K-Planes: Explicit Radiance Fields in Space, Time, and Appearance

Sara Fridovich-Keil*
UC Berkeley
sfk@berkeley.edu

Giacomo Meanti*
Istituto Italiano di Tecnologia
giacomo.meanti@iit.it

Frederik Rahbæk Warburg
Technical University of Denmark
frwa@dtu.dk

Benjamin Recht
UC Berkeley
brecht@berkeley.edu

Angjoo Kanazawa
UC Berkeley
kanazawa@berkeley.edu

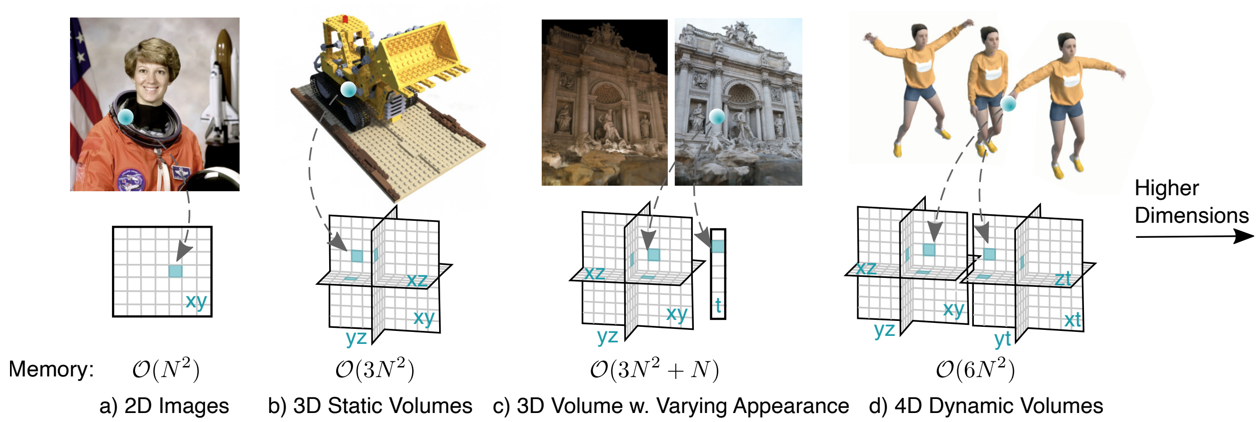


Figure 1. **Planar factorization of d-dimensional spaces.** We propose a simple planar factorization for volumetric rendering that naturally extends to arbitrary-dimensional spaces, and that scales gracefully with dimension in both optimization time and model size. We show the advantages of our approach on 3D static volumes, 3D photo collections with varying appearances, and 4D dynamic videos.

Abstract

We introduce k-planes, a white-box model for radiance fields in arbitrary dimensions. Our model uses $\binom{d}{2}$ ("d-choose-2") planes to represent a d-dimensional scene, providing a seamless way to go from static ($d=3$) to dynamic ($d=4$) scenes. This planar factorization makes adding dimension-specific priors easy, e.g. temporal smoothness and multi-resolution spatial structure, and induces a natural decomposition of static and dynamic components of a scene. We use a linear feature decoder with a learned color basis that yields similar performance as a nonlinear black-box MLP decoder. Across a range of synthetic and real, static and dynamic, fixed and varying appearance scenes, k-planes yields competitive and often state-of-the-art reconstruction fidelity with low memory usage, achieving 1000x compression over a full 4D grid, and fast optimization with a pure PyTorch implementation. For video results and code, please see `sarafridov.github.io/K-Planes`.

* equal contribution

1. Introduction

Recent interest in dynamic radiance fields demands representations of 4D volumes. However, storing a 4D volume directly is prohibitively expensive due to the curse of dimensionality. Several approaches have been proposed to factorize 3D volumes for static radiance fields, but these do not easily extend to higher dimensional volumes.

We propose a factorization of 4D volumes that is simple, interpretable, compact, and yields fast training and rendering. Specifically, we use six planes to represent a 4D volume, where the first three represent space and the last three represent space-time changes, as illustrated in Fig. 1(d). This decomposition of space and space-time makes our model interpretable, *i.e.* dynamic objects are clearly visible in the space-time planes, whereas static objects only appear in the space planes. This interpretability enables dimension-specific priors in time and space.

More generally, our approach yields a straightforward, prescriptive way to select a factorization of any dimension with 2D planes. For a d-dimensional space, we use $k=\binom{d}{2}$ ("d-*choose*-2") *k-planes*, which represent every pair of dimensions — for example, our model uses $\binom{4}{2} = 6$ *hex-planes* in 4D and reduces to $\binom{3}{2} = 3$ *tri-planes* in 3D. Choosing any other set of planes would entail either using more than k planes and thus occupying unnecessary memory, or using fewer planes and thereby forfeiting the ability to represent some potential interaction between two of the d dimensions. We call our model k-planes; Fig. 1 illustrates its natural application to both static and dynamic scenes.

Most radiance field models entail some black-box components with their use of MLPs. Instead, we seek a simple model whose functioning can be inspected and understood. We find two design choices to be fundamental in allowing k-planes to be a white-box model while maintaining reconstruction quality competitive with or better than previous black-box models [15, 27]: (1) Features from our k-planes are *multiplied* together rather than added, as was done in prior work [5, 6], and (2) our linear feature decoder uses a learned basis for view-dependent color, enabling greater adaptivity including the ability to model scenes with variable appearance. We show that an MLP decoder can be replaced with this linear feature decoder only when the planes are multiplied, suggesting that the former is involved in both view-dependent color and determining spatial structure.

Our factorization of 4D volumes into 2D planes leads to a high compression level without relying on MLPs, using 200 MB to represent a 4D volume whose direct representation at the same resolution would require more than 300 GB, a compression rate of three orders of magnitude. Furthermore, despite not using any custom CUDA kernels, k-planes trains orders of magnitude faster than prior implicit models and on par with concurrent hybrid models.

In summary, we present the first white-box, interpretable model capable of representing radiance fields in arbitrary dimensions, including static scenes, dynamic scenes, and scenes with variable appearance. Our k-planes model achieves competitive performance across reconstruction quality, model size, and optimization time across these varied tasks, without any custom CUDA kernels.

2. Related Work

K-planes is an interpretable, explicit model applicable to static scenes, scenes with varying appearances, and dynamic scenes, with compact model size and fast optimization time. Our model is the first to yield all of these attributes, as illustrated in Tab. 1. We further highlight that k-planes satisfies this in a simple framework that naturally extends to arbitrary dimensions.

Spatial decomposition. NeRF [21] proposed a fully implicit model with a large neural network queried many times during optimization, making it slow and essentially a black-box. Several works have used geometric representations to reduce the optimization time. Plenoxels [9] proposed a

	Static	Appearance	Dynamic	Fast	Compact	Explicit
NeRF	✓	✗	✗	✗	✓	✗
NeRF-W	✓	✓	✗	✗	✓	✗
DVGO	✓	✗	✗	✓	✗	✗
Plenoxels	✓	✗	✗	✓	✗	✓
Instant-NGP, TensoRF	✓	✗	✗	✓	✓	✗[1]
DyNeRF, D-NeRF	–	✗	✓	✗	✓	✗
TiNeuVox, Tensor4D	–	✗	✓	✓	✓	✗
MixVoxels, V4D	–	✗	✓	✓	✗	✗
NeRFPlayer	–	✗	✓	✓	✓[2]	✗
K-planes hybrid (Ours)	✓	✓	✓	✓	✓	✗
K-planes explicit (Ours)	✓	✓	✓	✓	✓	✓

[1] TensoRF offers both hybrid and explicit versions, with a small quality gap [2] NerfPlayer offers models at different sizes, the smallest of which has $<$ 100 million parameters but the largest of which has $>$ 300 million parameters

Table 1. **Related work overview.** The k-planes model works for a diverse set of scenes and tasks (static, varying appearance, and dynamic). It has a low memory usage (compact) and fast training and inference time (fast). Here "fast" includes any model that can optimize within a few ($<$ 6) hours on a single GPU, and "compact" denotes models that use less than roughly 100 million parameters. "Explicit" denotes white-box models that do not rely on MLPs.

fully explicit model with trilinear interpolation in a 3D grid, which reduced the optimization time from hours to a few minutes. However, their explicit grid representation of 3D volumes, and that of DVGO [30], grows exponentially with dimension, making it challenging to scale to high resolution and completely intractable for 4D dynamic volumes.

Hybrid methods [6, 22, 30] retain some explicit geometric structure, often compressed by a spatial decomposition, alongside a small MLP feature decoder. Instant-NGP [22] proposed a multiresolution voxel grid encoded implicitly via a hash function, allowing fast optimization and rendering with a compact model. TensoRF [6] achieved similar model compression and speed by replacing the voxel grid with a tensor decomposition into planes and vectors. In a generative setting, EG3D [5] proposed a similar spatial decomposition into three planes, whose values are added together to represent a 3D volume.

Our work is inspired by the explicit modeling of Plenoxels as well as these spatial decompositions, particularly the triplane model of [5], the tensor decomposition of [6], and the multiscale grid model of [22]. We also draw inspiration from Extreme MRI [23], which uses a multiscale low-rank decomposition to represent 4D dynamic volumes in magnetic resonance imaging. These spatial decomposition

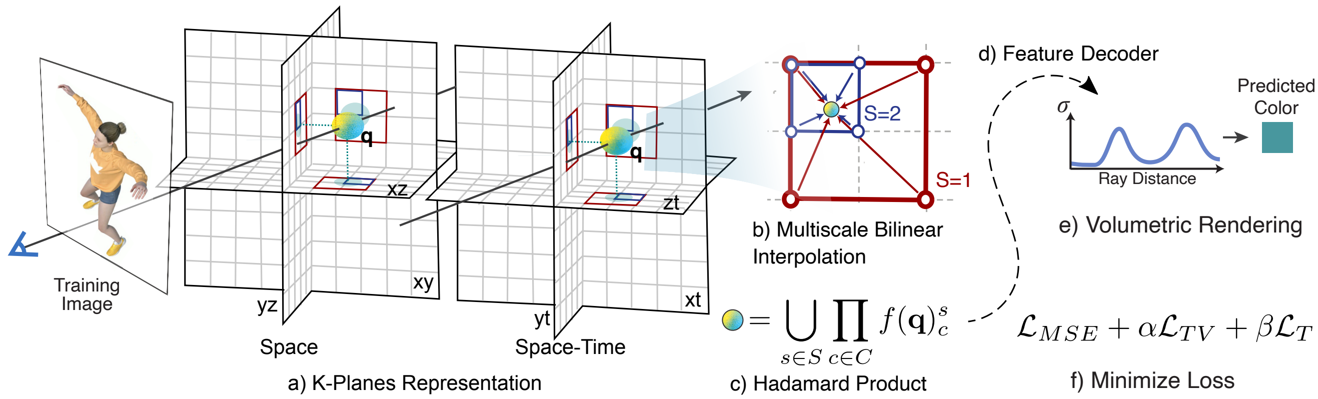


Figure 2. **Method overview.** (a) Our k-planes representation factorizes 4D dynamic volumes into six planes, three for space and three for spatiotemporal variations. To obtain the value of a 4D point $\mathbf{q} = (x, y, z, t)$, we first project the point into each plane, in which we (b) do multiscale bilinear interpolation. (c) The interpolated values are multiplied and then concatenated over the S scales. (d) These features are decoded either with a small MLP or our explicit linear decoder. (e) We follow the standard volumetric rendering formula to predict ray color and density. The model is optimized by (f) minimizing the reconstruction loss with simple regularization in space and time.

methods have been shown to offer a favorable balance of memory efficiency and optimization time for static scenes. However, it is not obvious how to extend these factorizations to 4D volumes in a memory-efficient way. K-planes defines a unified framework that enables efficient and interpretable factorizations of 3D and 4D volumes and trivially extends to even higher dimensional volumes.

Dynamic volumes. Applications such as Virtual Reality (VR) and Computed Tomography (CT) often require the ability to reconstruct 4D volumes. Several works have proposed extensions of NeRF to dynamic scenes. The two most common schemes are (1) modeling a deformation field on top of a static *canonical* field [7, 8, 16, 24, 27, 33, 40], or (2) directly learning a radiance field conditioned on time [11, 15, 16, 25, 38]. The former makes decomposing static and dynamic components easy [37, 40], but struggles with changes in scene topology (e.g. when a new object appears), while the latter makes disentangling static and dynamic objects hard. A third strategy is to choose a representation of 3D space and repeat it at each timestep (e.g. NeRFPlayer [29]), resulting in a model that ignores space-time interactions and can become impractically large for long videos.

Further, some of these models are fully implicit [15, 27] and thus suffer from extremely long training times (e.g. DyNeRF used 8 GPUs for 1 week to train a single scene), as well as being completely black-box. Others use partially explicit decompositions for video [8, 10, 13, 17, 18, 28, 29, 34], usually combining some voxel or spatially decomposed feature grid with one or more MLP components for feature decoding and/or representing scene dynamics. Most closely related to k-planes is Tensor4D [28], which uses 9 planes to decompose 4D volumes. K-planes is less redundant (*e.g.* Tensor4D includes two yt planes), does not rely on multiple MLPs, and offers a simpler factorization that naturally generalizes to static and dynamic scenes. Our method combines a fully explicit representation with a built-in decomposition of static and dynamic components, the ability to handle arbitrary topology and lighting changes over time, fast optimization, and compactness.

Appearance embedding. Reconstructing large environments from photographs taken with varying illumination is another domain in which implicit methods have shown appealing results, but hybrid and explicit approaches have not yet gained a foothold. NeRF-W [19] was the first to demonstrate photorealistic view synthesis in such environments. They augment a NeRF-based model with a learned global appearance code per frame, enabling it to explain away changes in appearance, such as time of day. With Generative Latent Optimization (GLO) [4], these appearance codes can further be used to manipulate the scene appearance by interpolation in the latent appearance space. Block-NeRF [31] employs similar appearance codes.

We show that our k-planes representation can also effectively reconstruct these unbounded environments with varying appearance. We similarly extend our model – either the learned color basis in the fully explicit version, or the MLP decoder in the hybrid version – with a global appearance code to disentangle global appearance from a scene without affecting geometry. To the best of our knowledge, ours is both the first fully explicit and the first hybrid method to successfully reconstruct these challenging scenes.

3. K-planes model

We propose a simple and interpretable model for representing scenes in arbitrary dimensions. Our representation yields low memory usage and fast training and rendering. The k-planes factorization, illustrated in Fig. 2, models a d-dimensional scene using $k = \binom{d}{2}$ planes representing every combination of two dimensions. For example, for static 3D scenes, this results in *tri-planes* with $\binom{3}{2} = 3$ planes representing xy, xz, and yz. For dynamic 4D scenes, this results in *hex-planes*, with $\binom{4}{2} = 6$ planes including the three space-only planes and three space-time planes xt, yt, and zt. Should we wish to represent a 5D space, we could use $\binom{5}{2} = 10$ *deca-planes*. In the following section, we describe the 4D instantiation of our k-planes factorization.

3.1. Hex-planes

The hex-planes factorization uses six planes. We refer to the space-only planes as $\mathbf{P}_{xy}$, $\mathbf{P}_{xz}$, and $\mathbf{P}_{yz}$, and the space-time planes as $\mathbf{P}_{xt}$, $\mathbf{P}_{yt}$, and $\mathbf{P}_{zt}$. Assuming symmetric spatial and temporal resolution N for simplicity of illustration, each of these planes has shape $N \times N \times M$, where M is the size of stored features that capture the density and view-dependent color of the scene.

We obtain the features of a 4D coordinate $\boldsymbol{q} = (i, j, k, \tau)$ by normalizing its entries between $[0, N)$ and projecting it onto these six planes

$$f(\boldsymbol{q})_c = \psi\big(\mathbf{P}_c, \pi_c(\boldsymbol{q})\big), \tag{1}$$

where π_c projects $\boldsymbol{q}$ onto the c'th plane and ψ denotes bilinear interpolation of a point into a regularly spaced 2D grid. We repeat Eq. (1) for each plane $c \in C$ to obtain feature vectors $f(\boldsymbol{q})_c$. We combine these features over the six planes using the Hadamard product (elementwise multiplication) to produce a final feature vector of length M

$$f(\boldsymbol{q}) = \prod_{c \in C} f(\boldsymbol{q})_c. \tag{2}$$

These features will be decoded into color and density using either a linear decoder or an MLP, described in Sec. 3.3.

Why Hadamard product? In 3D, k-planes reduces to the tri-plane factorization, which is similar to [5] except that the elements are multiplied. A natural question is why we multiply rather than add, as has been used in prior work with tri-plane models [5, 26]. Fig. 3 illustrates that combining the planes by multiplication allows k-planes to produce spatially localized signals, which is not possible with addition.

This selection ability of the Hadamard product produces substantial rendering improvements for linear decoders and modest improvement for MLP decoders, as shown in Tab. 2. This suggests that the MLP decoder is involved in both view-dependent color and determining spatial structure. The Hadamard product relieves the feature decoder of this extra task and makes it possible to reach similar performance using a linear decoder solely responsible for view-dependent color.

3.2. Interpretability

The separation of space-only and space-time planes makes the model interpretable and enables us to incorporate dimension-specific priors. For example, if a region of

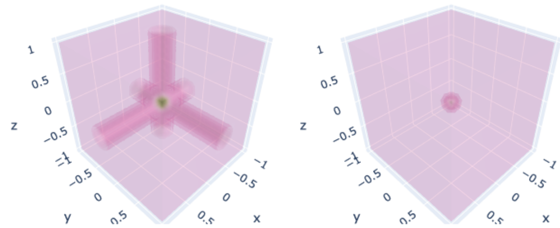


Figure 3. **Addition versus Hadamard product.** Elementwise addition of plane features (left) compared to multiplication (right), in a triplane example. A single entry in each plane is positive and the rest are zero, selecting a single 3D point by multiplication but producing intersecting lines by addition. This selection ability of multiplication improves the expressivity of our explicit model.

Plane Combination	Explicit	Hybrid	# params ↓
Multiplication	35.29	35.75	33M
Addition	28.78	34.80	33M

Table 2. **Ablation study over Hadamard product.** Multiplication of plane features yields a large improvement in PSNR ↑ for our explicit model, whereas our hybrid model can use its MLP decoder to partially compensate for the less expressive addition of planes. This experiment uses the static *Lego* scene [21] with 3 scales: 128, 256, and 512, and 32 features per scale.

the scene never moves, its temporal component will always be 1 (the multiplicative identity), thereby just using the features from the space planes. This offers compression benefits since a static region can easily be identified and compactly represented. Furthermore, the space-time separation improves interpretability, *i.e.* we can track the changes in time by visualizing the elements in the time-space planes that are not 1. This simplicity, separation, and interpretability make adding priors straightforward.

Multiscale planes. To encourage spatial smoothness and coherence, our model contains multiple copies at different spatial resolutions, for example 64, 128, 256, and 512. Models at each scale are treated separately, and the M-dimensional feature vectors from different scales are concatenated together before being passed to the decoder. The red and blue squares in Fig. 2a-b illustrate bilinear interpolation with multiscale planes. Inspired by the multiscale hash mapping of Instant-NGP [22], this representation efficiently encodes spatial features at different scales, allowing us to reduce the number of features stored at the highest resolution and thereby further compressing our model. Empirically, we do not find it necessary to represent our time dimension at multiple scales.

Total variation in space. Spatial total variation regularization encourages sparse gradients (with L1 norm) or smooth gradients (with L2 norm), encoding priors over edges being either sparse or smooth in space. We encourage this in 1D over the spatial dimensions of each of our space-time planes and in 2D over our space-only planes:

$$\mathcal{L}_{TV}(\mathbf{P}) = \frac{1}{|C|n^2} \sum_{c,i,j} \left(\|\mathbf{P}_c^{i,j} - \mathbf{P}_c^{i-1,j}\|_2^2 + \|\mathbf{P}_c^{i,j} - \mathbf{P}_c^{i,j-1}\|_2^2 \right), \tag{3}$$

where i, j are indices on the plane's resolution. Total variation is a common regularizer in inverse problems and was used in Plenoxels [9] and TensoRF [6]. We use the L2 version in our results, though we find that either L2 or L1 produces similar quality.

Smoothness in time. We encourage smooth motion with a 1D Laplacian (second derivative) filter

$$\mathcal{L}_{smooth}(\mathbf{P}) = \frac{1}{|C|n^2} \sum_{c,i,t} \|\mathbf{P}_c^{i,t-1} - 2\mathbf{P}_c^{i,t} + \mathbf{P}_c^{i,t+1}\|_2^2, \tag{4}$$

to penalize sharp "acceleration" over time. We only apply this regularizer on the time dimension of our space-time planes. Please see the appendix for an ablation study.

Sparse transients. We want the static part of the scene to be modeled by the space-only planes. We encourage this separation of space and time by initializing the features in the space-time planes as 1 (the multiplicative identity) and using an ℓ_1 regularizer on these planes during training:

$$\mathcal{L}_{sep}(\mathbf{P}) = \sum_c \|\mathbf{1} - \mathbf{P}_c\|_1, \qquad c \in \{xt, yt, zt\}. \tag{5}$$

In this way, the space-time plane features of the k-planes decomposition will remain fixed at 1 if the corresponding spatial content does not change over time.

3.3. Feature decoders

We offer two methods to decode the M-dimensional temporally- and spatially-localized feature vector $f(\boldsymbol{q})$ from Eq. (2) into density, σ, and view-dependent color, $\boldsymbol{c}$.

Learned color basis: a linear decoder and explicit model. Plenoxels [9], Plenoctrees [39], and TensoRF [6] proposed models where spatially-localized features are used as coefficients of the spherical harmonic (SH) basis, to describe view-dependent color. Such SH decoders can give both high-fidelity reconstructions and enhanced interpretability compared to MLP decoders. However, SH coefficients are difficult to optimize, and their expressivity is limited by the number of SH basis functions used (often limited 2nd degree harmonics, which produce blurry specular reflections).

Instead, we replace the SH functions with a learned basis, retaining the interpretability of treating features as coefficients for a linear decoder yet increasing the expressivity of the basis and allowing it to adapt to each scene, as was proposed in NeX [36]. We represent the basis using a small MLP that maps each view direction $\boldsymbol{d}$ to red $b_R(\boldsymbol{d}) \in \mathbb{R}^M$, green $b_G(\boldsymbol{d}) \in \mathbb{R}^M$, and blue $b_B(\boldsymbol{d}) \in \mathbb{R}^M$ *basis vectors*. The MLP serves as an adaptive drop-in replacement for the spherical harmonic basis functions repeated over the three color channels. We obtain the color values

$$\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{d}) = \bigcup_{i \in \{R,G,B\}} f(\boldsymbol{q}) \cdot b_i(\boldsymbol{d}), \tag{6}$$

where $\cdot$ denotes the dot product and $\cup$ denotes concatenation. Similarly, we use a learned basis $b_\sigma \in \mathbb{R}^M$, independent of the view direction, as a linear decoder for density:

$$\sigma(\boldsymbol{q}) = f(\boldsymbol{q}) \cdot b_\sigma. \tag{7}$$

Predicted color and density values are finally forced to be in their valid range by applying the sigmoid to $\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{d})$, and the exponential (with truncated gradient) to $\sigma(\boldsymbol{q})$.

MLP decoder: a hybrid model. Our model can also be used with an MLP decoder like that of Instant-NGP [22] and DVGO [30], turning it into a hybrid model. In this version, features are decoded by two small MLPs, one g_σ that maps the spatially-localized features into density σ and additional features $\hat{f}$, and another g_{RGB} that maps $\hat{f}$ and the embedded view direction $\gamma(\boldsymbol{d})$ into RGB color

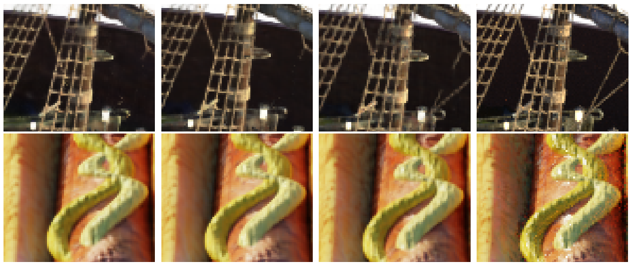
$$\begin{aligned} \sigma(\boldsymbol{q}), \hat{f}(\boldsymbol{q}) &= g_\sigma(f(\boldsymbol{q})) \\ \boldsymbol{c}(\boldsymbol{q}, \boldsymbol{d}) &= g_{RGB}(\hat{f}(\boldsymbol{q}), \gamma(\boldsymbol{d})). \end{aligned} \tag{8}$$

As in the linear decoder case, the predicted density and color values are finally normalized via exponential and sigmoid, respectively.

Global appearance. We also show a simple extension of our k-planes model that enables it to represent scenes with consistent, static geometry viewed under varying lighting or appearance conditions. Such scenes appear in the Phototourism [14] dataset of famous landmarks photographed at different times of day and in different weather. To model this variable appearance, we augment k-planes with an M-dimensional vector for each training image $1, \ldots, T$. Similar to NeRF-W [19], we optimize this per-image feature vector and pass it as an additional input to either the MLP learned color basis b_R, b_G, b_B, in our explicit version, or to the MLP color decoder g_{RGB}, in our hybrid version, so that it can affect color but not geometry.

3.4. Optimization details

Contraction and normalized device coordinates. For forward-facing scenes, we apply normalized device coordinates (NDC) [21] to better allocate our resolution while enabling unbounded depth. We also implement an ℓ_∞ version (rather than ℓ_2) of the scene contraction proposed in Mip-NeRF 360 [2], which we use on the unbounded Phototourism scenes.



(a) Ours-explicit (b) Ours-hybrid (c) TensoRF (d) Ground truth

Figure 4. **Zoomed qualitative results on static NeRF scenes.** Visual comparison of k-planes, TensoRF [6], and the ground truth, on *ship* (top) and *hotdog* (bottom).

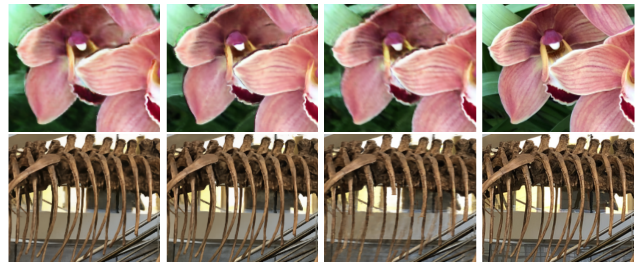
Proposal sampling. We use a variant of the proposal sampling strategy from Mip-NeRF 360 [2], with a small instance of k-planes as density model. Proposal sampling works by iteratively refining density estimates along a ray, to allocate more points in the regions of higher density. We use a two-stage sampler, resulting in fewer samples that must be evaluated in the full model and in sharper details by placing those samples closer to object surfaces. The density models used for proposal sampling are trained with the histogram loss [2].

Importance sampling. For multiview dynamic scenes, we implement a version of the importance sampling based on temporal difference (IST) strategy from DyNeRF [15]. During the last portion of optimization, we sample training rays proportionally to the maximum variation in their color within 25 frames before or after. This results in higher sampling probabilities in the dynamic region. We apply this strategy after the static scene has converged with uniformly sampled rays. In our experiments, IST has only a modest impact on full-frame metrics but improves visual quality in the small dynamic region. Note that importance sampling cannot be used for monocular videos or datasets with moving cameras.

4. Results

We demonstrate the broad applicability of our planar decomposition via experiments in three domains: static scenes (both bounded 360° and unbounded forward-facing), dynamic scenes (forward-facing multi-view and bounded 360° monocular), and Phototourism scenes with variable appearance. For all experiments, we report the metrics PSNR (pixel-level similarity) and SSIM[1] [35] (structural similarity), as well as approximate training time and number of parameters (in millions), in Tab. 3. Blank entries in Tab. 3 denote baseline methods for which the corresponding information is not readily available. Full per-scene results may be found in the appendix.

[1]Note that among prior work, some evaluate using an implementation of SSIM from MipNeRF [1] whereas others use the scikit-image implementation, which tends to produce higher values. For fair comparison on each dataset we make a best effort to use the same SSIM implementation as the relevant prior work.



(a) Ours-explicit (b) Ours-hybrid (c) TensoRF (d) Ground truth

Figure 5. **Zoomed qualitative results on static LLFF scenes.** Visual comparison of k-planes, TensoRF [6], and the ground truth, on *orchids* (top) and *T-rex* (bottom).

4.1. Static scenes

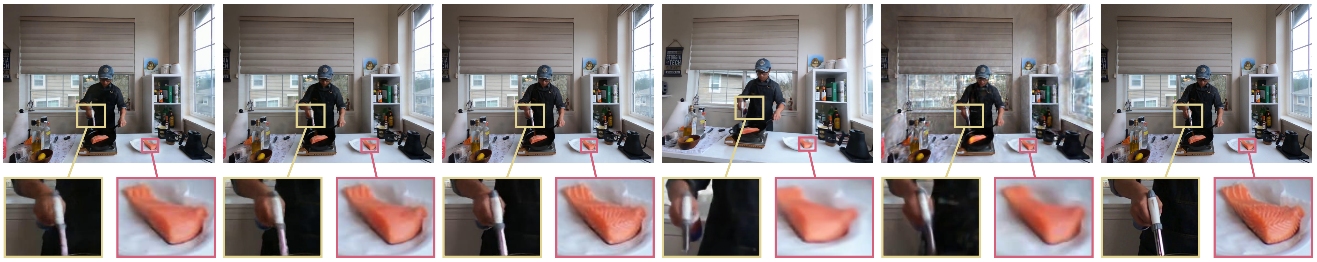
We first demonstrate our triplane model on the bounded, 360°, synthetic scenes from NeRF [21]. We use a model with three symmetric spatial resolutions $N \in \{128, 256, 512\}$ and feature length $M = 32$ at each scale; please see the appendix for ablation studies over these hyperparameters. The explicit and hybrid versions of our model perform similarly, within the range of recent results on this benchmark. Fig. 4 shows zoomed-in visual results on a small sampling of scenes. We also present results of our triplane model on the unbounded, forward-facing, real scenes from LLFF [20]. Our results on this dataset are similar to the synthetic static scenes; both versions of our model match or exceed the prior state-of-the-art, with the hybrid version achieving slightly higher metrics than the fully explicit version. Fig. 5 shows zoomed-in visual results on a small sampling of scenes.

4.2. Dynamic scenes

We evaluate our hexplane model on two dynamic scene datasets: a set of synthetic, bounded, 360°, monocular videos from D-NeRF [27] and a set of real, unbounded, forward-facing, multiview videos from DyNeRF [15].

The D-NeRF dataset contains eight videos of varying duration, from 50 frames to 200 frames per video. Each timestep has a single training image from a different viewpoint; the camera "teleports" between adjacent timestamps [12]. Standardized test views are from novel camera positions at a range of timestamps throughout the video. Both our explicit and hybrid models outperform D-NeRF in both quality metrics and training time, though they do not surpass very recent hybrid methods TiNeuVox [8] and V4D [10], as shown in Fig. 7.

The DyNeRF dataset contains six 10-second videos recorded at 30 fps simultaneously by 15-20 cameras from a range of forward-facing view directions; the exact number of cameras varies per scene because a few cameras produced miscalibrated videos. A central camera is reserved



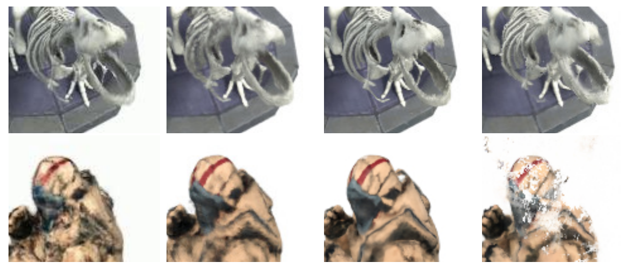
Ours-explicit Ours-hybrid DyNeRF MixVoxels Neural Volumes Ground truth

Figure 6. **Qualitative video results.** Our hexplane model rivals the rendering quality of state-of-the-art neural rendering methods. Our renderings were obtained after at most 4 hours of optimization on a single GPU whereas DyNeRF trained for a week on 8 GPUs. MixVoxels frame comes from a slightly different video rendering, and is thus slightly shifted.

	PSNR ↑	SSIM ↑	Train Time ↓	# Params ↓
NeRF [21] (static, synthetic)				
Ours-explicit	32.21	0.960	38 min	33M
Ours-hybrid	32.36	0.962	38 min	33M
Plenoxels [9]	31.71	0.958	11 min	∼500M
TensoRF [6]	33.14	0.963	17 min	18M
I-NGP [22]	33.18	-	5 min	∼ 16M
LLFF [20] (static, real)				
Ours-explicit	26.78	0.841	33 min	19M
Ours-hybrid	26.92	0.847	33 min	19M
Plenoxels	26.29	0.839	24 min	∼500M
TensoRF	26.73	0.839	25 min	45M
D-NeRF [27] (dynamic, synthetic)				
Ours-explicit	31.05	0.97	52 min	37M
Ours-hybrid	31.61	0.97	52 min	37M
D-NeRF	29.67	0.95	48 hrs	1-3M
TiNeuVox [8]	32.67	0.97	30 min	∼12M
V4D [10]	33.72	0.98	4.9 hrs	275M
DyNeRF [15] (dynamic, real)				
Ours-explicit	30.88	0.960	3.7 hrs	51M
Ours-hybrid	31.63	0.964	1.8 hrs	27M
DyNeRF [15]	[1]29.58	-	1344 hrs	7M
LLFF [20]	[1]23.24	-	-	-
MixVoxels-L [34]	30.80	0.960	1.3 hrs	125M
Phototourism [14] (variable appearance)				
Ours-explicit	22.25	0.859	35 min	36M
Ours-hybrid	22.92	0.877	35 min	36M
NeRF-W [19]	27.00	0.962	384 hrs	∼2M
NeRF-W (public)[2]	19.70	0.764	164 hrs	∼2M
LearnIt [32]	19.26	-	-	-

[1] DyNeRF and LLFF only report metrics on the *flame salmon* video (the first 10 seconds); average performance may be higher as this is one of the more challenging videos. [2] Open-source version https://github.com/kwea123/nerf_pl/tree/nerfw where we re-implemented test-time optimization as for k-planes.

Table 3. **Results.** Averaged metrics over all scenes in the respective datasets. Note that Phototourism scenes use MS-SSIM (multiscale structural similarity) instead of SSIM. K-planes timings are based on a single NVIDIA A30 GPU. Please see the appendix for per-scene results and the website for video reconstructions.



(a) Ours-explicit (b) TiNeuVox (c) V4D (d) D-NeRF

Figure 7. **Zoomed qualitative results on scenes from D-NeRF [27].** Visual comparison of k-planes, D-NeRF [27], TiNeuVox [8] and V4D [10], on *t-rex* (top) and *hook* (bottom).

for evaluation, and training uses frames from the remaining cameras. Both our methods again produce similar quality metrics to prior state-of-the-art, including recent hybrid method MixVoxels [34], with our hybrid method achieving higher quality metrics. See Fig. 6 for a visual comparison.

4.2.1 Decomposing time and space

One neat consequence of our planar decomposition of time and space is that it naturally disentangles dynamic and static portions of the scene. The static-only part of the scene can be obtained by setting the three time planes to one (the multiplicative identity). Subtracting the static-only rendered image from the full rendering (i.e. with the time plane parameters not set to 1), we can reveal the dynamic part of the scene. Fig. 9 shows this decomposition of time and space. This natural volumetric disentanglement of a scene into static and dynamic regions may enable many applications across augmented and virtual reality [3].

We can also visualize the time planes to better understand where motion occurs in a video. Fig. 8 shows the averaged features learned by the xt plane in our model for the *flame salmon* and *cut beef* DyNeRF videos, in which we can identify the motions of the hands in both space and time. The xt plane learns to be sparse, with most entries equal to the multiplicative identity, due to a combination of our sparse transients prior and the true sparsity of motion in the video. For example, in the left side of Fig. 8 one of the

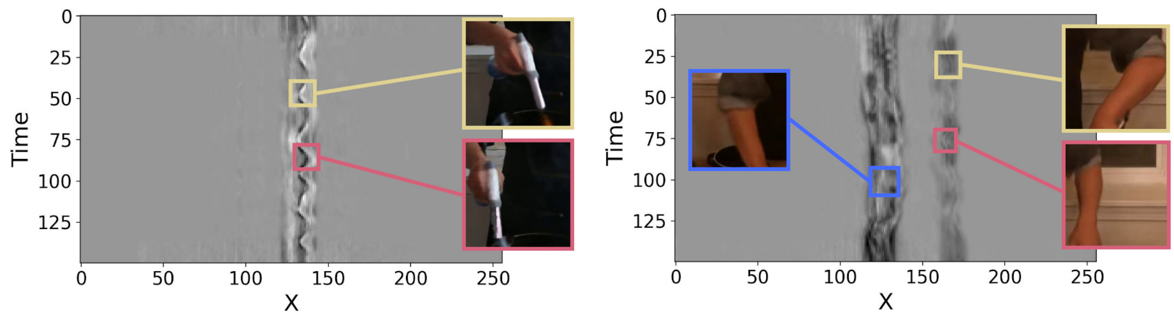


Figure 8. **Visualization of a time plane.** The xt plane highlights the dynamic regions in the scene. The wiggly patterns across time correspond to the motion of the person's hands and cooking tools, in the *flame salmon* scene (left) where only one hand moves and the *cut beef* scene (right) where both hands move.

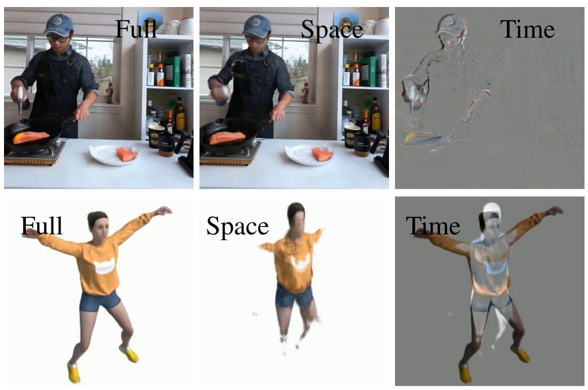


Figure 9. **Decomposition of space and time.** K-planes (left) naturally decomposes a 3D video into static and dynamic components. We render the static part (middle) by setting the time planes to the identity, and the remainder (right) is the dynamic part. Top shows the *flame salmon* multiview video [15] and bottom shows the *jumping jacks* monocular video [27].

cook's arms contains most of the motion, while in the right side both arms move. Having access to such an explicit representation of time allows us to add time-specific priors.

4.3. Variable appearance

Our variable appearance experiments use the Phototourism dataset [14], which includes photos of well-known landmarks taken by tourists with arbitrary view directions, lighting conditions, and transient occluders, mostly other tourists. Our experimental conditions parallel those of NeRF-W [19]: we train on more than a thousand tourist photographs and test on a standard set that is free of transient occluders. Like NeRF-W, we evaluate on test images by optimizing our per-image appearance feature on the left half of the image and computing metrics on the right half. Visual comparison to prior work is shown in the appendix.

Also similar to NeRF-W [4, 19], we can interpolate in the appearance code space. Since only the color decoder (and not the density decoder) takes the appearance code as input, our approach is guaranteed not to change the geometry, regardless of whether we use our explicit or our hybrid model. Fig. 10 shows that our planar decomposition with a 32-dimensional appearance code is sufficient to accurately capture global appearance changes in the scene.

Figure 10. **Appearance interpolation**. Like NeRF-W [19], we can interpolate our appearance code to alter the visual appearance of landmarks. We show three test views from the *Trevi fountain* with appearance codes corresponding to day and night.

5. Conclusions

We introduced a simple yet versatile method to decompose a d-dimensional space into $\binom{d}{2}$ planes, which can be optimized directly from indirect measurements and scales gracefully in model size and optimization time with increasing dimension, without any custom CUDA kernels. We demonstrated that the proposed k-planes decomposition applies naturally to reconstruction of static 3D scenes as well as dynamic 4D videos, and with the addition of a global appearance code can also extend to the more challenging task of unconstrained scene reconstruction. K-planes is the first explicit, simple model to demonstrate competitive performance across such varied tasks.

Acknowledgments. Many thanks to Matthew Tancik, Ruilong Li, and other members of KAIR for helpful discussion and pointers. We also thank the DyNeRF authors for their response to our questions about their method.

References

[1] 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 *ICCV*, pages 5835–5844. IEEE, 2021. 6

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, pages 5460–5469. IEEE, 2022. 5, 6

[3] Sagie Benaim, Frederik Warburg, Peter Ebert Christensen, and Serge Belongie. Volumetric disentanglement for 3d scene manipulation, 2022. 7

[4] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks, 2017. 3, 8

[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, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, pages 16102–16112, 2022. 2, 4

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 2, 5, 6, 7

[7] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, pages 14304–14314, 2021. 3

[8] 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. In *SIGGRAPH Asia 2022 Conference Papers*. ACM, nov 2022. 3, 6, 7

[9] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5491–5500. IEEE, 2022. 2, 5, 7

[10] Wanshui Gan, Hongbin Xu, Yi Huang, Shifeng Chen, and Naoto Yokoya. V4d: Voxel for 4d novel view synthesis, 2022. 3, 6, 7

[11] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *ICCV*, pages 5712–5721, 2021. 3

[12] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NeurIPS*, 2022. 6

[13] 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 *ACCV*, 2022. 3

[14] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *Int. J. Comput. Vis.*, 129(2):517–547, 2021. 5, 7, 8

[15] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video. In *CVPR*, pages 5511–5521, 2022. 2, 3, 6, 7, 8

[16] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 3

[17] 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:2205.15723*, 2022. 3

[18] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM TOG*, 38(4):65:1–65:14, July 2019. 3

[19] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 3, 5, 7, 8

[20] 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 TOG*, 38(4):29:1–29:14, 2019. 6, 7

[21] 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*, pages 405–421. Springer, 2020. 2, 4, 5, 6, 7

[22] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4), 2022. 2, 4, 5, 7

[23] Frank Ong, Xucheng Zhu, Joseph Y. Cheng, Kevin M. Johnson, Peder E. Z. Larson, Shreyas S. Vasanawala, and Michael Lustig. Extreme mri: Large-scale volumetric dynamic imaging from continuous non-gated acquisitions. *Magnetic Resonance in Medicine*, 84(4):1763–1780, 2020. 2

[24] 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 *ICCV*, 2021. 3

[25] 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. *ACM TOG*, 40(6), dec 2021. 3

[26] Songyou Peng, Michael Niemeyer, Lars M. Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, Lecture Notes in Computer Science, pages 523–540, 2020. 4

[27] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 2, 3, 6, 7, 8

[28] 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, 2022. 3

[29] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields, 2022. 3

[30] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2, 5

[31] 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. 3

[32] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, pages 2846–2855. Computer Vision Foundation / IEEE, 2021. 7

[33] 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 *ICCV*. IEEE, 2021. 3

[34] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis, 2022. 3, 7

[35] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 6

[36] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion, 2021. 5

[37] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. D^2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video, 2022. 3

[38] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, pages 9421–9431, 2021. 3

[39] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, pages 5732–5741, 2021. 5

[40] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. STaR: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *CVPR*, 2021. 3