DMiT: Deformable Mipmapped Tri-Plane Representation for Dynamic Scenes –*Supplementary Material*–

Jing-Wen Yang[1,2], Jia-Mu Sun[1,2], Yong-Liang Yang[3], Jie Yang[1], Ying Shan[4], Yan-Pei Cao[5,4], and Lin Gao (✉)[1,2]

[1] Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, China
[2] University of Chinese Academy of Sciences, China
[3] Department of Computer Science, University of Bath, United Kingdom
[4] Tencent AI Lab, China [5] VAST, China

In the supplementary material, we will first present some additional implementation details of our proposed method **DMiT** in Appendix A. More rendering results from both synthetic and real-world datasets as well as the corresponding per-scene evaluation metrics will be provided in Appendix B. Moreover, we will conduct a detailed analysis on our decomposed canonical space in Appendix C. Finally, more ablation results (Appendix D) as well as further discussion including failure cases (Appendix E) will be provided respectively.

A Implementation Details

A.1 Deformation Network

To construct the projection from each observation to a shared canonical space, a deformation network is employed, which is built upon the tiny-cuda-nn framework [16]. The deformation network is configured with a width parameter of $W = 128$ and a depth parameter of $D = 8$. The input of the network consists of the positions of the sample points obtained using the cone-casting approach outlined in the main paper, together with their respective timestamps. To address the issue of overfitting in the deformation network, we employ a frequency annealing strategy in the context of positional encoding. This strategy is discussed in detail in Section 3.3 of the main paper. To provide more clarification, the D-NeRF and multi-scale D-NeRF experiments have established a maximum encoding dimension of 10 for space and 6 for time. In the HyperNeRF dataset, spatial information is represented using a maximum dimension of 10, while time is represented using a maximum dimension of 8. This annealing strategy is set to terminate at iteration 10K for synthetic experiments and 80K for real-world experiments. In D-NeRF experiments, an additional one-layer network is employed for time encoding. This network transforms the time-embedded feature into a time-aware feature with a dimension of 30. Subsequently, the featurized spatial

✉ Corresponding author is Lin Gao(gaolin@ict.ac.cn).

and temporal information is fed into the Multi-Layer Perceptron (MLP) to forecast the spatial displacement of each sampled point. As a result, the relationship between the observation space and canonical space can be established.

A.2 Tiny MLP in Canonical Space

The shallow MLP responsible for predicting density σ and color $\mathbf{c}$, is implemented using the tiny-cuda-nn library [16] due to its efficient inference capabilities, as mentioned in [7]. The MLP has a width W of 128 and a depth D of 6 layers. The initial two layers are specifically designed to estimate the density value σ and a geometric feature vector $\mathbf{f}_{\text{geo}}$ with a dimension of 15. These layers take the Tri-Mip encoded feature $\mathbf{f}$ as input, as explained in Section 3.2 of the main paper. The channel dimension of the mipmapped planes $\mathcal{M}$ is set to 16, resulting in a concatenated feature dimension of 48. Subsequently, the last layers of the MLP will predict the associated color $\mathbf{c}$ based on the provided geometric feature $\mathbf{f}_{\text{geo}}$ and the encoded view direction $\mathbf{d}$, represented by spherical harmonics.

A.3 Optimization

The trainable parameters of the whole framework consist of the model weights Θ_d derived from the deformation network Φ_d, the model weights Θ_c obtained from the canonical MLP along with parameters of the time encoding network, and the mipmapped tri-plane features $\mathcal{M}$. The networks in this study are initialized following the approach proposed by [6]. Additionally, the mipmap features are uniformly initialized within the range from -0.01 to 0.01. To optimize the static components of the canonical representation, we employ the AdamW optimizer [13] with a base learning rate of 1×10^{-3}. For the joint deformation and geometry refinement (Sec. 3.3 of the main paper), the three mipmap feature planes undergo bilinear upsampling. Additionally, a distinct AdamW optimizer is employed to enable a more adaptable schedule. The initial learning rate of this optimizer is 1×10^{-2} (*i.e.*, $10\times$ the learning rate of the static optimizer) for faster convergence and reconstruction. The scheduling of both optimizers is performed by the MultiStepLR function from the PyTorch library [20]. To ensure a consistent number of sampled spheres for cone casting, a dynamic batch-size method is employed, as described in [17]. In our pipeline, the deformable components are optimized using an AdamW optimizer, which incorporates a warmup step and a cosine-decayed scheduler.

B Detailed Results

Apart from the results shown in the main paper, we additionally provide more quantitative and qualitative results to further demonstrate the competitiveness of our method against other state-of-the-art methods. Detailed per-scene metrics and visual comparisons of rendering results on each dataset can be found in the following pages.

Table 1: Quantitative per-scene results on the test set of the original D-NeRF dataset. **Bold** means the best performance and underline means the second best performance.

Methods	Train. ↓	PSNR ↑								
		lego	*jumpingjacks*	*bouncingballs*	*hook*	*standup*	*trex*	*hellwarrior*	*mutant*	*Average*
D-NeRF [21]	1d	21.76	32.56	38.20	29.49	33.47	31.45	24.90	31.25	30.39
TiNeuVox [3]	**28m**	24.56	33.61	39.80	30.54	34.45	31.22	27.00	31.82	31.63
K-Planes-hybrid [4]	1h	25.21	31.06	39.31	27.85	32.60	30.15	23.97	32.38	30.32
K-Planes-explicit [4]	1h	**25.25**	29.63	37.53	27.33	31.66	29.71	23.82	32.08	29.63
Tensor4D [22]	10h	23.21	25.33	30.94	26.09	26.62	25.46	21.74	26.30	25.71
4DGS [27]	40m	24.82	33.50	38.72	31.23	35.60	32.23	26.37	35.96	32.30
D3DGS [30]	35m	24.90	37.42	41.24	**37.31**	**44.25**	37.84	**41.30**	**42.56**	**38.35**
SC-GS [8]	1h	25.00	**38.11**	**43.48**	36.69	40.28	**38.87**	32.40	39.43	36.78
RealTime4DGS† [29]	-	-	-	-	-	-	-	-	-	34.09
V4D [5]	6h	25.19	34.04	41.87	29.93	35.14	33.34	26.12	33.32	32.37
HexPlane [2]	1h	24.79	30.95	39.57	27.64	32.87	30.21	23.50	32.05	30.20
Ours	30m	24.72	34.19	43.04	33.03	37.17	35.25	29.16	37.16	34.22

Methods	Train. ↓	SSIM ↑								
		lego	*jumpingjacks*	*bouncingballs*	*hook*	*standup*	*trex*	*hellwarrior*	*mutant*	*Average*
D-NeRF [21]	1d	0.836	0.974	0.983	0.961	0.980	0.971	0.948	0.971	0.953
TiNeuVox [3]	**28m**	0.904	0.977	0.992	0.959	0.979	0.966	0.963	0.960	0.963
K-Planes-hybrid [4]	1h	**0.947**	0.970	0.993	0.947	0.978	0.972	0.947	0.971	0.966
K-Planes-explicit [4]	1h	0.946	0.963	0.989	0.940	0.973	0.970	0.944	0.969	0.962
Tensor4D [22]	10h	0.890	0.948	0.978	0.939	0.959	0.936	0.931	0.940	0.940
4DGS [27]	40m	0.931	0.981	0.993	0.970	0.986	0.980	0.964	0.986	0.974
D3DGS [30]	35m	0.942	0.989	0.995	0.986	0.995	**0.993**	**0.986**	**0.995**	0.985
SC-GS [8]	1h	0.941	**0.992**	**0.996**	**0.990**	**0.996**	**0.993**	0.985	**0.995**	**0.986**
RealTime4DGS† [29]	-	-	-	-	-	-	-	-	-	0.980
V4D [5]	6h	0.945	0.980	**0.996**	0.961	0.985	0.980	0.961	0.972	0.973
HexPlane [2]	1h	0.935	0.972	0.993	0.949	0.979	0.970	0.948	0.967	0.964
Ours	30m	0.942	0.986	**0.996**	0.984	0.992	0.989	0.978	0.990	0.982

Methods	Train. ↓	LPIPS_v ↓								
		lego	*jumpingjacks*	*bouncingballs*	*hook*	*standup*	*trex*	*hellwarrior*	*mutant*	*Average*
D-NeRF [21]	1d	0.167	0.043	0.112	0.121	0.022	0.042	0.071	0.028	0.076
TiNeuVox [3]	**28m**	0.107	0.041	0.044	0.061	0.032	0.052	0.080	0.048	0.058
K-Planes-hybrid [4]	1h	**0.047**	0.051	0.035	0.070	0.033	0.036	0.090	0.037	0.050
K-Planes-explicit [4]	1h	0.049	0.062	0.044	0.075	0.042	0.039	0.094	0.040	0.056
Tensor4D [22]	10h	0.132	0.079	0.067	0.093	0.061	0.089	0.118	0.087	0.091
4DGS [27]	40m	0.064	0.028	0.031	0.034	0.020	0.029	0.056	0.021	0.036
D3DGS [30]	35m	0.049	0.018	**0.020**	0.019	0.010	**0.013**	0.034	**0.007**	0.021
SC-GS [8]	1h	0.051	**0.012**	0.022	**0.013**	**0.007**	0.015	**0.027**	0.008	**0.019**
RealTime4DGS† [29]	-	-	-	-	-	-	-	-	-	-
V4D [5]	6h	0.049	0.031	0.022	0.053	0.023	0.029	0.065	0.036	0.038
HexPlane [2]	1h	0.064	0.046	0.029	0.064	0.032	0.044	0.081	0.043	0.050
Ours	30m	0.049	0.022	0.023	0.019	0.013	0.018	0.038	0.012	0.024

Methods	Train. ↓	LPIPS_a ↓								
		lego	*jumpingjacks*	*bouncingballs*	*hook*	*standup*	*trex*	*hellwarrior*	*mutant*	*Average*
D-NeRF [21]	1d	0.060	0.029	0.027	0.027	0.013	0.024	0.041	0.016	0.030
TiNeuVox [3]	**28m**	0.068	0.031	0.014	0.042	0.021	0.041	0.059	0.039	0.039
K-Planes-hybrid [4]	1h	**0.031**	0.037	0.013	0.049	0.021	0.030	0.072	0.027	0.035
K-Planes-explicit [4]	1h	0.032	0.049	0.020	0.057	0.030	0.034	0.075	0.030	0.041
Tensor4D [22]	10h	0.114	0.093	0.046	0.101	0.063	0.101	0.149	0.090	0.095
4DGS [27]	40m	0.042	0.018	0.012	0.022	0.011	0.020	0.036	0.011	0.021
D3DGS [30]	35m	0.033	0.012	**0.004**	0.011	0.005	**0.006**	0.019	**0.003**	0.012
SC-GS [8]	1h	0.034	**0.007**	**0.004**	**0.007**	**0.003**	0.008	**0.014**	0.004	**0.010**
RealTime4DGS† [29]	-	-	-	-	-	-	-	-	-	-
V4D [5]	6h	0.032	0.019	0.005	0.034	0.013	0.021	0.045	0.024	0.024
HexPlane [2]	1h	0.044	0.032	0.009	0.049	0.021	0.035	0.066	0.035	0.036
Ours	30m	0.032	0.015	**0.004**	0.012	0.007	0.013	0.027	0.006	0.014

Single-scale D-NeRF Dataset We conducted an evaluation of several approaches using three metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [25], and Learned Perceptual Image Patch Similarity (LPIPS) [32]. The results are presented in Tab. 1. All methods are trained with full-resolution images (800×800) from the training set and rendered at full resolution for both quantitative evaluation and qualitative comparisons. The cost of training time is achieved from our experiments on one NVIDIA GeForce RTX 3090 GPU. For method marked with "†" [29], we directly use the metrics from their paper. In the single-scale dataset, though anti-aliasing feature is not a dominant factor of the final rendering results, our method consistently surpasses all NeRF-based baseline methods [2–5,21,22] across all measured criteria, thanks to its ability to reconstruct high-quality details. However, concurrent works [8, 27, 29, 30] utilized 3DGS [9] along with its differentiable rasterizer, achieving efficient dynamic scene representation. Our proposed method outperforms [27, 29], but not [8, 30]. Though these two works have greatly enhanced the rendering quality on the monocular single-scale synthetic dataset, they suffer from aliasing artifacts on the multi-scale dataset captured at various distances, as mentioned in [9] . Please refer to the section below for comparisons on the multi-scale D-NeRF dataset. Although TiNeuVox [3] requires slightly less training time compared to our method, our proposed method (**DMiT**) demonstrates a significantly faster inference time, almost 7× faster than TiNeuVox. This speed improvement is achieved while maintaining a higher rendering quality, which may be attributed to the efficiency and compactness of our deformable and mipmapped tri-plane representation. While achieving impressive efficiency as a NeRF-based framework, the rendering speed is not in real-time like 3DGS-based methods [8, 27, 29, 30].

Multi-scale D-NeRF Dataset To demonstrate the effectiveness of our method for anti-aliasing, we list per-scene results in Tab. 2 by averaging over four resolutions. In addition, we present the comparison of rendering results of both full-resolution and low-resolution with baseline methods [3, 4, 21, 22, 27, 30]. These comparisons are illustrated in Fig. 1 and Fig. 2, where K-Planes-H denotes the hybrid architecture of K-Planes [4] and K-Planes-E denotes the explicit architecture. We also provide a low-resolution rendering comparison labeled with the corresponding PSNR/SSIM metrics in Fig. 3. The proposed methodology accurately depicts dynamic scenes with enhanced level of details at varying distances, closely approximating the ground truth (GT). Other baseline methods either lose fine-grained details or suffer from noticeable artifacts. 3DGS-based methods [27, 30] generate blurry images when rendering high-resolution images (corresponding to small capture distances) and dilated objects when rendering low-resolution images (corresponding to large capture distances) as shown in Fig. 1 and Fig. 2. These artifacts have been thoroughly discussed in [31]. Our supplementary video provides a more comprehensive observation of the improved visual quality achieved by the utilization of anti-aliasing techniques in dynamic scene rendering.

Table 2: Quantitative per-scene results on the test set of the multi-scale D-NeRF dataset. **Bold** means the best performance and underline means the second best performance.

Methods	Train. ↓	PSNR ↑								
		lego	*jumpingjacks*	*bouncingballs*	*hook*	*standup*	*trex*	*hellwarrior*	*mutant*	*Average*
D-NeRF [21]	1d	23.46	30.18	28.75	29.69	33.36	28.39	23.79	31.99	28.70
TiNeuVox [3]	**28m**	25.14	33.27	36.74	30.94	34.11	32.12	27.77	32.74	31.60
K-Planes-hybrid [4]	1h	24.95	26.47	31.32	27.02	29.24	29.22	21.90	30.69	27.60
K-Planes-explicit [4]	1h	25.04	25.90	30.53	26.52	28.98	28.69	21.60	30.54	27.23
Tensor4D [22]	10h	23.21	25.38	29.32	25.78	26.14	26.34	21.81	25.95	25.49
4DGS [27]	40m	25.11	32.73	36.16	29.14	32.27	30.43	26.82	31.29	30.49
D3DGS [30]	35m	25.67	34.91	38.54	31.34	34.57	34.28	29.14	32.96	32.68
HexPlane [2]	1h	25.04	27.58	33.72	27.11	30.85	29.76	22.84	31.54	28.56
V4D [5]	6h	25.20	33.13	36.36	29.87	34.19	32.88	26.59	33.45	31.46
SC-GS [8]	1h	23.16	**35.41**	38.47	31.79	34.46	25.94	29.26	32.83	31.42
RealTime4DGS [29]	-	24.34	30.95	33.84	27.98	32.20	29.31	23.84	31.80	29.28
Ours	30m	**25.92**	35.30	**42.63**	**34.29**	**38.17**	**37.47**	**30.04**	**38.37**	**35.27**

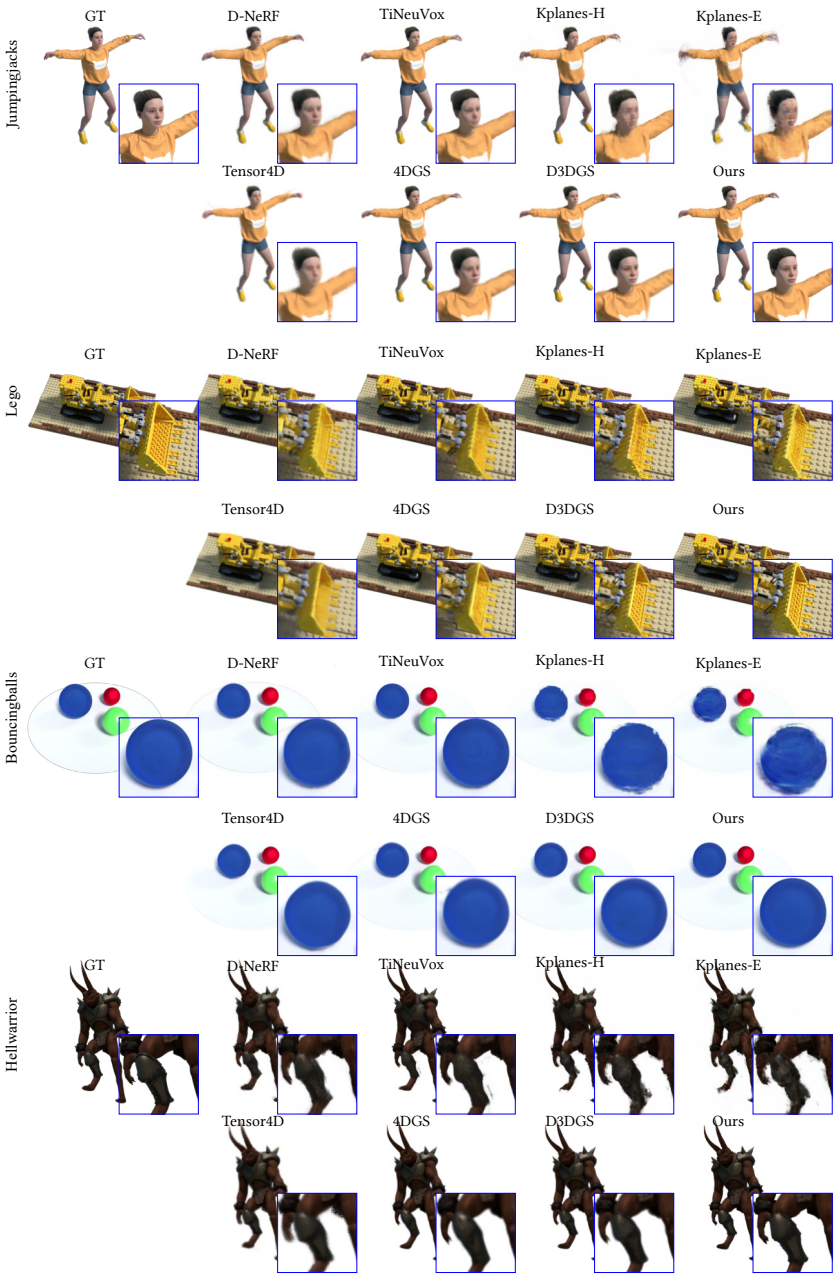
Methods	Train. ↓	SSIM ↑								
		lego	*jumpingjacks*	*bouncingballs*	*hook*	*standup*	*trex*	*hellwarrior*	*mutant*	*Average*
D-NeRF [21]	1d	0.867	0.961	0.958	0.946	0.972	0.953	0.935	0.962	0.944
TiNeuVox [3]	**28m**	0.920	0.981	0.970	0.972	0.985	0.975	0.967	0.977	0.968
K-Planes-hybrid [4]	1h	0.935	0.947	0.959	0.949	0.970	0.962	0.918	0.974	0.952
K-Planes-explicit [4]	1h	0.934	0.937	0.955	0.942	0.966	0.959	0.912	0.973	0.947
Tensor4D [22]	10h	0.876	0.935	0.947	0.939	0.954	0.932	0.924	0.942	0.931
4DGS [27]	40m	0.915	0.980	0.976	0.964	0.981	0.971	0.963	0.974	0.966
D3DGS [30]	35m	0.935	0.988	0.988	0.978	0.989	0.987	0.975	0.984	0.978
HexPlane [2]	1h	0.931	0.951	0.963	0.951	0.978	0.966	0.935	0.976	0.957
V4D [5]	6h	0.936	0.982	0.969	0.970	0.988	0.979	0.963	0.983	0.971
SC-GS [8]	1h	0.915	**0.989**	0.988	0.980	0.989	0.943	0.976	0.984	0.970
RealTime4DGS [29]	-	0.896	0.972	0.977	0.958	0.982	0.966	0.939	0.978	0.958
Ours	30m	**0.948**	0.988	**0.995**	**0.988**	**0.993**	**0.992**	**0.979**	**0.993**	**0.985**

Methods	Train. ↓	LPIPS_v ↓								
		lego	*jumpingjacks*	*bouncingballs*	*hook*	*standup*	*trex*	*hellwarrior*	*mutant*	*Average*
D-NeRF [21]	1d	0.095	0.060	0.066	0.063	0.033	0.049	0.085	0.041	0.062
TiNeuVox [3]	**28m**	0.077	0.030	0.054	0.044	0.024	0.044	0.063	0.033	0.046
K-Planes-hybrid [4]	1h	0.056	0.059	0.057	0.069	0.031	0.049	0.119	0.031	0.059
K-Planes-explicit [4]	1h	0.058	0.086	0.062	0.076	0.040	0.056	0.123	0.033	0.067
Tensor4D [22]	10h	0.105	0.076	0.078	0.076	0.050	0.077	0.102	0.069	0.079
4DGS [27]	40m	0.078	0.030	0.045	0.042	0.024	0.037	0.051	0.034	0.042
D3DGS [30]	35m	0.051	0.019	0.029	0.028	0.015	0.020	0.038	0.022	0.028
HexPlane [2]	1h	0.059	0.048	0.052	0.057	0.025	0.043	0.081	0.033	0.050
V4D [5]	6h	0.054	0.025	0.049	0.042	0.019	0.032	0.056	0.027	0.038
SC-GS [8]	1h	0.082	**0.018**	0.028	0.027	0.015	0.068	0.037	0.024	0.037
RealTime4DGS [29]	-	0.102	0.042	0.045	0.053	0.022	0.041	0.080	0.028	0.052
Ours	30m	**0.036**	0.019	**0.016**	**0.016**	**0.010**	**0.010**	**0.034**	**0.008**	**0.019**

Methods	Train. ↓	LPIPS_v ↓								
		lego	*jumpingjacks*	*bouncingballs*	*hook*	*standup*	*trex*	*hellwarrior*	*mutant*	*Average*
D-NeRF [21]	1d	0.065	0.044	0.047	0.050	0.031	0.042	0.058	0.035	0.047
TiNeuVox [3]	**28m**	0.048	0.020	0.022	0.023	0.014	0.031	0.035	0.021	0.027
K-Planes-hybrid [4]	1h	0.043	0.047	0.034	0.038	0.021	0.039	0.076	0.020	0.040
K-Planes-explicit [4]	1h	0.044	0.068	0.037	0.043	0.027	0.044	0.080	0.021	0.045
Tensor4D [22]	10h	0.087	0.082	0.051	0.069	0.052	0.080	0.111	0.063	0.074
4DGS [27]	40m	0.057	0.033	0.026	0.047	0.028	0.037	0.056	0.036	0.040
D3DGS [30]	35m	0.036	0.021	0.015	0.029	0.017	0.017	0.038	0.022	0.024
HexPlane [2]	1h	0.043	0.036	0.023	0.033	0.017	0.034	0.053	0.021	0.033
V4D [5]	6h	0.039	0.016	0.022	0.023	0.011	0.025	0.030	0.014	0.023
SC-GS [8]	1h	0.070	0.019	0.014	0.028	0.018	0.073	0.041	0.024	0.036
RealTime4DGS [29]	-	0.088	0.046	0.030	0.059	0.025	0.039	0.083	0.026	0.049
Ours	30m	**0.025**	**0.013**	**0.004**	**0.009**	**0.005**	**0.007**	**0.020**	**0.004**	**0.011**


Standup
T-Rex
Mutant
Hook
GT
D-NeRF
TiNeuVox
Kplanes-H
Kplanes-E
Tensor4D
4DGS
D3DGS
Ours

(a)

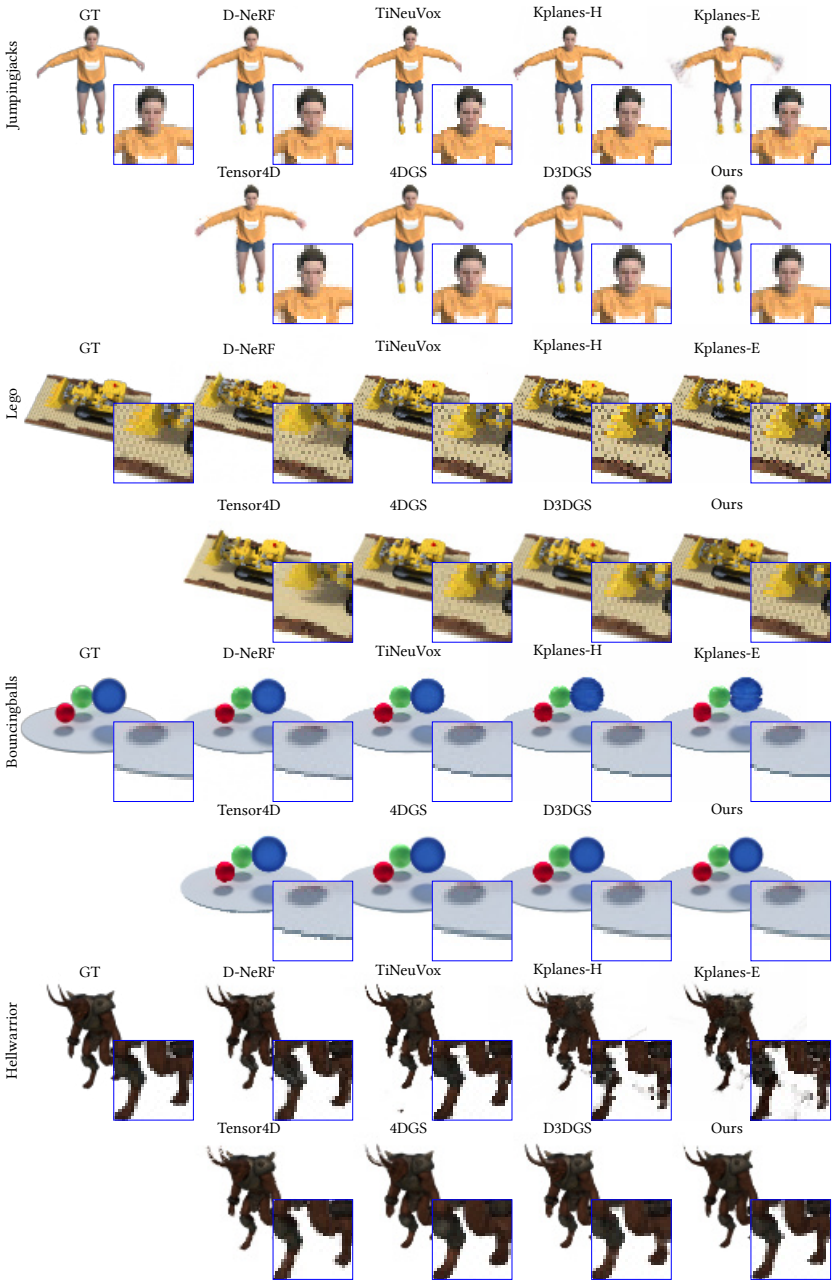


(b)

Fig. 1: Qualitative comparison of reconstruction results between our method and baseline methods on the multi-scale D-NeRF dataset.


Standup
Mutant
T-Rex
Hook
GT
D-NeRF
TiNeuVox
Kplanes-H
Kplanes-E
Tensor4D
4DGS
D3DGS
Ours

(a)



(b)

Fig. 2: Qualitative comparison of low-resolution rendering results between our method and baseline methods on the multi-scale D-NeRF dataset for visualization of the anti-aliasing effects.

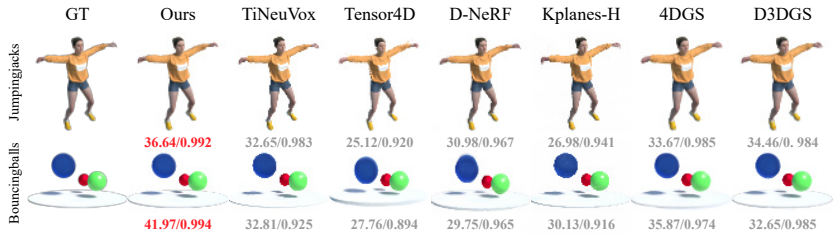
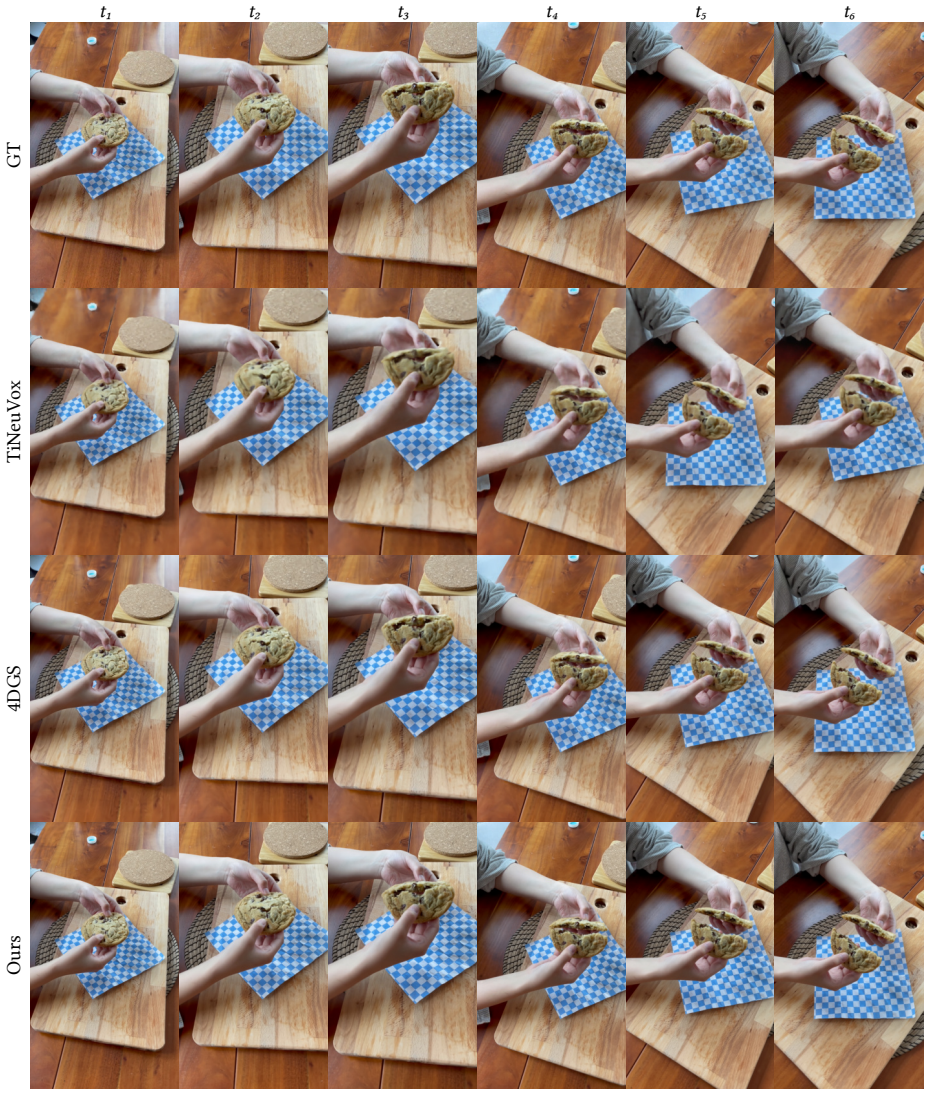


Fig. 3: Qualitative comparison of low-resolution rendering results between our method and baseline methods on the multi-scale D-NeRF dataset with PSNR/SSIM metrics shown at the bottom.

HyperNeRF Dataset Per-scene evaluation metrics under PSNR, LPIPS [32] and MS-SSIM [26] are provided in Tab. 3 with additional results of several highly related methods on this dataset for detailed comparisons. Metrics marked with "†" are adopted from the corresponding paper. The quantitative experiments were conducted following the procedures in [3, 19], with training and rendering performed at a resolution of 960x540 pixels, which is half of the standard 1080p resolution. In addition, it should be noted that the images used for qualitative comparisons undergo training and rendering processes at the same resolution as those used in quantitative tests. This is due to limitation of our experimental device, which has a memory capacity of 24GB, preventing the successful loading of full-resolution images for testing.

Despite requiring 32× less training time compared to HyperNeRF [19], our method produces comparable high-quality rendering results with fine details and glossy appearance well represented, as shown in Fig. 6 of the main paper. In contrast, TiNeuvox [3] tends to provide blurry results and have limitations in accurately restoring intricate textual patterns with high-frequency components, despite its somewhat faster convergence rate. When compared to synthetic scenes like the D-NeRF dataset, inaccuracies in camera parameters provided by the HyperNeRF dataset may negatively impact reconstruction quality. However, the experiments also demonstrate the effectiveness of our method in handling scenes with inaccurate camera registration. Furthermore, previous works [3, 19] have shown that metrics such as PSNR and MS-SSIM may not accurately reflect the perceptual quality of images. Conversely, LPIPS has been found to align closely with human perception. Our approach demonstrates superior performance in terms of average PSNR while being comparable to HyperNeRF [19] and Nerfies [18] using the LPIPS metric with 'ALEX' as the underlying backbone. Additionally, our **DMiT** requires much less training time and enables faster inference.

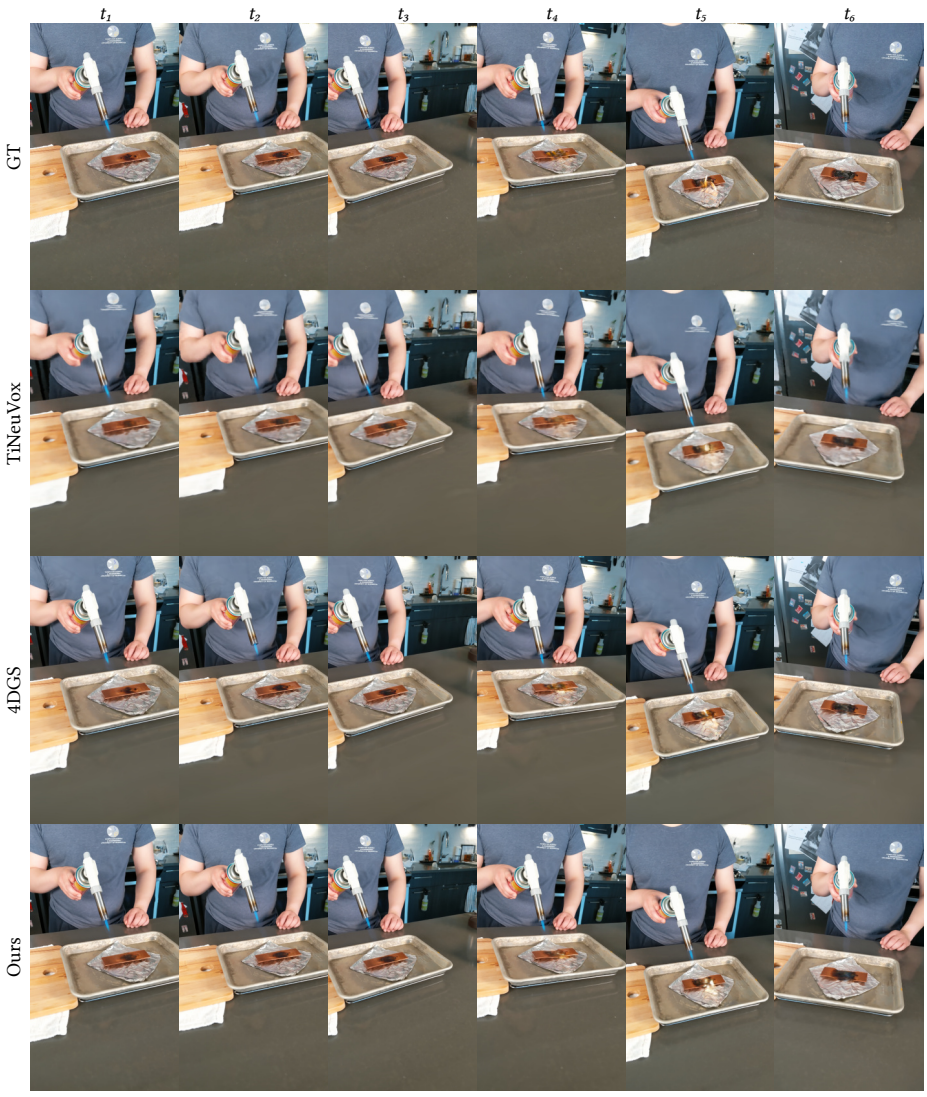
NeRF-DS Dataset We further evaluate our method on monocular real-world dataset NeRF-DS [28] , which consists of common dynamic specular scenes. The detailed per-scene metrics PSNR, LPIPS [32] and SSIM [25] against several SOTA baselines [3, 8, 27, 28, 30] are provided in Tab. 4. More qualitative ren-



(a) *split-cookie*



(b) *vrig-chicken*



(c) *torchocolate*

Fig. 4: Qualitative comparison of time interpolation and novel view synthesis results between our method and baseline methods on three scenes from the HyperNeRF dataset.

Table 3: Quantitative per-scene results on the test set of 5 scenes from the HyperNeRF dataset. **Bold** means the best performance and underline means the second best performance.

Methods	Train. ↓	PSNR ↑					
		torchocolate	*hand*	*cut-lemon*	*3dprinter*	*chicken*	*Average*
HyperNeRF† [19]	∼32h	**28.00**	**30.07**	**31.80**	20.00	26.90	27.35
Neural Volumes† [12]	hours	24.60	29.30	28.80	16.20	17.60	23.30
NSFF† [11]	hours	22.30	24.90	28.00	27.70	26.90	25.96
Nerfies† [18]	hours	27.80	29.90	30.80	20.60	26.70	27.16
4DGS [27]	3h	27.61	27.72	29.69	21.85	28.36	27.05
D3DGS [30]	2.5h	27.52	29.09	28.10	20.35	22.78	25.57
TiNeuVox [3]	**40m**	26.85	27.41	28.49	22.77	28.37	26.78
V4D [5]	8h	26.26	27.60	29.41	**23.34**	28.48	27.02
NeRFPlayer† [23]	-	-	-	-	22.90	26.30	24.60
Ours	1.5h	26.34	29.62	30.94	22.21	**29.88**	**27.80**

Methods	Train. ↓	SSIM ↑					
		torchocolate	*hand*	*cut-lemon*	*3dprinter*	*chicken*	*Average*
HyperNeRF† [19]	∼32h	**0.962**	**0.950**	0.956	0.821	0.948	**0.927**
Neural Volumes† [12]	hours	0.917	0.912	0.951	0.665	0.615	0.812
NSFF† [11]	hours	0.883	0.797	0.904	**0.947**	0.944	0.895
Nerfies† [18]	hours	0.959	0.940	0.946	0.830	0.943	0.924
4DGS [27]	3h	0.957	0.877	0.927	0.808	0.929	0.900
D3DGS [30]	2.5h	0.959	0.935	0.917	0.759	0.727	0.859
TiNeuVox [3]	**40m**	0.933	0.856	0.887	0.820	0.916	0.882
V4D [5]	8h	0.933	0.875	0.911	0.840	0.930	0.898
NeRFPlayer† [23]	-	-	-	-	0.810	0.905	0.858
Ours	1.5h	0.953	0.943	**0.958**	0.823	**0.954**	0.926

Methods	Train. ↓	$LPIPS_a$ ↓					
		torchocolate	*hand*	*cut-lemon*	*3dprinter*	*chicken*	*Average*
HyperNeRF† [19]	∼32h	0.172	0.150	0.210	0.111	0.079	**0.144**
Neural Volumes† [12]	hours	0.189	0.213	0.190	0.330	0.336	0.252
NSFF† [11]	hours	0.253	0.329	0.238	0.125	0.106	0.210
Nerfies† [18]	hours	0.169	0.171	0.223	**0.108**	**0.078**	0.150
4DGS [27]	3h	0.199	0.299	0.337	0.271	0.226	0.266
D3DGS [30]	2.5h	**0.162**	0.163	0.310	0.290	0.194	0.224
TiNeuVox [3]	**40m**	0.306	0.417	0.463	0.338	0.347	0.374
V4D [5]	8h	0.291	0.334	0.378	0.312	0.247	0.312
NeRFPlayer† [23]	-	-	-	-	-	-	-
Ours	1.5h	0.186	**0.139**	**0.146**	0.177	0.115	0.153

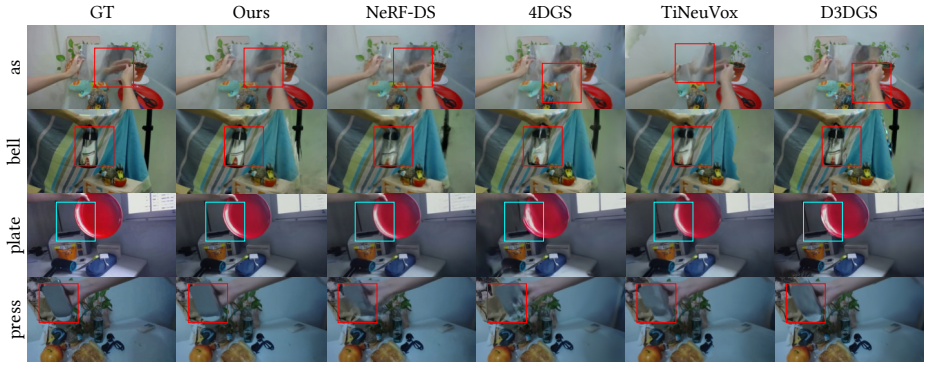


Fig. 5: Qualitative comparison of reconstruction results between our method and baseline methods on the NeRF-DS dataset.

Table 4: Quantitative per-scene results on the test set of all scenes from NeRF-DS dataset. **Bold** means the best performance and underline means the second best performance.

Methods	Train. ↓	PSNR ↑							
		as	*basic*	*bell*	*cup*	*plate*	*press*	*sieve*	*Average*
TiNeuVox [3]	**40m**	21.97	**20.79**	22.60	20.31	20.46	24.88	21.62	21.80
NeRF-DS [28]	8h	25.03	19.82	23.31	25.04	20.68	25.61	25.78	23.61
4DGS [27]	3h	25.27	19.26	22.78	20.29	17.90	23.92	25.27	22.10
D3DGS [30]	2h	25.98	19.61	25.35	24.47	20.16	25.21	25.39	23.74
SC-GS [8]	3h	26.20	19.60	25.10	24.50	20.20	**26.60**	26.00	24.03
Ours	1h	**26.47**	20.58	**25.49**	**25.35**	**20.76**	26.56	**26.13**	**24.48**

Methods	Train. ↓	SSIM ↑							
		as	*basic*	*bell*	*cup*	*plate*	*press*	*sieve*	*Average*
TiNeuVox [3]	**40m**	0.835	**0.818**	0.814	0.825	0.798	0.862	0.826	0.825
NeRF-DS [28]	8h	0.883	0.817	0.817	0.875	0.813	0.875	0.888	0.853
4DGS [27]	3h	0.853	0.763	0.808	0.709	0.733	0.803	0.854	0.789
D3DGS [30]	2h	0.874	0.782	0.844	0.880	0.802	0.857	0.869	0.844
SC-GS [8]	3h	-	-	-	-	-	-	-	-
Ours	1h	**0.897**	0.814	**0.857**	**0.885**	**0.821**	**0.898**	**0.889**	**0.866**

Methods	Train. ↓	LPIPS_a ↓							
		as	*basic*	*bell*	*cup*	*plate*	*press*	*sieve*	*Average*
TiNeuVox [3]	**40m**	0.293	0.138	0.167	0.250	0.221	0.178	0.241	0.213
NeRF-DS [28]	8h	0.128	**0.126**	0.134	0.118	**0.149**	0.143	**0.099**	0.128
4DGS [27]	3h	0.153	0.155	0.142	0.402	0.299	0.182	0.144	0.207
D3DGS [30]	2h	0.127	0.131	0.109	0.112	0.181	0.130	0.100	0.127
SC-GS [8]	3h	-	-	-	-	-	-	-	-
Ours	1h	**0.104**	0.127	**0.101**	**0.106**	0.152	**0.118**	0.100	**0.115**

dering results are shown in Fig. 5. All methods are trained using full-resolution RGB images with their released official implementation. Metrics of SC-GS [8] are sourced from their paper, where SSIM/LPIPS(ALEX) metrics are not provided. Note that NeRF-DS [28] also utilized GT front masks as additional inputs. Our proposed method restores more intricate details of specular appearance and renders with fewer artifacts while maintaining more stable deformation across frames.

PlenopticVideo Dataset The PlenopticVideo dataset [10] is a multi-view dataset captured using 21 cameras simultaneously. The central camera is used for testing, while 18 selected cameras are used for training. Following [10], we evaluate our methods using the first 300 frames downscaled by a factor of 2. PSNR, MS-SSIM [26] and LPIPS [32] are chosen as evaluation criteria. Please refer to Tab. 5 for per-scene metrics of all scenes excluding *coffee martini*. Due to our machine's limited CPU memory, we did not evaluate all baseline methods ourselves except for HyperReel [1], 4DGS [27] and RealTime4DGS [29](marked with "*"). Note that NeRFPlayer [23] only reported metrics on SSIM [25] instead of MS-SSIM [26]. Our proposed method is trained every 60 frames due to insufficient CPU memory for loading all training data simultaneously. As demonstrated in [1], the reported SSIM may be higher than expected due to inconsistent evaluation libraries or inappropriate parameter settings. Despite SSIM metrics being less reliable, our proposed method still outperforms other baselines in PSNR and MS-SSIM. Though [10] has a lower LPIPS score than our method, it requires approximately 1344 GPU hours to train one full model, while our methods only need 1.5 hours for each training on 60 frames. In addition, more rendering results

Table 5: Quantitative per-scene results on the test set of the PlenopticVideo dataset. **Bold** means the best performance and underline means the second best performance.

Methods	PSNR ↑					
	cook spinach	*cut roasted beef*	*flame salmon*	*flame steak*	*sear steak*	*Average*
K-Planes-E [4]	32.19	31.93	28.71	31.80	31.89	31.30
LLFF [15]	-	-	23.24	-	-	23.24
DyNeRF [10]	-	-	29.58	-	-	29.58
MixVoxels-L [24]	31.61	31.30	29.92	31.21	31.43	31.09
HexPlane [2]	31.86	32.71	29.26	31.92	32.09	31.57
NeRFPlayer [23]	30.58	29.35	**31.65**	31.93	29.13	30.53
HyperReel* [1]	31.98	**32.88**	28.37	**32.12**	**32.34**	31.54
4DGS* [27]	31.42	32.04	26.03	30.24	28.85	29.71
RealTime4DGS* [29]	**32.58**	32.69	29.38	32.02	31.99	**31.73**
Ours	32.24	32.52	30.05	31.82	32.02	**31.73**

Methods	SSIM ↑					
	cook spinach	*cut roasted beef*	*flame salmon*	*flame steak*	*sear steak*	*Average*
K-Planes-E [4]	0.968	0.965	0.942	0.970	0.971	0.963
LLFF [15]	-	-	0.848	-	-	0.848
DyNeRF [10]	-	-	0.961	-	-	0.961
MixVoxels-L [24]	0.965	0.965	0.945	0.970	0.971	0.963
HexPlane [2]	0.966	0.970	0.960	0.976	0.972	0.969
NeRFPlayer [23]	0.929	0.908	0.940	0.950	0.908	0.927
HyperReel* [1]	0.965	0.964	0.945	0.974	0.976	0.965
4DGS* [27]	0.967	0.968	0.919	0.971	0.962	0.957
RealTime4DGS* [29]	0.972	**0.975**	0.961	0.970	0.970	0.969
Ours	**0.975**	**0.975**	**0.966**	**0.977**	**0.978**	**0.974**

Methods	LPIPS_a ↓					
	cook spinach	*cut roasted beef*	*flame salmon*	*flame steak*	*sear steak*	*Average*
K-Planes-E [4]	-	-	-	-	-	-
LLFF [15]	-	-	0.235	-	-	0.235
DyNeRF [10]	-	-	**0.083**	-	-	**0.083**
MixVoxels-L [24]	0.122	**0.088**	0.116	0.088	0.080	0.099
HexPlane [2]	0.097	0.094	0.097	0.081	0.079	0.090
NeRFPlayer [23]	0.113	0.144	0.098	0.088	0.138	0.116
HyperReel* [1]	0.104	0.098	0.151	0.093	0.089	0.107
4DGS* [27]	0.110	0.111	0.160	0.097	0.109	0.117
RealTime4DGS* [29]	0.099	0.097	0.084	0.110	0.115	0.101
Ours	**0.090**	0.089	0.094	**0.080**	**0.078**	0.086

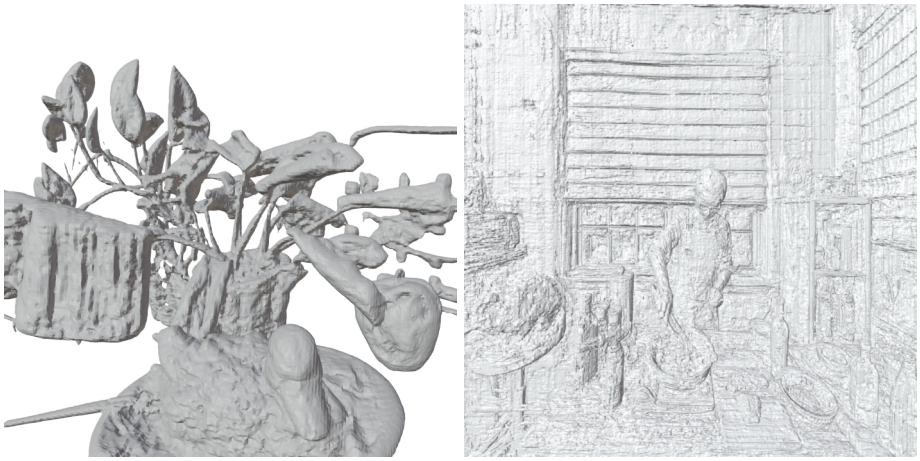


Fig. 6: Qualitative comparison of reconstruction results between our method and baseline methods on the PlenopticVideo dataset.



(a) canonical geometry **(b)** mixing of all time steps **(c)** comparison

Fig. 7: Comparisons between the learned canonical geometry and mixing of renderings across all training time steps of *T-Rex* scene from D-NeRF.



(a) *cup* **(b)** *cook spinach*

Fig. 8: Canonical geometry visualization of one real-world scene *cup* from NeRF-DS [28](Fig. 8a) and one real-world scene *cook spinach* from PlenopticVideo [10](Fig. 8b).

and comparisons are provided in Fig. 6. Our proposed method can recover more fine-grained details and generate rendering results closest to the ground truth.

C Canonical Space Analysis

As shown in Fig. 4(b) in the main paper, the learned canonical geometry of our proposed method outperforms baseline methods for two main reasons. Firstly, the powerful encoding scheme from Tri-MipRF [7] effectively generates intricate and highly detailed geometry, as demonstrated in Fig. 4 of their main paper. Secondly, our effective joint refinement strategy enables the separation of motion and geometry, resulting in a valid and detailed canonical space and better deformation results. In contrast, D-NeRF [21] produces poor canonical results even with an explicitly defined canonical frame. Since this frame only constitutes

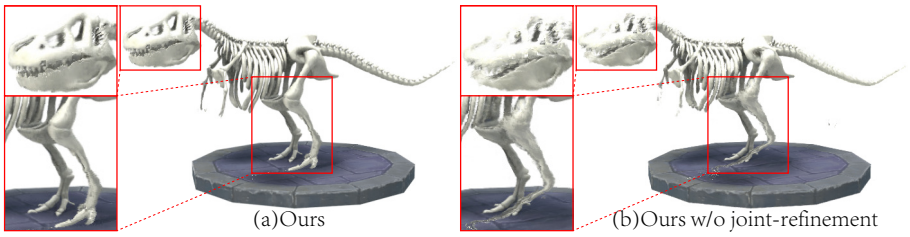


Fig. 9: Ablation of joint refinement. Zoom in for details.

a small portion of the training set, D-NeRF struggles to represent canonical geometry aligned with the chosen canonical frame. Please refer to Fig. 4(b) in the main paper for the comparison of canonical spaces between the two methods.

To gain a deeper comprehension of the learned canonical space, we fused it with the novel-view-synthesis results of the *T-Rex* scene from D-NeRF at all training timestamps, while maintaining a consistent camera position. Please refer to Fig. 7 for visualizations comparing the canonical geometry to the mixed renderings across all time steps. The learned canonical space does not represent a specific time step in the training set but rather serves as an "average" of all time steps. This averaged canonical space enables effective deformation convergence and the decoupling of motion and geometry. Additionally, we provide more canonical geometry visualization results of real-world scenes in Fig. 8. Note that the geometry results from PlenopticVideo seem squashed due to the application of NDC contraction, as done in previous works [1, 2, 4]. The quality of geometry reconstruction is not the main contribution of our work, as it is not built upon SDF. The primary purpose of the canonical geometry visualizations is to verify the efficacy of the suggested method in separating dynamics from static geometry. As shown in Fig. 8, floaters or specular appearance severely affect the quality of canonical geometry. We leaver this for the future work to achieve higher-quality canonical geometry reconstruction of dynamic scenes with specular objects or objects undergoing complex non-rigid motions.

D Additional Ablation results

We further provide visualized ablation results to evaluate the warm-up stage and the joint-refinement strategy introduced in Sec. 3.3 of the main paper. We present reconstruction results of *T-Rex* from D-NeRF in Fig. 9 demonstrate the impact of the proposed joint-refinement strategy. Results without joint refinement exhibit noticeable fuzzy artifacts and deformation inaccuracies, as highlighted in red boxes. Note that the warm-up stage of deformation does not aim to resolve the convergence gap between deformation and geometry. The progressive resolution growth of tri-miplanes has already applied implicit regularization to address the fast convergence of canonical space. The warm-up stage is designed to prevent instability in deformation during the upsampling of tri-miplanes, which may otherwise lead to inferior renderings, as shown in Fig. 10.



(a) Ours w/o deformation warm-up (b) Ours w/ deformation warm-up (c) GT close-up

Fig. 10: Comparisons between *hellwarrior* scene from D-NeRF using the proposed methods without deformation warm-up stage(a), with deformation warm-up stage(b) and zoom-in of GT(c).

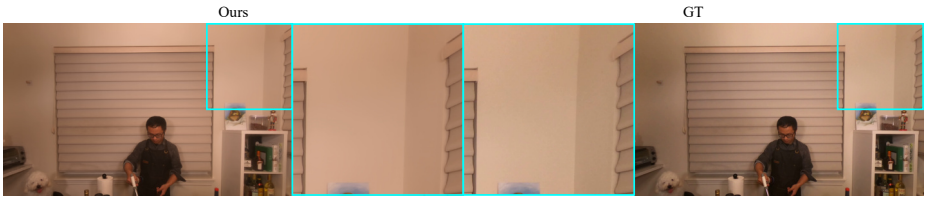


Fig. 11: Qualitative comparison between our rendering result and GT of scene *flame steak* from the PlenopticVideo dataset [10] with zoom-in comparison of the wall. Our proposed method generate a darker wall color compared to the GT.

E Discussions

Overfitting issues compared to multiplane methods. Methods like Hexplane [2] and K-Planes [4] also suffer from overfitting but for different reasons. These methods are based on time-conditioned representations rather than deformation-based approaches utilizing a canonical space, thus avoiding the coupling issue between deformation networks and canonical space geometry. However, the time-conditioned multi-plane features and functions can still overfit the input viewpoints and timesteps. Despite employing Total Variational loss regularization and a coarse-to-fine strategy to mitigate this, overfitting artifacts persist, as shown in Appendix B and the failure cases of Hexplane in their supplementary material.

Failure cases. As discussed in Sec. 5 of the main paper, our proposed method may produce false color when using unconstrained input. Please refer to Fig. 11 for a comparison between our rendering results and the GT images of the *flame steak* scene from the PlenopticVideo dataset [10]. The color of the white wall in our renderings appears slightly darker than in the GT images. This discrepancy is primarily due to non-uniform lighting conditions across different camera views in this dataset, leading to inconsistent wall colors in the training images. To address issues caused by unconstrained training data, it might be beneficial to

assign an appearance latent code as in [14] or introduce an explicit illumination model.

References

1. Attal, B., Huang, J.B., Richardt, C., Zollhoefer, M., Kopf, J., O'Toole, M., Kim, C.: Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16610–16620 (2023)
2. Cao, A., Johnson, J.: Hexplane: A fast representation for dynamic scenes. IEEE/CVF Conference on Computer Vision and Pattern Recognition (2023)
3. Fang, J., Yi, T., Wang, X., Xie, L., Zhang, X., Liu, W., Nießner, M., Tian, Q.: Fast dynamic radiance fields with time-aware neural voxels. In: SIGGRAPH Asia 2022 Conference Papers (2022)
4. Fridovich-Keil, S., Meanti, G., Warburg, F.R., Recht, B., Kanazawa, A.: K-planes: Explicit radiance fields in space, time, and appearance. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12479–12488 (2023)
5. Gan, W., Xu, H., Huang, Y., Chen, S., Yokoya, N.: V4d: Voxel for 4d novel view synthesis. IEEE Transactions on Visualization and Computer Graphics (2023)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256. JMLR Workshop and Conference Proceedings (2010)
7. Hu, W., Wang, Y., Ma, L., Yang, B., Gao, L., Liu, X., Ma, Y.: Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In: ICCV (2023)
8. Huang, Y.H., Sun, Y.T., Yang, Z., Lyu, X., Cao, Y.P., Qi, X.: Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4220–4230 (2024)
9. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (July 2023)
10. Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Newcombe, R., et al.: Neural 3d video synthesis from multi-view video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5521–5531 (2022)
11. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6498–6508 (2021)
12. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: learning dynamic renderable volumes from images. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019)
13. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2019)
14. Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7210–7219 (2021)

15. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019)
16. Müller, T.: tiny-cuda-nn (4 2021), https://github.com/NVlabs/tiny-cuda-nn
17. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics **41**(4), 102:1–102:15 (Jul 2022)
18. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: IEEE/CVF International Conference on Computer Vision. pp. 5865–5874 (2021)
19. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: HyperNeRF: a higher-dimensional representation for topologically varying neural radiance fields. ACM Transactions on Graphics **40**(6), 1–12 (2021)
20. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
21. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-NeRF: Neural radiance fields for dynamic scenes. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10318–10327 (2021)
22. Shao, R., Zheng, Z., Tu, H., Liu, B., Zhang, H., Liu, Y.: Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16632–16642 (2023)
23. Song, L., Chen, A., Li, Z., Chen, Z., Chen, L., Yuan, J., Xu, Y., Geiger, A.: Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. IEEE Transactions on Visualization and Computer Graphics **29**(5), 2732–2742 (2023)
24. Wang, F., Tan, S., Li, X., Tian, Z., Song, Y., Liu, H.: Mixed neural voxels for fast multi-view video synthesis. in 2023 ieee. In: CVF International Conference on Computer Vision (ICCV). pp. 19649–19659 (2023)
25. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004)
26. Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. In: The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003. vol. 2, pp. 1398–1402. Ieee (2003)
27. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4d gaussian splatting for real-time dynamic scene rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20310–20320 (2024)
28. Yan, Z., Li, C., Lee, G.H.: Nerf-ds: Neural radiance fields for dynamic specular objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8285–8295 (2023)
29. Yang, Z., Yang, H., Pan, Z., Zhang, L.: Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In: International Conference on Learning Representations (ICLR) (2024)

30. Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., Jin, X.: Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20331–20341 (2024)
31. Yu, Z., Chen, A., Huang, B., Sattler, T., Geiger, A.: Mip-splatting: Alias-free 3d gaussian splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19447–19456 (2024)
32. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 586–595 (2018)