Supplementary Material:
“HexPlane: A Fast Representation for Dynamic Scenes”

Contents

1. General Discussions

1.1. Broader Impacts

Our work aims to design an explicit representation for dynamic 3D scenes. We only reconstruct existing scenes and render images from different viewpoints and timesteps. Therefore, we don’t generate any new scenes or deceive contents which don’t exist before. Our current method is not intended and can also not be used to create fake materials, which could mislead others.

We use *Plenoptic Video dataset* [7] in our experiments, which contains human faces in videos. This dataset is a public dataset with License: CC-BY-NC 4.0 with consent.

Our method is hundreds of times faster than existing methods, consuming significantly less computation resources. Considering the GPU resource usages, our method could save considerably carbon emission.

1.2. Limitations and Future Directions

For comprehensively understanding HexPlane, we discuss its limitations and potential future improvements.

The ultimate goal of our paper is to propose and validate an explicit representation for dynamic scenes instead of purchasing SOTA numbers. To this end, we intend to make HexPlanes simple and general, making minor assumptions about the scenes and not introducing complicated tricks to improve performance. This principle leads to elegant solutions while potentially limiting performance as well. In the following, we will discuss these in detail.

Many methods [3, 11–13] use deformation and canonical fields to represent dynamic 3D scenes with monocular videos, where spacetime points are mapped into a static 3D scene represented by a canonical field. Again, we don’t employ deformation fields in our design since this assumption is not always held in the real world, especially for scenes with typology changes and new-emerging content. But this design is very effective for monocular videos since it introduces a solid information-sharing mechanism and allows learning 3D structures from very sparse views. HexPlane uses an inherent basis-sharing mechanism to cope with sparse observations. Although this design is shown to be powerful in our experiments, it is still less effective than the aforementioned deformation field, leading to degraded results for scenes with extremely sparse observations. Introducing deformation fields into HexPlane, like using HexPlane to represent deformation fields, would be an appealing improvement for monocular videos.

Similarly, category-specific priors like 3DMM [2] or SMPL [8] are even more powerful than deformation fields, which enormously improve results but are hardly limited to particular scenes. Combining these ideas with HexPlane for specific scenes would be very interesting.

Existing works demonstrated that explicit representations are prone to giving artifacts and require strong regularizations for good results, which also holds in HexPlane. There are color jittering and artifacts in the synthesized results, demanding stronger regularizations and other tricks to improve results further. Special spacetime regularizations and other losses like optical flow loss would be an interesting future direction to explore. Also, instead of simply representing everything using spherical coordinates in the paper, we could have a foreground and background model like NeRF++ [15], where the background is modeled in spher-

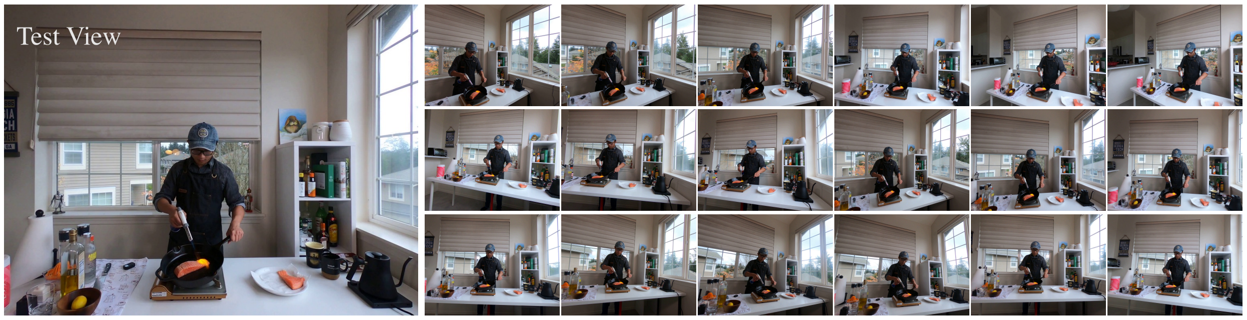


Figure 1. **Train and Test View of Plenoptic Video Dataset [7].** Plenoptic Video Dataset has 18 train views and 1 test view.

ical coordinates. Having separate foreground background models could noticeably improve the results. Moreover, rather than using the same basis for representing a long video, using a different basis for different video clips may give better results. We believe HexPlane could be further improved with these adjustments.

Besides dynamic novel view synthesis, we believe HexPlane could be utilized in a broader range of research, like dynamic scene generation or edits.

1.3. License

We provide licenses of assets used in our paper.

Plenoptic Video Dataset [7]. We evaluate our method on all all public scenes of Plenoptic Video dataset [7], except a-synchronize scene "coffee-martini". The dataset is in https://github.com/facebookresearch/Neural_3D_Video with License CC-BY-NC 4.0

D-NeRF Dataset [13]. We use D-NeRF dataset provided in https://github.com/albertpumarola/D-NeRF.

iPhone Dataset [4]. The iPhone dataset is provided in https://github.com/KAIR-BAIR/dycheck/, licensed under Apache-2.0 license.

Plenoptic Video Dataset Baselines [7]. For all baselines in this dataset, we use numbers reported in the original paper since these models are not publicly available.

D-NeRF Dataset Baselines [13]. D-NeRF model is in https://github.com/albertpumarola/D-NeRF and Tineuvox model [3] is in https://github.com/hustvl/TiNeuVox. Tineuvox is licensed under Apache License 2.0.

2. Training Details and More Results

2.1. Plenoptic Video Dataset [7].

Plenoptic Video Dataset [7] is a multi-view real-world video dataset, where each video is 10-second long. The training and testing views are shown in Figure 1.

We have $R_1 = 48, R_2 = 24, R_3 = 24$ for appearance HexPlance, where R_1, R_2, R_3 are basis numbers for $XY-ZT, XZ-YT, YZ-XT$ planes. For opacity HexPlane, we set $R_1 = 24, R_2 = 12, R_3 = 12$. We have different R_1, R_2, R_3 since scenes in this dataset are almost face-forwarding, demanding better representation along the XY plane. The scene is modeled using *normalized device coordinate* (NDC) [10] with min boundaries $[-2.5, -2.0, 0.0]$ and max boundaries $[2.5, 2.0, 1.0]$.

Instead of giving the same grid resolutions along X, Y, Z axes, we adjust them based on their boundary distances. That is, we give larger grid resolution to axis ranging a longer distance, like X axis from -2.5 to 2.5, and provide smaller grid resolution to axis going a shorter length, like Z axis from 0 to 1. The ratio of grid resolutions for different axes is the same as their distance ratios, while the total grid size number is manually controlled.

During training, HexPlane starts with a space grid size of 64^3 and doubles its resolution at 70k, 140k, and 210k to 512^3. The emptiness voxel is calculated at 50k and 100k iterations. The learning rate for feature planes is 0.02, and the learning rate for $\mathbf{V}^{RF}$ and neural network is 0.001. All learning rates are exponentially decayed. We use Adam [5] for optimization with $\beta_1 = 0.9, \beta_2 = 0.99$. We apply Total Variational loss on all feature planes with $\lambda = 0.0005$ for spatial axes and $\lambda = 0.001$ for temporal axes.

We follow the hierarch training pipeline as [7]. *HexPlane* in Table 1 uses 650k iterations, with 300k stage one training, 250k stage two training and 100k stage three training. *HexPlane†* uses 100k iterations in total, with 10k stage one training, 50k stage two training and 40k stage three training. According to [7], stage one is a global-median-based weighted sampling with $\gamma = 0.001$; stage two is also a global-median-based weighted sampling with $\gamma = 0.02$; stage three is a temporal-difference-based weighted sampling with $\alpha = 0.1$.

In evaluation, D-SSIM is computed as $\frac{1-\text{MS-SSIM}}{2}$ and LPIPS [16] is calculated using AlexNet [6]. We use default settings for Just-Objectionable-Difference (JOD) [9].

Each scene results are in Table 1, and more visualizations are in Figure 2. We found that *HexPlane* gives visually more smooth results than *HexPlane†*. Since we don't have

Table 1. **Results of Plenoptic Video Dataset [7].** We report results of each scene.

Model	Flame Salmon				Cook Spinach				Cut Roasted Beef			
	PSNR↑	D-SSIM ↓	LPIPS ↓	JOD ↑	PSNR↑	D-SSIM ↓	LPIPS ↓	JOD ↑	PSNR↑	D-SSIM ↓	LPIPS ↓	JOD ↑
HexPlane	29.470	0.018	0.078	8.16	32.042	0.015	0.082	8.32	32.545	0.013	0.080	8.59
HexPlane†	29.263	0.020	0.097	8.14	31.860	0.017	0.097	8.25	32.712	0.015	0.094	8.37
Model	**Flame Steak**				**Sear Steak**				**Average**			
	PSNR↑	D-SSIM ↓	LPIPS ↓	JOD ↑	PSNR↑	D-SSIM ↓	LPIPS ↓	JOD ↑	PSNR↑	D-SSIM ↓	LPIPS ↓	JOD ↑
HexPlane	32.080	0.011	0.066	8.61	32.387	0.011	0.070	8.66	31.705	0.014	0.075	8.47
HexPlane†	31.924	0.012	0.081	8.51	32.085	0.014	0.079	8.51	31.569	0.016	0.090	8.36

Table 2. **Per-Scene Results of D-NeRF Dataset [13].** We report results of each scene.

Model	Hell Warrior			Mutant			Hook		
	PSNR↑	SSIM↑	LPIPS ↑	PSNR↑	SSIM↑	LPIPS ↑	PSNR↑	SSIM↑	LPIPS ↑
T-NeRF	23.19	0.93	0.08	30.56	0.96	0.04	27.21	0.94	0.06
D-NeRF	25.02	0.95	0.06	31.29	0.97	0.02	29.25	0.96	0.11
TiNeuVox-S	27.00	0.95	0.09	31.09	0.96	0.05	29.30	0.95	0.07
TiNeuVox-B	28.17	0.97	0.07	33.61	0.98	0.03	31.45	0.97	0.05
HexPlane	24.24	0.94	0.07	33.79	0.98	0.03	28.71	0.96	0.05
Model	**Bouncing Balls**			**Lego**			**T-Rex**		
	PSNR↑	SSIM↑	LPIPS ↑	PSNR↑	SSIM↑	LPIPS ↑	PSNR↑	SSIM↑	LPIPS ↑
T-NeRF	37.81	0.98	0.12	23.82	0.90	0.15	30.19	0.96	0.13
D-NeRF	38.93	0.98	0.10	21.64	0.83	0.16	31.75	0.97	0.03
TiNeuVox-S	39.05	0.99	0.06	24.35	0.88	0.13	29.95	0.96	0.06
TiNeuVox-B	40.73	0.99	0.04	25.02	0.92	0.07	32.70	0.98	0.03
HexPlane	39.69	0.99	0.03	25.22	0.94	0.04	30.67	0.98	0.03
Model	**Stand Up**			**Jumping Jacks**			**Average**		
	PSNR↑	SSIM↑	LPIPS ↑	PSNR↑	SSIM↑	LPIPS ↑	PSNR↑	SSIM↑	LPIPS ↑
T-NeRF	31.24	0.97	0.02	32.01	0.97	0.03	29.51	0.95	0.08
D-NeRF	32.79	0.98	0.02	32.80	0.98	0.03	30.50	0.95	0.07
TiNeuVox-S	32.89	0.98	0.03	32.33	0.97	0.04	30.75	0.96	0.07
TiNeuVox-B	35.43	0.99	0.02	34.23	0.98	0.03	32.64	0.97	0.04
HexPlane	34.36	0.98	0.02	31.65	0.97	0.04	31.04	0.97	0.04

baseline results, we don't explore new evaluation metrics.

2.2. D-NeRF Dataset [13].

We have $R_1 = R_2 = R_3 = 48$ for appearance HexPlane since it has 360° videos. For opacity HexPlane, we set $R_1 = R_2 = R_3 = 24$. The bounding box has max boundaries $[1.5, 1.5, 1.5]$ and min boundaries $[-1.5, -1.5, -1.5]$.

During training, HexPlane starts with space grid size of 32^3 and upsamples its resolution at 3k, 6k, 9k to 200^3. The emptiness voxel is calculated at 4k and 10k iterations. Total training iteration is 25k. The learning rate for feature planes are 0.02, and learning rate for $\mathbf{V}^{RF}$ and neural network is 0.001. All learning rates are exponentially decayed. We use Adam [5] for optimization with $\beta_1 = 0.9, \beta_2 = 0.99$. During evaluation, the LPIPS is computed using VGG-Net [14] following previous works. We show per-scene quantitative results in Table 2 and visualizations in Figure 3.

2.3. iPhone dataset [4].

2.4. Ablation Details.

For a fair comparison, we fix all settings in ablations. **Volume Basis** represents 4D volumes as the weighted summation of a set of shared 3D volumes as Eq 2 in main paper, where each 3D volume is represented in Eq 1 format to save memory. The 3D volume $\mathbf{V}_t$ at time t is then:

$$\begin{aligned}\mathbf{V}_t &= \sum_{i=1}^{R_t} \mathbf{f}(t)_i \cdot \hat{\mathbf{V}}_i \\ &= \sum_{i=1}^{R_t} \mathbf{f}(t)_i (\sum_{r=1}^{R_1} \mathbf{M}_{r,i}^{XY} \circ \mathbf{v}_{r,i}^{Z} \circ \mathbf{v}_{r,i}^{1} + \sum_{r=1}^{R_2} \mathbf{M}_{r,i}^{XZ} \\ &\circ \mathbf{v}_{r,i}^{Y} \circ \mathbf{v}_{r,i}^{2} + \sum_{r=1}^{R_3} \mathbf{M}_{r,i}^{YZ} \circ \mathbf{v}_{r,i}^{X} \circ \mathbf{v}_{r,i}^{3})\end{aligned} \quad (1)$$

Similarly, we use a piece-wise linear function to approximate $\mathbf{f}(t)$. In experiments, we set $R_1 = R_2 = R_3 = 16$ for appearance HexPlane and $R_1 = R_2 = R_3 = 8$ for opacity

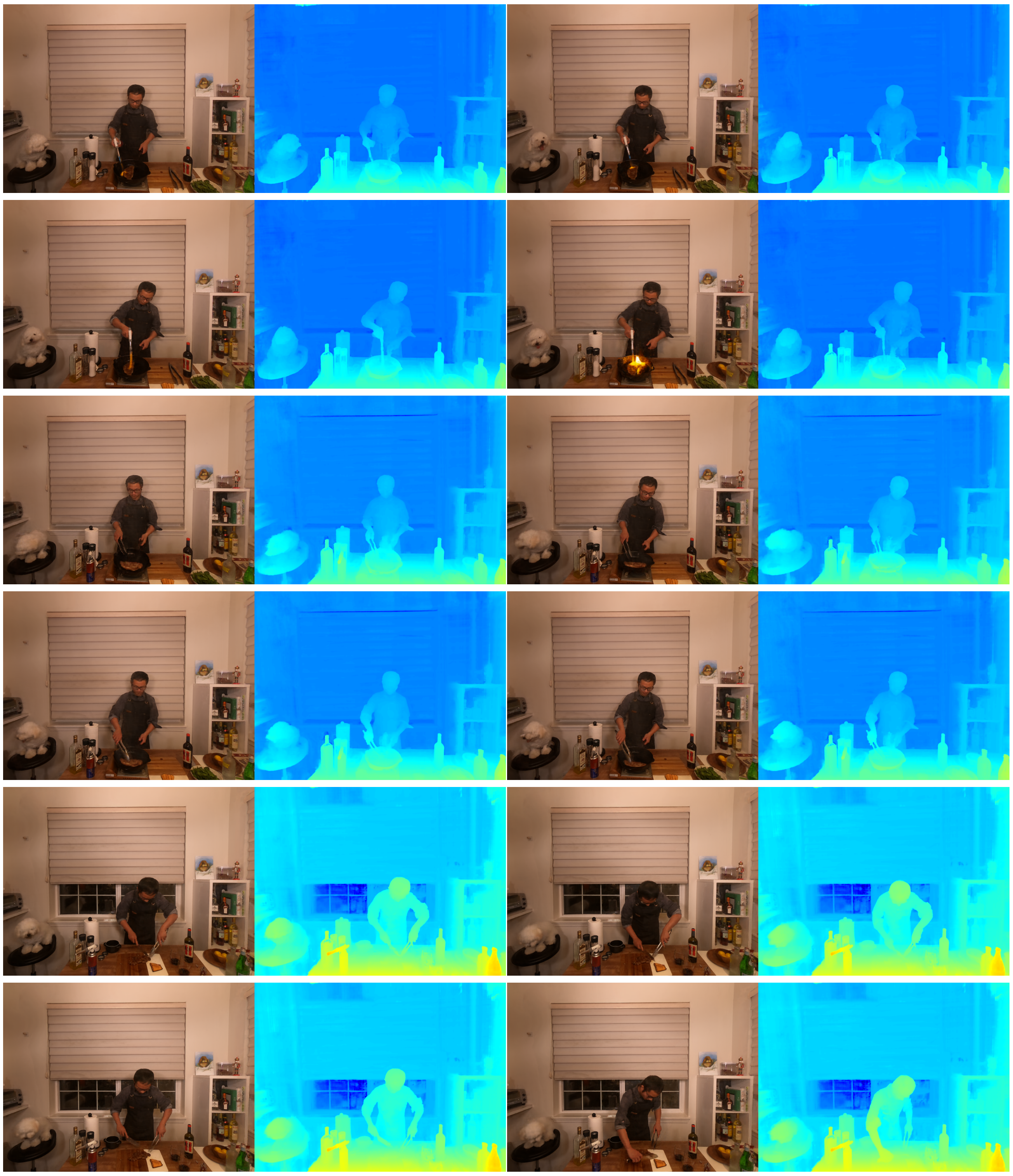


Figure 2. **View Synthesis Results and Depths at Test View on Plenoptic Video Dataset [7].**

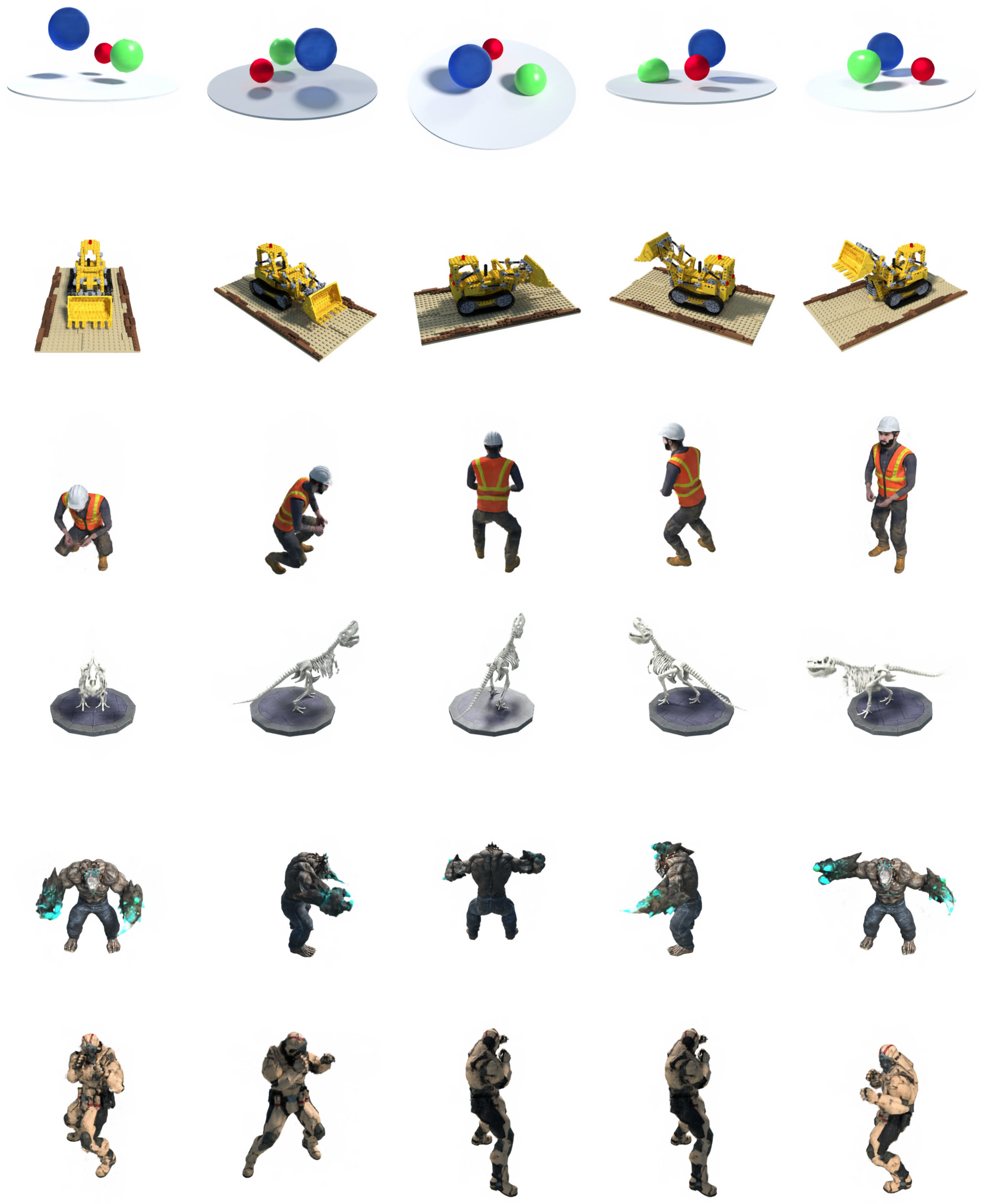


Figure 3. **View Synthesis Results on D-NeRF Dataset [13].**

HexPlane. We evaluate $R_t = 8, 12, 16$ in experiments.
VM-T (Vector, Matrix and Time) uses Eq 3 in main paper to represent 4D volumes.

$$\mathbf{V}_t = \sum_{r=1}^{R_1} \mathbf{M}_r^{XY} \circ \mathbf{v}_r^{Z} \circ \mathbf{v}_r^{1} \cdot \mathbf{f}_r^{1}(t) + \sum_{r=1}^{R_2} \mathbf{M}_r^{XZ} \circ \mathbf{v}_r^{Y} \circ \mathbf{v}_r^{2} \cdot \mathbf{f}_r^{2}(t) + \sum_{r=1}^{R_3} \mathbf{M}_r^{ZY} \circ \mathbf{v}_r^{X} \circ \mathbf{v}_r^{3} \cdot \mathbf{f}_r^{3}(t) \tag{2}$$

We evaluate $R_1 = R_2 = R_3 = 24, 48, 96$.
CP Decom. (CANDECOMP Decomposition) represents 4D volumes using a set of vectors for each axis.

$$\mathbf{V}_t = \sum_{r=1}^{R} \mathbf{v}_r^{X} \circ \mathbf{v}_r^{Y} \circ \mathbf{v}_r^{Z} \circ \mathbf{v}_r \cdot \mathbf{f}_r(t) \tag{3}$$

$\mathbf{v}^X, \mathbf{v}^Y, \mathbf{v}^Z$ are feature vectors corresponding to X, Y, Z axes. We evaluate $R = 48, 96, 192, 384$ in experiments.

2.5. Fusion Ablations

We provide complete results of fusion ablations in Table 3. For *Fusion-One* and *Fusion-Two*, we choose one fusion method from *Concat*, *Sum*, and *Multiply*, and enumerate all combinations of fusion methods. Besides that, we also explore to regress opacities from MLPs like [1]. In this setting, we sample opacity features, 8-dim feature vectors from HexPlane and regress opacity values from another MLP. Using MLP to regress opacities could substantially boost the the results for all designs, at the cost of slower rendering speeds.

Please also note that we found different fusion designs expect different weight initializations for feature planes. For *Multiply-Multiply*, *Concat-Multiply*, we initialize features with gaussian noise with mean 0.5 and scale 0.9. For the others, we initialize features with mean 0.0 and scale 0.1. To this end, different fusion designs may expect various various hyper-parameter tuning and slight adjustments to get optimal results. We encounrage readers to explore these settings and find the best settings for their own applications.

2.6. Visualization of Feature Planes.

We visualize each channel of XT, ZT feature plane for opacity HexPlane in Figure 4. This HexPlane is trained in *Flame Salmon* scene in Plenoptic Video Dataset [7].

3. Failure Cases

HexPlane doesn't always give satisfactory results. It generates degraded results when objects move too fast or there are too few observations to synthesis details. Figure 5 shows failure cases and corresponding ground-truth images.

4. Failed Designs for Dynamic Scenes

Although HexPlane is a simple and elegant solution, it is not the instant solution we had for this task. In this section, we discuss other designs we tried. These designs could model the dynamic scenes while their qualities and speeds are not comparable to HexPlane. We discuss these "failed" designs, hoping they could inspire future work.

4.1. Fixed Basis for Time Axis

In Eq 2 of main paper, we use $\mathbf{f}(t)$ as the coefficients of basis volumes at time t. Its could be further expressed as:

$$\mathbf{V_t} = \sum_{i=1}^{R_t} \mathbf{f}(t)_i \cdot \hat{\mathbf{V}}_i = \hat{\mathbf{V}} \cdot \mathbf{f}(t) \tag{4}$$

where the second $\cdot$ is matrix-vector production; $\hat{\mathbf{V}} \in \mathbb{R}^{XYZFR_t}$ is the stack of $\{\hat{\mathbf{V}}_1, \dots, \hat{\mathbf{V}}_{R_t}\}$; $\mathbf{f}(t) \in \mathbb{R}^{R_t}$ is a function of t. An interesting perspective to understand $\hat{\mathbf{V}}$ is: instead of storing static features with shape $\mathbb{R}^F$, every spatial point in 3D volume contains a feature matrix with shape $\mathbb{R}^{FR_t}$. And feature vectors at specific time t could be computed by inner product between $\mathbf{f}(t)$ and feature matrix. That is, $\mathbf{f}(t)$ is a set of basis functions w.r.t to time t and feature matrix contains coefficients of basis functions to approximate feature value changes along with time. Following the traditional approach of basis functions for time series, we use a set of sine/cosine functions as $\mathbf{f}(t)$.

While in practice, we found this implementation couldn't work since it requires enormous GPU memories. For instance, with $X = Y = Z = 128, R_t = 32, F = 27$, it uses 7GB to store such a representation and around 30GB during training because of back-propagation and keeping auxiliary terms of Adam. And it is extremely slow because of reading/writing values in memories.

Therefore, we apply tensor decomposition to reduce memory usages by factorizing volumes into matrixes $\mathbf{M}^{XY}, \mathbf{M}^{XZ}, \mathbf{M}^{YZ}$ and vectors $\mathbf{v}^X, \mathbf{v}^Y, \mathbf{v}^Z$ following Eq 1. Similarly, we add additional R_t dimension to matrixes $\mathbf{M}$ and vectors $\mathbf{v}$, leading to $\mathbf{MT}_r^{XY} \in \mathbb{R}^{XYR_T}, \mathbf{vT}_r^X \in \mathbb{R}^{XR_T}$. When calculating features from the representation, $\mathbf{f}(t)$ is first multiplied with $\mathbf{MT}_i^{XY}$ and $\mathbf{vT}_i^X$ along the last dimension to get $\mathbf{M}$ and $\mathbf{v}$ at this time steps. We then calculate features using resulted $\mathbf{v}$ and $\mathbf{M}$ following Eq 1.

$\mathbf{f}(t)$ is designed to be like positional encoding, $\mathbf{f}(t) = [1, \sin(t), \cos(t), \sin(2*t), \cos(2*t), \sin(4*t), \cos(4*t), \cdots]$. We also try to use Legendre polynomials or Chebyshev polynomials to represent $\mathbf{f}(t)$. During training, we use weighted $L1$ loss to regularize $\mathbf{MT}_r, \mathbf{vT}_r$, and assign higher weights for high-frequency coefficients to keep results smooth. We also use the smoothly bandwidth annealing trick in [11] during training, which gradually introducing high-frequency components.

Table 3. **Ablations on Feature Fusions Designs.** We show results with various fusion designs on D-NeRF dataset. HexPlane could work with other fusion mechanisms, showing its robustness.

		Opacity without MLP Regression			Opacity with MLP Regression		
Fusion-One	Fusion-Two	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Multiply	Concat	31.042	0.968	0.039	31.477	0.969	0.037
	Sum	31.023	0.967	0.039	31.318	0.969	0.038
	Multiply	30.345	0.966	0.041	31.094	0.968	0.038
Sum	Concat	25.428	0.931	0.084	29.240	0.954	0.057
	Sum	25.227	0.928	0.090	28.024	0.946	0.067
	Multiply	30.585	0.965	0.044	30.934	0.966	0.041
Concat	Concat	25.057	0.928	0.073	30.173	0.961	0.049
	Sum	24.915	0.925	0.077	27.971	0.946	0.066
	Multiply	30.299	0.965	0.041	30.874	0.971	0.036

This design could model dynamic scenes while it suffers from severe color jittering and distortions. Compared to HexPlane, it has additional matrix-vector production, which reduces overall speeds.

4.2. Frequency-Domain Methods

We also tried another method from the frequency domain, which is orthogonal to the HexPlane idea. The notations of this section are slightly inconsistent with the notations of the main paper.

According to Fourier Theory, the value at (x, y, z, t) spacetime point could be represented in its frequency domain (we ignore feature dimension here for simplicity):

$$\mathbf{D}(x,y,z,t) = \sum_{u=1}^{U}\sum_{v=1}^{V}\sum_{w=1}^{W}\sum_{k=1}^{K} \widetilde{\mathbf{D}}(u,v,w,k) \cdot e^{-j2\pi(\frac{ux}{U}+\frac{vy}{V}+\frac{wz}{W}+\frac{kt}{K})} \tag{5}$$

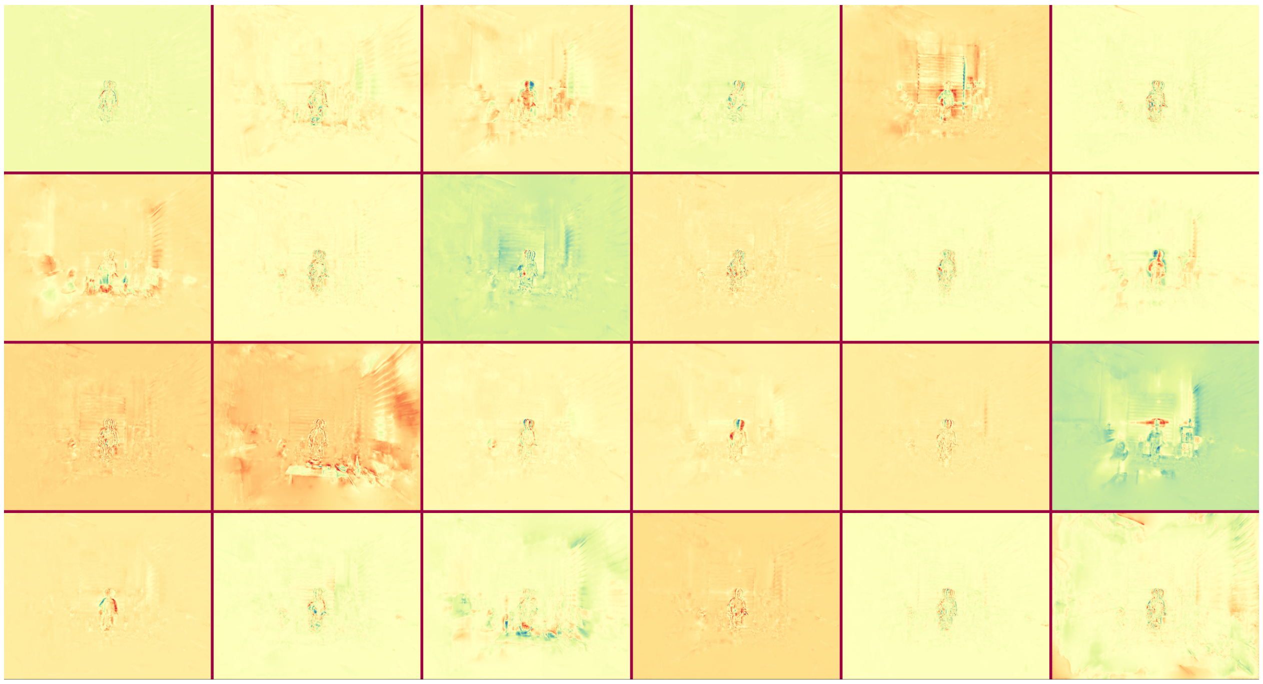
$\widetilde{\mathbf{D}}$ is another 4D volume storing frequency weights, having the same size as $\mathbf{D}$. Storing $\widetilde{\mathbf{D}}$ is memory-consuming, and similarly, we apply tensor decomposition on this volume.

$$\begin{aligned}
\mathbf{D}(x,y,z,t) &= \sum_{u=1}^{U}\sum_{v=1}^{V}\sum_{w=1}^{W}\sum_{k=1}^{K}\sum_{r=1}^{R} \widetilde{\mathbf{v}}^{U}(u)_r \cdot \widetilde{\mathbf{v}}^{V}(v)_r \cdot \\
&\widetilde{\mathbf{v}}^{W}(w)_r \cdot \widetilde{\mathbf{v}}^{K}(k)_r \cdot e^{-j2\pi(\frac{ux}{U}+\frac{vy}{V}+\frac{wz}{W}+\frac{kt}{K})} \\
&= \sum_{u=1}^{U}\sum_{v=1}^{V}\sum_{w=1}^{W}\sum_{k=1}^{K}\sum_{r=1}^{R} (\widetilde{\mathbf{v}}^{U}(u)_r \cdot e^{-j2\pi\frac{ux}{U}})(\widetilde{\mathbf{v}}^{V}(v)_r \cdot e^{-j2\pi\frac{vy}{V}}) \\
&\cdot (\widetilde{\mathbf{v}}^{W}(w)_r e^{-j2\pi\frac{wz}{W}}) \cdot (\widetilde{\mathbf{v}}^{K}(k)_r e^{-j2\pi\frac{kt}{K}})) \\
&= \sum_{r=1}^{R}(\sum_{u=1}^{U} \widetilde{\mathbf{v}}^{U}(u)_r \cdot e^{-j2\pi\frac{ux}{U}}) \cdot \sum_{v=1}^{V}(\widetilde{\mathbf{v}}^{V}(v)_r \cdot e^{-j2\pi\frac{vy}{Y}}) \cdot \\
&(\sum_{w=1}^{W} \widetilde{\mathbf{v}}^{W}(w)_r e^{-j2\pi\frac{wz}{W}}) \cdot (\sum_{k=1}^{K} \widetilde{\mathbf{v}}^{K}(k)_r e^{-j2\pi\frac{kt}{K}})
\end{aligned} \tag{6}$$

where $\widetilde{\mathbf{v}}^{U}, \widetilde{\mathbf{v}}^{V}, \widetilde{\mathbf{v}}^{W}, \widetilde{\mathbf{v}}^{K}$ are the decomposed vectors from $\widetilde{\mathbf{D}}$ along U, V, W, K axes using CANDECOMP Decomposition, which axes are related to x, y, z, t in time domain. Instead of storing the 4D frequency volume and computing values by traversing all elements inside this volume using Eq 5, we decompose 4D volumes into many single vectors and calculate values by summation along each axis, significantly reducing computations.

Similarly, we apply weight $L1$ and smoothly bandwidth annealing trick on vector weights. We also try wavelet series instead of Fourier series, and other decompositions. We found this method leads to less-saturated colors and degraded details, which is shown in videos.

Also, this method replaces grid sampling of HexPlane by inner product, which is less efficient and leads to slow speeds.


Feature Map Visualization on XY Plane

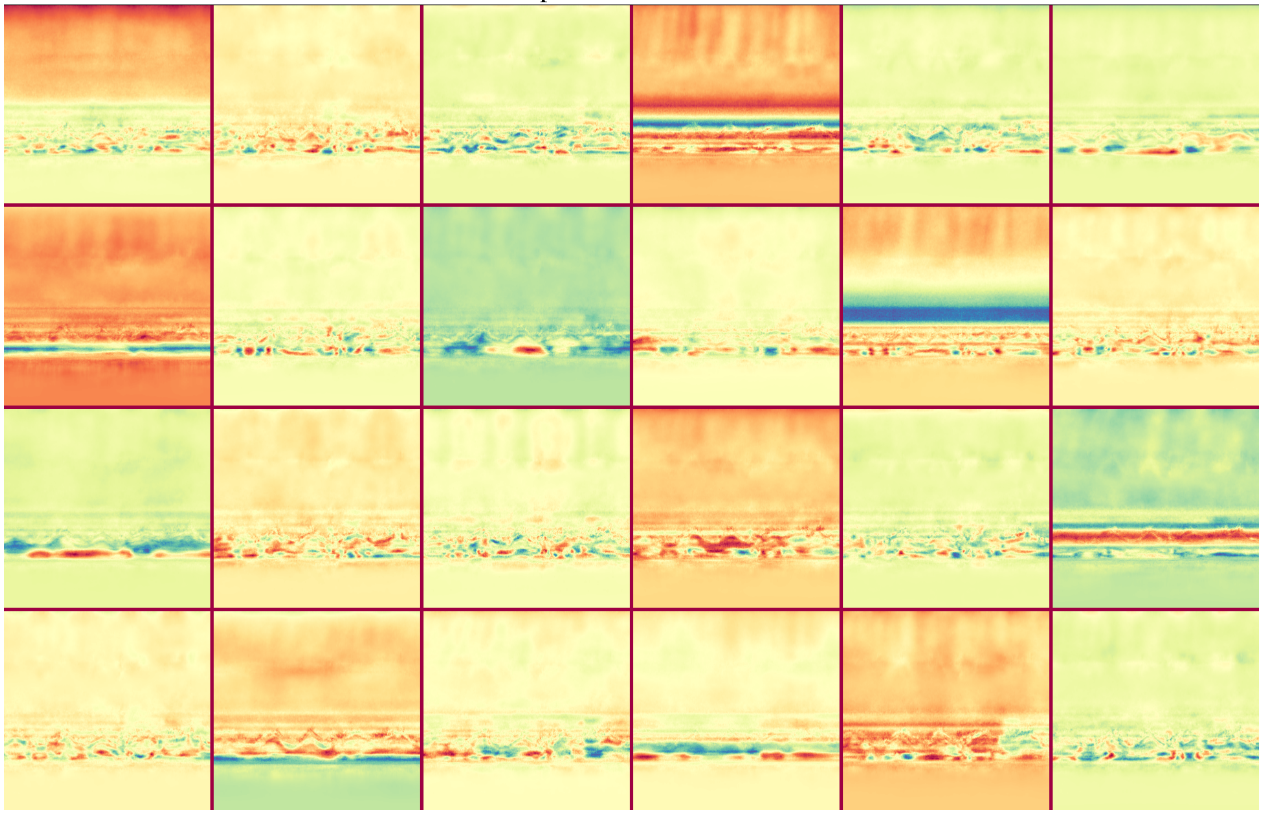

Feature Map Visualization on ZT Plane

Figure 4. **Feature Map Visualization on *Flame Salmon* Scene.**

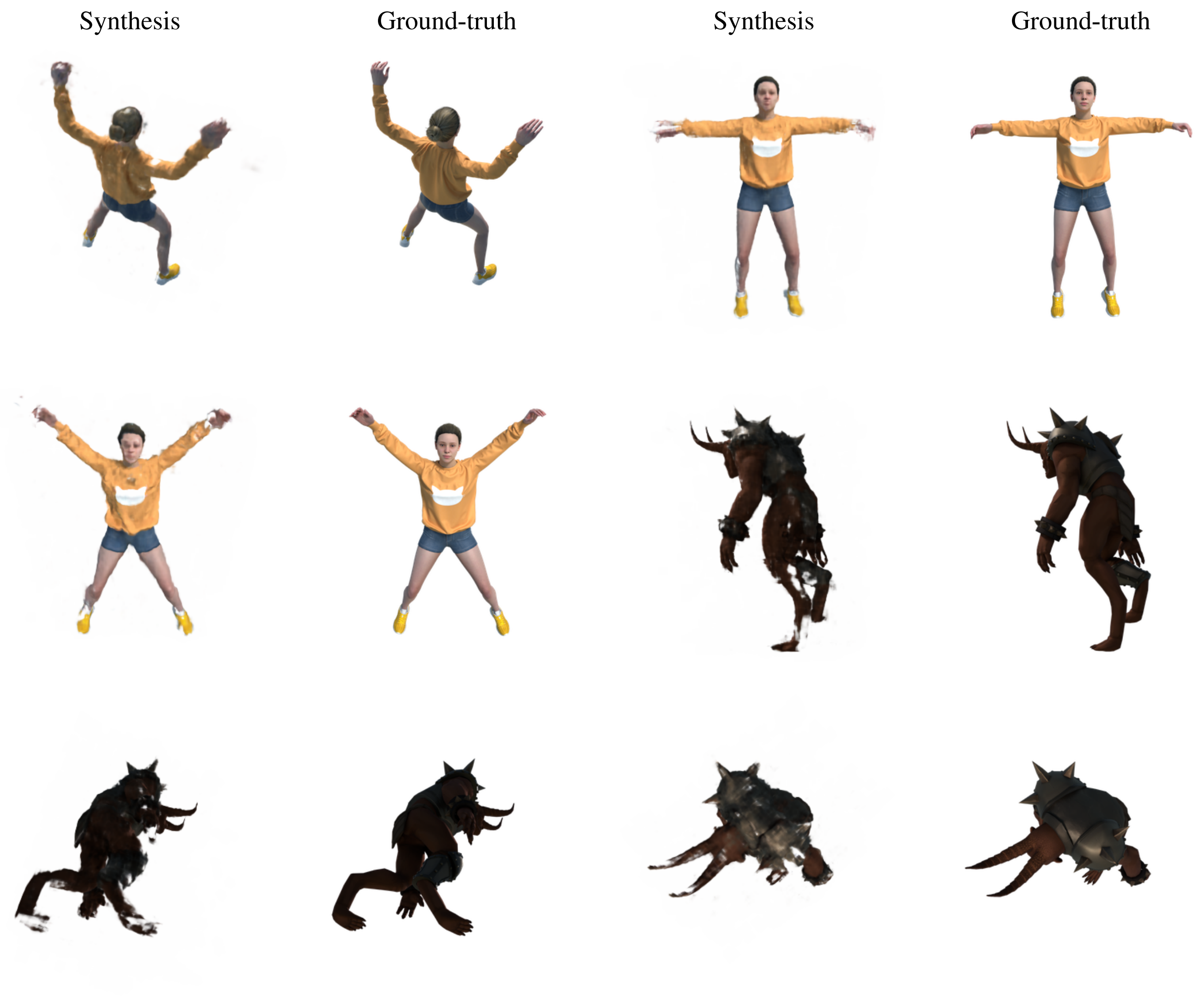


Figure 5. **Failure Cases from HexPlane.**

References

[1] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. 6

[2] Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, et al. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (TOG)*, 39(5):1–38, 2020. 1

[3] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. *arXiv preprint arXiv:2205.15285*, 2022. 1, 2

[4] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NeurIPS*, 2022. 1, 2, 3

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2, 3

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 2

[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, 2, 3, 4, 6

[8] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015. 1

[9] Rafał K Mantiuk, Gyorgy Denes, Alexandre Chapiro, Anton Kaplanyan, Gizem Rufo, Romain Bachy, Trisha Lian, and Anjul Patney. Fovvideovdp: A visible difference predictor for wide field-of-view video. *ACM Transactions on Graphics (TOG)*, 40(4):1–19, 2021. 2

[10] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

[11] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 1, 6

[12] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 1

[13] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 1, 2, 3, 5

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[15] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 1

[16] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 2