

A.4. Hi-Res Studio Capture

Multi-View Facial Capture: We employ multi-view hi-res facial captures. We can synthesize hi-resolution novel views with a few minutes of training without any modification and without using any expert knowledge such as facial details, foreground-background etc. Figure 13 shows novel views synthesized and facial details (such as hair, eyes, wrinkles, teeth, etc.) captured using a model trained for a specific subject.

Multi-View Full Body Capture: Our approach also enables us to synthesize full-bodies from hi-res multi-view captures. Once again, we did not use any human-body specific information. Figure 16 shows novel views synthesized and body details captured using a model trained for a specific subject.

Ability to Generalize: An important aspect of our approach is to enable generalization to unseen time instants and unknown subjects. We train a model for one time instant of one subject and can use it to synthesize new views for unknown time instants. We show extreme facial expressions and unseen subjects in Figure 14. We also contrast

Figure 13. **Hi-Res Facial Details:** Our approach allows us to capture hi-res facial details. We show novel views synthesized for various subjects and emphasize different regions on the face to show details such as hair, eyes, teeth, and skin details.

8 sequences	**PSNR**↑	**MCSSIM**↑	**LPIPS** ↓
`4032×3024`			
NeRF			
`vanilla`	25.192 ± 3.681	0.881 ± 0.063	0.396 ± 0.084
2M iterations	**25.666** ± **3.833**	**0.887** ± **0.062**	0.372 ± 0.080
Naive Composition	17.147 ± 2.878	0.687 ± 0.134	0.528 ± 0.116
Naive Composition++	18.280 ± 2.852	0.732 ± 0.124	0.475 ± 0.118
Ours (10 minutes)			
$K = 50, N = 50$	22.561 ± 3.361	0.848 ± 0.078	**0.347** ± **0.085**
$K = 100, N = 100$	22.951 ± 3.564	0.854 ± 0.077	0.361 ± 0.087
$K = 200, N = 200$	22.930 ± 3.612	0.854 ± 0.078	0.380 ± 0.096
$K = ALL, N = 200$*	21.650 ± 2.605	0.841 ± 0.072	0.416 ± 0.079
Ours (50 minutes)			
$K = 50, N = 50$	22.335 ± 3.316	0.839 ± 0.086	0.355 ± 0.099
$K = 100, N = 100$	23.020 ± 3.500	0.851 ± 0.079	0.356 ± 0.093
$K = 200, N = 200$	23.237 ± 3.673	0.853 ± 0.082	0.369 ± 0.105
$K = ALL, N = 200$*	21.650 ± 2.605	0.841 ± 0.072	0.400 ± 0.090
SRN [40]	21.147 ± 3.140	0.821 ± 0.078	0.594 ± 0.113
LLFF [26]	23.334 ± 3.315	0.863 ± 0.064	0.431 ± 0.091
NeRF	25.076 ± 3.432	0.871 ± 0.062	0.439 ± 0.103
NeX	25.430 ± 3.503	0.881 ± 0.058	0.387 ± 0.077

Table 9. **Real forward-facing dataset:** We study our approach on the original resolution of the 8 real sequences from Mildenhall et al. [26]. We also add the results of $4\times$ bi-linearly upsampled results from NeX [45] on these sequences. Our approach underperform PSNR and SSIM but competitive LPIPS score.

13 sequences	**PSNR**↑	**SSIM**↑	**LPIPS** ↓
`num-views=10`			
Ours (MS)	18.460 ± 4.099	0.656 ± 0.129	0.451 ± 0.167
Ours	**19.439** ± **4.375**	**0.697** ± **0.128**	**0.410** ± **0.177**
`num-views=20`			
Ours (MS)	22.414 ± 4.197	0.766 ± 0.126	0.289 ± 0.147
Ours.	**23.651** ± **4.045**	**0.813** ± **0.096**	**0.241** ± **0.120**
`num-views=30`			
Ours (MS)	24.191 ± 4.219	0.803 ± 0.122	0.243 ± 0.137
Ours	**25.357** ± **3.709**	**0.846** ± **0.081**	**0.201** ± **0.094**
`num-views=40`			
Ours (MS)	24.832 ± 4.110	0.822 ± 0.117	0.218 ± 0.125
Ours	**26.083** ± **3.691**	**0.865** ± **0.072**	**0.178** ± **0.077**
`num-views=50`			
Ours (MS)	25.529 ± 4.212	0.836 ± 0.112	0.198 ± 0.116
Ours	**26.829** ± **3.621**	**0.878** ± **0.064**	**0.161** ± **0.070**

Table 10. **Varying Camera Parameter for Synthetic Multi-View Sequences of Unbounded Scenes:** We make two settings of camera parameters: (1) MS, computed using Agisoft Metashape; and (2) using ground truth camera parameters provided with synthetic sequence. We vary the number of views to synthesize target views using synthetic multi-view data. The held-out sequences are fixed in these analysis. We observe that performance improves with better camera parameter estimation.

the results of generalization with a subject-specific model in Figure 15. We observe that the learned model generalizes well except for the clothing in the bottom part of the images. We posit that there isn't sufficient coverage from multi-views in that area. However, an exemplar model learned for a specific subject is able to capture the details. We leave the reader with an open philosophical question as to whether we should think about generalization if we can learn an exemplar model for a given data distribution in a few seconds?

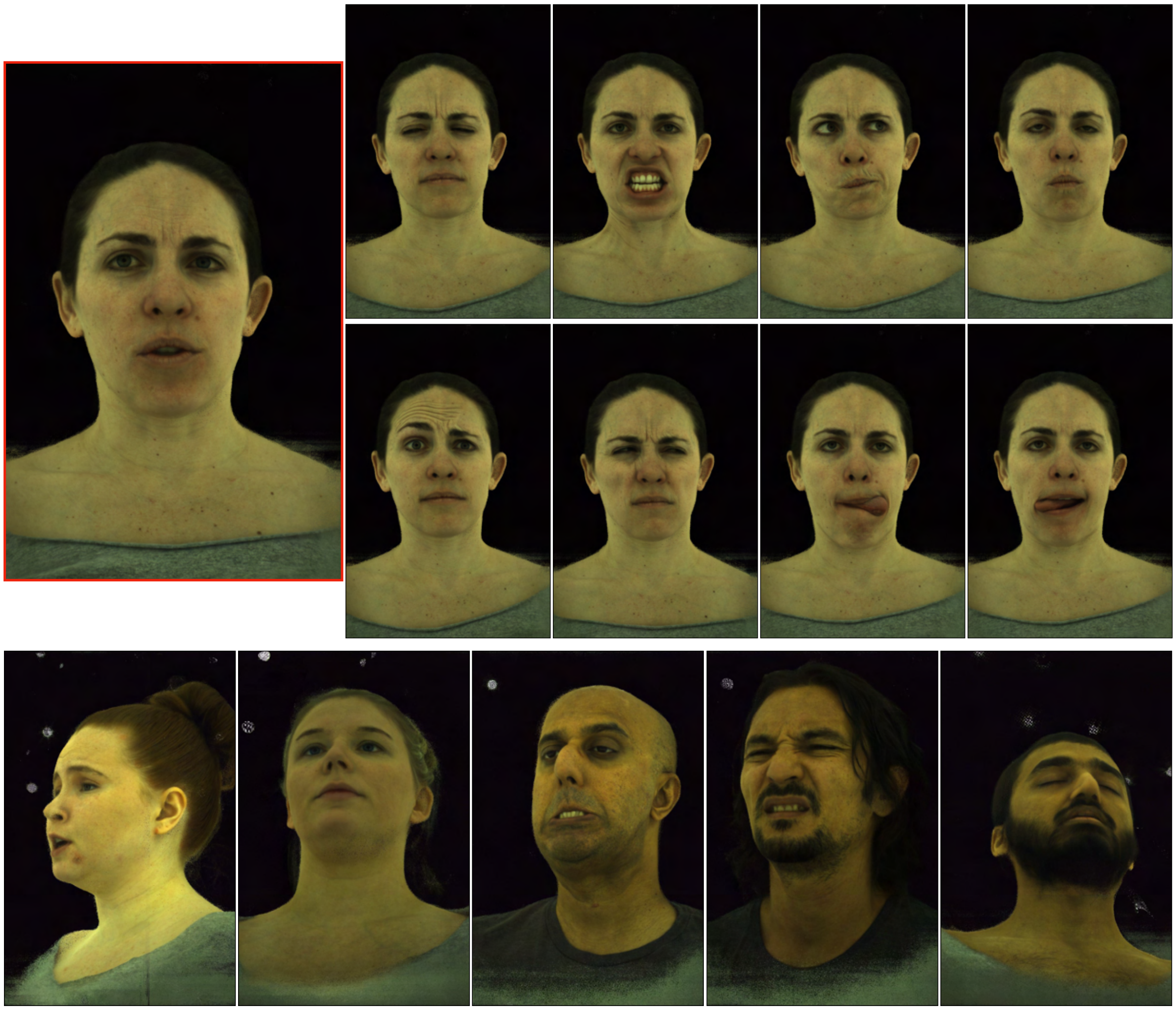


Figure 14. **Generalization to unseen time instants and unseen subjects:** The model is trained on a single time instant – shown on top-left. Our model generalizes to unseen expressions (top-right) and unseen subjects (bottom row).

Exemplar Generalization Exemplar Generalization

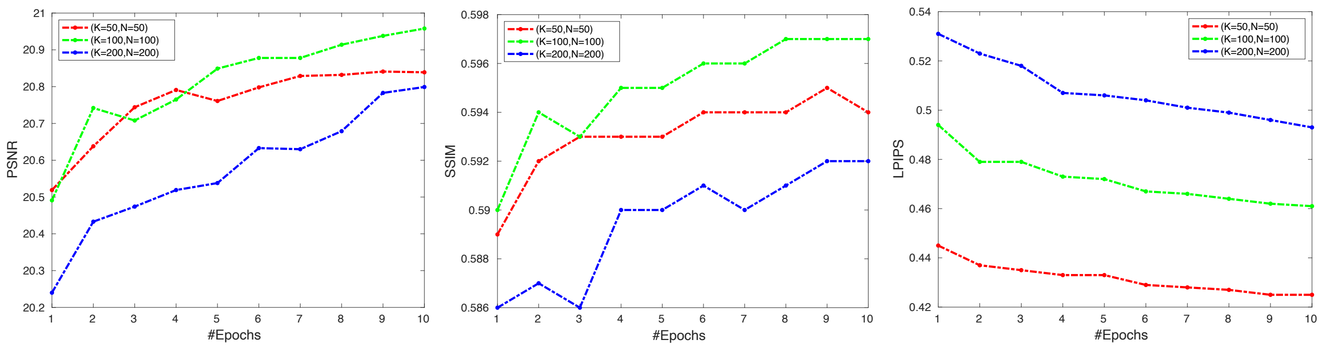
Figure 15. **Contrasting Exemplar Models and Generalization:** We contrast the results from exemplar model with the results obtained using a model that has never seen these subjects. We term it **Generalization** here.

Figure 16. **Hi-Res Body Synthesis:** Our approach allows us to synthesize high quality novel views of human bodies. In the bottom-row, we zoom to see the details captured on the face for each of three subjects. Best viewed in electronic format.

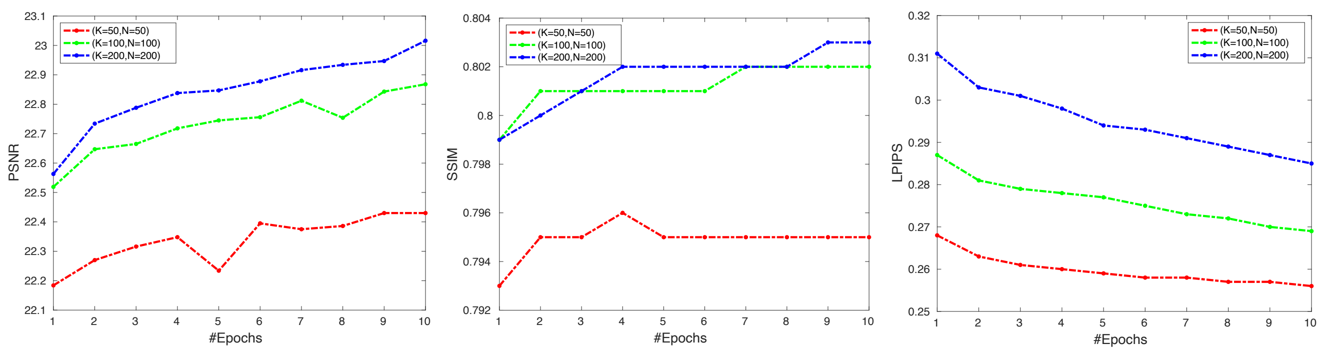
A.5. Convergence Analysis

We study the convergence properties of our approach using 12 LLFF sequences [26] (Appendix A.2) and Shiny Dataset [45]. We show the plots in Figure 17 for model training in the first 10 epochs, i.e. from 60 seconds to 600 seconds. We observe that our model gets close to convergence in the first few seconds. Crucially, our approach obtains competitive results to prior work on the Shiny dataset within 60 seconds of training as compared to 64 hours for NeRF [27] on full-resolution and $24 - 30$ hours of training of NeX [45] on one-fourth resolution. We also study convergence using 24 sparse and unconstrained multi-view sequences (Appendix A.1). Training an epoch on these sequences roughly take 10 seconds because these are sparse. We observe that model gets close to the best performance in the first 10 seconds of training.

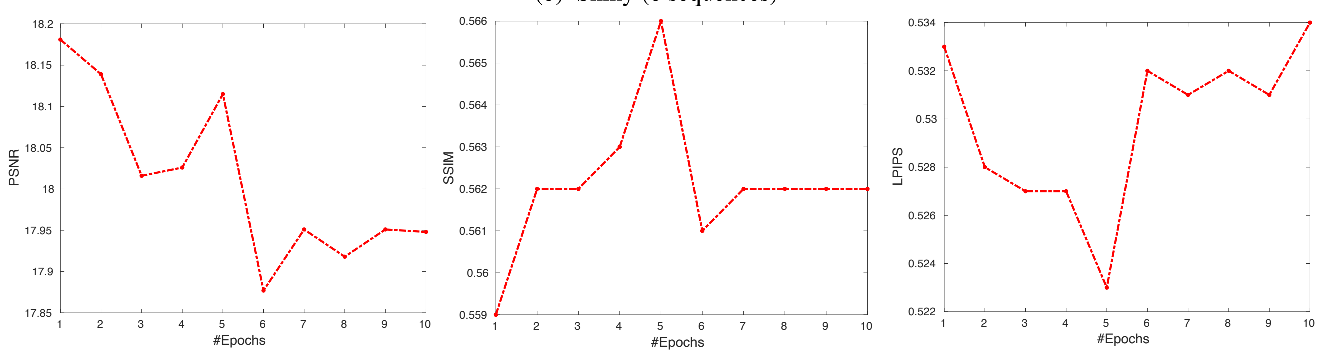
We also provide the raw data used in the analysis. We use 24 sparse and unconstrained multi-view sequences (Sec A.1) from Open4D [3]. Training an epoch on these se-



(a) LLFF (12 sequences)



(b) Shiny (8 sequences)



(c) Sparse and Unconstrained Multi-Views (24 sequences)

Figure 17. **Convergence Analysis:** We study the convergence properties of our approach using 12 LLFF sequences [26] and 8 sequences from Shiny Dataset [45]. We also study convergence using 24 sparse and unconstrained multi-view sequences. We observe that our model gets close to convergence in the first few seconds. Note the difference in values on y-axis is small.

quences roughly take 10 seconds because these are sparse. Table 11 shows the performance of our model for 10 epochs (from 10 seconds to roughly 2 minutes). We also use two hi-res (12 MP) datasets for these analysis: **(1)** 12 sequences (Sec A.2) from LLFF dataset [26]; and **(2)** 8 sequences from Shiny dataset [45]. We compute the performance of the models for the first 10 epochs, i.e., from 60 to 600 seconds of training. We follow the three settings (as in Sec 4.2) where we vary the number of stereo-pairs (K) and number of 3D points (N): **(1)** ($K = 50, N = 50$); **(2)** ($K = 100, N = 100$); and **(3)** ($K = 200, N = 200$). Table 12, Table 13, and Table 14 shows the performance for 12 sequences from LLFF. Table 15, Table 16, and Table 17 shows the performance for 8 sequences from the Shiny dataset. We observe that our approach gets close to convergence within the first 60 seconds of training in all the settings.

B. 4D View Synthesis

We use temporal sequences from Open4D dataset [3] for these analysis. Figure 18 shows different things we can do

24 sequences	PSNR↑	SSIM↑	LPIPS ↓
Num-Epochs			
1	18.181 ± 1.519	0.559 ± 0.079	0.533 ± 0.066
2	18.139 ± 1.378	0.562 ± 0.077	0.528 ± 0.062
3	18.016 ± 1.461	0.562 ± 0.077	0.527 ± 0.063
4	18.026 ± 1.420	0.563 ± 0.076	0.527 ± 0.063
5	18.115 ± 1.459	0.566 ± 0.076	0.523 ± 0.061
6	17.877 ± 1.409	0.561 ± 0.075	0.532 ± 0.061
7	17.951 ± 1.456	0.562 ± 0.076	0.531 ± 0.059
8	17.918 ± 1.511	0.562 ± 0.077	0.532 ± 0.061
9	17.951 ± 1.475	0.562 ± 0.076	0.531 ± 0.058
10	17.948 ± 1.472	0.562 ± 0.077	0.534 ± 0.061
NeRF [27]	13.693 ± 2.050	0.317 ± 0.094	0.713 ± 0.089
DS-NeRF [6]	14.531 ± 2.603	0.316 ± 0.099	0.757 ± 0.040
LLFF [26]	15.187 ± 2.166	0.384 ± 0.082	0.602 ± 0.090

Table 11. **Sparse and Unconstrained Multi-Views** : We follow the evaluation criterion in Table 1. We observe that our model gets the best performance in the the first 10 seconds of training. We contrast the performance of NeRF and DS-NeRF which takes 420 minutes of training on a single NVIDIA V100 GPU. We also show the performance of LLFF which is an off-the-shelf method and does not require training.

12 sequences	PSNR↑	SSIM↑	LPIPS ↓
Num-Epochs			
1	20.519 ± 2.805	0.589 ± 0.137	0.445 ± 0.076
2	20.638 ± 2.736	0.592 ± 0.137	0.437 ± 0.076
3	20.744 ± 2.772	0.593 ± 0.137	0.435 ± 0.076
4	20.791 ± 2.783	0.593 ± 0.137	0.433 ± 0.075
5	20.761 ± 2.774	0.593 ± 0.137	0.433 ± 0.076
6	20.798 ± 2.787	0.594 ± 0.136	0.429 ± 0.076
7	20.829 ± 2.807	0.594 ± 0.136	0.428 ± 0.074
8	20.832 ± 2.803	0.594 ± 0.136	0.427 ± 0.075
9	20.841 ± 2.802	0.595 ± 0.136	0.425 ± 0.074
10	20.839 ± 2.798	0.594 ± 0.136	0.426 ± 0.075
NeRF-2M	21.741 ± 2.985	0.602 ± 0.147	0.584 ± 0.087

Table 12. **LLFF-12 sequences and** $(K = 50, N = 50)$: We use 50 stereo-pairs and 50 3D points. We follow the evaluation criterion in Table 3. We observe that our model gets close to the best performing model in the the first 60 seconds of training. For reference, we also show the performance of NeRF which takes 64 hours of training on a single NVIDIA V100 GPU.

with 4D view synthesis. Figure 19 contrasts our approach with naive composition.

B.1. Unseeen Temporal Sequences

We use all the available views of the following 5 publicly available temporal sequences. Figure 20 contrasts our approach with Open4D on unseen temporal sequences. We observe better qualitative results. Our approach is able to capture details such as human faces consistently better than Open4D. Crucially, our approach does not require explicit foreground-background modeling and can work with arbitrary temporal sequences.

12 sequences	PSNR↑	SSIM↑	LPIPS ↓
Num-Epochs			
1	20.491 ± 2.966	0.590 ± 0.140	0.494 ± 0.079
2	20.742 ± 2.779	0.594 ± 0.138	0.479 ± 0.079
3	20.708 ± 2.851	0.593 ± 0.139	0.479 ± 0.080
4	20.765 ± 2.829	0.595 ± 0.138	0.473 ± 0.078
5	20.849 ± 2.783	0.595 ± 0.137	0.472 ± 0.079
6	20.878 ± 2.787	0.596 ± 0.137	0.467 ± 0.079
7	20.878 ± 2.807	0.596 ± 0.137	0.466 ± 0.078
8	20.914 ± 2.806	0.597 ± 0.137	0.464 ± 0.078
9	20.938 ± 2.801	0.597 ± 0.136	0.462 ± 0.079
10	20.958 ± 2.805	0.597 ± 0.136	0.461 ± 0.080
NeRF-2M	21.741 ± 2.985	0.602 ± 0.147	0.584 ± 0.087

Table 13. **LLFF-12 sequences and** $(K = 100, N = 100)$: We use 100 stereo-pairs and 100 3D points. We follow the evaluation criterion in Table 3. We observe that our model gets close to the best performing model in the the first 60 seconds of training. For reference, we also show performance of a NeRF model that takes 64 hours of training on a single NVIDIA V100 GPU.

12 sequences	PSNR↑	SSIM↑	LPIPS ↓
Num-Epochs			
1	20.240 ± 2.955	0.586 ± 0.141	0.531 ± 0.083
2	20.433 ± 2.859	0.587 ± 0.139	0.523 ± 0.082
3	20.474 ± 2.842	0.586 ± 0.138	0.518 ± 0.083
4	20.519 ± 2.808	0.590 ± 0.137	0.507 ± 0.082
5	20.538 ± 2.816	0.590 ± 0.137	0.506 ± 0.081
6	20.633 ± 2.795	0.591 ± 0.136	0.504 ± 0.081
7	20.630 ± 2.825	0.590 ± 0.136	0.501 ± 0.082
8	20.679 ± 2.841	0.591 ± 0.136	0.499 ± 0.081
9	20.783 ± 2.777	0.592 ± 0.136	0.496 ± 0.081
10	20.799 ± 2.772	0.592 ± 0.136	0.493 ± 0.081
NeRF-2M	21.741 ± 2.985	0.602 ± 0.147	0.584 ± 0.087

Table 14. **LLFF-12 sequences and** $(K = 200, N = 200)$: We use 200 stereo-pairs and 200 3D points. We follow the evaluation criterion in Table 3. We observe that our model gets close to the best performing model in the the first 60 seconds of training. For reference, we also show performance of a NeRF model that takes 64 hours of training on a single NVIDIA V100 GPU.

Sequences

`WFD-01`: Training - `{0011:0411}`. Testing - `{0412:0511}`.
`WFD-02`: Training - `{0400:0800}`. Testing - `{0801:0900}`.
`JiuJitsu`: Training - `{0001:0400}`. Testing - `{0401:0500}`.
`Gangnam`: Training - `{0100:0400}`. Testing - `{0401:0500}`.

8 sequences	PSNR↑	MCSSIM↑	LPIPS ↓
Num-Epochs			
1	22.184 ± 4.211	0.793 ± 0.142	0.268 ± 0.110
2	22.270 ± 4.321	0.795 ± 0.142	0.263 ± 0.108
3	22.316 ± 4.372	0.795 ± 0.141	0.261 ± 0.107
4	22.348 ± 4.379	0.796 ± 0.141	0.260 ± 0.107
5	22.234 ± 4.399	0.795 ± 0.141	0.259 ± 0.107
6	22.395 ± 4.542	0.795 ± 0.141	0.258 ± 0.107
7	22.375 ± 4.579	0.795 ± 0.142	0.258 ± 0.017
8	22.386 ± 4.625	0.795 ± 0.142	0.257 ± 0.107
9	22.430 ± 4.677	0.795 ± 0.142	0.257 ± 0.107
10	22.430 ± 4.740	0.795 ± 0.142	0.256 ± 0.107
NeRF [27]	22.009 ± 3.148	0.757 ± 0.156	0.487 ± 0.180
NeRF-2M	21.457 ± 3.657	0.751 ± 0.155	0.498 ± 0.153
NeX [45]	22.292 ± 3.137	0.774 ± 0.152	0.423 ± 0.156

Table 15. **Shiny dataset and** $(K = 50, N = 50)$: We use 50 stereo-pairs and 50 3D points. We follow the evaluation criterion in Table 7. We observe that our model gets close to the best performing model in the first 60 seconds of training. For reference, we also show the performance of NeRF models. We also show the performance of NeX models take 24-30 hours of training for one-fourth resolution.

8 sequences	PSNR↑	MCSSIM↑	LPIPS ↓
Num-Epochs			
1	22.519 ± 4.197	0.799 ± 0.142	0.287 ± 0.126
2	22.647 ± 4.264	0.801 ± 0.140	0.281 ± 0.123
3	22.665 ± 4.305	0.801 ± 0.140	0.279 ± 0.123
4	22.718 ± 4.347	0.801 ± 0.140	0.278 ± 0.123
5	22.745 ± 4.369	0.801 ± 0.141	0.277 ± 0.123
6	22.756 ± 4.437	0.801 ± 0.141	0.275 ± 0.123
7	22.812 ± 4.499	0.802 ± 0.141	0.273 ± 0.123
8	22.754 ± 4.401	0.802 ± 0.141	0.272 ± 0.123
9	22.843 ± 4.455	0.802 ± 0.141	0.270 ± 0.123
10	22.868 ± 4.588	0.802 ± 0.141	0.269 ± 0.123
NeRF [27]	22.009 ± 3.148	0.757 ± 0.156	0.487 ± 0.180
NeRF-2M	21.457 ± 3.657	0.751 ± 0.155	0.498 ± 0.153
NeX [45]	22.292 ± 3.137	0.774 ± 0.152	0.423 ± 0.156

Table 16. **Shiny dataset and** $(K = 100, N = 100)$: We use 100 stereo-pairs and 100 3D points. We follow the evaluation criterion in Table 7. We observe that our model gets close to the best performing model in the first 60 seconds of training. For reference, we also show the performance of NeRF models. We also show the performance of NeX models take 24-30 hours of training for one-fourth resolution.

`Birds`: Training - {`0309:0709`}. Testing - {`0710:0809`}.

B.2. Held-out Camera Views

We held-out one camera view from the following 5 publicly available temporal sequences. Figure 21 contrasts our approach with Open4D on held-out camera views. Once again, we observe that our approach is able to capture details (facial and body details) better than Open4D.

Sequences

`WFD-01`: time - {`0011:0511`}. Test CAM-ID: {`4`}.
`WFD-02`: time - {`0400:0900`}. Test CAM-ID: {`4`}.
`JiuJitsu`: time - {`0001:0500`}. Test CAM-ID: {`0`}.
`Gangnam`: time - {`0100:0500`}. Test CAM-ID: {`4`}.
`Birds`: time - {`0309:0809`}. Test CAM-ID: {`7`}.

---

8 sequences	PSNR↑	MCSSIM↑	LPIPS ↓
Num-Epochs			
1	22.563 ± 4.269	0.799 ± 0.147	0.311 ± 0.141
2	22.734 ± 4.385	0.800 ± 0.146	0.303 ± 0.138
3	22.788 ± 4.413	0.801 ± 0.145	0.301 ± 0.137
4	22.838 ± 4.428	0.802 ± 0.145	0.298 ± 0.137
5	22.847 ± 4.467	0.802 ± 0.145	0.294 ± 0.135
6	22.878 ± 4.478	0.802 ± 0.145	0.293 ± 0.135
7	22.916 ± 4.543	0.802 ± 0.145	0.291 ± 0.134
8	22.934 ± 4.571	0.802 ± 0.145	0.289 ± 0.133
9	22.947 ± 4.603	0.803 ± 0.144	0.287 ± 0.133
10	23.016 ± 4.698	0.803 ± 0.144	0.285 ± 0.132
NeRF [27]	22.009 ± 3.148	0.757 ± 0.156	0.487 ± 0.180
NeRF-2M	21.457 ± 3.657	0.751 ± 0.155	0.498 ± 0.153
NeX [45]	22.292 ± 3.137	0.774 ± 0.152	0.423 ± 0.156

Table 17. **Shiny dataset and** $(K = 200, N = 200)$: We use 200 stereo-pairs and 200 3D points. We follow the evaluation criterion in Table 7. We observe that our model gets close to the best performing model in the first 60 seconds of training. For reference, we also show the performance of NeRF models. We also show the performance of NeX models take 24-30 hours of training for one-fourth resolution.

Open4D-24 sequences	**PSNR↑**	**SSIM↑**	**LPIPS ↓**
`no gamma`	17.034 ± 2.663	0.539 ± 0.099	0.539 ± 0.075
`no spatial`	18.387 ± 2.308	0.569 ± 0.089	0.527 ± 0.066
`no entropy`	17.893 ± 1.481	0.551 ± 0.074	0.573 ± 0.064
`direct MLP`	17.905 ± 1.808	0.562 ± 0.081	0.546 ± 0.064
`full`	17.948 ± 1.472	0.562 ± 0.077	0.534 ± 0.061
LLFF-12 sequences	**PSNR↑**	**SSIM↑**	**LPIPS ↓**
`no gamma`	18.831 ± 2.904	0.579 ± 0.133	0.444 ± 0.070
`no spatial`	20.536 ± 2.798	0.594 ± 0.135	0.429 ± 0.074
`no entropy`	20.792 ± 2.777	0.595 ± 0.136	0.435 ± 0.075
`direct MLP`	20.816 ± 2.796	0.593 ± 0.135	0.423 ± 0.076
`full`	20.834 ± 2.784	0.594 ± 0.136	0.426 ± 0.075
Shiny-8 sequences	**PSNR↑**	**MCSSIM↑**	**LPIPS ↓**
`no gamma`	17.724 ± 2.313	0.765 ± 0.138	0.288 ± 0.094
`no spatial`	21.047 ± 3.177	0.791 ± 0.140	0.266 ± 0.097
`no entropy`	22.529 ± 4.787	0.796 ± 0.142	0.258 ± 0.110
`direct MLP`	22.419 ± 4.757	0.794 ± 0.142	0.259 ± 0.105
`full`	22.430 ± 4.748	0.795 ± 0.142	0.256 ± 0.108

Table 18. : We study the influence of different components on our approach and see their benefits in our approach.

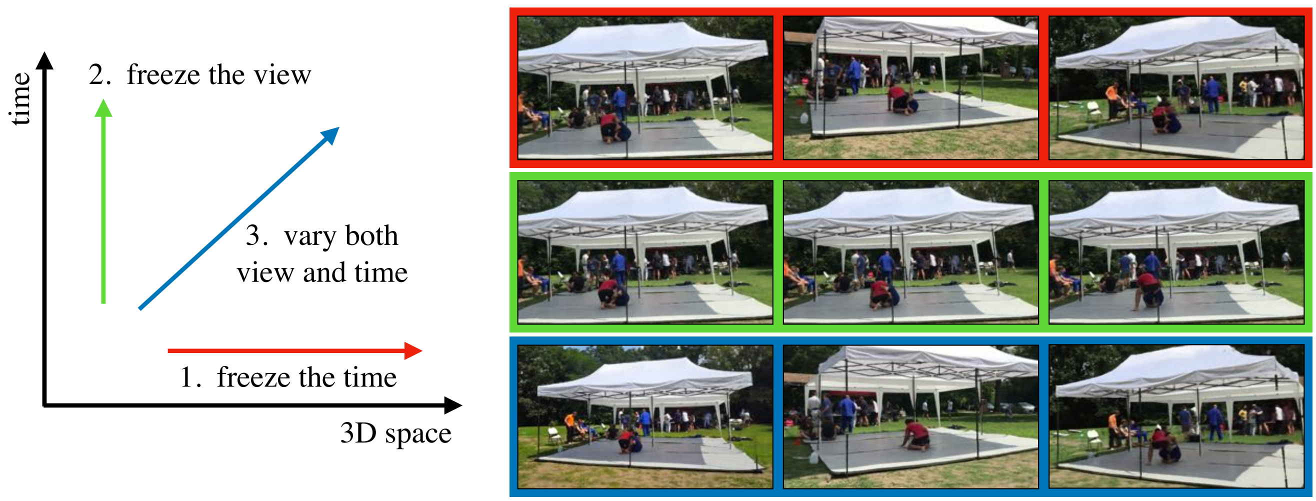


Figure 18. **4D view synthesis:** We demonstrate our approach for 4D view synthesis on the challenging Open4D dataset [3]. Without any background-foreground modeling or any modification, our approach learns to perform 4D visualization of dynamic events. **(1)**. We can freeze the time/event and move the view. **(2)**. We can freeze the view and see the event happening. **(3)**. We can vary both view and time.

Figure 19. **Naive Composition vs. Neural Composition for 4D View Synthesis:** We contrast the performance of naive composition using depth ordering with neural pixel composition for unseen temporal sequences. We observe that neural composition allows us to generate more realistic views in contrast to the naive composition.

C. More Analysis

We run more analysis on our model for various settings and study their impact on performance of our approach. In

Open4D Ours Ground Truth

Open4D Ours Ground Truth

Figure 20. **Unseen Temporal Sequences:** We contrast Open4D with ours for unseen temporal sequences. We observe that our approach allows us to capture details (such as details on human faces) consistently better than Open4D.

these experiments, we train the model for 10 epochs using LLFF-12 sequences (Sec A.2) and Shiny Dataset [45], and we use $K = 50$ stereo-pairs and $N = 50$ 3D points. We also use 24 sparse and unconstrained sequences from Open4D (Sec A.1).

Number of Filters: We vary the number of filters in our MLP model, $n_f = \{16, 32.64, 128, 256, 512\}$. Our default setting is $n_f = 256$. Table 19 shows the performance for Open4D-24 sequences, LLFF-12 sequences and Shiny dataset. The performance improves as we increase the number of filters. The use of $n_f = 256$ is a good balance between performance and size of model. We also observe that we can make extremely compact model at the loss of slight performance.

Number of Layers: We vary the number of layers in our MLP model, $n_l = \{1, 2, 3, 4, 5, 6\}$. Our default setting is $n_l = 5$. Table 20 shows the performance for Open4D-24 sequences, LLFF-12 sequences and Shiny dataset respectively.

Influence of Gamma: We use γ as a correction term that helps us to obtain sharp outputs. Table 18 (first row)) shows the performance for Open4D-24 sequences, LLFF-12 sequences and Shiny dataset. We observe that the additional γ term helps in inpainting the missing information.

Influence of Spatial Information: The second row in Table 18 shows the performance of our approach without using spatial information as an input to MLP. We observe that using spatial information enables us to provide smooth outputs and better inpaints missing information.

Influence of Uncertainty/Entropy: The third row in Ta-

Open4D Ours Ground Truth

Open4D Ours Ground Truth

Open4D Ours Ground Truth

Figure 21. **Held-Out Camera Views:** We contrast Open4D with ours for held-out camera views. Once again, we observe that our approach allows us to capture consistent details (such as details on human faces) better than Open4D.

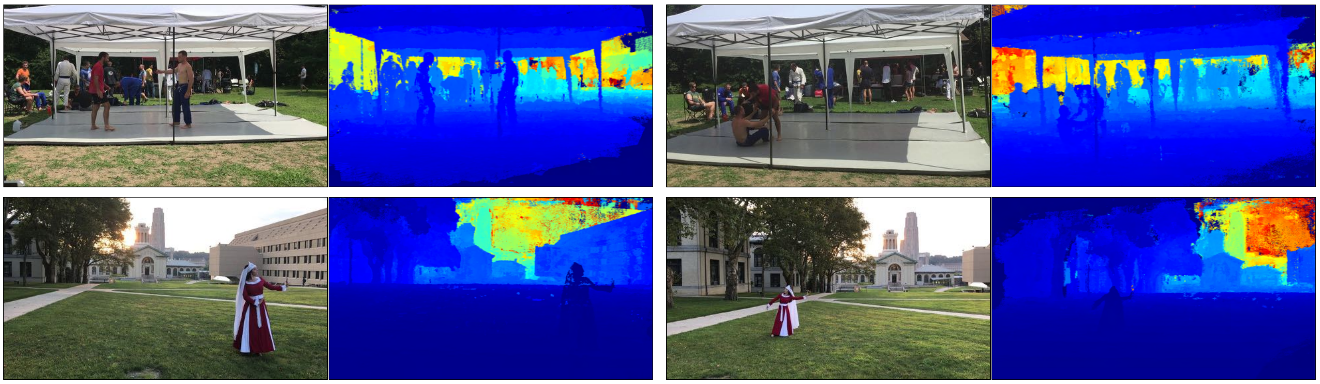


Figure 22. **Depth maps using learned MLPs**: We show depth map for images for various sequences. We use the learned MLPs to select the depth value corresponding to the max α_i value from an array of depth values for a pixel. The "jet blue" color corresponds to missing depth values for these images (e.g., the bottom right edge on the depth map of the first image).

Figure 23. **Dense 3D reconstruction from sparse views**: We show dense 3D point clouds computed using our approach for a specific time instant for four unconstrained multi-view sequences [3]. A user can easily explore the region by navigating the point clouds. We show random views of the point clouds.

Open4D-24 sequences	**PSNR↑**	**SSIM↑**	**LPIPS ↓**
Num-Filters			
16	17.716 $\pm$ 1.485	0.555 $\pm$ 0.079	0.538 $\pm$ 0.067
32	17.775 $\pm$ 1.453	0.557 $\pm$ 0.081	0.536 $\pm$ 0.069
64	17.828 $\pm$ 1.584	0.556 $\pm$ 0.082	0.541 $\pm$ 0.068
128	17.985 $\pm$ 1.561	0.561 $\pm$ 0.079	0.535 $\pm$ 0.066
`default` = 256	17.948 $\pm$ 1.472	0.562 $\pm$ 0.077	0.534 $\pm$ 0.061
512	18.091 $\pm$ 1.707	0.564 $\pm$ 0.081	0.534 $\pm$ 0.067
LLFF-12 sequences	**PSNR↑**	**SSIM↑**	**LPIPS ↓**
Num-Filters			
16	20.320 $\pm$ 2.401	0.591 $\pm$ 0.135	0.441 $\pm$ 0.07
32	20.667 $\pm$ 2.755	0.592 $\pm$ 0.136	0.433 $\pm$ 0.075
64	20.737 $\pm$ 2.818	0.594 $\pm$ 0.136	0.429 $\pm$ 0.075
128	20.771 $\pm$ 2.785	0.594 $\pm$ 0.136	0.428 $\pm$ 0.075
`default` = 256	20.834 $\pm$ 2.784	0.594 $\pm$ 0.136	0.426 $\pm$ 0.075
512	20.833 $\pm$ 2.781	0.594 $\pm$ 0.135	0.422 $\pm$ 0.075
Shiny-8 sequences	**PSNR↑**	**MCSSIM↑**	**LPIPS ↓**
Num-Filters			
16	21.202 $\pm$ 4.283	0.787 $\pm$ 0.144	0.273 $\pm$ 0.110
32	22.058 $\pm$ 4.150	0.794 $\pm$ 0.141	0.262 $\pm$ 0.105
64	22.188 $\pm$ 4.322	0.795 $\pm$ 0.142	0.261 $\pm$ 0.106
128	20.371$\pm$ 4.600	0.795 $\pm$ 0.142	0.259 $\pm$ 0.109
`default` = 256	22.430 $\pm$ 4.748	0.795 $\pm$ 0.142	0.256 $\pm$ 0.108
512	22.516 $\pm$ 4.809	0.795 $\pm$ 0.143	0.254 $\pm$ 0.108

Table 19. **Number of Filters**: We follow the evaluation criterion in Table 1 for Open4D-24 sequences, Table 3 for LLFF-12 sequences and Table 7 for Shiny-8 sequences. The performance improves as we increase the number of filters. We use $n_f = 256$ as a good balance between performance and size of model.

ble 18 shows the performance of our approach without using the uncertainty of the depth estimates ($\mathfrak{H}$). Using uncertainty provides slightly better performance.

Direct MLP: Finally, we observe the benefits of using depth explicitly in computing α to do a proper color composition. The fourth row in Table 18 shows the performance for Open4D-24 sequences, LLFF-12 sequences and Shiny dataset. We observe that using depth explicitly allows to do better view synthesis.

Open4D-24 sequences	**PSNR↑**	**SSIM↑**	**LPIPS ↓**
Num-Layers			
1	17.601 $\pm$ 1.779	0.527 $\pm$ 0.086	0.587 $\pm$ 0.101
2	18.014 $\pm$ 1.688	0.549 $\pm$ 0.081	0.555 $\pm$ 0.082
3	17.971 $\pm$ 1.621	0.555 $\pm$ 0.081	0.541 $\pm$ 0.073
4	18.066 $\pm$ 1.565	0.559 $\pm$ 0.081	0.535 $\pm$ 0.065
`default` = 5	17.948 $\pm$ 1.472	0.562 $\pm$ 0.077	0.534 $\pm$ 0.061
6	17.996 $\pm$ 1.669	0.562 $\pm$ 0.079	0.534 $\pm$ 0.067
LLFF-12 sequences	**PSNR↑**	**SSIM↑**	**LPIPS ↓**
Num-Layers			
1	20.433 $\pm$ 2.972	0.573 $\pm$ 0.138	0.450 $\pm$ 0.090
2	20.707 $\pm$ 2.874	0.588 $\pm$ 0.136	0.432 $\pm$ 0.081
3	20.833 $\pm$ 2.805	0.593 $\pm$ 0.136	0.424 $\pm$ 0.076
4	20.828 $\pm$ 2.818	0.594 $\pm$ 0.136	0.424 $\pm$ 0.075
`default` = 5	20.834 $\pm$ 2.784	0.594 $\pm$ 0.136	0.426 $\pm$ 0.075
6	20.820 $\pm$ 2.775	0.594 $\pm$ 0.136	0.426 $\pm$ 0.075
Shiny-8 sequences	**PSNR↑**	**MCSSIM↑**	**LPIPS ↓**
Num-Layers			
1	22.334 $\pm$ 4.485	0.793 $\pm$ 0.142	0.256 $\pm$ 0.108
2	22.447 $\pm$ 4.659	0.794 $\pm$ 0.143	0.254 $\pm$ 0.108
3	22.412 $\pm$ 4.688	0.795 $\pm$ 0.143	0.254 $\pm$ 0.108
4	20.391$\pm$ 4.730	0.794 $\pm$ 0.142	0.255 $\pm$ 0.108
`default` = 5	22.430 $\pm$ 4.748	0.795 $\pm$ 0.142	0.256 $\pm$ 0.108
6	22.367 $\pm$ 4.657	0.795 $\pm$ 0.143	0.256 $\pm$ 0.108

Table 20. **Number of Layers**: We follow the evaluation criterion in Table 1 for Open4D-24 sequences, Table 3 for LLFF-12 sequences and Table 7 for Shiny-8 sequences. We use $n_l = 5$ in this work.

References

[1] Edward H Adelson, James R Bergen, et al. *The plenoptic function and the elements of early vision*, volume 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, 1991. 1, 2

[2] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *ICCV*, 2009. 3

[3] Aayush Bansal, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa Narasimhan. 4d visualization of dynamic events from unconstrained multi-view videos. In *CVPR*, 2020. 3, 5, 7, 8, 11, 12, 19, 20, 23, 26

[4] 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 *ICCV*, 2021. 2

[5] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis from sparse views of novel scenes. In *CVPR*, 2021. 2

[6] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *CVPR*, 2022. 3, 5, 12, 21

[7] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, 2021. 3

[8] Yasutaka Furukawa, Carlos Hernandez, et al. *Multi-view stereo: A tutorial*. Foundation and Trends in Computer Graphics and Vision, 2015. 3

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 8

[10] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *SIGGRAPH*, 1996. 2

[11] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3, 4

[12] Jared Heinly. *Toward Efficient and Robust Large-Scale Structure-from-Motion Systems*. PhD thesis, The University of North Carolina at Chapel Hill, 2015. 3

[13] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6, 13

[14] Yoonwoo Jeong, Seokjun Ahn, Christopehr Choy, Animashree Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *ICCV*, 2021. 2

[15] Takeo Kanade and PJ Narayanan. Historical perspectives on 4d virtualized reality. In *CVPR Workshops*, 2006. 3

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[17] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 2017. 3

[18] Marc Levoy and Pat Hanrahan. Light field rendering. In *Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1996. 2

[19] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, 2021. 2

[20] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 2019. 1, 2

[21] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Trans. Graph.*, 2021. 3

[22] L McMillan. An image-based approach to three-dimensional computer graphics. *Ph. D. Dissertation, UNC Computer Science*, 1999. 1

[23] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH*, 1995. 1, 2

[24] Alexandre Meyer and Fabrice Neyret. Interactive volumetric textures. In *Eurographics Workshop on Rendering Techniques*, pages 157–168. Springer, 1998. 3

[25] Marko Mihajlovic, Aayush Bansal, Michael Zollhoefer, Siyu Tang, and Shunsuke Saito. KeypointNeRF: Generalizing image-based volumetric avatars using relative spatial encoding of keypoints. In *ECCV*, 2022. 2

[26] 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.*, 2019. 3, 4, 5, 6, 11, 13, 17, 19, 20, 21

[27] 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 *ECCV*, 2020. 1, 2, 3, 5, 6, 11, 12, 13, 19, 21, 22

[28] Ren Ng. *Light field photography with a hand-held plenoptic camera*. PhD thesis, Stanford University, 2005. 2

[29] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 2

[30] Manuel M Oliveira and Gary Bishop. Image-based objects. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 191–198, 1999. 3

[31] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Trans. Graph.*, 2017. 3

[32] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *CVPR*, 2021. 2, 3

[33] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, 2020. 2, 3

[34] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, 2021. 2, 3

[35] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth pri-

ors for neural radiance fields from sparse input views. In *CVPR*, 2022. 2

[36] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 3

[37] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 3

[38] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998. 1, 3

[39] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *Visual Communications and Image Processing*, 2000. 2

[40] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Neurips*, 2019. 17

[41] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 2006. 3

[42] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. State of the Art on Neural Rendering. *Computer Graphics Forum (EG STAR 2020)*, 2020. 2

[43] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in neural rendering, 2021. 2

[44] 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 *CVPR*, 2021. 2

[45] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, 2021. 6, 7, 11, 13, 17, 19, 20, 22, 24

[46] Liwen Wu, Jae Yong Lee, Anand Bhattad, Yuxiong Wang, and David Forsyth. Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In *CVPR*, 2022. 2

[47] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *CVPR*, 2019. 2, 4, 6, 11

[48] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. Pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 2

[49] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[50] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5, 6, 13

[51] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Trans. Graph.*, 2018. 3