HexPlane: A Fast Representation for Dynamic Scenes

Ang Cao Justin Johnson

University of Michigan, Ann Arbor

{ancao, justincj}@umich.edu

Abstract

Modeling and re-rendering dynamic 3D scenes is a challenging task in 3D vision. Prior approaches build on NeRF and rely on implicit representations. This is slow since it requires many MLP evaluations, constraining real-world applications. We show that dynamic 3D scenes can be explicitly represented by six planes of learned features, leading to an elegant solution we call HexPlane. A HexPlane computes features for points in spacetime by fusing vectors extracted from each plane, which is highly efficient. Pairing a HexPlane with a tiny MLP to regress output colors and training via volume rendering gives impressive results for novel view synthesis on dynamic scenes, matching the image quality of prior work but reducing training time by more than $100\times$. *Extensive ablations confirm our HexPlane design and show that it is robust to different feature fusion mechanisms, coordinate systems, and decoding mechanisms. HexPlane is a simple and effective solution for representing 4D volumes, and we hope they can broadly contribute to modeling spacetime for dynamic 3D scenes.*[1]

1. Introduction

Reconstructing and re-rendering 3D scenes from a set of 2D images is a core vision problem which can enable many AR/VR applications. The last few years have seen tremendous progress in reconstructing *static* scenes, but this assumption is restrictive: the real world is *dynamic*, and in complex scenes motion is the norm, not the exception.

Many current approaches for representing dynamic 3D scenes rely on *implicit* representations, building on NeRF [42]. They train a large multi-layer perceptron (MLP) that inputs the position of a point in space and time, and outputs either the color of the point [28, 29] or a *deformation* to a canonical static scene [16, 49, 50, 54]. In either case, rendering images from novel views is expensive since each generated pixel requires many MLP evaluations. Training is similarly slow, requiring up to days of GPU time to model a single dynamic scene; this computational bottleneck prevents these methods from being widely applied.

[1] Project page: https://caoang327.github.io/HexPlane.

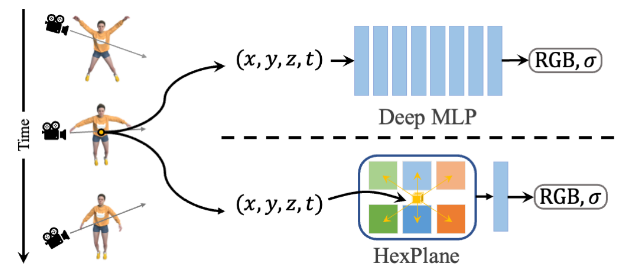

Figure 1. **HexPlane for Dynamic 3D Scenes.** Instead of regressing colors and opacities from a deep MLP, we explicitly compute features for points in spacetime via HexPlane. Pairing with a tiny MLP, it allows above $100\times$ speedups with matching quality.

Several recent methods for modeling *static* scenes have demonstrated tremendous speedups over NeRF through the use of *explicit* and *hybrid* methods [7, 43, 66, 81]. These methods use an explicit spatial data structure that stores explicit scene data [14, 81] or features that are decoded by a tiny MLP [7, 43, 66]. This decouples a model's *capacity* from its *speed*, and allows high-quality images to be rendered in realtime [43]. While effective, these methods have thus far been applied only to static scenes.

In this paper, we aim to design an explicit representation of dynamic 3D scenes, building on similar advances for static scenes. To this end, we design a *spatial-temporal* data structure that stores scene data. It must overcome two key technical challenges. First is *memory usage*. We must model all points in both space and time; naïvely storing data in a dense 4D grid would scale with the fourth power of grid resolution which is infeasible for large scenes or long durations. Second is *sparse observations*. Moving a single camera through a static scene can give views that densely cover the scene; in contrast, moving a camera through a dynamic scene gives just one view per timestep. Treating timesteps independently may give insufficient scene coverage for high-quality reconstruction, so we must instead share information across timesteps.

We overcome these challenges with our novel HexPlane architecture. Inspired by factored representations for static scenes [5, 7, 51], a HexPlane decomposes a 4D spacetime grid into six *feature planes* spanning each pair of coordinate axes (*e.g.* XY, ZT). A HexPlane computes a feature vector for a 4D point in spacetime by projecting the point onto each feature plane, then aggregating the six resulting feature vectors. The fused feature vector is then passed to a tiny MLP which predicts the color of the point; novel views can then be rendered via volume rendering [42].

Despite its simplicity, a HexPlane provides an elegant solution to the challenges identified above. Due to its factored representation, a HexPlane's memory footprint only scales quadratically with scene resolution. Furthermore, each plane's resolution can be tuned independently to account for scenes requiring variable capacity in space and time. Since some planes rely only on *spatial* coordinates (*e.g.* XY), by construction a HexPlane encourages sharing information across disjoint timesteps.

Our experiments demonstrate that HexPlane is an effective and highly efficient method for novel view synthesis in dynamic scenes. On the challenging Plenoptic Video dataset [28] we match the image quality of prior work but improve training time by $>100\times$; we also outperform prior approaches on a monocular video dataset [54]. Extensive ablations validate our HexPlane design and demonstrate that it is robust to different feature fusion mechanisms, coordinate systems (rectangular vs. spherical), and decoding mechanisms (spherical harmonics vs. MLP).

HexPlane is a simple, explicit, and general representation for dynamic scenes. It makes minimal assumptions about the underlying scene, and does not rely on deformation fields or category-specific priors. Besides improving and accelerating view synthesis, we hope HexPlane will be useful for a broad range of research in dynamic scenes [61].

2. Related Work

Neural Scene Representations. Using neural networks to implicitly represent 3D scenes [39, 46, 62, 63, 67, 75] has achieved exciting progress recently. NeRF [42] and its variants [2, 3, 40, 44, 69, 71, 80, 87] show impressive results on novel view synthesis [9, 75, 82, 94] and many other applications including 3D reconstruction [38, 67, 85, 89, 95], semantic segmentation [25, 55, 93], generative model [5, 6, 10, 45, 58, 77], and 3D content creation [1, 22, 30, 48, 53, 72, 86].

Implicit neural representations exhibit remarkable rendering quality, but they suffer from slow rendering speeds due to the numerous costly MLP evaluations required for each pixel. To address this challenge, many recent papers propose *hybrid* representations that combine a fast explicit scene representation with learnable neural network components, providing significant speedups over purely implicit methods. Various explicit representations have been investigated, including sparse voxels [14, 34, 59, 66], low-rank components [5, 7, 31, 51], point clouds [4, 21, 79, 92, 96] and others [8, 36, 43, 68, 90]. However, these approaches assume static 3D scenes, leaving explicit representations for dynamic scenes unexplored. This paper provides an explicit model for dynamic scenes, substantially accelerating prior methods that rely on fully implicit methods.

Neural Rendering for Dynamic Scenes. Representing dynamic scenes by neural radiance fields is an essential extension of NeRF, enabling numerous real-world applications [27, 47, 52, 65, 78, 84, 91]. One line of research represents dynamic scenes by extending NeRF with an additional time dimension (T-NeRF) or additional latent code [16, 28, 29, 76]. Despite the ability to represent general typology changes, they suffer from a severely under-constrained problem, requiring additional supervision like depths, optical flows or dense observations for decent results. Another line of research employs individual MLPs to represent a deformation field and a canonical field [11, 49, 50, 54, 70, 83], where the canonical field depicts a static scene, and the deformation field learns coordinate maps to the canonical space over time. We propose a simple yet elegant solution for dynamic scene representation using six feature planes, making minimal assumptions about the underlying scene.

Recently, *MAV3D* [61] adopted our design for text-to-4D dynamic scene generation, demonstrating an exciting direction for dynamic scenes beyond reconstruction.

Accelerating NeRFs. Many works have been proposed to accelerate NeRF at diverse stages. Some methods improve *inference* speeds of trained NeRFs by optimizing the computation [18, 20, 56, 81]. Others reduce the *training* times by learning a generalizable model [9, 24, 74, 75]. Recently, rendering speeds during *both stages* are substantially reduced by using explicit-implicit representations [5, 7, 14, 33, 43, 66]. In line with this idea, we propose an explicit representation for dynamic fields to accelerate dynamic NeRFs.

Very recently, several concurrent works have aimed to accelerate dynamic NeRFs. [12, 15, 19, 32, 73] use *time-aware* MLPs to regress spacetime points' colors or deformations from canonical spaces. However, they remain partially implicit for dynamic fields, as they rely on MLPs with time input to obtain spacetime features. In contrast, our paper proposes a more elegant and efficient explicit representation for dynamic fields without using time-aware MLPs. Like [26], *NeRFPlayer* [64] uses a highly compact 3D grid at each time step for 4D field representation , which results in substantial memory costs for lengthy videos.

Tensor4D [60] shares a similar idea as ours, which represents dynamic scenes with 9 planes and multiple MLPs. *D-TensoRF* [23] regards dynamic fields as 5D tensors and applies CP/MM decomposition on them for compact representation. Our paper is most closely related to *K-Planes* [13], which also employs six feature planes for representation.

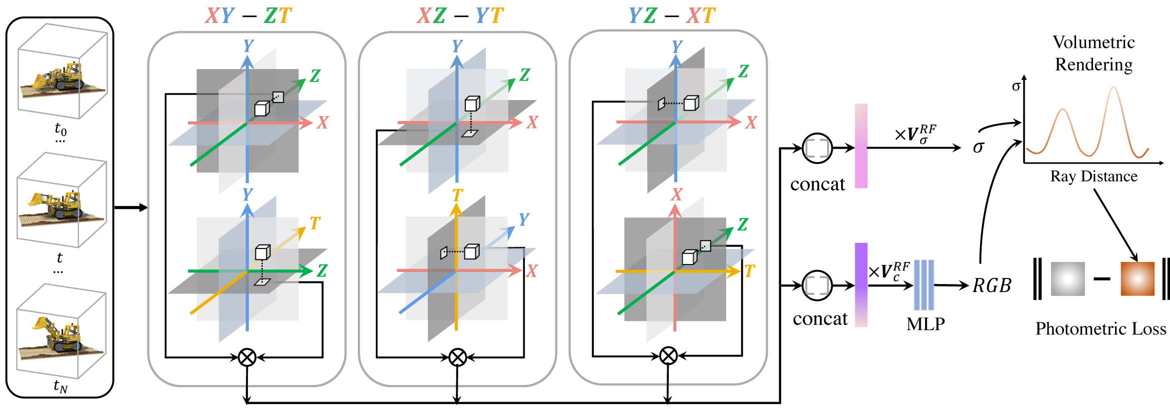

Figure 2. **Method Overview.** HexPlane contains six feature planes spanning each pair of coordinate axes (e.g. XY , ZT). To compute features of points in spacetime, it multiplies feature vectors extracted from paired planes and concatenated multiplied results into a single vector, which are then multiplied by $\mathbf{V}^{RF}$ for final results. RGB colors are regressed from point features using a tiny MLP and images are synthesized via volumetric rendering. HexPlane and the MLP are trained by photometric loss between rendered and target images.

3. Method

Given a set of posed and timestamped images of a dynamic scene, we aim to fit a model to the scene that allows rendering new images at novel poses and times. Like NeRF [42], a model gives color and opacity for points in spacetime; images are rendered via differentiable volumetric rendering along rays. The model is trained using photometric loss between rendered and ground-truth images.

Our main contribution is a new *explicit* representation for dynamic 3D scenes, which we combine with a small *implicit* MLP to achieve novel view synthesis in dynamic scenes. An input spacetime point is used to efficiently query the explicit representation for a feature vector. A tiny MLP receives the feature along with the point coordinates and view direction and regresses an output RGB color for the point. Figure 2 shows an overview of the model.

Designing an explicit representation for dynamic 3D scenes is challenging. Unlike static 3D scenes which are often modeled by point clouds, voxels, or meshes, explicit representations for dynamic scenes have been underexplored. We show how the key technical challenges of *memory usage* and *sparse observations* can be overcome by our simple HexPlane representation.

3.1. 4D Volumes for Dynamic 3D Scenes

A dynamic 3D scene could be naïvely represented as a 4D volume $\mathbf{D}$ comprising independent static 3D volumes per time step $\{\mathbf{V_1}, \mathbf{V_2}, \cdots, \mathbf{V_T}\}$. However this design suffers from two key problems. First is *memory consumption*: a naïve 4D volume is very memory intensive, requiring $O(N^3TF)$ space where N, T, and F are the spatial resolution, temporal resolution, and feature size. Storing a volume of RGB colors (F=3) with N=512, T=32 in `float32` format takes 48GB of memory.

The second problem is *sparse observations*. A single camera moving through a static scene can capture dozens or hundreds of images. In dynamic scenes capturing multiple images per timestep requires multiple cameras, so we typically have only a few views per timestep; these sparse views are insufficient for independently modeling each timestep, so we must share information between timesteps.

We reduce *memory consumption* using *factorization* [5, 7] which has been previously applied to 3D volumes. We build on TensoRF [7] which decomposes a 3D volume $\mathbf{V} \in \mathbb{R}^{XYZF}$ [1] as a sum of vector-matrix outer products:

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

where $\circ$ is outer product; $\mathbf{M}_r^{XY} \circ \mathbf{v}_r^{Z} \circ \mathbf{v}_r^{1}$ is a low-rank component of $\mathbf{V}$; $\mathbf{M}_r^{XY} \in \mathbb{R}^{XY}$ is a matrix spanning the X and Y axes, and $\mathbf{v}^{Z} \in \mathbb{R}^{Z}, \mathbf{v}_r^{i} \in \mathbb{R}^{F}$ are vectors along the Z and F axes. R_1, R_2, R_3 are the number of low-rank components. With $R = R_1 + R_2 + R_3 \ll N$, this design reduces memory usage from $O(N^3TF)$ to $O(RN^2T)$.

3.2. Linear Basis for 4D Volume

Factorization helps reduce memory usage, but factoring an independent 3D volume per timestep still suffers from sparse observations and does not share information across time. To solve this problem, we can represent the 3D volume $\mathbf{V_t}$ at time t as the weighted sum of a set of shared 3D basis volumes $\{\hat{\mathbf{V}}_1, \ldots, \hat{\mathbf{V}}_{R_t}\}$; then

$$\mathbf{V_t} = \sum_{i=1}^{R_t} \mathbf{f}(t)_i \cdot \hat{\mathbf{V}}_i \tag{2}$$

[1] To simplify notation, we write $\mathbb{R}^{X\times Y}$ as $\mathbb{R}^{XY}$ in this paper.

where $\cdot$ is a scalar-volume product, R_t is the number of shared volumes, and $\mathbf{f}(t) \in \mathbb{R}^{R_t}$ gives weights for the shared volumes as a function of t. Shared volumes allow information to be shared across time. In practice each $\hat{\mathbf{V}}_i$ is represented as a TensoRF as in Equation 1 to save memory.

Unfortunately, we found in practice (and will show with experiments) that shared volumes are still too costly; we can only use small values for R_t without exhausting GPU memory. Since each shared volume is a TensoRF, it has its own independent $\mathbf{M}_r^{XY}, \mathbf{v}_r^{Z}$, *etc.*; we can further improve efficiency by sharing these low-rank components across all shared volumes. The 3D volume $\mathbf{V}_t$ at time t is then

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

where each $\mathbf{f}^i(t) \in \mathbb{R}^{R_i}$ gives a vector of weights for the low-rank components at each time t.

In this formulation, $\mathbf{f}^i(t)$ captures the model's dependence on time. The correct mathematical form for $\mathbf{f}^i(t)$ is not obvious. We initially framed $\mathbf{f}^i(t)$ as a learned combination of sinusoidal or other fixed basis functions, with the hope that this could make periodic motion easier to learn; however we found this inflexible and hard to optimize. $\mathbf{f}^i(t)$ could be an arbitrary nonlinear mapping, represented as an MLP; however this would be slow. As a pragmatic tradeoff between flexibility and speed, we represent $\mathbf{f}^i(t)$ as a learned piecewise linear function, implemented by linearly interpolating along the first axis of a learned $T \times R_i$ matrix.

3.3. HexPlane Representation

Equation 3 fully decouples the spatial and temporal modeling of the scene: $\mathbf{f}^i(t)$ models time and other terms model space. However in real scenes space and time are entangled; for example a particle moving in a circle is difficult to model under Equation 3 since its x and y positions are best modeled *separately* as functions of t. This motivates us to replace $\mathbf{v}_r^Z \cdot \mathbf{f}^1(t)_r$ in Equation 3 with a joint function of t and z, similarly represented as a piecewise linear function; this can be implemented by bilinear interpolation into a learned tensor of shape $Z \times T \times R_1$. Applying the same transform to all similar terms then gives our HexPlane representation, which represents a 4D feature volume $V \in \mathbb{R}^{XYZTF}$ as:

$$\mathbf{D} = \sum_{r=1}^{R_1} \mathbf{M}_r^{XY} \circ \mathbf{M}_r^{ZT} \circ \mathbf{v}_r^{1} + \sum_{r=1}^{R_2} \mathbf{M}_r^{XZ} \circ \mathbf{M}_r^{YT} \circ \mathbf{v}_r^{2} + \sum_{r=1}^{R_3} \mathbf{M}_r^{YZ} \circ \mathbf{M}_r^{XT} \circ \mathbf{v}_r^{3} \tag{4}$$

where each $\mathbf{M}_r^{AB} \in \mathbb{R}^{AB}$ is a learned plane of features. This formulation displays a beautiful symmetry, and strikes a balance between representational power and speed.

We can alternatively express a HexPlane as a function $\mathbf{D}$ which maps a point (x, y, z, t) to an F-dimensional feature:

$$\begin{aligned} D(x,y,z,t) = &(\mathbf{P}_{xy\bullet}^{XYR_1} \odot \mathbf{P}_{zt\bullet}^{ZTR_1})\mathbf{V}^{R_1F} \\ &+(\mathbf{P}_{xz\bullet}^{XZR_2} \odot \mathbf{P}_{yt\bullet}^{YTR_2})\mathbf{V}^{R_2F} \\ &+(\mathbf{P}_{yz\bullet}^{YZR_3} \odot \mathbf{P}_{xt\bullet}^{XTR_3})\mathbf{V}^{R_3F} \end{aligned} \tag{5}$$

where $\odot$ is an elementwise product; the superscript of each bold tensor represents its shape, and $\bullet$ in a subscript represents a slice so each term is a vector-matrix product. $\mathbf{P}^{XYR_1}$ stacks all $\mathbf{M}_r^{XY}$ to a 3D tensor, and $\mathbf{V}^{R_1F}$ stacks all $\mathbf{v}_r^1$ to a 2D tensor; other terms are defined similarly. Coordinates x, y, z, t are real-valued, so subscripts denote bilinear interpolation. This design reduces memory usage to $O(N^2R + NTR + RF)$.

We can stack all $\mathbf{V}^{R_iF}$ into $\mathbf{V}^{RF}$ and rewrite Eq 5 as

$$[\mathbf{P}_{xy\bullet}^{XYR_1} \odot \mathbf{P}_{zt\bullet}^{ZTR_1}; \mathbf{P}_{xz\bullet}^{XZR_2} \odot \mathbf{P}_{yt\bullet}^{YTR_2}; \mathbf{P}_{yz\bullet}^{YZR_3} \odot \mathbf{P}_{xt\bullet}^{XTR_3}]\mathbf{V}^{RF} \tag{6}$$

where ; concatenates vectors. As shown in Figure 2, a HexPlane comprises three pairs of feature planes; each pair has a spatial and a spatio-temporal plane with orthogonal axes (*e.g.* XY/ZT). Querying a HexPlane is fast, requiring just six bilinear interpolations and a vector-matrix product.

3.4. Optimization

We represent dynamic 3D scenes using the proposed HexPlane, which is optimized by photometric loss between rendered and target images. For point (x, y, z, t), its opacity and appearance feature are quired from HexPlane, and the final RGB color is regressed from a tiny MLP with appearance feature and view direction as inputs. With points' opacities and colors, images are rendered via volumetric rendering. The optimization objective is:

$$\mathcal{L} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} \|C(\mathbf{r}) - \hat{C}(\mathbf{r})\|_2^2 + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}} \tag{7}$$

$\mathcal{L}_{\text{reg}}$, λ_{reg} are regularization and its weight; $\mathcal{R}$ is the set of rays and $C(\mathbf{r}), \hat{C}(\mathbf{r})$ are rendered and GT colors of ray $\mathbf{r}$.

Color Regression. To save computations, we query points' opacities directly from one HexPlane, and query appearance features of points with high opacities from another separate HexPlane. Queried features and view directions are fed into a tiny MLP for RGB colors. An MLP-free design is also feasible with spherical harmonics coefficients as features.

Regularizer. Dynamic 3D reconstruction is a severely ill-posed problem, needing strong regularizers. We apply Total Variational (TV) loss on planes to force the spatial-temporal continuity, and depth smooth loss in [44] to reduce artifacts.

Coarse to Fine Training. A coarse-to-fine scheme is also employed like [7, 81], where the resolution of grids gradually grows during training. This design accelerates the training and provides an implicit regularization on nearby grids.

Figure 3. **High-Quality Dynamic Novel View Synthesis on Plenoptic Video dataset [28].** The proposed HexPlane could effectively represent dynamic 3D scenes with complicated motions and render high-quality results with faithful details at various timesteps and unseen viewpoints. We show several samples of input video sequences and synthesis results using a cyclic camera trajectory.

Emptiness Voxel. We keep a tiny 3D voxel indicating the emptiness of scene regions and skip points in empty regions. Since many regions are empty, it is helpful for acceleration. To get this voxel, we evaluate points' opacities across time steps and reduce them to a single voxel with maximum opacities. Although keeping several voxels for various time intervals improves speeds, we only keep one for simplicity.

4. Experiments

We evaluate HexPlane, our proposed explicit representation, on dynamic novel view synthesis tasks with challenging datasets, comparing its performance and speed to state-of-the-art methods. Through extensive ablation studies, we explore its advantages and demonstrate its robustness to different feature fusion mechanisms, coordinate systems, and decoding mechanisms. As our objective is to demonstrate the effectiveness of this simple design, we prioritize HexPlane's simplicity and generality without implementing intricate tricks for performance enhancement.

4.1. Dynamic Novel View Synthesis Results

For a comprehensive evaluation, we use two datasets with distinct settings: the high-resolution, multi-camera *Plenoptic Video dataset* [28], with challenging dynamic content and intricate visual details; the monocular *D-NeRF dataset* [54], featuring synthetic objects. *Plenoptic Video dataset* assesses HexPlane's representational capacity for long videos with complex motions and fine details, while *D-NeRF dataset* tests its ability to handle monocular videos and extremely sparse observations (with teleporting [17]).

Plenoptic Video dataset [28] is a real-world dataset captured by a multi-view camera system using 21 GoPro at 2028 $\times$ 2704 (2.7K) resolution and 30 FPS. Each scene comprises 19 synchronized, 10-second videos, with 18 designated for training and one for evaluation. This dataset is suitable to test the representation ability as it features complex and challenging dynamic content such as highly specular, translucent, and transparent objects; topology changes; moving self-casting shadows; fire flames and strong view-dependent effects for moving objects; and so on.

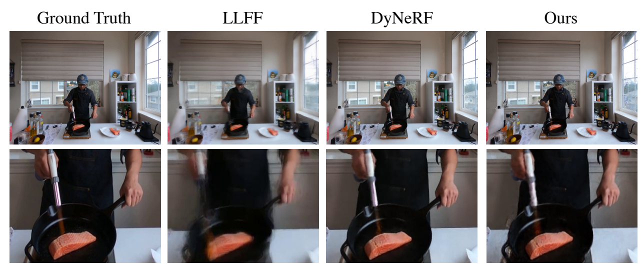

Figure 4. **Visual Comparison of Synthesis Results.** Since DyNeRF [28] model is not publicly available, we compare our results to images provided the paper. With visually similar results, our proposed HexPlane is over 100 $\times$ faster than DyNeRF.

For a fair comparison, we adhere to the same training and evaluation pipelines as DyNeRF [28] with slight changes due to GPU resources. [28] trains its model on 8 V100 GPUs for a week, with 24576 batch size for 650K iterations. We train our model on a single 16GB V100 GPU, with a 4096 batch size and the same iteration numbers, which is 6$\times$ fewer sampling. We follow the same importance sampling design and hierarch training as [28], with 512 spatial grid sizes and 300 time grid sizes. The scene is in NDC [42].

As shown in Figure 3, HexPlane delivers high-quality dynamic novel view synthesis across various times and viewpoints. It accurately models real-world scenes with intricate motions and challenging visual features, such as flames, showcasing its robust representational capabilities.

Quantitative comparisons with SOTA methods are in Table 1, with baseline results from [28] paper. PSNR, struc-

Table 1. **Quantitative Comparisons on Plenoptic Video dataset [28].** We report synthesis quality, training times (measured in GPU hours) with speedups relative to DyNeRF [28]. With 672× speedups, HexPlane†with fewer training iterations has comparable quantitative results to DyNeRF. And HexPlane trained with the same iterations noticeably outperforms DyNeRF. Baseline methods are evaluated on a particular scene, and we also report average results on all public scenes (*-all*). Best and Second results are in highlight.

Model	Steps	PSNR↑	D-SSIM↓	LPIPS↓	JOD ↑	Training Time↓	Speeds-up ↑
Neural Volumes [35]	-	22.800	0.062	0.295	6.50	-	-
LLFF [41]	-	23.239	0.076	0.235	6.48	-	-
NeRF-T [28]	-	28.449	0.023	0.100	7.73	-	-
DyNeRF [28]	650k	29.581	0.020	0.099	8.07	1344h	1×
HexPlane	650k	29.470	0.018	0.078	8.16	12h	112×
HexPlane†	100k	29.263	0.020	0.097	8.14	2h	672 ×
HexPlane-all	650k	31.705	0.014	0.075	8.47	12h	112 ×
HexPlane†-all	100k	31.569	0.016	0.089	8.36	2h	672 ×

Table 2. **Quantitative Results on D-NeRF dataset [54].** Without deformation, HexPlane has comparable or better results compared to other deformation-based methods, and is noticeably faster.

Model	Deform.	PSNR↑	SSIM ↑	LPIPS ↓	Training Time↓
T-NeRF [54]		29.51	0.95	0.08	-
D-NeRF [54]	✓	30.50	0.95	0.07	20 hours
TiNeuVox-S [12]	✓	30.75	0.96	0.07	12m 10s
TiNeuVox-B [12]	✓	**32.67**	**0.97**	**0.04**	49m 46s
HexPlane (ours)		31.04	**0.97**	**0.04**	**11m 30s**

tural dissimilarity index measure (DSSIM) [57], perceptual quality measure LPIPS [88] and video visual difference measure Just-Objectionable-Difference (JOD) [37] are evaluated for comprehensive study. Besides results on the "flame salmon" scene like [28], we also report average results on all public scenes except the unsynchronized one, referred to *HexPlane-all*. We also train a model with fewer training iterations as *HexPlane†*.

As shown in Table 1, *HexPlane†* achieves comparable performance to DyNeRF [28] while substantially faster (672× speedups), highlighting the benefits of employing explicit representations. DyNeRF uses a giant MLP and per-frame latent codes to represent dynamic scenes, which is slow due to tremendous MLP evaluations. When trained with the same iteration number, *HexPlane* outperforms DyNeRF in all metrics except PSNR, while being above 100 × faster. Although explicit representations typically demand significant memory for their rapid processing speeds due to explicit feature storage, HexPlane occupies a mere 200MB for the entire model. This relatively compact size is suitable for most GPUs. Given its fast speed, we believe this tradeoff presents an attractive option.

Since the model of DyNeRF is not publicly available, it is hard to compare the visual results directly. We download images from the original paper and find the most matching images in our results, which are compared in Figure 4.

D-NeRF dataset [54] is a monocular video dataset with 360° observations for synthetic objects. Dynamic 3D reconstruction for monocular video is challenging since only one observation is available each time. Current SOTA methods for monocular video usually have a deformation field and a static canonical field, where points in dynamic scenes are mapped to positions in the canonical field. The mapping (deformation) is represented by another MLP.

The underlying assumption of deformation field design is that there are no topology changes, which does not always hold in the real world while holding in this dataset. Again, to keep HexPlane general enough, we do not assume deformation, the same as T-NeRF [54]. We use this dataset to validate the ability to work with monocular videos.

We show quantitative results in Table 2. For fairness, all training times are re-measured on the same 2080TI GPU. Our HexPlane distinctly outperforms other methods even without introducing the deformation field, demonstrating the inherent ability to deal with sparse observations due to the shared basis. Again, our method is hundreds of times faster than MLP-based designs like D-NeRF and T-NeRF.

Tineuvox [12] is a recent work for accelerating D-NeRF, replacing canonical space MLP with a highly-optimized sparse voxel Cuda kernel and keeping an MLP to represent deformation. Therefore, it still uses explicit representation for static scenes while our target is dynamic scenes. Without any custom Cuda kernels, our method is faster and better than its light version and achieves the same LPIPS and SSIM as its bigger version, which takes longer time to train.

4.2. Ablations and Analysis

We run deep introspections to HexPlane by answering questions with extensive ablations. Ablations are conducted mainly on D-NeRF [54] dataset because of efficiency.

How does HexPlane compare to others? We compare HexPlane with other designs mentioned in the *Method Section* in Table 3, where each method has various basis numbers R: (1). *Volume Basis* represents 4D volumes as weighted summation of a set of shared 3D volumes as Eq 2, which 3D volume is represented as Eq 1; (2). *VM-T* (vector, matrix and time) uses Eq 3 representing 4D volumes;

Table 3. **Quantitative Results for Different Factorizations.** Various factorization designs are evaluated on D-NeRF dataset with different R (basis number). HexPlane achieves the best quality and speed among all methods.

Model	R	PSNR↑	SSIM↑	LPIPS↓	Training Time↓
Volume Basis	8	30.460	0.965	0.045	**18m 04s**
	12	30.587	0.966	0.043	24m 06s
	16	**30.631**	**0.967**	**0.042**	29m 20s
VM-T	24	30.329	0.962	0.051	**14m 36s**
	48	30.657	0.965	0.048	15m 58s
	96	**30.744**	**0.966**	**0.045**	17m 03s
CP Decom.	48	28.370	0.942	0.083	**10m 31s**
	96	29.371	0.951	0.070	11m 03s
	192	30.086	0.957	0.063	11m 33s
	384	**30.302**	**0.959**	**0.059**	13m 06s
HexPlane	24	30.886	0.966	0.042	**10m 27s**
	48	**31.042**	**0.968**	**0.039**	11m 30s

Table 4. **Ablations on Feature Planes Designs.** We remove and swap HexPlane's planes and show results on D-NeRF dataset.

Model	PSNR↑	SSIM↑	LPIPS↓	Training Time↓
Spatial Planes	20.369	0.879	0.148	9m 02s
Spatial-Temporal Planes	21.112	0.879	0.148	9m 29s
DoublePlane (XY-ZT)	30.370	0.961	0.054	8m 04s
HexPlane-Swap	28.562	0.954	0.056	11m 44s
HexPlane	31.042	0.968	0.039	11m 30s

(3). *CP Decom.* (CANDECOMP Decomposition) follows [7], which represents 4D volumes using a set of vectors for each axis. Implementation details are shown in Supp.

HexPlane gives optimal performance among all methods, illustrating the advantages of spatial-temporal planes. Compared to other methods, spatial-temporal planes allow HexPlane to model motions effectively with a small basis number R, leading to higher efficiency as well. Increasing R used for representation leads to better results while also resulting in more computations. We also notice that an unsuitable large R may lead to the overfitting problem, which instead harms synthesis quality on novel views.

Could variants of HexPlane work? HexPlane has excellent symmetry as it contains all pairs of coordinate axes. By breaking this symmetry, we evaluate other variants in Table 4. *Spatial Planes* only have three spatial planes: $\mathbf{P}^{XY}, \mathbf{P}^{XZ}, \mathbf{P}^{YZ}$, and *Spatial-Temporal Planes* contain the left three spatial-temporal planes; *DoublePlane* contains only one group of paired planes, i.e. $\mathbf{P}^{XY}, \mathbf{P}^{ZT}$; *HexPlane-Swap* groups planes with repeated axes like $\mathbf{P}^{XY}, \mathbf{P}^{XT}$. We report their performance and speeds.

As shown in the table, neither *Spatial Planes* nor *Spatial-Temporal Planes* could represent dynamic scenes alone, indicating both are essential for representations. *HexPlane-Swap* achieves inferior results since its axes are not complementary, losing features from the particular axis. *DoublePlane* is less effective than HexPlane since HexPlane contains more comprehensive spatial-temporal modes.

Table 5. **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.

Fusion-One	Fusion-Two	PSNR↑	SSIM↑	LPIPS↓
Multiply	Concat	31.042	0.968	0.039
	Sum	31.023	0.967	0.039
	Multiply	30.345	0.966	0.041
Sum	Concat	25.428	0.931	0.084
	Sum	25.227	0.928	0.090
	Multiply	30.585	0.965	0.044

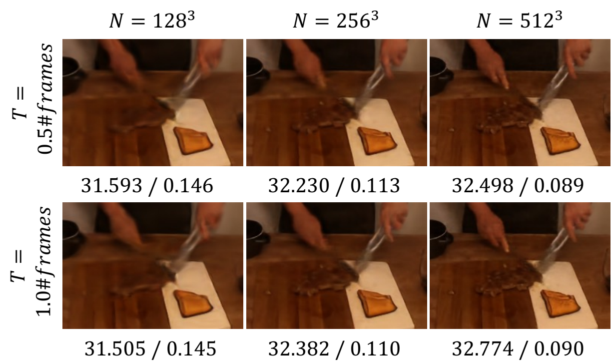

Figure 5. **Synthesis Results with Different Spacetime Grid Resolutions.** We show zoomed in synthesis results on Plenoptic Video dataset with space grid resolution ranging from 128^3 to 512^3 and time grid ranging from half to one of the video frame number. PSNR and LPIPS of the scene are reported below each images.

How does grid resolution affect results? We show qualitative results with various spacetime grid resolutions in Figure 5 and report its PSNR/LPIPS below zoomed-in images. Besides the space grid ranging from 128^3 to 512^3, we compare results with different time grid resolutions, ranging from half to the same as video frames. Higher resolutions of the space grid lead to better synthesis quality, shown by both images and metrics. HexPlane results are not noticeably affected by a smaller time grid resolution.

4.3. Robustness of HexPlane Designs

In addition to its performance and efficiency, this section demonstrates HexPlane's robustness to diverse design choices, resulting in a highly adaptable and versatile framework. This flexibility allows for its applications across a wide range of tasks and research directions.

Various Feature Fusion Mechanisms. In HexPlane, feature vectors from each plane are extracted and subsequently fused into a single vector, which are multiplied by matrix $\mathbf{V}^{RF}$ later for final results. During fusion, features from paired planes are first element-wise multiplied (*fusion one*) and then concatenated into a single one (*fusion two*). We explore other fusion designs beyond this *Multiply-Concat*.

Table 5 shows that *Multiply-Concat* is not the sole viable design. *Sum-Multiply* and swapped counterpart *Multiply-*

Table 6. **Dynamic View Synthesis without MLPs.** HexPlane-SH is a pure explicit model without MLPs on D-NeRF dataset, which stores spherical harmonics (SH) as appearance features and directly regress RGB from it rather than MLPs. HexPlane-SH gives reasonable results and faster than HexPlane with MLP.

Model	PSNR↑	SSIM↑	LPIPS↓	Training Time↓
HexPlane	31.042	0.968	0.039	11m 30s
HexPlane-SH	29.284	0.952	0.056	10m 42s

NDC Spherical NDC Spherical

Figure 6. **Synthesis Results at Extreme Views for NDC and Spherical Coordinates.** Scenes represented in NDC are assumed to be bounded along x, y axes, whose boundaries are observable at extreme views(top-left and top-right corners), leading to incorrect geometries and artifacts. Using spherical coordinate, our HexPlane could seamlessly represent dynamic unbounded scenes.

Sum both yield good results, albeit not optimal,[2] highlighting an intriguing symmetry between multiplication and addition. *Multiply-Multiply* also produces satisfactory outcomes, while *Sum-Sum* or *Sum-Concat* fail, illustrating the capacity limitations of addition compared to multiplication. Overall, HexPlane is remarkably robust to various fusion designs. We show complete results and analysis in Supp.

Spherical Harmonics Color Decoding. Instead of regressing colors from MLPs, we evaluate a pure explicit model in Table 6 without MLPs. Spherical harmonics (SH) [81] coefficients are computed directly from HexPlanes, and decoded to RGBs with view directions. Using SH allows faster rendering speeds at a slightly reduced quality. We find that optimizing SH for dynamic scenes is more challenging compared to [7, 14], which is an interesting future direction.

Spherical Coordinate for Unbounded Scenes. HexPlane is limited to bounded scenes because grid sampling fails for out-of-boundary points, which is a common issue among explicit representations. Even normalized device coordinates (NDC) [42] still require bounded x, y values and face-forwarding assumptions. This limitation constrains the usage for real-world videos, leading to artifacts and incorrect geometries as shown in Figure 6.

To address it, we re-parameterize (x, y, z, t) into spherical coordinate (θ, ϕ, r, t) and build HexPlane with θ, ρ, r, t axes, where $r = 1/\sqrt{x^2 + y^2 + z^2}$, θ, ϕ is the polar angle and azimuthal angle. During rendering, points are sampled with r linearly placed between 0 and 1. Without any special adjustments, HexPlane can represent dynamic fields with spherical coordinates, and deliver satisfactory results, which provides a solution for modeling unbounded scenes and exhibits robustness to different coordinate systems.

[2]Further tuning of initialization/other factors may lead to better results.

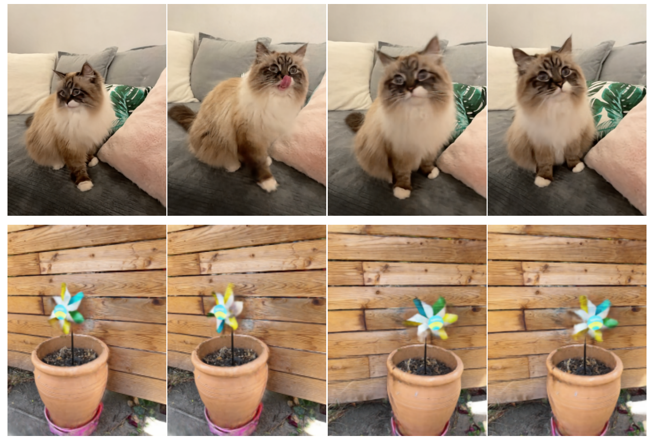

Figure 7. **Dynamic Novel View Synthesis on Videos Captured by iPhone.** We test HexPlane on casual videos captured by iPhone [17] and show synthesis results across novel timesteps and views. *Row one* are results with interpolated camera poses, while *Row two* shows results with extrapolated viewpoints, which are significantly distinct from camera poses used for video captures.

4.4. View Synthesis Results on Real Captured Video

We test HexPlane with monocular videos captured by iPhone from [17], whose camera trajectories are relatively casual and closer to real-world use cases. We show synthesis results in 7. Without any deformation or category-specific priors, our method could give realistic synthesis results on these real-world monocular videos, faithfully modeling static backgrounds, casual motions of cats, typology changes (cat's tongue), and fine details like cat hairs.

5. Conclusion

We propose HexPlane, an explicit representation for dynamic 3D scenes using six feature planes, which computes features of spacetime points via sampling and fusions. Compared to implicit representations, it could achieve comparable or even better synthesis quality for dynamic novel view synthesis, with over hundreds of times accelerations.

In this paper, we aim to keep HexPlane neat and general, preventing introducing deformation, category-specific priors, or other specific tricks. Using these ideas to make HexPlane better and faster would be an appealing future direction. Also, using HexPlane in other tasks except for dynamic novel view synthesis, e.g., spatial-temporal generation, would be interesting to explore [61]. We hope HexPlane could contribute to a broad range of research in 3D.

Acknowledgments Toyota Research Institute provided funds to support this work but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We thank Shengyi Qian for the title suggestion, David Fouhey, Mohamed El Banani, Ziyang Chen, Linyi Jin and for helpful discussions and feedbacks.

References

[1] Chong Bao, Bangbang Yang, Zeng Junyi, Bao Hujun, Zhang Yinda, Cui Zhaopeng, and Zhang Guofeng. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2

[4] Ang Cao, Chris Rockwell, and Justin Johnson. Fwd: Real-time novel view synthesis with forward warping and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15713–15724, 2022. 2

[5] 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. 2, 3

[6] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809, 2021. 2

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022. 1, 2, 3, 4, 7, 8

[8] Anpei Chen, Zexiang Xu, Xinyue Wei, Siyu Tang, Hao Su, and Andreas Geiger. Factor fields: A unified framework for neural fields and beyond. *arXiv preprint arXiv:2302.01226*, 2023. 2

[9] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021. 2

[10] Yu Deng, Jiaolong Yang, Jianfeng Xiang, and Xin Tong. Gram: Generative radiance manifolds for 3d-aware image generation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10663–10673, 2022. 2

[11] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14304–14314. IEEE Computer Society, 2021. 2

[12] 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. 2, 6

[13] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. *arXiv preprint arXiv:2301.10241*, 2023. 2

[14] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 1, 2, 8

[15] Wanshui Gan, Hongbin Xu, Yi Huang, Shifeng Chen, and Naoto Yokoya. V4d: Voxel for 4d novel view synthesis. *ArXiv*, abs/2205.14332, 2022. 2

[16] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021. 1, 2

[17] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NeurIPS*, 2022. 5, 8

[18] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14346–14355, 2021. 2

[19] Xiang Guo, Guanying Chen, Yuchao Dai, Xiaoqing Ye, Jiadai Sun, Xiao Tan, and Errui Ding. Neural deformable voxel grid for fast optimization of dynamic view synthesis. In *Asian Conference on Computer Vision*, 2022. 2

[20] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021. 2

[21] Di Huang, Sida Peng, Tong He, Xiaowei Zhou, and Wanli Ouyang. Ponder: Point cloud pre-training via neural rendering. *arXiv preprint arXiv:2301.00157*, 2022. 2

[22] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. 2022. 2

[23] Hankyu Jang and Daeyoung Kim. D-tensorf: Tensorial radiance fields for dynamic scenes. *arXiv preprint arXiv:2212.02375*, 2022. 2

[24] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. Geonerf: Generalizing nerf with geometry priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18365–18375, 2022. 2

[25] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *arXiv*, 2022. 2

[26] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3d video synthesis. *ArXiv*, abs/2210.14831, 2022. 2

[27] Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhofer, Jurgen Gall, Angjoo Kanazawa, and Christoph Lassner.

Tava: Template-free animatable volumetric actors. *ArXiv*, abs/2206.08929, 2022. 2

[28] 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, 5, 6

[29] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 1, 2

[30] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. *arXiv preprint arXiv:2211.10440*, 2022. 2

[31] S. Lionar, Daniil Emtsev, Dusan Svilarkovic, and Songyou Peng. Dynamic plane convolutional occupancy networks. *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1828–1837, 2020. 2

[32] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Devrf: Fast deformable voxel radiance fields for dynamic scenes. *ArXiv*, abs/2205.15723, 2022. 2

[33] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *ArXiv*, abs/2007.11571, 2020. 2

[34] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2

[35] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019. 6

[36] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason M. Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (TOG)*, 40:1 – 13, 2021. 2

[37] 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. 6

[38] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 2

[39] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 2

[40] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P Srinivasan, and Jonathan T Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16190–16199, 2022. 2

[41] 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)*, 38(4):1–14, 2019. 6

[42] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 3, 5, 8

[43] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 1, 2

[44] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022. 2, 4

[45] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[46] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020. 2

[47] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2856–2865, 2021. 2

[48] Hao Ouyang, Bo Zhang, Pan Zhang, Hao Yang, Jiaolong Yang, Dong Chen, Qifeng Chen, and Fang Wen. Real-time neural character rendering with pose-guided multiplane images. *ArXiv*, abs/2204.11820, 2022. 2

[49] 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, 2

[50] 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, 2

[51] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy

networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 2

[52] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9050–9059, 2021. 2

[53] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 2

[54] 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, 5, 6

[55] Shengyi Qian, Alexander Kirillov, Nikhila Ravi, Devendra Singh Chaplot, Justin Johnson, David F Fouhey, and Georgia Gkioxari. Recognizing scenes from novel viewpoints. *arXiv preprint arXiv:2112.01520*, 2021. 2

[56] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14315–14325, 2021. 2

[57] Umme Sara, Morium Akter, and Mohammad Shorif Uddin. Image quality assessment through fsim, ssim, mse and psnr—a comparative study. *Journal of Computer and Communications*, 7(3):8–18, 2019. 6

[58] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33:20154–20166, 2020. 2

[59] Katja Schwarz, Axel Sauer, Michael Niemeyer, Yiyi Liao, and Andreas Geiger. Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids. In *Advances in Neural Information Processing Systems*, 2022. 2

[60] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d : Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. *ArXiv*, abs/2211.11610, 2022. 2

[61] Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, and Yaniv Taigman. Text-to-4d dynamic scene generation. *ArXiv*, abs/2301.11280, 2023. 2, 8

[62] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019. 2

[63] Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *ArXiv*, abs/1906.01618, 2019. 2

[64] Liangchen Song, Anpei Chen, Zhong Li, Z. Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *ArXiv*, abs/2210.15947, 2022. 2

[65] Shih-Yang Su, Frank Yu, Michael Zollhoefer, and Helge Rhodin. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. In *NeurIPS*, 2021. 2

[66] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 1, 2

[67] Jiaming Sun, Xi Chen, Qianqian Wang, Zhengqi Li, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. Neural 3d reconstruction in the wild. *ACM SIGGRAPH 2022 Conference Proceedings*, 2022. 2

[68] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. *ACM SIGGRAPH 2022 Conference Proceedings*, 2022. 2

[69] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8238–8248, 2022. 2

[70] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021. 2

[71] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. *arXiv preprint arXiv:2112.03907*, 2021. 2

[72] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 2

[73] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis. *ArXiv*, abs/2212.00190, 2022. 2

[74] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. Fourier plenoctrees for dynamic radiance field rendering in real-time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13524–13534, June 2022. 2

[75] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021. 2

[76] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint

video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021. 2

[77] Jiaxin Xie, Hao Ouyang, Jingtan Piao, Chenyang Lei, and Qifeng Chen. High-fidelity 3d gan inversion by pseudo-multi-view optimization. *arXiv preprint arXiv:2211.15662*, 2022. 2

[78] Hongyi Xu, Thiemo Alldieck, and Cristian Sminchisescu. H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. In *NeurIPS*, 2021. 2

[79] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 2

[80] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Tsung-Yi Lin, Alberto Rodriguez, and Phillip Isola. NeRF-Supervision: Learning dense object descriptors from neural radiance fields. In *IEEE Conference on Robotics and Automation (ICRA)*, 2022. 2

[81] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *arXiv*, 2021. 1, 2, 4, 8

[82] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 2

[83] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13144–13152, 2021. 2

[84] Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. Editable free-viewpoint video using a layered neural representation. *ACM Transactions on Graphics (TOG)*, 40:1 – 18, 2021. 2

[85] Jason Y. Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. In *NeurIPS*, 2021. 2

[86] Kai Zhang, Nicholas I. Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *ECCV*, 2022. 2

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

[88] 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. 6

[89] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5439–5448, 2022. 2

[90] Yuqi Zhang, Guanying Chen, and Shuguang Cui. Efficient large-scale scene representation with a hybrid of high-resolution grid and plane features. *arXiv preprint arXiv:2303.03003*, 2023. 2

[91] Fuqiang Zhao, Wei Yang, Jiakai Zhang, Pei-Ying Lin, Yingliang Zhang, Jingyi Yu, and Lan Xu. Humannerf: Generalizable neural human radiance field from sparse inputs. *ArXiv*, abs/2112.02789, 2021. 2

[92] Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J Black, and Otmar Hilliges. Pointavatar: Deformable point-based head avatars from videos. *arXiv preprint arXiv:2212.08377*, 2022. 2

[93] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[94] Zhizhuo Zhou and Shubham Tulsiani. Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction. *arXiv preprint arXiv:2212.00792*, 2022. 2

[95] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022. 2

[96] Yiming Zuo and Jia Deng. View synthesis with sculpted neural points. *ArXiv*, abs/2205.05869, 2022. 2