SparseCam4D: Spatio-Temporally Consistent 4D Reconstruction from Sparse Cameras

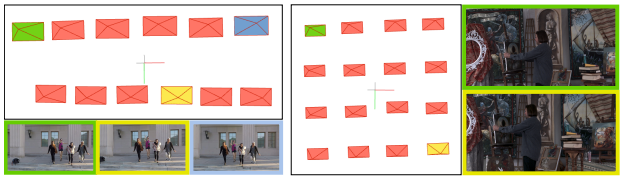
Supplementary Material

1. Implementation Details

Our framework is implemented in PyTorch, using the RealTime4DGS codebase as the foundation. To assess the versatility of our method, we did not apply dataset-specific tuning; instead, a unified training schedule was adopted across all datasets. For the optimization of 4D Gaussians, we strictly follow the official RealTime4DGS implementation. The Adam optimizer is employed throughout. The learning rate for each feature plane starts at 1.6×10^{-3} and exponentially decays to 1.6×10^{-4}. The tiny MLP decoder is trained with an initial learning rate of 1.6×10^{-4}, which decays to 1.6×10^{-5}. In the distortion field, we configure the base resolution of the spatial axes to 64. The basis resolution for temporal axis is set according to the total number of frames in the training sequence, while the pose axis is set to the number of generated views. For the multi-resolution feature structure, we employ a scale factor of 2 for each feature plane. We also find that expanding this factor from 2 to [2,4] will contribute to rendering quality at the expense of higher training cost. To stabilize the optimization process, we introduce a warm-up stage during the first 1000 iterations. In this phase, only the 4D Gaussian primitives are optimized, while both the distortion field and camera pose optimization remain frozen. After this warm-up stage, joint optimization over the distortion field, camera poses, and Gaussian attributes is enabled. This staged training scheme prevents instability in the early iterations and ensures smoother convergence.

1.1. Camera Selection

1.1.1. Training Cameras



(a) Jumping (b) Painter

Figure 1. **Visualization of selected training cameras.** The non-red frustums denote the cameras used for training, corresponding to the images shown at the bottom and on the right respectively. The remaining red frustums indicate the cameras used for evaluation.

We construct a minimal subset of cameras C_s from the original set C (usually containing 12-21 cameras) to serve as the training views. The subset C_s is required to satisfy two conditions: 1) C_s should sufficiently cover the scene content observed by C; 2) Since VDM will interpolate additional viewpoints within C_s, the per-frame overlap among cameras in C_s is expected to be similar in order to ensure uniform scene coverage. Based on these criteria, we propose a greedy selection strategy that jointly considers visible point coverage and field-of-view overlap. The algorithm begins by choosing the camera located at the top-left

Algorithm 1: Camera Subset Selection with Overlap-Constrained Greedy Expansion

Input:
$P[i]$: Visible point sets for each camera i
overlap$[i][j]$: overlap matrix between camera i and camera j, defined as $\text{overlap}[i][j] = \frac{|P_i \cap P_j|}{|P_i \cup P_j|}$
$o_{\min}$: overlap threshold
τ: target coverage
μ: overlap weight
Output: Selected camera subset for training C_s

foreach *camera* i **do**
  $G_{\text{nbr}}[i] \leftarrow \{j \mid \text{overlap}[i][j] \geq o_{\min}, j \neq i\}$
end
$P_{\text{total}} \leftarrow \bigcup_i P[i]$ `// all visible points`
$U \leftarrow P_{\text{total}}$ `// uncovered points`
$c_0 \leftarrow$ top-left camera
$C_s \leftarrow \{c_0\}$
$U \leftarrow U \setminus P[c_0]$
$c_{\text{cur}} \leftarrow c_0$
while $1 - |U|/|P_{total}| < \tau$ **do**
  $\mathcal{N} \leftarrow \{v \in G_{\text{nbr}}[c_{\text{cur}}] \mid v \notin C_s\}$
  if $\mathcal{N} = \emptyset$ **then**
    break
  end
  foreach $v \in \mathcal{N}$ **do**
    $\text{gain}(v) \leftarrow |P[v] \cap U|$
    $\text{score}(v) \leftarrow \text{gain}(v) \cdot (1 + \mu \cdot \text{overlap}[c_{\text{cur}}][v])$
  end
  $v^\star \leftarrow \arg\max_{v \in \mathcal{N}} \text{score}(v)$
  if $|P[v^\star] \cap U| = 0$ **then**
    break
  end
  $C_s \leftarrow C_s \cup \{v^\star\}$
  $U \leftarrow U \setminus P[v^\star]$
  $c_{\text{cur}} \leftarrow v^\star$
end
return C_s

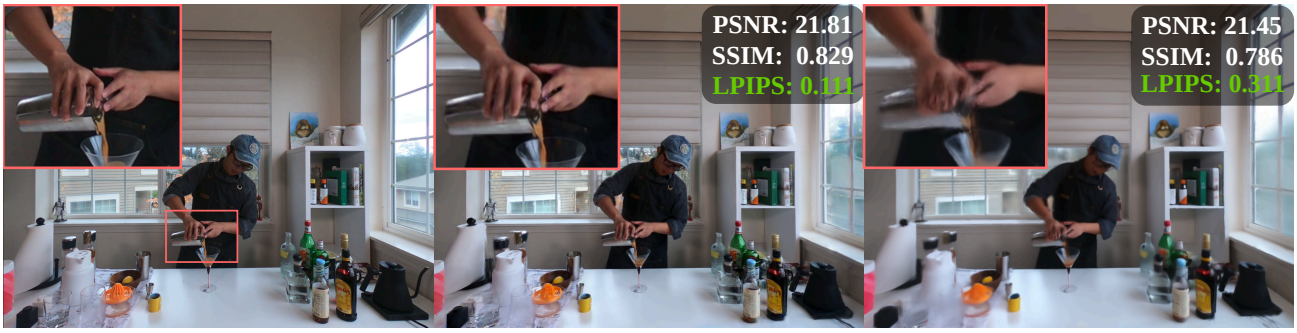


Figure 2. **Reliability of PSNR vs. LPIPS in Ablation Experiments.** We find that directly using generated-view reconstruction introduces severe oversmoothness that, paradoxically, favors PSNR computation, preventing PSNR from accurately reflecting changes in reconstruction quality. In contrast, LPIPS effectively captures the variations in reconstruction quality under these ablation settings.

position as the starting view. At each iteration, we construct a set of candidate cameras whose overlap with the current camera exceeds the minimum adjacency threshold o_{min}. For each candidate v, we compute a score that balances newly covered points and the geometric consistency with the current camera:

$$\text{score}(v) = |P[v] \cap U| \cdot (1 + \mu \cdot \text{overlap}[c_{cur}, v]), \quad (1)$$

where U denotes the set of currently uncovered points and $\mu = 0.05$ is the overlap weight. The first term encourages maximal expansion of scene coverage, while the second ensures that consecutive selected cameras maintain sufficient overlap for reliable view interpolation using generative models. The candidate with the highest score is added to the selected subset, and the uncovered set is updated accordingly. The process iterates until the target coverage ratio is achieved or no valid candidates remain, yielding the final camera subset. We summarize the details in Algorithm 1. Tab. 6-8 provides camera index references for training and evaluation. Fig. 1 illustrates the selected training cameras from two example scenes *Jumping* and *Painter*, with the training cameras highlighted in different colors.

1.1.2. Evaluation Cameras

To fully reflect the rendering quality of each method under the sparse-view setting, we do not adopt the original approach of evaluation using only one single selected camera. Instead, evaluation is performed on all remaining cameras.

2. Metrics selection in Ablation Study

In the ablation study, SSIM and LPIPS are adopted as the primary evaluation metrics instead of PSNR. This choice is motivated by two considerations. First, SSIM and LPIPS are more sensitive to structural details, with LPIPS in particular capturing perceptual differences in texture, sharpness, and local geometry, whereas PSNR primarily reflects pixel-wise color discrepancies and is less aligned with perceptual image quality. Second, as illustrated in Fig. 2, the rendered images produced without distortion field exhibit substantial generative inconsistency, leading to overly smooth and blurred outputs. Such behavior tends to favor PSNR, resulting in only a marginal difference. In contrast, SSIM and LPIPS, especially LPIPS, capture the degradation of structural fidelity and perceptual quality, revealing the significant discrepancy.

3. Additional Evaluation Results

3.1. Evaluation under Different Sparsity Levels.

We compare our method with the baseline 4DGS approach, *i.e.* 4DGaussian, under different sparsity levels, and the results are reported in Tab. 1. Our method outperforms the baseline across all camera-view configurations, further demonstrating its practical robustness.

Table 1. **Performance with different number of camera views.** We set 4DGaussian as the baseline model for dynamic scene reconstruction and evaluate the rendering quality with different number of training views.

	3 views			6 views			9 views		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
4DGaussian	16.72	0.435	0.546	19.86	0.581	0.448	21.41	0.601	0.430
Ours	21.56	0.656	0.264	23.58	0.687	0.243	24.72	0.751	0.205

3.2. Efficiency Comparison with RealTime4DGS

Since our implementation is based on the RealTime4DGS codebase, we compare our method against it in terms of rendering fps to further evaluate efficiency. As shown in Tab 2, our approach maintains the similar rendering speed as RealTime4DGS. Notably, the proposed distortion field is discarded after training, so inference complexity remains comparable to RealTime4DGS.

3.3. Per-scene Breakdown Results

In Tab. 4, Tab. 3, and Tab. 5, we provide a breakdown of the metrics for different scenes in the Neural 3D Video, Tech-

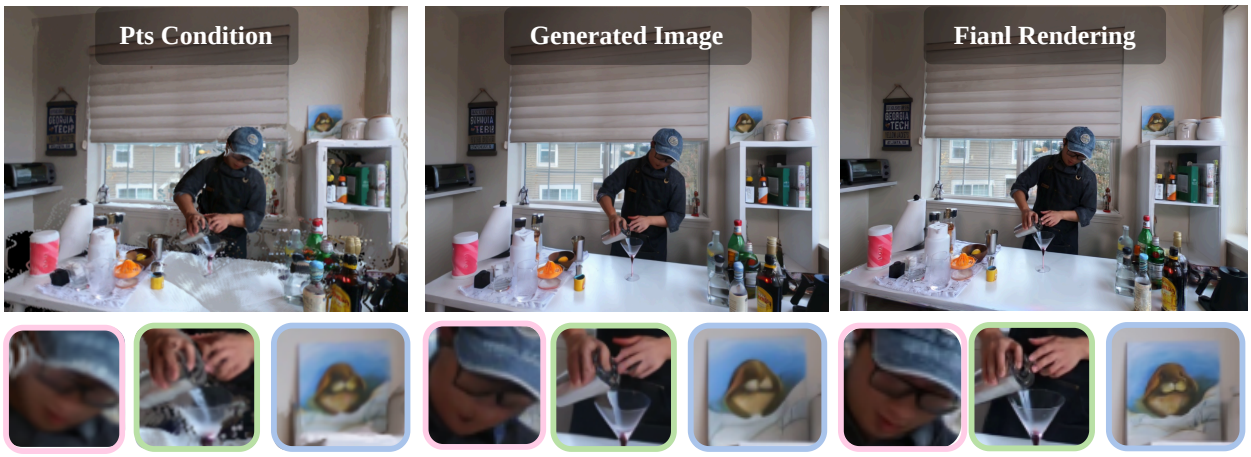


Figure 3. Visualization of the point-cloud rendering condition, generation prior, and final results.

Table 2. **Efficiency Comparison with RealTime4DGS.**

	Coffee Martini	Playground	Average
RealTime4DGS	11.48fps	11.51fps	11.50fps
Ours	11.82fps	11.97fps	11.90fps

nicolor, and Nvidia Dynamic Scenes datasets, respectively. As shown in the quantitative results, our method demonstrates clear superiority over the other five approaches. This indicates that our method effectively leverages the generated priors while mitigating the inconsistency they may introduce. In addition, since our evaluation is conducted on all cameras excluding those used for training, we provide more detailed per-camera metrics, as reported in Tab. 6-8.

3.4. Additional Qualitative Results

In Fig. 6, Fig. 7-10, and Fig. 12, 11, we showcase additional rendering comparisons with those methods aforementioned at different timesteps. Our method consistently recovers high-quality and coherent renderings over time, demonstrating its effectiveness in dynamic scenes with sparse observation. Please zoom in for more details.

4. More Details for Generated Images

In our main experiments, we adopt ViewCrafter to provide additional observations, from which 20–25 generated views are uniformly sampled for training. ViewCrafter conditions its diffusion model on point-cloud renderings obtained from Dust3R, given two input images along with the target trajectory. However, we find that the quality of generated images is highly dependent on the quality of the point-cloud renderings. To improve the generation quality, we replace Dust3r with a stronger 3D foundation model, VGGT, to produce more reliable conditioning signals. Despite this improvement, the generated sequences still exhibit spatio-temporal inconsistencies that hinder their direct use for 4D reconstruction, as illustrated in Fig 3. Such inconsistencies can lead to severe over-blurring in the reconstructed results, as shown in Fig. 2 (right).

As for the ablation study with ReCamMaster, unlike ViewCrafter, which leverages point-cloud renderings as trajectory priors, ReCamMaster directly conditions on camera parameters to generate a video sequence of 81 frames. Without explicit geometric constraints, the generated images from vanilla ReCamMaster suffer from severe hallucinations, making them unsuitable for reconstruction. To address this, we finetune ReCamMaster by incorporating point-cloud renderings as an additional conditioning signal, analogous to ViewCrafter.

4.1. Failure Case

Although our method substantially mitigates the inter-frame inconsistencies caused by distortions in generated images, the final performance still partially depends on the quality of the generated images themselves. As illustrated in Fig. 4, when the generated images are of poor quality—typically due to out-of-domain inputs or generative hallucinations—such as the severely deformed human body and the additional ghosting artifacts shown in the figure, addressing only the inter-image inconsistency is insufficient to achieve satisfactory reconstruction results. Without considering training cost, incorporating the generative model into the reconstruction pipeline for joint or iterative optimization is a promising direction for future exploration.

5. More visualization

To gain an intuitive understanding of the STDF training results, we visualize the full feature maps (16 channels) of each plane in the CoffeMartini scene in Fig. 5. The activated regions vary across different dimensions, indicating the intertwined nature of spatial and temporal deformations.

To further interpret the outputs, we render the deformation predictions of STDF as an additional attribute field. This allows us to clearly observe the regions in the scene

Figure 4. **Visualization of one failure case.** From left to right are the source image and point-cloud–rendering condition provided to ViewCrafter, the corresponding generated image, and a Gaussian rendering near that generated image. Since ViewCrafter is primarily trained on scene-centric data with few human subjects, this example suffers from an out-of-domain issue. In addition, the quality of the point-cloud–rendering condition is relatively low. These factors jointly lead to low-quality generated images, which in turn degrade the Gaussian reconstruction quality on the human body.

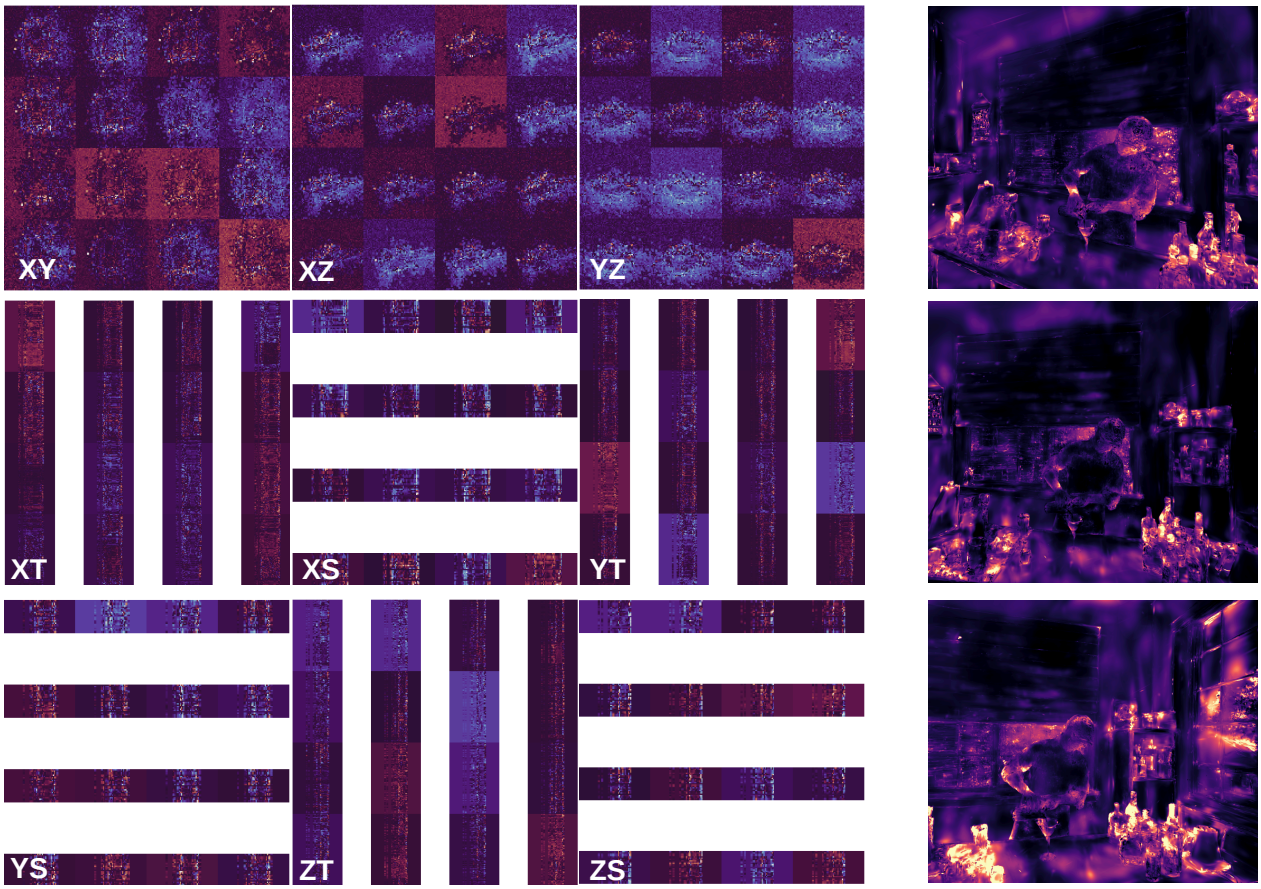


Figure 5. Visualization of Spatio-Temporal Distortion Field

where distortions occur. Through this visualization approach, we can intuitively perceive the spatio-temporal distortions present in the 4D scene.

Figure 6. **Additional qualitative comparison results on *Technicolor* Dataset.** We highlight the geometric completeness of background structures, bottom decorations, and the red bridge across different viewpoints, as well as the temporal consistency of dynamic regions such as human faces, the 'Happy Birthday' text, and the toy train.

Figure 7. **Additional qualitative comparison results on *Neural 3D Video* Dataset (part I).**

Figure 8. **Additional qualitative comparison results on *Neural 3D Video* Dataset (part II).**

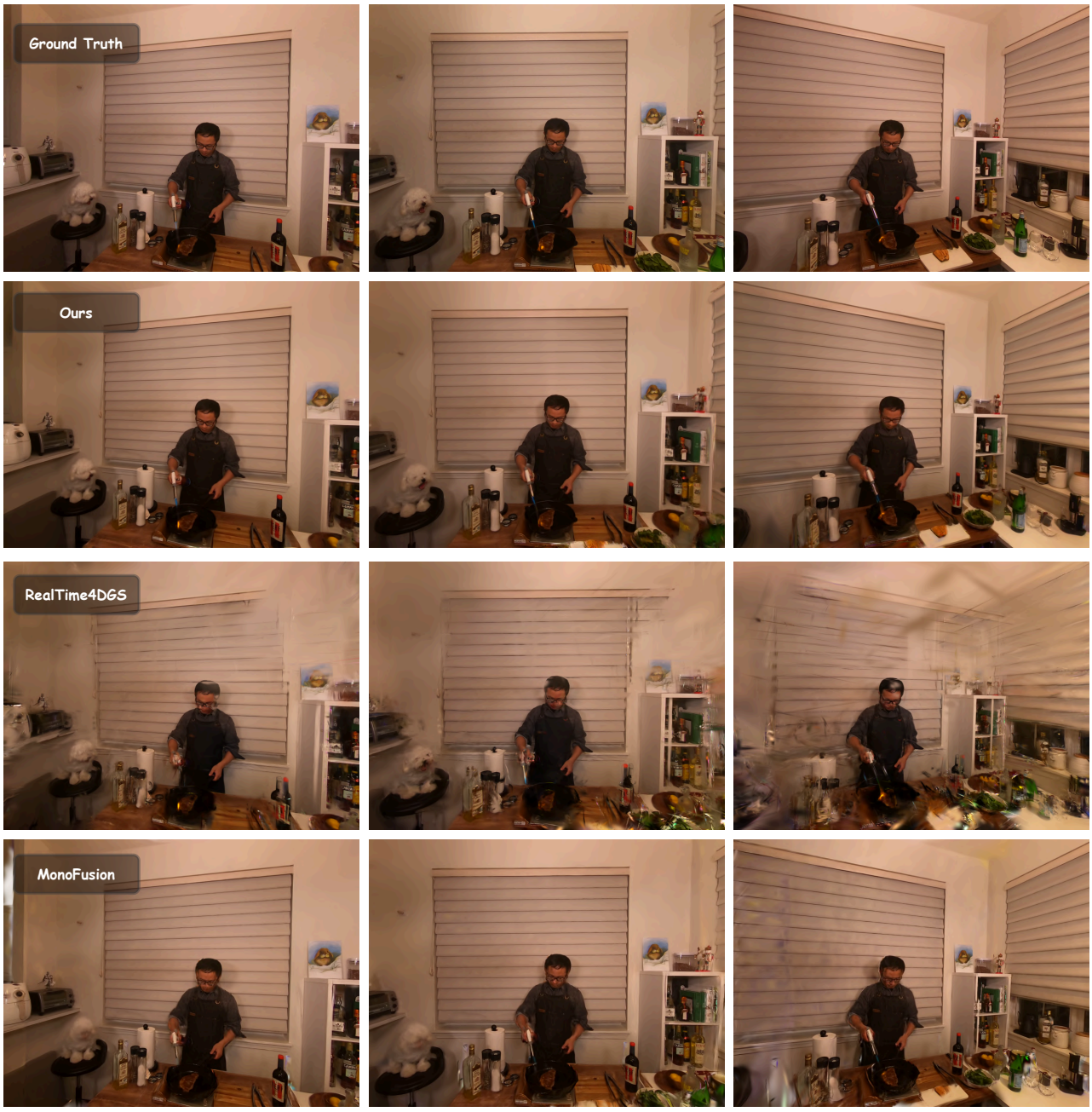


Figure 9. **Additional qualitative comparison results on *Neural 3D Video* Dataset (part III).**

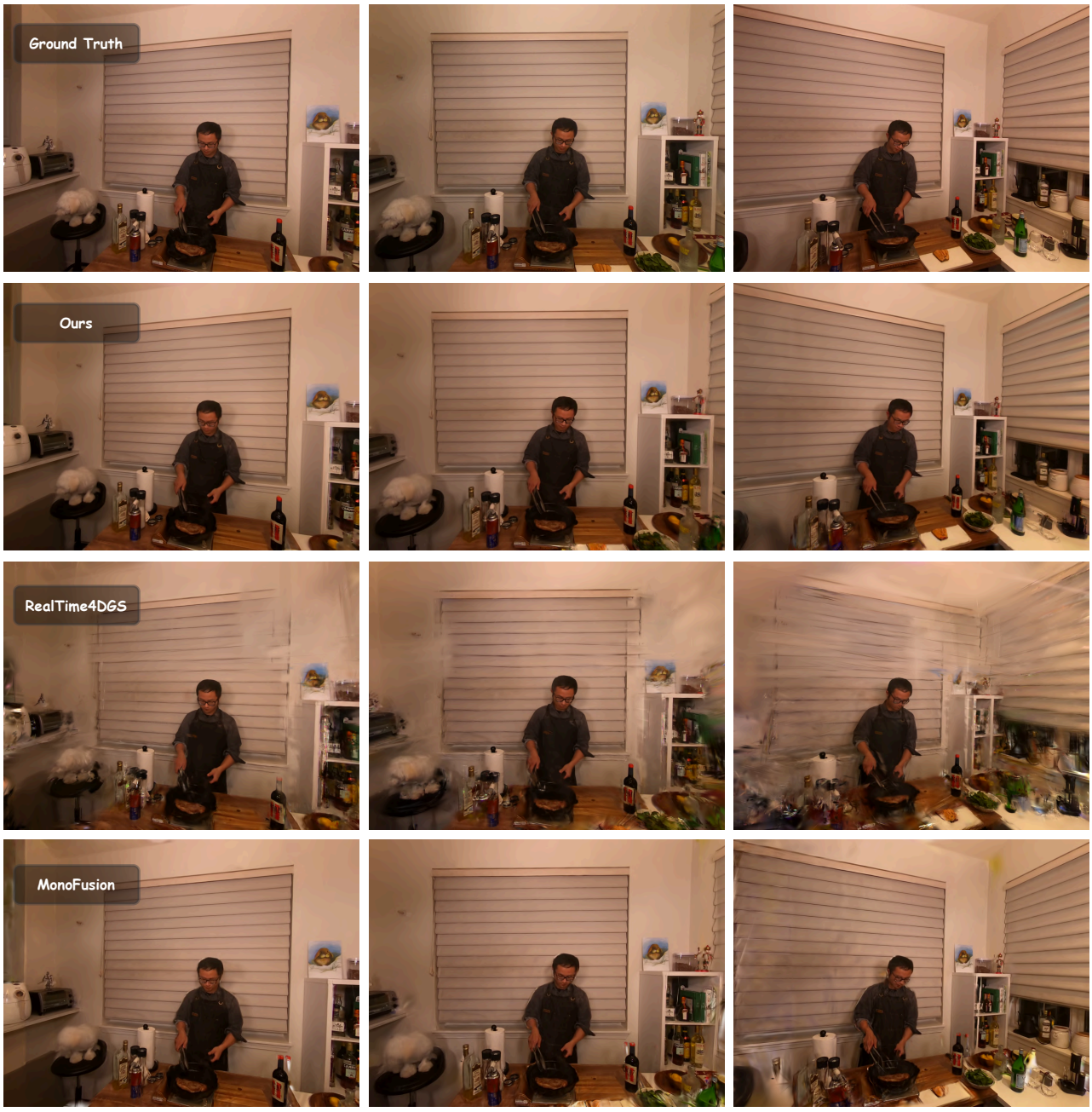


Figure 10. **Additional qualitative comparison results on *Neural 3D Video* Dataset (part IV).**

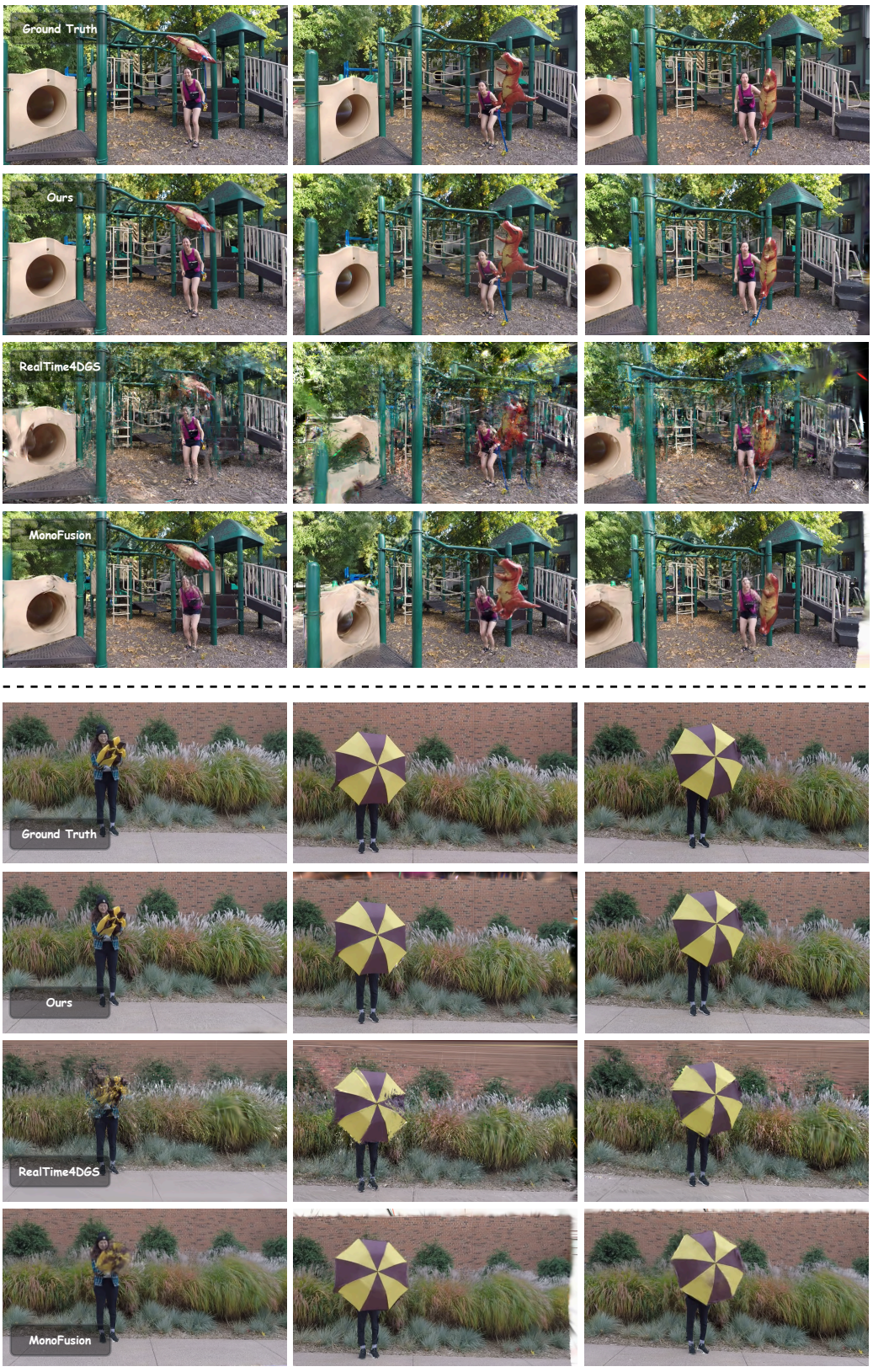


Figure 11. **Additional qualitative comparison results on *Nvidia Dynamic Scenes* Dataset (part I).** In the top panel, we emphasize the geometric completeness of the left archway and the right escalator across different viewpoints, as well as the temporal consistency of dynamic regions such as the human body, balloon, and the blue tether. In the bottom panel, we highlight the fine details of background grass, shoes, and umbrella textures, together with accurate motion fitting under fast movement in the leftmost column.

Figure 12. **Additional qualitative comparison results on *Nvidia Dynamic Scenes* Dataset (part II).**

Table 3. PSNR, SSIM, and LPIPS of our framework on Technicolor, with cam00, cam09 and cam15 served as training views.

Technicolor - PSNR						
	Fabien	Theater	Train	Birthday	Painter	Average
HyperReel	13.62	14.24	15.44	12.28	15.13	14.14
4DGaussians	18.31	17.53	16.72	11.97	16.48	16.20
4DRotor	15.21	16.00	14.01	12.42	16.62	14.85
RealTime4DGS	13.00	16.27	16.59	18.45	18.76	16.53
MonoFusion	17.86	19.48	15.12	16.47	20.92	17.97
Ours	25.23	22.44	21.56	20.11	26.39	23.15

Technicolor - SSIM						
	Fabien	Theater	Train	Birthday	Painter	Average
HyperReel	0.653	0.434	0.351	0.341	0.484	0.453
4DGaussians	0.704	0.552	0.435	0.335	0.499	0.505
4DRotor	0.625	0.479	0.270	0.307	0.450	0.426
RealTime4DGS	0.520	0.451	0.438	0.586	0.556	0.510
MonoFusion	0.737	0.590	0.392	0.524	0.648	0.578
Ours	0.814	0.686	0.656	0.683	0.802	0.728

Technicolor - LPIPS						
	Fabien	Theater	Train	Birthday	Painter	Average
HyperReel	0.514	0.666	0.595	0.664	0.642	0.616
4DGaussians	0.492	0.539	0.546	0.620	0.564	0.552
4DRotor	0.574	0.571	0.589	0.612	0.560	0.581
RealTime4DGS	0.614	0.623	0.505	0.437	0.529	0.542
MonoFusion	0.415	0.380	0.356	0.339	0.271	0.352
Ours	0.340	0.379	0.264	0.274	0.236	0.299

Table 4. PSNR, SSIM, and LPIPS of our framework on Neural 3D Video, with cam01, cam05 and cam10 served as training views.

Neural 3D Video - PSNR							
	coffee martini	cook spinach	cut roasted beef	flame salmon	flame steak	sear steak	Average
HyperReel	14.07	16.44	17.04	10.64	17.57	17.28	15.63
4DGaussians	14.49	17.94	18.34	14.83	19.53	18.57	17.40
4DRotor	15.86	19.65	18.51	16.10	19.69	19.05	18.20
RealTime4DGS	15.07	17.95	17.04	14.65	19.57	18.86	17.31
MonoFusion	15.59	19.71	19.15	15.42	20.02	19.91	18.43
Ours	18.40	23.93	22.43	18.92	23.30	23.56	21.91

Neural 3D Video - SSIM							
	coffee martini	cook spinach	cut roasted beef	flame salmon	flame steak	sear steak	Average
HyperReel	0.520	0.586	0.635	0.452	0.644	0.633	0.582
4DGaussian	0.593	0.664	0.696	0.611	0.736	0.716	0.673
4DRotor	0.649	0.741	0.715	0.651	0.753	0.744	0.708
RealTime4DGS	0.590	0.670	0.640	0.569	0.708	0.700	0.649
MonoFusion	0.643	0.780	0.777	0.630	0.788	0.784	0.738
Ours	0.729	0.832	0.792	0.726	0.820	0.818	0.789

Neural 3D Video - LPIPS							
	coffee martini	cook spinach	cut roasted beef	flame salmon	flame steak	sear steak	Average
HyperReel	0.540	0.499	0.447	0.604	0.466	0.459	0.500
4DGaussians	0.401	0.320	0.325	0.362	0.258	0.270	0.320
4DRotor	0.406	0.334	0.359	0.371	0.334	0.336	0.357
RealTime4DGS	0.471	0.428	0.459	0.485	0.405	0.412	0.442
MonoFusion	0.367	0.228	0.236	0.354	0.229	0.231	0.270
Ours	0.298	0.232	0.259	0.291	0.237	0.238	0.258

Table 5. PSNR, SSIM, and LPIPS of our framework on Nvidia Dynamic Scenes, with cam01, cam06 and cam10 as training views.

Nvidia Dynamic Scenes - PSNR							
	jumping	balloon1	balloon3	playground	skating	umbrella	Average
HyperReel	24.45	15.46	22.58	15.41	21.05	24.74	19.88
4DGaussians	17.38	16.96	14.56	12.75	19.70	19.52	16.81
4DRotor	21.90	14.55	20.33	10.95	26.28	22.29	19.38
RealTime4DGS	17.85	18.36	19.07	13.96	18.52	19.71	17.91
MonoFusion	20.72	20.31	19.81	16.86	23.55	20.07	20.22
Ours	24.36	25.31	25.33	19.64	28.73	25.42	24.81

Nvidia Dynamic Scenes - SSIM							
	jumping	balloon1	balloon3	playground	skating	umbrella	Average
HyperReel	0.782	0.350	0.632	0.281	0.669	0.629	0.528
4DGaussians	0.525	0.372	0.264	0.166	0.607	0.299	0.372
4DRotor	0.731	0.311	0.497	0.137	0.814	0.556	0.508
RealTime4DGS	0.585	0.533	0.509	0.267	0.649	0.333	0.479
MonoFusion	0.687	0.563	0.542	0.499	0.819	0.428	0.590
Ours	0.793	0.810	0.791	0.672	0.906	0.794	0.794

Nvidia Dynamic Scenes - LPIPS							
	jumping	balloon1	balloon3	playground	skating	umbrella	Average
HyperReel	0.256	0.537	0.266	0.491	0.450	0.238	0.396
4DGaussians	0.454	0.473	0.609	0.608	0.449	0.498	0.516
4DRotor	0.245	0.601	0.392	0.670	0.166	0.260	0.389
RealTime4DGS	0.450	0.371	0.380	0.499	0.422	0.432	0.426
MonoFusion	0.241	0.151	0.174	0.206	0.134	0.248	0.192
Ours	0.170	0.127	0.142	0.190	0.121	0.148	0.150

Table 6. **PSNR, SSIM, and LPIPS of our framework on Technicolor.** Three cameras (cam00, cam09 and cam15) are served as training views for scene *Fabien*, *Theater* and *Train*, while only two cameras (cam00 and cam15) are training views for scene *Birthday* and *Painter*.

Technicolor - PSNR

	Fabien	Theater	Train	Birthday	Painter	Average
cam01	25.581	21.231	23.456	19.562	26.322	23.23
cam02	21.886	19.203	21.202	18.227	25.736	21.251
cam03	18.844	18.5	19.468	17.692	27.939	20.488
cam04	28.676	23.089	23.59	20.991	27.423	24.754
cam05	28.162	23.422	23.833	20.837	27.249	24.701
cam06	23.838	20.949	21.0	19.824	25.58	22.238
cam07	20.735	19.916	19.228	19.306	25.717	20.98
cam08	28.175	23.997	23.649	20.245	25.817	24.377
cam10	25.706	23.722	20.932	20.962	26.613	23.587
cam11	24.939	23.549	21.013	21.682	27.519	23.741
cam12	25.122	23.568	20.321	19.482	23.795	22.457
cam13	27.505	25.954	21.626	20.643	26.436	24.433
cam14	28.818	24.666	20.962	21.986	26.913	24.669
Average	25.23	22.444	21.56	20.111	26.389	23.147

Technicolor - SSIM

	Fabien	Theater	Train	Birthday	Painter	Average
cam01	0.846	0.665	0.71	0.69	0.789	0.74
cam02	0.801	0.609	0.638	0.634	0.803	0.697
cam03	0.75	0.556	0.644	0.651	0.86	0.692
cam04	0.835	0.697	0.726	0.715	0.823	0.759
cam05	0.842	0.755	0.762	0.709	0.824	0.779
cam06	0.821	0.675	0.658	0.682	0.784	0.724
cam07	0.776	0.62	0.619	0.675	0.761	0.69
cam08	0.81	0.725	0.741	0.683	0.787	0.749
cam10	0.825	0.701	0.539	0.687	0.797	0.71
cam11	0.82	0.687	0.665	0.726	0.826	0.745
cam12	0.778	0.689	0.562	0.64	0.761	0.686
cam13	0.83	0.781	0.657	0.669	0.819	0.751
cam14	0.85	0.753	0.612	0.713	0.79	0.744
Average	0.814	0.686	0.656	0.683	0.802	0.728

Technicolor - LPIPS

	Fabien	Theater	Train	Birthday	Painter	Average
cam01	0.305	0.355	0.24	0.245	0.218	0.273
cam02	0.355	0.431	0.301	0.303	0.231	0.324
cam03	0.394	0.484	0.328	0.319	0.218	0.349
cam04	0.34	0.342	0.225	0.24	0.218	0.273
cam05	0.323	0.329	0.214	0.253	0.221	0.268
cam06	0.333	0.399	0.27	0.275	0.235	0.302
cam07	0.367	0.447	0.307	0.291	0.266	0.336
cam08	0.357	0.343	0.211	0.281	0.249	0.288
cam10	0.307	0.36	0.266	0.259	0.24	0.287
cam11	0.304	0.38	0.249	0.235	0.225	0.278
cam12	0.404	0.387	0.315	0.323	0.266	0.339
cam13	0.336	0.316	0.258	0.295	0.247	0.29
cam14	0.287	0.347	0.255	0.241	0.238	0.274
Average	0.34	0.379	0.264	0.274	0.236	0.299

Table 7. **PSNR, SSIM, and LPIPS of our framework on Neural 3D Video.** Three cameras (cam01, cam05 and cam10) are training views.

Neural 3D Video - PSNR

	coffee martini	cook spinach	cut roasted beef	flame salmon	flame steak	sear steak	Average
cam00	21.654	28.114	26.901	22.323	27.558	27.744	25.715
cam02	25.236	25.926	25.206	24.866	25.472	25.365	25.345
cam03	-	25.194	24.231	-	25.489	25.891	25.201
cam04	25.477	31.417	-	25.741	31.034	30.641	28.862
cam06	20.31	27.966	26.145	21.048	27.624	27.609	25.117
cam07	19.602	26.638	24.25	20.449	25.344	25.759	23.674
cam08	19.134	26.957	24.089	19.889	26.177	26.647	23.816
cam09	20.957	26.997	26.072	21.31	27.161	27.446	24.99
cam11	14.669	20.251	19.145	15.398	19.277	19.349	18.015
cam12	13.623	19.353	18.495	14.841	18.082	18.604	17.166
cam13	15.547	20.289	20.071	16.839	19.371	20.014	18.688
cam14	16.705	22.012	22.252	17.309	20.462	21.098	19.973
cam15	-	22.182	21.68	16.824	20.618	21.2	20.501
cam16	15.457	21.483	20.783	16.689	20.086	20.699	19.199
cam17	-	22.11	20.975	-	20.894	21.646	21.406
cam18	14.762	21.959	20.944	15.755	20.997	21.37	19.298
cam19	17.115	22.524	21.444	17.245	22.972	23.031	20.722
cam20	15.736	19.319	18.569	16.248	20.715	20.014	18.434
Average	18.399	23.927	22.426	18.923	23.296	23.563	21.905

Neural 3D Video - SSIM

	coffee martini	cook spinach	cut roasted beef	flame salmon	flame steak	sear steak	Average
cam00	0.832	0.907	0.897	0.841	0.918	0.915	0.885
cam02	0.875	0.917	0.906	0.87	0.907	0.905	0.897
cam03	-	0.919	0.893	-	0.903	0.911	0.906
cam04	0.889	0.941	-	0.884	0.945	0.94	0.92
cam06	0.767	0.898	0.862	0.774	0.899	0.894	0.849
cam07	0.735	0.876	0.814	0.754	0.859	0.861	0.816
cam08	0.735	0.874	0.8	0.754	0.874	0.87	0.818
cam09	0.786	0.892	0.868	0.8	0.893	0.892	0.855
cam11	0.673	0.751	0.712	0.643	0.735	0.723	0.706
cam12	0.65	0.717	0.684	0.627	0.689	0.69	0.676
cam13	0.67	0.772	0.76	0.676	0.756	0.753	0.731
cam14	0.726	0.815	0.804	0.719	0.792	0.787	0.774
cam15	-	0.797	0.772	0.691	0.779	0.774	0.762
cam16	0.669	0.772	0.74	0.676	0.744	0.746	0.724
cam17	-	0.772	0.728	-	0.741	0.752	0.748
cam18	0.613	0.767	0.716	0.629	0.739	0.732	0.699
cam19	0.667	0.785	0.758	0.656	0.79	0.792	0.741
cam20	0.646	0.795	0.757	0.62	0.799	0.787	0.734
Average	0.729	0.832	0.792	0.726	0.82	0.818	0.789

Neural 3D Video - LPIPS

	coffee martini	cook spinach	cut roasted beef	flame salmon	flame steak	sear steak	Average
cam00	0.21	0.173	0.185	0.195	0.167	0.17	0.183
cam02	0.184	0.16	0.173	0.179	0.169	0.17	0.172
cam03	-	0.169	0.186	-	0.179	0.177	0.178
cam04	0.181	0.155	-	0.17	0.151	0.151	0.161
cam06	0.25	0.183	0.207	0.235	0.179	0.182	0.206
cam07	0.273	0.183	0.223	0.252	0.191	0.192	0.219
cam08	0.266	0.182	0.222	0.249	0.178	0.18	0.213
cam09	0.219	0.15	0.168	0.204	0.146	0.147	0.172
cam11	0.37	0.317	0.339	0.372	0.326	0.329	0.342
cam12	0.388	0.324	0.346	0.387	0.344	0.343	0.355
cam13	0.358	0.281	0.291	0.347	0.286	0.292	0.309
cam14	0.313	0.252	0.261	0.312	0.259	0.264	0.277
cam15	-	0.265	0.285	0.327	0.271	0.276	0.284
cam16	0.357	0.278	0.304	0.345	0.294	0.293	0.312
cam17	-	0.274	0.308	-	0.291	0.288	0.29
cam18	0.384	0.278	0.311	0.372	0.291	0.296	0.322
cam19	0.352	0.277	0.294	0.348	0.268	0.27	0.301
cam20	0.373	0.279	0.307	0.367	0.27	0.272	0.311
Average	0.298	0.232	0.259	0.291	0.237	0.238	0.258

Table 8. **PSNR, SSIM, and LPIPS of our framework on Nvidia Dynamic Scenes.** Three cameras (cam01, cam06 and cam10) are training views.

Nvidia Dynamic Scenes - PSNR							
	jumping	balloon1	balloon3	playground	skating	umbrella	Average
cam02	25.867	26.076	28.318	19.736	32.098	26.946	26.507
cam03	23.858	25.564	27.681	19.201	26.698	27.409	25.069
cam04	25.41	25.337	26.352	18.441	30.907	27.033	25.58
cam05	26.698	26.248	27.455	20.261	32.017	28.36	26.84
cam07	24.477	25.839	27.519	19.161	29.468	22.384	24.808
cam08	25.006	24.963	21.572	19.643	30.531	22.493	24.035
cam09	27.679	27.486	23.516	19.941	33.348	23.373	25.89
cam11	20.699	25.297	24.761	20.925	21.651	25.361	23.116
cam12	19.521	20.987	20.808	19.436	21.827	25.385	21.328
Average	24.357	25.311	25.331	19.638	28.727	25.416	24.807

Nvidia Dynamic Scenes - SSIM							
	jumping	balloon1	balloon3	playground	skating	umbrella	Average
cam02	0.818	0.848	0.878	0.695	0.93	0.822	0.832
cam03	0.756	0.82	0.88	0.626	0.826	0.818	0.788
cam04	0.802	0.814	0.845	0.617	0.908	0.807	0.799
cam05	0.858	0.845	0.832	0.699	0.934	0.84	0.835
cam07	0.78	0.83	0.848	0.625	0.919	0.758	0.793
cam08	0.814	0.795	0.519	0.67	0.931	0.686	0.736
cam09	0.875	0.87	0.81	0.69	0.949	0.8	0.832
cam11	0.784	0.836	0.805	0.74	0.894	0.813	0.812
cam12	0.65	0.635	0.699	0.684	0.865	0.799	0.722
Average	0.793	0.81	0.791	0.672	0.906	0.794	0.794

Nvidia Dynamic Scenes - LPIPS							
	jumping	balloon1	balloon3	playground	skating	umbrella	Average
cam02	0.144	0.108	0.106	0.182	0.096	0.122	0.126
cam03	0.197	0.12	0.108	0.187	0.156	0.147	0.152
cam04	0.165	0.128	0.119	0.23	0.113	0.139	0.149
cam05	0.127	0.115	0.115	0.177	0.094	0.129	0.126
cam07	0.176	0.126	0.12	0.219	0.13	0.178	0.158
cam08	0.159	0.134	0.216	0.197	0.113	0.175	0.166
cam09	0.128	0.106	0.153	0.176	0.097	0.161	0.137
cam11	0.185	0.126	0.143	0.151	0.138	0.137	0.147
cam12	0.248	0.182	0.195	0.192	0.152	0.148	0.186
Average	0.17	0.127	0.142	0.19	0.121	0.148	0.15