Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis Supplementary Material

Zhan Li[1,2*] Zhang Chen[1†] Zhong Li[1†] Yi Xu[1]
[1] OPPO US Research Center [2] Portland State University
lizhan@pdx.edu zhang.chen@oppo.com zhong.li@oppo.com yi.xu@oppo.com
https://oppo-us-research.github.io/SpacetimeGaussians-website/

A. Overview

Within the supplementary material, we provide:

- Quantitative and qualitative comparisons to concurrent work in Appendix B.
- More ablation study in Appendix C.
- Additional discussions in Appendix D.
- Additional experiment details in Appendix E.
- Per-scene quantitative comparisons and more visual comparisons with other methods on the Neural 3D Video Dataset [7], Google Immersive Dataset [3] and Technicolor Dataset [12] in Appendix F.
- Real-time demos and dynamic comparisons in our video. Please refer to our website.

B. Comparisons with Concurrent Work

We compare with concurrent work [10, 15–17] on the Neural 3D Video Dataset [7] in Tab. 1. We also include Im4D [8] in this comparison since it is related to 4K4D [16]. Same with Table 1 in the main paper, we group DSSIM results into two categories (DSSIM$_1$: $data\_range$ is set to 1.0; DSSIM$_2$: $data\_range$ is set to 2.0).

Compared to methods [10, 15, 17] that similarly build upon Gaussian Splatting, our method achieves the best rendering quality and is among the fastest and most compact ones. Specifically, in terms of quality, our full model performs the best on all of PSNR, DSSIM and LPIPS. Meanwhile, our lite model also outperforms Dynamic 3DGS [10] and 4DGaussians [15] by a noticeable margin, and is only inferior to 4DGS [17].

Both our lite model and Dynamic 3DGS [10] can run at over 300 FPS on the Neural 3D Video Dataset. Although our full model is slower than these two, it is still faster than 4DGS [17] and 4DGaussians [15]. Compared with Dynamic 3DGS, our lite model takes about only six percent of model size and is 0.6 dB higher in PSNR. Meanwhile, the results of Dynamic 3DGS contain many time-varying floaters, which harm temporal consistency and visual quality. To illustrate this, we show the slices of a column of pixels across time in Fig. 1. In this visualization, temporal noises appear as sharp vertical lines or dots. It can be seen that the results of Dynamic 3DGS contain many such patterns. On the contrary, our results are free of these artifacts. One reason for this phenomenon is that Dynamic 3DGS requires per-frame training, while ours trains across a sequence of frames. As a result, our method can better preserve the temporal consistency across frames. Please refer to our video for dynamic comparisons.

Compared to Im4D [8] and 4K4D [16], both our full model and lite-version model achieve higher rendering quality and speed.

† Corresponding authors.
* Work done while Zhan was an intern at OPPO US Research Center.

C. More Ablation Study

C.1. Guided Sampling and Strategies of Adding Gaussians

We visualize the effects of guided sampling in Fig. 2. It can be seen that when without guided sampling, distant areas that are not well covered by SfM points will have very blurry rendering in both training and novel views. It reveals that it is challenging to pull Gaussians to these areas with gradient-based optimization and density control. On the other hand, with guided sampling applied, the renderings at these areas become much sharper for both training and novel views. Note that the color tone difference in the bottom two rows is caused by inconsistent white balance in the training views of the scene, which makes each model have slightly different color tone in the novel view.

We also compare our guided sampling with two other strategies. The first one randomly adds Gaussians in the whole space and the second one adds a sphere of Gaussians near the far points of our guided sampling. As shown in Tab. 2 rows 2-5, our method has over 0.7dB PSNR improvement.

Table 1. **Quantitative comparisons on the Neural 3D Video Dataset.** "FPS" is measured at *1352* $\times$ *1014* resolution. "Size" is the total model size for 300 frames. Some methods only report results on part of the scenes. For fair comparison, we additionally report our results under their settings. ^{3} only includes the *Cook Spinach*, *Cut Roasted Beef*, and *Sear Steak* scenes. ^{4} only includes the *Cut Roasted Beef* scene. For the LPIPS metric, no annotation means LPIPS$_{Alex}$, ^{V} denotes LPIPS$_{VGG}$. † denotes it is unclear which LPIPS or DSSIM is used from the corresponding paper.

Method	PSNR↑	DSSIM$_1$↓	DSSIM$_2$↓	LPIPS↓	FPS↑	Size↓
Dynamic 3DGS [10]	30.67	0.035	0.019	0.099	**460**	2772 MB
4DGaussians [15]	31.15	-	0.016 †	0.049 †	30	**90MB**
4DGS [17]	32.01	-	**0.014**	0.055	114	-
Ours	**32.05**	**0.026**	**0.014**	**0.044**	140	200 MB
Ours-Lite	31.59	0.027	0.015	0.047	310	103 MB
4DGaussians [15] ^{3}	32.62	0.023 †	-	-	-	-
Ours ^{3}	**33.53**	**0.020**	**0.010**	**0.034, 0.131** ^{V}	154	148MB
Ours-Lite ^{3}	33.36	**0.020**	0.011	0.036, 0.133 ^{V}	**330**	**83MB**
Im4D [8] ^{4}	32.58	-	0.015	0.208 ^{V}	-	-
4K4D [16] ^{4}	32.86	-	0.014	0.167 ^{V}	110	-
Ours ^{4}	33.52	**0.020**	**0.011**	**0.035, 0.133** ^{V}	151	154 MB
Ours-Lite ^{4}	**33.72**	0.021	**0.011**	0.038, 0.136 ^{V}	**338**	**80 MB**

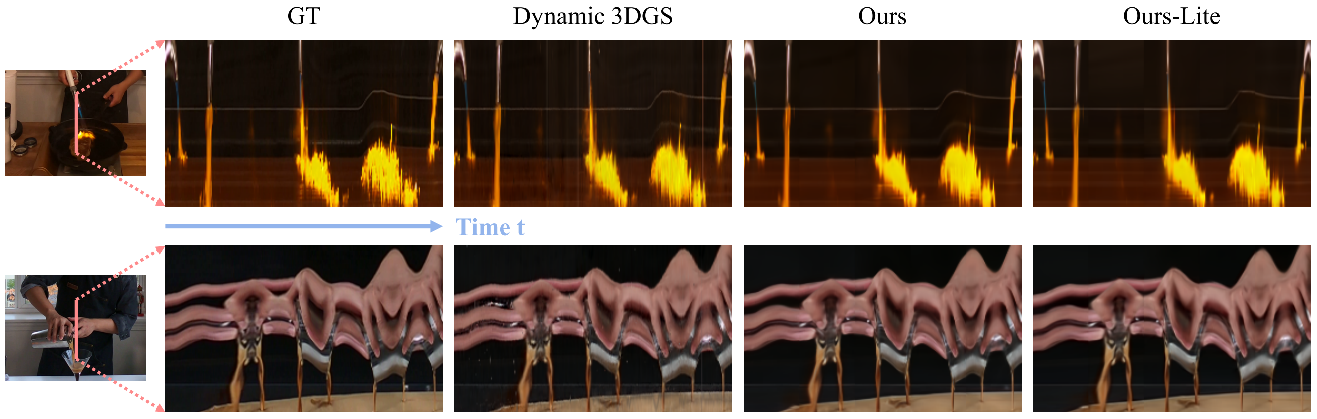

Figure 1. **Comparisons of temporal consistency on the Neural 3D Video Dataset**. From the test view video results of each method, we take a vertical column of 150 pixels across 250 frames and concatenate these columns horizontally. The resulting image patch is equivalent to a slice in the height-time space. Ours results are clearer than Dynamic 3DGS [10] and contain fewer temporal noises.

C.2. Analysis on More Scenes

Tab. 2 rows 4-9 extend the ablation study in Table 4 of the main paper to additional scenes from the Neural 3D Video Dataset and the Google Immersive Dataset. We can see that our proposed components remain effective under various camera setup and scene content.

C.3. Polynomial Orders and Replacing Polynomials with MLP

In this experiment, we alter the polynomial orders n_p, n_q and replace the polynomials with MLPs. Tab. 3 shows the results. Our choice of n_p, n_q and polynomials balances quality and storage.

Table 2. **Ablation of guided sampling and other components.** Conducted on the first 50 frames of *Flame Salmon* and *09_Exhibit* scenes.

	PSNR↑	DSSIM$_1$↓	LPIPS↓
Add random Gaussians during init	27.72	0.0455	0.0787
Add a sphere of Gaussians during init	29.13	0.0381	0.0690
w/o Guided Sampling	27.48	0.0453	0.0921
Ours-Full	**29.88**	**0.0373**	**0.0665**
w/o Temporal Opacity	28.82	0.0376	0.0673
w/o Polynomial Motion	28.35	0.0406	0.0688
w/o Polynomial Rotation	28.69	0.0455	0.0690
w/o Feature Splatting	28.05	0.0448	0.0754

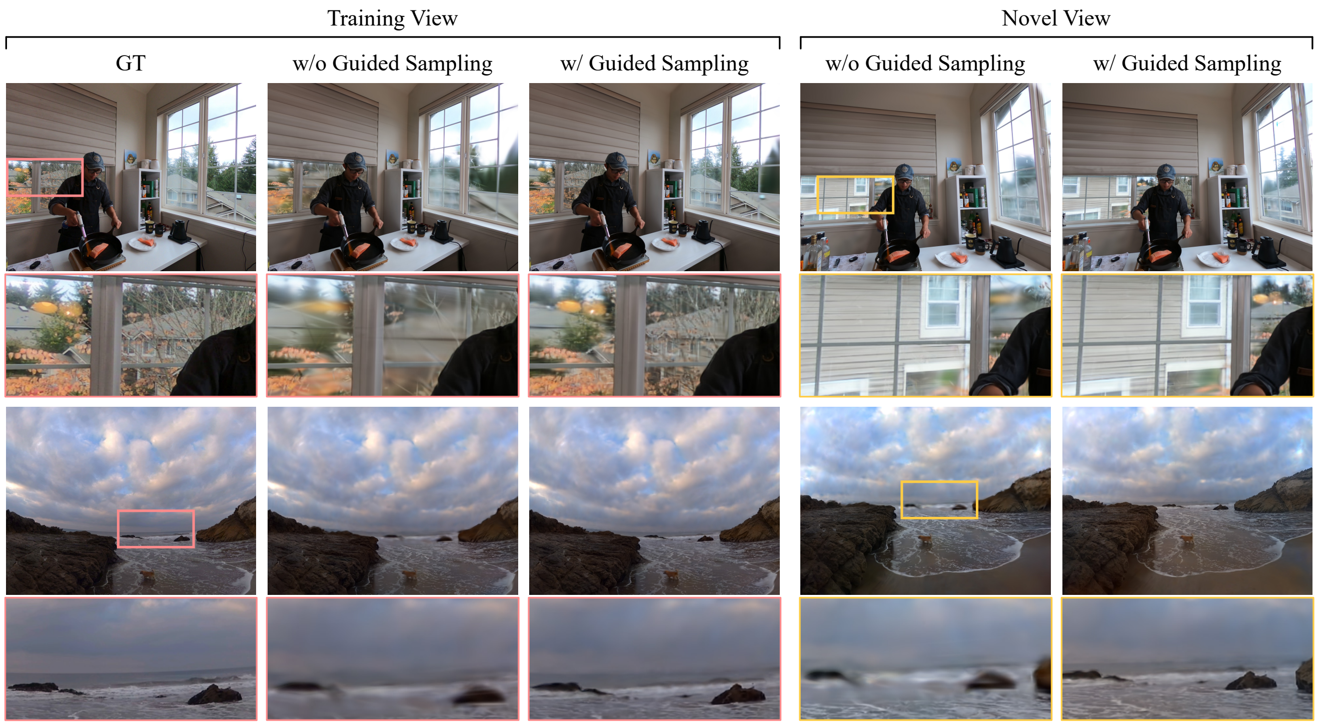

Figure 2. **Ablation on Guided Sampling.** With guided sampling, the rendering results contain less blurriness in both training and novel views.

Table 3. **Ablation of temporal function, polynomial orders, and features.** Conducted on the first 50 frames of *Theater* and *Sear Steak* scenes.

	Size (MB)↓	PSNR↑	DSSIM$_1$↓	LPIPS↓
Ours-Temporal-MLP	41.6	30.93	0.0428	0.0967
$n_p = 1$	33.4	32.24	0.0388	0.0823
$n_p = 2$	37.3	32.43	0.0379	0.0820
$n_p = 4$	44.3	**32.61**	**0.0374**	**0.0809**
$n_q = 2$	45.0	32.60	0.0377	0.0813
Ours-Full ($n_p = 3, n_q = 1$)	40.7	32.56	0.0376	0.0816
w/o $\mathbf{f}^{base}$	**31.9**	31.83	0.0408	0.0959
w/o $\mathbf{f}^{dir}$	38.5	32.03	0.0384	0.0849
w/o $\mathbf{f}^{time}$	39.1	32.03	0.0392	0.0818
Random init $\mathbf{f}^{base}$	36.3	32.14	0.0385	0.0841
Random init $\mathbf{f}^{dir}$	40.7	32.22	0.0379	0.0827
Random init $\mathbf{f}^{time}$	40.8	32.45	0.0378	0.0814

C.4. Feature Components

In this experiment, we ablate different features ($\mathbf{f}^{base}$, $\mathbf{f}^{dir}$ and $\mathbf{f}^{time}$) used in our full model. As shown in Tab. 3 rows 7-10 and Fig. 3, each component boosts rendering quality.

C.5. Initialization of Features

In our model, $\mathbf{f}^{base}$ and $\mathbf{f}^{dir}$ are initialized with the color of SfM points. $\mathbf{f}^{time}$ is initialized as zeros. The last three rows of Tab. 3 show an ablation of feature initialization, where our choice (Ours-Full) works better than random initialization.

Table 4. **Comparison with per-frame 3DGS and replacing the MLP in Ours-Full with SH.** Conducted on the first 50 frames of *Flame Salmon* and *Flame Steak* scenes. Ours-SH uses SH of order 0 to 3, as in 3DGS.

	Size (MB)↓	FPS↑	Train Time (min.)↓	PSNR↑	DSSIM$_1$↓	LPIPS↓
3DGS	5100	135	700	29.76	0.0311	0.0486
Ours-SH	118	132	37	31.37	0.0276	0.0470
Ours-Full	**37**	**145**	**35**	**31.66**	**0.0274**	**0.0467**

C.6. Feature Rendering vs. Spherical Harmonics

In this experiment, we replace our full model's feature rendering with spherical harmonics rendering, and refer to this baseline as Ours-SH. Tab. 4 and Fig. 4 show that Ours-Full has better quantitative and visual quality while having smaller model size than Ours-SH. For fair comparisons of FPS, all methods use PyTorch implementation for rendering.

C.7. Comparison with Per-Frame 3DGS

To validate the improvements of our method, we further compare with per-frame trained 3DGS. As shown in Tab. 4,

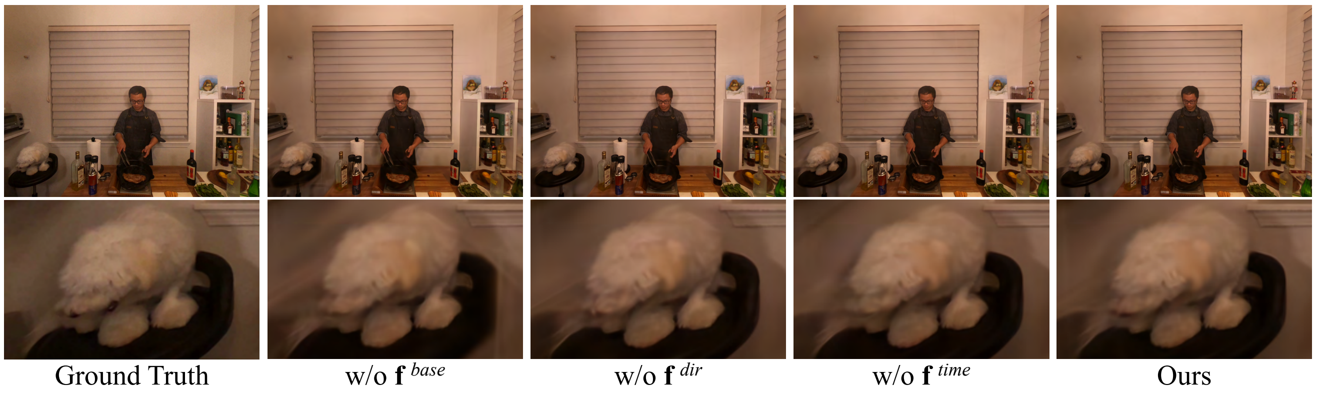

Figure 3. **Ablation on feature components.** Using all features produces the best visual quality.

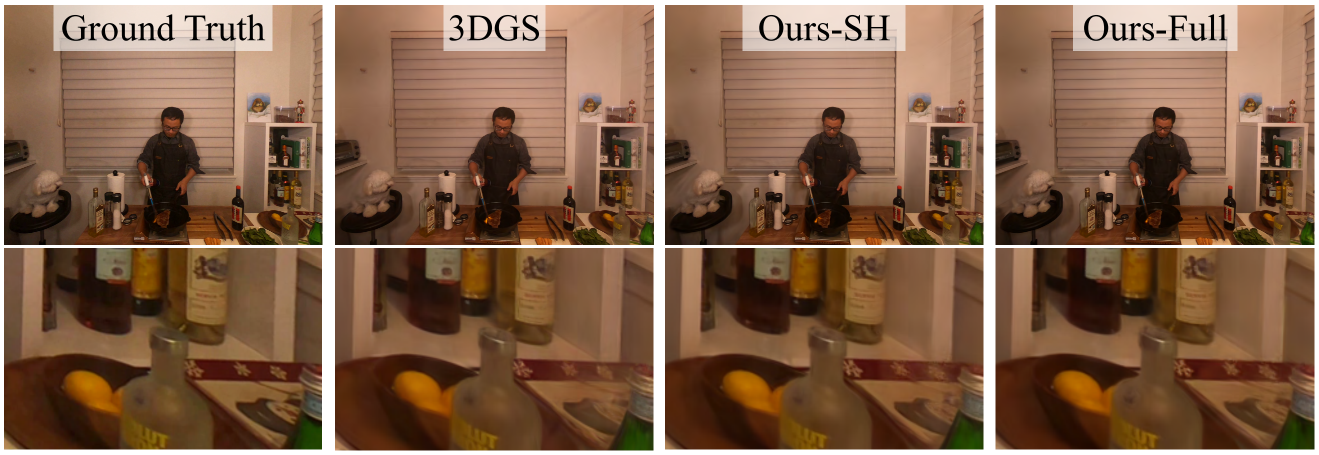

Figure 4. **Qualitative comparison of 3DGS [6], Ours-SH and Ours-Full.** Conducted on the *Flame Steak* scene from the Neural 3D Video Dataset [7]. 3DGS is trained per-frame. Ours-SH denotes replacing our feature rendering with spherical harmonics rendering in 3DGS [6].

our method has much smaller size and better rendering quality. Fig. 4 shows visual comparison.

C.8. Longer Video Sequence

In our experiments, following prior arts [2, 13], we train each model with 50-frame video sequence and arrange these models in series to render full-length sequences (typically 300 frames). In practice, this scheme can work for long videos at the cost of redundancy among models (*e.g.*, static parts of the scene are repeatedly modeled). Our method also supports using a single model to represent more frames. Here, we conduct an experiment that directly trains our model with 300 frames on the *Flame Salmon* scene from the Neural 3D Video Dataset [7]. As shown in Tab. 5, compared to six 50-frame models in series, our single 300-frame model can reduce the per-frame training time and model size by around 80% and 30% respectively. At the same time, the rendering quality is comparable. This is attributed to our temporal opacity formulation so that complex long-sequence motion can be represented by multiple simpler motion segments.

D. Discussions

Our method is able to model shadows and ambient occlusions, as demonstrated in Fig. 3 and Fig. 4. For complex motion, our temporal opacity allows using multiple Gaussians where each one only needs to fit a shorter and less complex motion segment. Generally, the size of temporal RBF is small for fast-changing volumetric objects (*e.g.*, flames) and large for static solid objects. Learned motion tends to be small for static objects and large for moving objects. Fig. 5 visualizes the temporal RBF and motion for an example scene. Note that our method does not apply additional regularization on motion.

For guided sampling, although it can alleviate the blurring in areas that are insufficiently covered by sparse point cloud, it cannot fully eliminate such artifacts. This is re-

Table 5. **Performance of longer sequence per model on the *Flame Salmon* scene from the Neural 3D Video Dataset.** We increase the training frames per model from 50 to 300. Longer sequence per model has smaller model size and shorter per-frame training time.

Video Length per Model	# of Models	Iterations per Model	PSNR↑	DSSIM$_1$↓	DSSIM$_2$↓	LPIPS↓	Per-Frame Training Time (sec.)↓	Per-Frame Size (MB)↓	Total Size (MB)↓
50 frames	6	12K	**29.48**	0.038	0.0224	**0.063**	20	1	300
300 frames	1	10K	29.17	**0.037**	**0.0222**	0.068	**3.7**	**0.7**	**216**

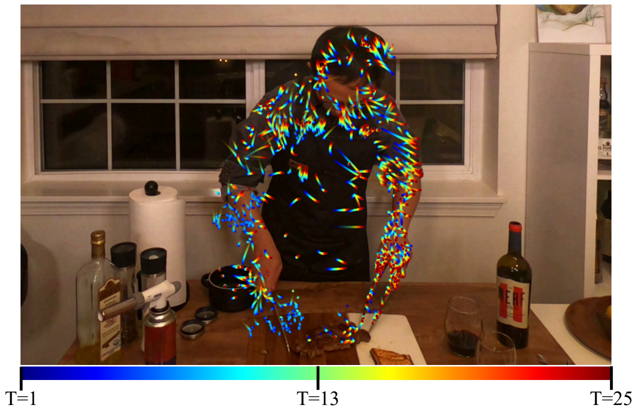

Figure 5. **Visualization of trajectory across 25 frames.** The background image is the ground truth at time $T = 25$. The color of a trajectory denotes timestamp, where dark blue corresponds to $T = 1$ and dark red corresponds to $T = 25$. To visualize temporal opacity along a trajectory, we set the alpha channel value of a segment based on temporal opacity (excluding the spatial opacity term σ_i^s). We only show moving objects in the scene.

flected in some challenging scenarios such as the far content outside windows in the *Coffee Martini* scene and the flame in the *02_Flames* scene. The reason is that we do not have accurate depth of these areas, hence need to spawn new Gaussians across a depth range. However, these Gaussians may not cover the exact correct locations, and the ones near the correct locations may also be pruned during subsequent training. A possible solution would be to leverage the depth priors from learning-based depth estimation methods.

E. Experiment Details

E.1. Baselines

Since the open source code of MixVoxels [14] does not contain the training config for MixVoxels-X, we use MixVoxels-L in our comparisons. We train HyperReel with their official code to generate visual examples. Note that the training time of HyperReel for each scene on the Technicolor Dataset is about 3.5 hours, while that of our full model on the Technicolor Dataset is only about 1 hour. For Dynamic 3DGS, its performance on the Neural 3D Video Dataset is not reported in their original paper. When applying their open source code to Neural 3D Video Dataset with default hyperparameters, the rendering quality is subpar. So we tune its hyperparameters to improve the performance on this dataset.

E.2. Camera Models

We use the original *centered-undistorted* camera model from 3DGS [6] for the Neural 3D Video Dataset. We implement the *uncentered-undistorted* camera model for the Technicolor Dataset. For the Google Immersive Dataset [3], to evaluate on the same distorted videos as [1, 13], we further adapt our method to fit the *uncentered-distorted* camera model with a differentiable image space warping, which maps perspective view to fish-eye distorted view. For the real-time demo, we retrain our models on the undistorted videos for simplicity. As there are black pixels in the undistorted images, we opt to use a mask to mask out the black pixels. Since image warping and masking are differentiable, our models can still be trained end-to-end.

E.3. Initialization

Following 3DGS [6], we use the sparse point cloud from COLMAP for initialization. Since the datasets provide camera intrinsics and extrinsics, we input them to COLMAP and call *point triangulator* to generate sparse points. The running time for *point triangulator* is much less than that of dense reconstruction. For the *Theater*, *Train* and *Birthday* scenes in Table 3 of the main paper and in Tab. 8, we only use 25 percent of SfM points from each frame (except the first frame whose SfM points are all used). The selection of SfM points is based on the distance between each point and its nearest neighbor. After sorting the distances, we keep the points with the longest distance to reduce redundancy. In the ablation study on the number of frames whose SfM points are used (Table 5 in the main paper), we use all the points in each sampled frames.

Features $\mathbf{f}^{base}$ and $\mathbf{f}^{dir}$ are initialized with the color of SfM points. $\mathbf{f}^{time}$ is initialized as zeros.

E.4. Density Control

During training, we conduct 12 times of cloning/splitting and over 50 times of pruning on the Technicolor dataset. Sparse points from multiple timestamps contain richer but more redundant information than sparse points from a single timestamp (or static points). Thus, after densification and guided sampling steps, we gradually prune Gaussians with small spatial opacity to keep the most representative Gaussians.

E.5. Guided Sampling

We uniformly sample from $s \times d$ to $7.5 \times d$ with small random noise. d is the max depth in a training view. s is set as 0.7 for most scenes.

E.6. Others

We apply *sigmoid* function to get the final RGB color for Neural 3D dataset and *clamp* function for the other two datasets in our full model. We use a linear form of the time variable and do not apply positional encoding on it. We use Nvidia RTX 3090 when reporting our rendering speed in the comparisons with other methods, and use Nvidia RTX 4090 in our real-time demos.

F. More Results

We provide per-scene quantitative comparisons on the Neural 3D Video Dataset [7] (Tab. 6), Google Immersive Dataset [3] (Tab. 7) and Technicolor Dataset [12] (Tab. 8). Our method outperforms the other baselines on most scenes. We also provide per-scene Gaussian numbers of trained 50-frame models in Tab. 9.

Figs. 6 to 8 show more visual comparisons of our full model and our lite-version model with NeRFPlayer [13], HyperReel [2], K-Planes [5], MixVoxels-L [14] and Dynamic 3DGS [10] on the Neural 3D Video Dataset [7].

Fig. 9 shows more visual comparisons on the Google Immersive Dataset [3]. We compare the results of our full model and lite-version model to NeRFPlayer [13] and HyperReel [2].

Fig. 10 shows visual comparisons on the Technicolor Dataset [12]. We compare the results of our full model and lite-version model to HyperReel [2].

The above visual comparisons demonstrate that our method preserves sharp details while containing fewer artifacts. Compared to our full model, the results of our lite-version model are slightly blurrier.

References

[1] Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *European Conference on Computer Vision*, pages 441–459. Springer, 2020. 5

[2] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. HyperReel: High-fidelity 6-DoF video with ray-conditioned sampling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 4, 6, 7, 8

[3] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)*, 39(4):86–1, 2020. 1, 5, 6, 8, 9, 13

[4] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 7

[5] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 6, 7

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4):1–14, 2023. 4, 5

[7] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022. 1, 4, 6, 7, 8, 9, 10, 11, 12

[8] Haotong Lin, Sida Peng, Zhen Xu, Tao Xie, Xingyi He, Hujun Bao, and Xiaowei Zhou. Im4d: High-fidelity and real-time novel view synthesis for dynamic scenes. *arXiv preprint arXiv:2310.08585*, 2023. 1, 2

[9] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, 2019. 7

[10] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 1, 2, 6, 7

[11] 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 Transactions on Graphics (TOG)*, 2019. 7

[12] Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbiriou, Frederic Babon, Matthieu Hog, Remy Gendrot, Tristan Langlois, Olivier Bureller, Arno Schubert, et al. Dataset and pipeline for multi-view light-field video. In *Proceedings of the IEEE conference on computer vision and pattern recognition Workshops*, pages 30–40, 2017. 1, 6, 8, 9, 14

[13] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023. 4, 5, 6, 7, 8

[14] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, Yafei Song, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19706–19716, 2023. 5, 6, 7

Table 6. **Per-scene quantitative comparisons on the Neural 3D Video Dataset [7].** Some methods only report part of the scenes. [1] only includes the *Flame Salmon* scene. [2] excludes the *Coffee Martini* scene. "-" denotes results that are unavailable in prior work.

Method	Avg.	*Coffee Martini*	*Cook Spinach*	*Cut Roasted Beef*	*Flame Salmon*	*Flame Steak*	*Sear Steak*
PSNR↑							
Neural Volumes [9] [1]	22.80	-	-	-	22.80	-	-
LLFF [11] [1]	23.24	-	-	-	23.24	-	-
DyNeRF [7] [1]	29.58	-	-	-	29.58	-	-
HexPlane [4] [2]	31.71	-	32.04	32.55	29.47	32.08	32.39
NeRFPlayer [13]	30.69	**31.53**	30.56	29.35	31.65	31.93	29.13
HyperReel [2]	31.10	28.37	32.30	32.92	28.26	32.20	32.57
K-Planes [5]	31.63	29.99	32.60	31.82	30.44	32.38	32.52
MixVoxels-L [14]	31.34	29.63	32.25	32.40	29.81	31.83	32.10
MixVoxels-X [14]	31.73	30.39	32.31	32.63	**30.60**	32.10	32.33
Dynamic 3DGS [10]	30.67	26.49	32.97	30.72	26.92	33.24	33.68
Ours	**32.05**	28.61	**33.18**	33.52	29.48	**33.64**	**33.89**
Ours-Lite	31.59	27.49	32.92	**33.72**	28.67	33.28	33.47
DSSIM$_1$↓							
NeRFPlayer [13]	0.034	**0.0245**	0.0355	0.0460	**0.0300**	0.0250	0.0460
HyperReel [2]	0.036	0.0540	0.0295	0.0275	0.0590	0.0255	0.0240
Dynamic 3DGS [10]	0.035	0.0557	0.0263	0.0295	0.0512	0.0233	0.0224
Ours	**0.026**	0.0415	**0.0215**	**0.0205**	0.0375	**0.0176**	**0.0174**
Ours-Lite	0.027	0.0437	0.0218	0.0209	0.0387	0.0179	0.0177
DSSIM$_2$↓							
Neural Volumes [9] [1]	0.062	-	-	-	0.062	-	-
LLFF [11] [1]	0.076	-	-	-	0.076	-	-
DyNeRF [7] [1]	0.020	-	-	-	**0.020**	-	-
K-Planes [5]	0.018	0.0235	0.0170	0.0170	0.0235	0.0150	0.0130
MixVoxels-L [14]	0.017	0.0244	0.0162	0.0157	0.0255	0.0144	0.0122
MixVoxels-X [14]	0.015	**0.0232**	0.0160	0.0146	0.0233	0.0137	0.0121
Dynamic 3DGS [10]	0.019	0.0332	0.0129	0.0161	0.0302	0.0113	0.0105
Ours	**0.014**	0.0250	**0.0113**	**0.0105**	0.0224	**0.0087**	**0.0085**
Ours-Lite	0.015	0.0270	0.0118	0.0112	0.0244	0.0097	0.0095
LPIPS$_{Alex}$↓							
Neural Volumes [9] [1]	0.295	-	-	-	0.295	-	-
LLFF [11] [1]	0.235	-	-	-	0.235	-	-
DyNeRF [7] [1]	0.083	-	-	-	0.083	-	-
HexPlane [4] [2]	0.075	-	0.082	0.080	0.078	0.066	0.070
NeRFPlayer [13]	0.111	0.085	0.113	0.144	0.098	0.088	0.138
HyperReel [2]	0.096	0.127	0.089	0.084	0.136	0.078	0.077
MixVoxels-L [14]	0.096	0.106	0.099	0.088	0.116	0.088	0.080
MixVoxels-X [14]	0.064	0.081	0.062	0.057	0.078	0.051	0.053
Dynamic 3DGS [10]	0.099	0.139	0.087	0.090	0.122	0.079	0.079
Ours	**0.044**	**0.069**	**0.037**	**0.036**	**0.063**	**0.029**	**0.030**
Ours-Lite	0.047	0.075	0.038	0.038	0.068	0.031	0.031

[15] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 1, 2

[16] Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 4k4d: Real-time 4d view synthesis at 4k resolution. *arXiv preprint arXiv:2310.11448*, 2023. 1, 2

[17] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and render-

Table 7. **Per-scene quantitative comparisons on the Google Immersive Dataset [3].**

Method	Avg.	*01_Welder*	*02_Flames*	*04_Truck*	*09_Exhibit*	*10_Face_Paint_1*	*11_Face_Paint_2*	*12_Cave*
PSNR↑								
NeRFPlayer [13]	25.8	25.568	26.554	27.021	24.549	27.772	27.352	21.825
HyperReel [2]	28.8	25.554	**30.631**	27.175	**31.259**	29.305	27.336	**30.063**
Ours	**29.2**	**26.844**	30.566	**27.308**	29.336	**30.588**	**29.895**	29.610
Ours-Lite	27.5	25.499	29.505	24.204	27.973	28.646	28.456	27.977
DSSIM$_1$↓								
NeRFPlayer [13]	0.076	0.0910	0.0790	0.0615	0.0655	0.0420	0.0490	0.1425
HyperReel [2]	0.063	0.1050	0.0475	0.0760	0.0485	0.0435	0.0605	0.0595
Ours	**0.042**	**0.0504**	**0.0349**	**0.0524**	**0.0447**	**0.0240**	**0.0320**	**0.0543**
Ours-Lite	0.051	0.0585	0.0546	0.0684	0.0516	0.0271	0.0326	0.0630
LPIPS$_{Alex}$↓								
NeRFPlayer [13]	0.196	0.289	0.154	0.164	0.151	0.147	0.152	0.314
HyperReel [2]	0.193	0.281	0.159	0.223	0.140	0.139	0.195	0.214
Ours	**0.081**	**0.098**	**0.059**	**0.087**	**0.073**	**0.055**	0.063	**0.133**
Ours-Lite	0.095	0.119	0.070	0.115	0.087	0.067	**0.062**	0.143

Table 8. **Per-scene quantitative comparisons on the Technicolor Dataset [12].**

Method	Avg.	*Birthday*	*Fabien*	*Painter*	*Theater*	*Trains*
PSNR↑						
DyNeRF [7]	31.8	29.20	32.76	35.95	29.53	31.58
HyperReel [2]	32.7	29.99	34.70	35.91	**33.32**	29.74
Ours	**33.6**	**32.09**	**35.70**	**36.44**	30.99	**32.58**
Ours-Lite	33.0	31.59	35.28	35.95	30.12	32.17
DSSIM$_1$↓						
HyperReel [2]	0.047	0.0390	0.0525	0.0385	**0.0525**	0.0525
Ours	**0.040**	**0.0290**	**0.0471**	**0.0366**	0.0596	**0.0294**
Ours-Lite	0.044	0.0330	0.0522	0.0382	0.0634	0.0324
DSSIM$_2$↓						
DyNeRF [7]	0.021	0.0240	**0.0175**	**0.0140**	0.0305	0.0190
Ours	**0.019**	**0.0153**	0.0179	0.0146	**0.0287**	**0.0168**
Ours-Lite	0.021	0.0175	0.0201	0.0154	0.0312	0.0185
LPIPS$_{Alex}$↓						
DyNeRF [7]	0.140	0.0668	0.2417	0.1464	0.1881	0.0670
HyperReel [2]	0.109	0.0531	0.1864	0.1173	**0.1154**	0.0723
Ours	**0.084**	**0.0419**	**0.1141**	**0.0958**	0.1327	**0.0372**
Ours-Lite	0.097	0.0532	0.1359	0.0989	0.1487	0.0492

ing with 4d gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2024. 1, 2

Table 9. **Per-scene Gaussian numbers (K) on three datasets.** For each scene, the number is averaged over 50-frame models.

	Avg.	*01_Welder*	*02_Flames*	*04_Truck*	*09_Exhibit*	*10_Face_Paint_1*	*11_Face_Paint_2*	*12_Cave*
Google Immersive Dataset [3]	427	571	389	374	484	285	249	629
	Avg.	*Coffee Martini*	*Cook Spinach*	*Cut Roasted Beef*	*Flame Salmon*	*Flame Steak*	*Sear Steak*	
Neural 3D Video Dataset [7]	215	262	189	169	319	177	176	
	Avg.	*Birthday*	*Fabien*	*Painter*	*Theater*	*Trains*		
Technicolor Dataset [12]	374	379	295	412	313	470		

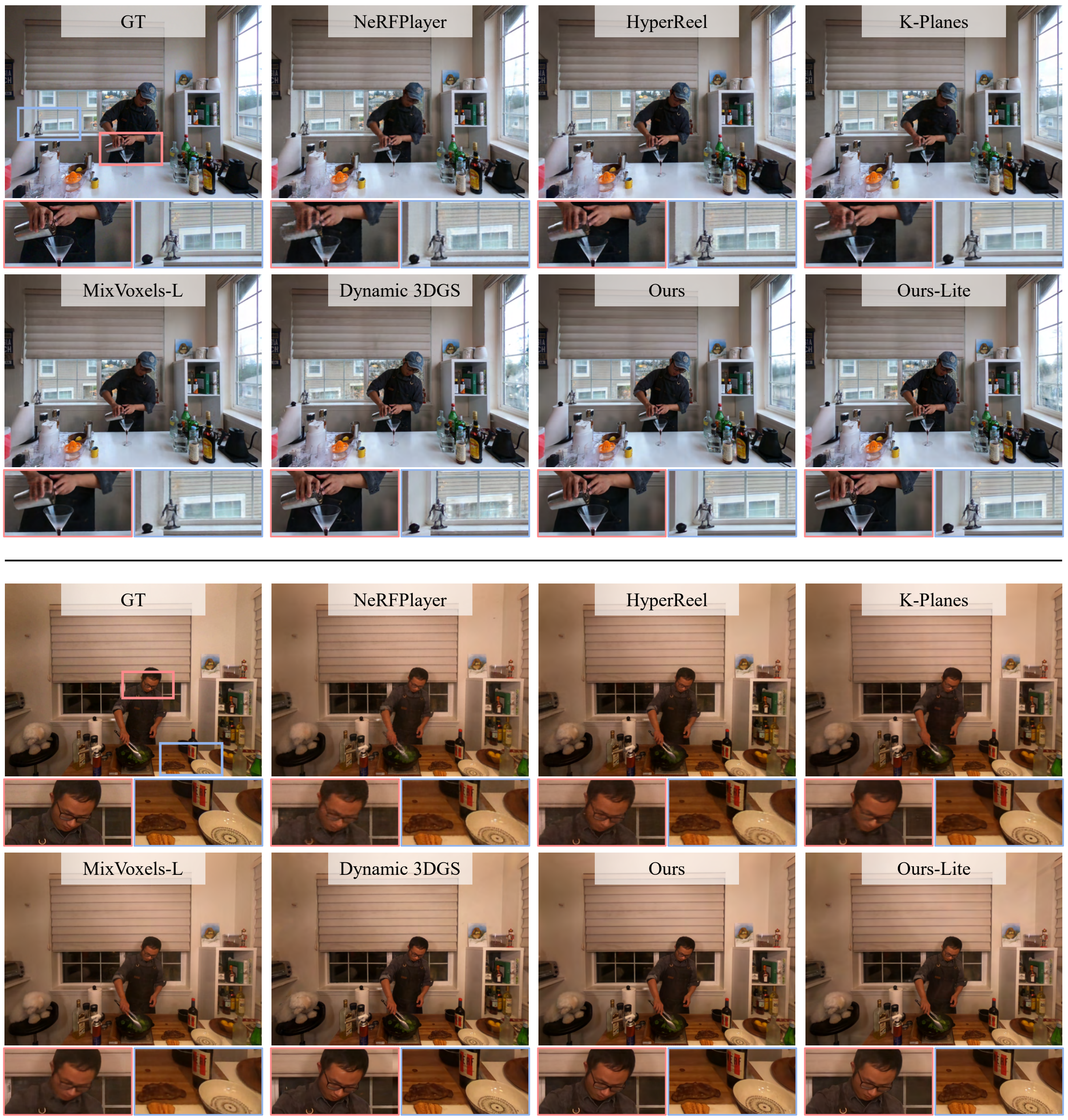

Figure 6. **Qualitative comparisons on the Neural 3D Video Dataset [7].** To be continued in the next page.

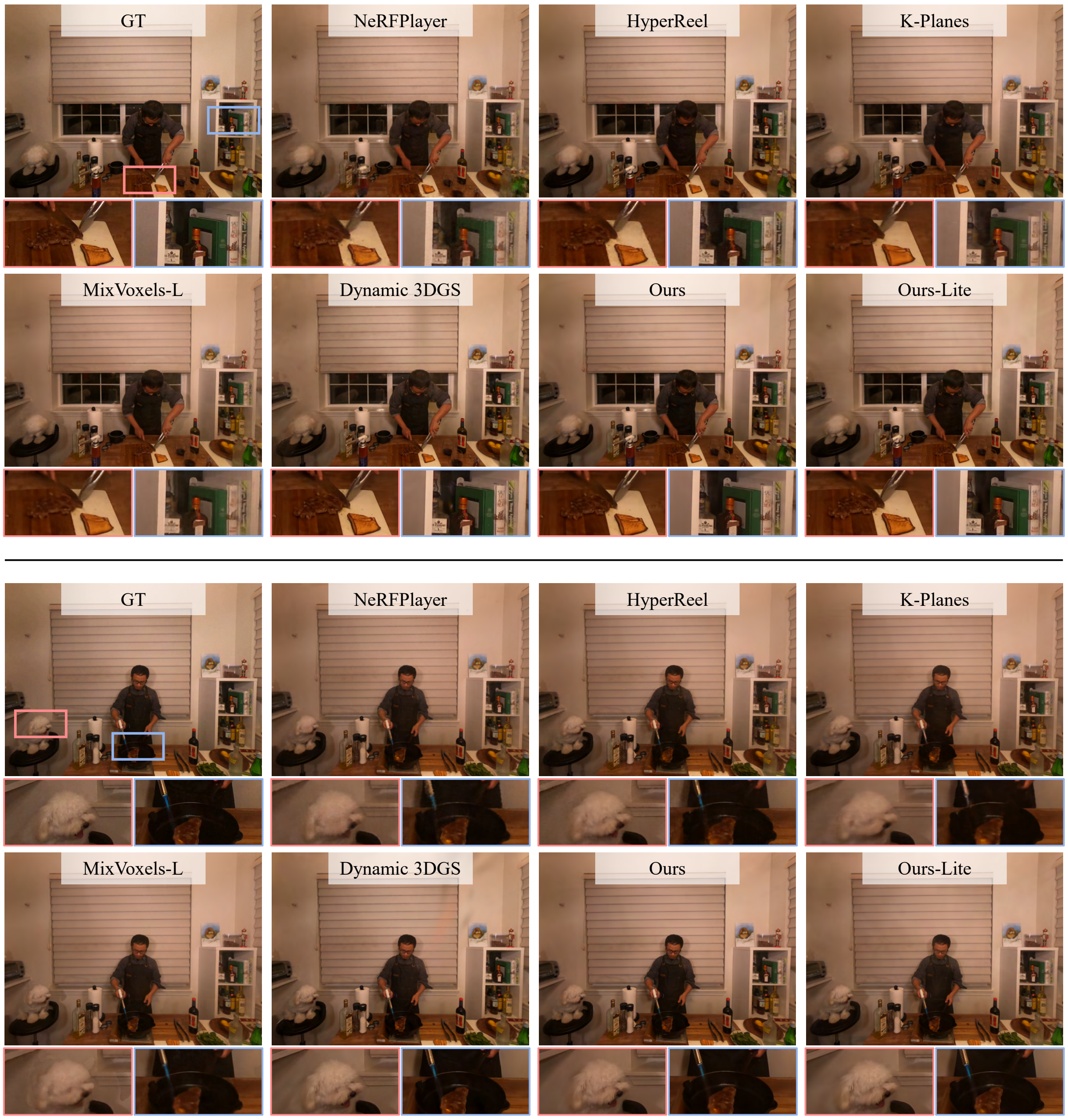

Figure 7. **Qualitative comparisons on the Neural 3D Video Dataset** [7]**.** To be continued in the next page.

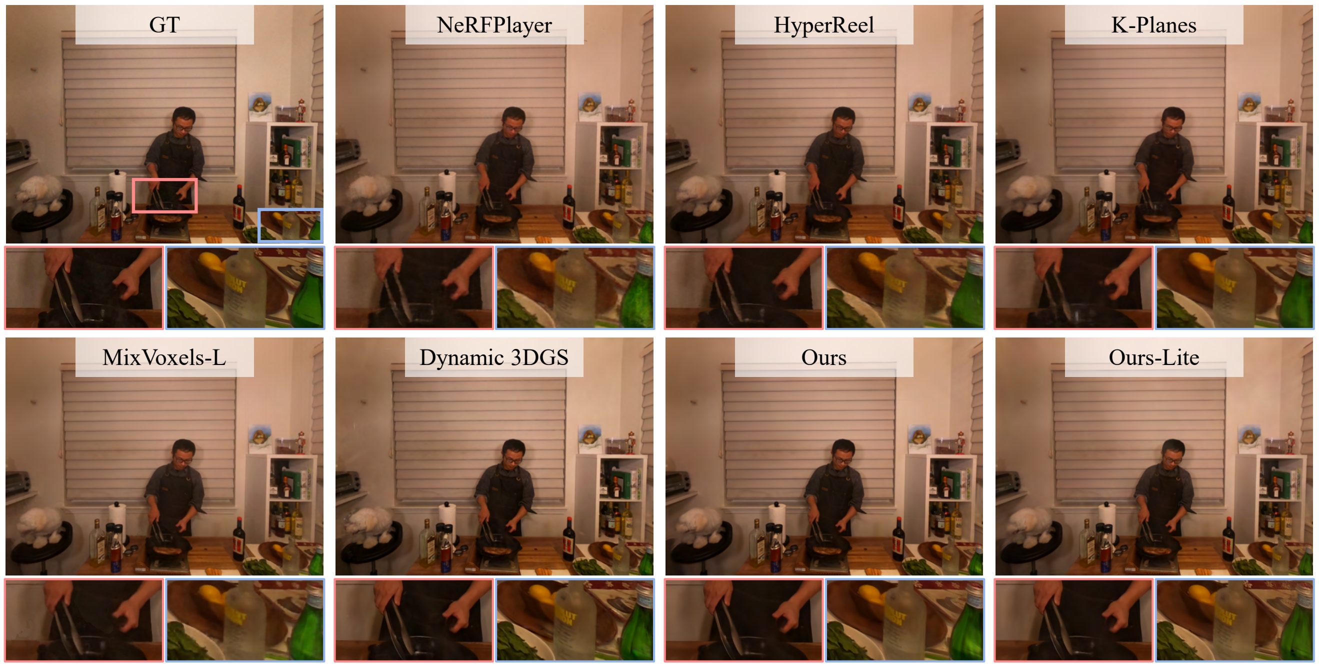

Figure 8. **Qualitative comparisons on the Neural 3D Video Dataset [7].**

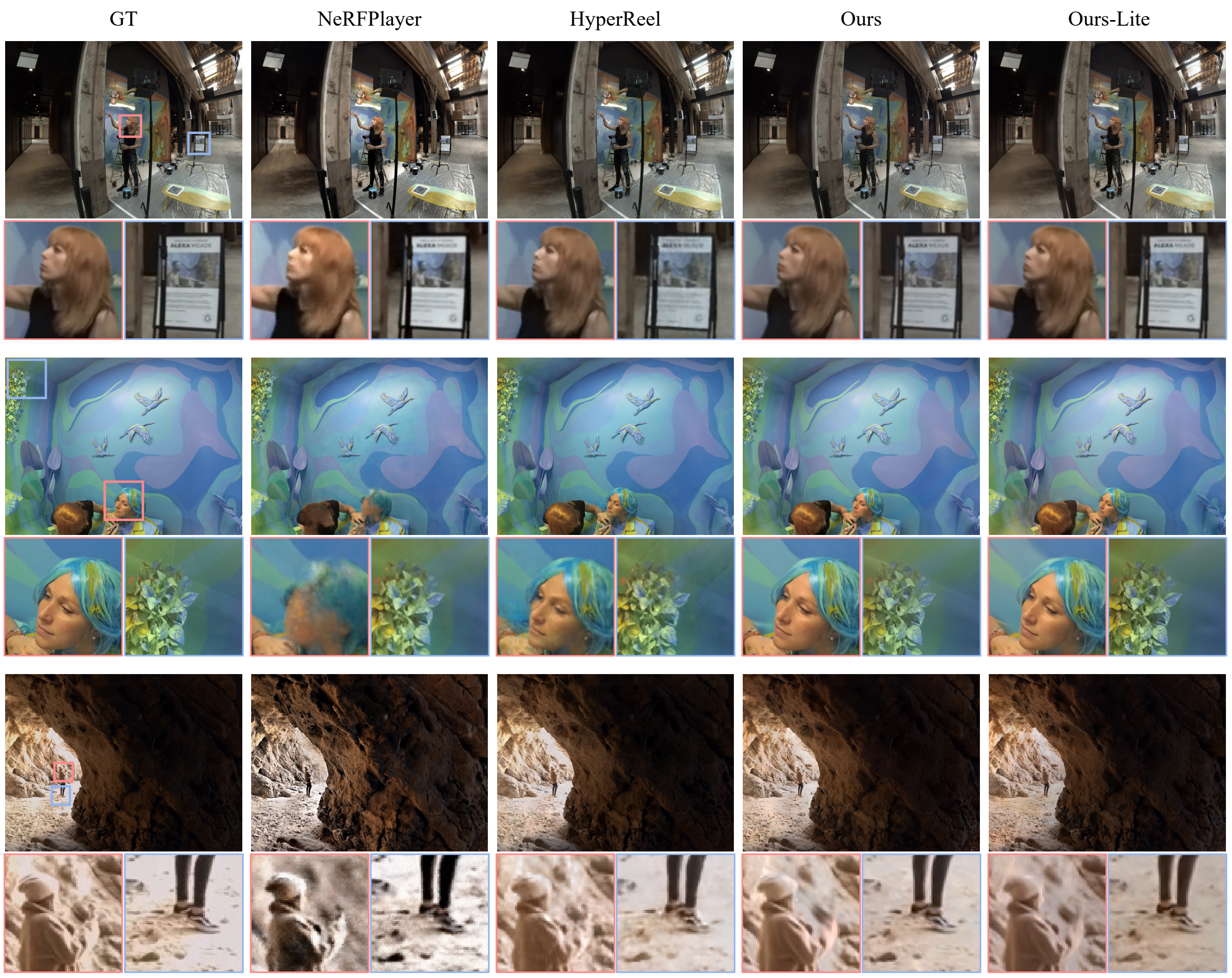

Figure 9. **Qualitative comparisons on the Google Immersive Dataset [3].**

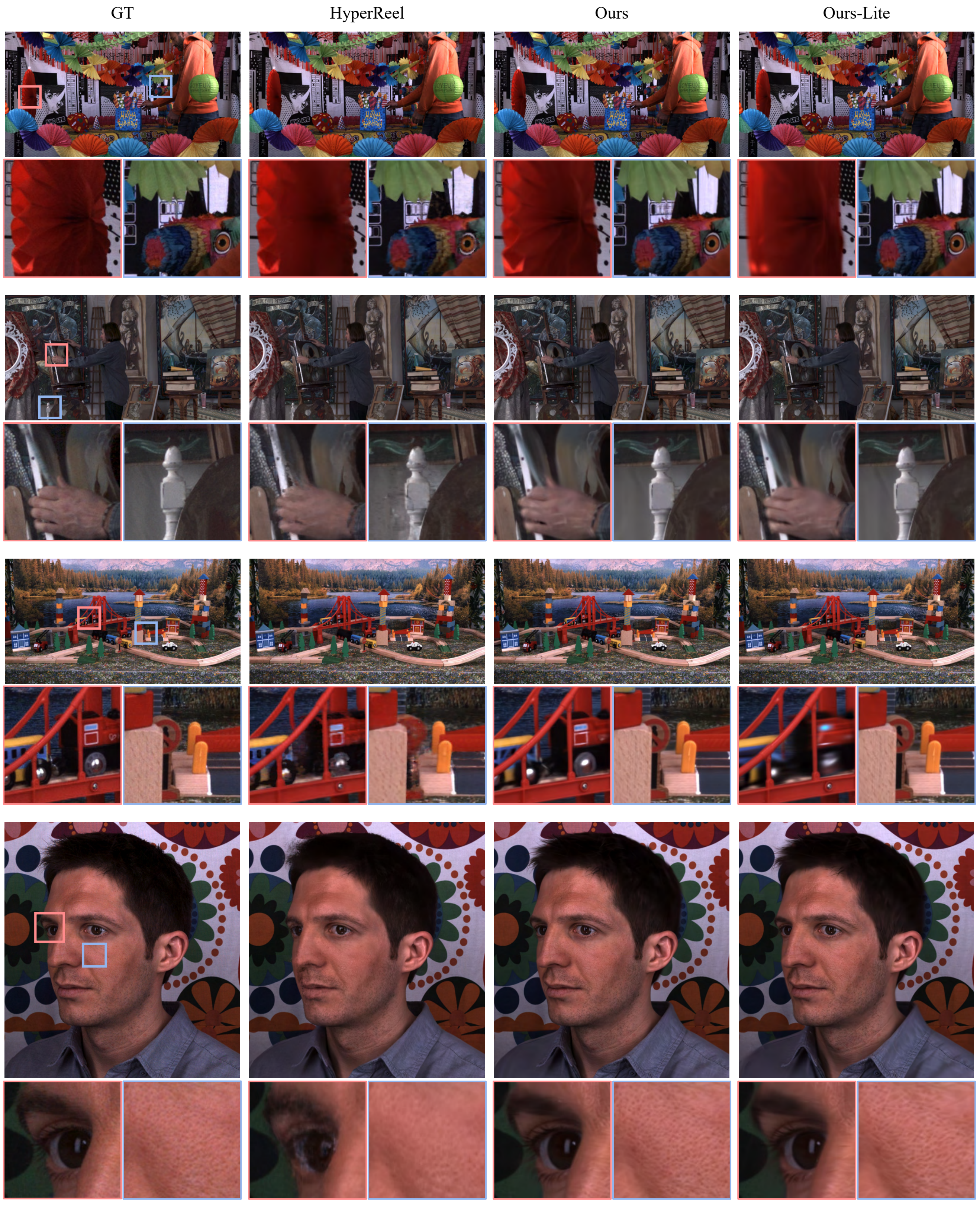

Figure 10. **Qualitative comparisons on the Technicolor Dataset [12].**