WavePlanes: A compact Wavelet representation for Dynamic Neural Radiance Fields

Adrian Azzarelli
Visual Information Laboratory
University of Bristol
a.azzarelli@bristol.ac.uk

Nantheera Anantrasirichai
Visual Information Laboratory
University of Bristol
n.anantrasirichai@bristol.ac.uk

David R Bull
Visual Information Laboratory
University of Bristol
dave.bull@bristol.ac.uk

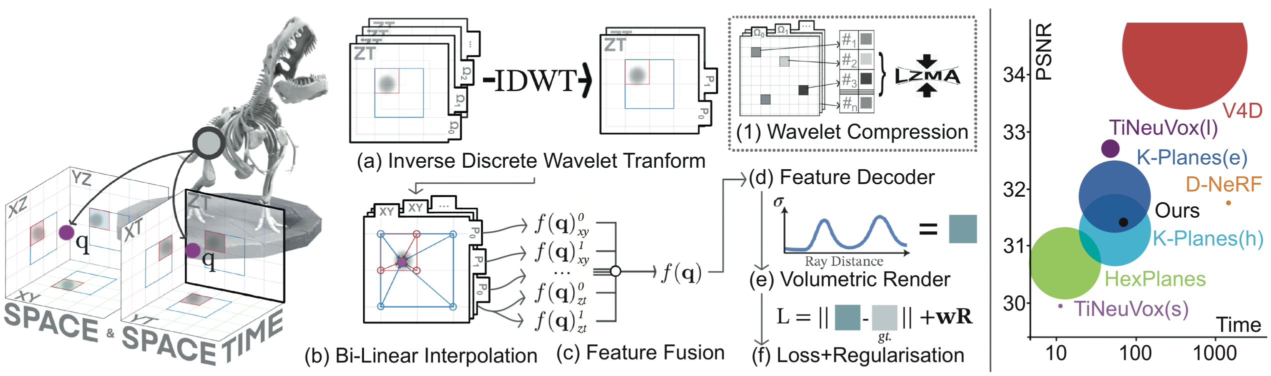


Figure 1. **Method overview.** We represent N-level wavelet coefficients as $\mathbf{\Omega}^s_c$. (a) The wavelet representation is transformed using the inverse discrete wavelet transform (IDWT). (b) 4-D samples are projected onto each plane and bi-linearly interpolated over multiple scales. (c) Features sampled from different planes and multiple scales are fused into a single feature. (d) Features are linearly decoded. (e) A 2-D pixel is rendered from the 3-D volume using the NeRF volumetric rendering function. (f) A loss and weighted regularization is used for training. *After training:* (1) We compress non-zero wavelet coefficients. **Model Comparison**: Our model is small, fast and performant. Time is measured in minutes and model size is represented by point radius.

Abstract

Dynamic Neural Radiance Fields (Dynamic NeRF) enhance NeRF technology to model moving scenes. However, they are resource intensive and challenging to compress. To address this issue, this paper presents WavePlanes, a fast and more compact explicit model. We propose a multi-scale space and space-time feature plane representation using N-level 2-D wavelet coefficients. The inverse discrete wavelet transform reconstructs N feature signals at varying detail, which are linearly decoded to approximate the color and density of volumes in a 4-D grid. Exploiting the sparsity of wavelet coefficients, we compress a Hash Map containing only non-zero coefficients and their locations on each plane. This results in a compressed model size of ∼12 MB. Compared with state-of-the-art plane-based models, WavePlanes is up to 15× smaller, less computationally demanding and achieves comparable results in as little as one hour of training - without requiring custom CUDA code or high performance computing resources. Additionally, we propose new feature fusion schemes that work as well as previously proposed schemes while providing greater interpretability. Our code is available at: ~~*https://github.com/...*~~

1. Introduction

Neural Radiance Fields (NeRFs) have gained significant attention due to their ability to generate high-resolution 3D scenes with accurate novel view synthesis. The traditional

NeRF is designed for static scenes and struggles to model objects and scenes with motion. Dynamic NeRFs extend these capabilities to address this issue, broadening applications in computer graphics by supporting the generation of realistic object and scene animations. While static NeRF frameworks yield impressive results, dynamic NeRF frameworks still face numerous limitations, particularly associated with complexity, scalability and stream-ability.

Recent advancements in plane-based representations, including matrix-vector and plane decomposed 3-D space, have demonstrated promising results for static scenes [5, 13, 22]. These representations have also shown success in modelling dynamic environments [3, 10] and demonstrate substantial improvements to training speed. However, the significance and interpretability of the temporal component (represented by an axis on a 2-D feature planes) have not been fully realized, particularly in scenarios with high spatial detail or fast motion. Consequently, these models tend to be large and challenging to compress, making tasks such as 4-D video streaming problematic. For static scenes, wavelet-based plane representations have been used to alleviate these problems [19]. However this approach has not been explored for dynamic scenes.

In this paper, we propose the use of wavelets as a basis for a 4-D decomposed plane representation, as demonstrated in Figure 1. Our motivation is to provide a highly compressible representation, which could benefit data-intensive transmission tasks such as 4-D video streaming.

Two design choices are proposed to improve the processing time, performance and final model size:

- Firstly, we shift the reconstructed space-time features by +1, prior to feature fusion. This maximizes the likely hood of static empty space represented by the fused features while preserving sparsity in the decomposed wavelet representation. This consequently enables our proposed wavelet-based compression scheme.
- Secondly, we offer modifications to previously proposed *time smoothness* and *time sparsity* (TS) regularizers in [10]. Substituting the time smoothness regularizer, we propose smoothing spatial features on the space-time planes. This improves performance on scenes with rapid changes in motion. We also modify the time sparsity function to work directly on the wavelet coefficient representation. This eliminates the need for signal reconstruction during regularization and noticeably reduces the training time.

The proposed compression scheme (Figure 1 (1)) exploits the sparsity of our representation to further reduce our model size without using an MLP and without affecting performance. This is accomplished by filtering the wavelet coefficients using hard thresholding, constructing a Hash Map containing non-zero coefficients and their locations on each wavelet plane, and then compressing it using a lossless algorithm.

Finally, we investigate alternative feature fusion methods (Figure 1 (c)), and its relevance to model interpretability. Up to this point, only the element-wise feature factorization and addition were tested [10]. Hence, we extend the investigation and propose two feature fusion methods capable of competing with the prior factorization-only scheme while offering better interpretability.

In summary, we contribute:

1. A wavelet-based plane representation for modelling dynamic scenes as explicit 4-D volumes; offering a compressible and fast model capable of competing with the state-of-the-art.
2. New regularizers for plane-based approaches.
3. A novel compression scheme that achieves state-of-the-art model size.
4. Alternative feature fusion schemes with more interpretability and comparable performance to prior schemes.

2. Related Work

Dynamic NeRFs. Representing 3-D volumes in time is currently accomplished by one of the following techniques:

1. *Jointly learning a 3-D "canonical" (static) and 3-D deformation field*. This anchors the canonical field to a point in time, typically $t = 0$s, where the deformation field linearly shifts canonical field volumes in space, conditioned on time, as proposed in D-NeRF [18] and Dy-NeRF [14]. It disentangles time-conditioned spatial features in a way that preserves volumetric consistency. However, they are not capable of discovering view-dependant effects under motion. NeRF-DS [27] addresses this by conditioning color features on surface position and orientation. Unfortunately, this approach still fails to adapt to temporal changes in scene topology.

2. *Decomposing 4-D into representational components*, such as matrix-vectors and planes. This approach was introduced by Tensor4D [21], which offers a 9-plane candecomp/parafac [4, 12] /matrix-matrix (MM) decomposition. Subsequently, K-Planes [10] and HexPlanes [3] proposed a 6-plane and 6-plane+6-vector decomposition, respectively, along with various feature-plane fusion schemes. HexPlanes uses a multiplicative-additive approach similar to Tensor4D, while K-Planes demonstrates that a purely multiplicative approach is preferred over an additive approach.

3. *Using static representations as a set of key-frames.* This approach relies on the robustness and compactness of static NeRF representations, along with the effectiveness of a key-frame interpolation strategy. NeRFPlayer [23] decomposes features into static, deforming and "new areas", where the latter is used to represent temporal changes

that occur after the first key frame, such as water gradually mixing with coffee. HyperReel [1] proposes a hierarchical key-frame interpolation strategy that is compact and fast as a result of using TensoRF [5] for modelling each key-frame. However, it is limited by computational requirements needed to load the model. There is also an overlap with approach (2), as HumanRF [13] utilizes a key-framed MM decomposition for dynamic human modelling; storing and selecting a set of space-only (static) feature planes for at each time frame.

4. *Directly learning a 4-D field conditioned on time.* This groups the remaining work. DynIBaR [15] proposes a novel ray-conditioning scheme for dynamic videos, where features in each frame are conditioned on the predicted features of surrounding frames. V4D [11] proposes a texture driven, time-conditioned, voxel-based model where pixel-level refinement and a conditional positional encoder are used to handle over-smoothing caused by the total variation regularizer.

Each approach has its own set of benefits and drawbacks. For instance, using MLPs results in small models size but long optimization times. Whereas explicit approaches, such as K-Planes, offer faster optimization though require significantly more storage, as indicated in Figure 1. As a plane-based approach our method is fast, yet we are not limited by model size.

Wavelet NeRFs. Using wavelets to model static scenes is a recent development and has demonstrated the ability to enhance the compactness of existing representations [19, 20, 26]. WIRE [20] proposes better-suited activation functions for implicit neural representations and achieves a reduction of $p/\sqrt{2}$ w.r.t the number of trainable parameters by proposing a complex Gabor wavelet activation. The authors in [19] propose a tri-plane decomposed wavelet-based representation. To improve compression, an additional mask for each plane is learnt to exploit the sparsity of the tri-plane representation, which reduces the model size to around 2MB. We have yet to see this applied to modelling dynamic scenes. Hence, in this paper we introduce the first wavelet-based approach for modelling 4-D volumes.

K=4-Planes Background. We extend the K-Planes (explicit) model, which learns 4-D feature volumes in space and time, $f(\mathbf{q})$ where $\mathbf{q} = (x, y, z, t)$, as a set of multi-scale 2-D feature planes, $\mathbf{P}_c^s$ for $c \in C = [xy, xz, yz, xt, yt, zt]$ and $s \in S = [64, 128, 256, 512]$ with a constant scale for time. To sample features, $\mathbf{q}$ is normalized and projected onto each plane using $f_c(\mathbf{q}) = \psi(\mathbf{P}_c, \pi_c(\mathbf{q}))$, where π_c denotes the planar projection and ψ denotes the bi-linear interpolation of the projected point on a 2-D grid. Subsequently a 4-D feature volume is reconstructed using the Hardman Product (HP) (an element-wise operator), in Equation 1.

$$f(\mathbf{q}) = [\![\prod_{c \in C} f_c^s(\mathbf{q})]\!]^{s \in S} \tag{1}$$

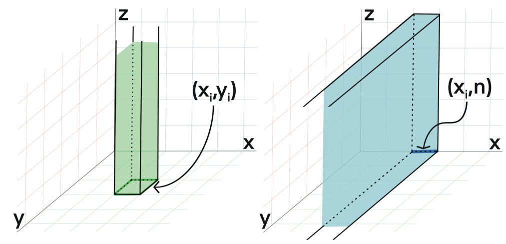


Figure 2. The volumetric regions at time $t = n$ affected by a single feature on P_{xy} (a tile) and P_{xt} (an edge), respectively; for the HP fusion scheme.

Where $\prod$ applies the HP operator fusing plane features and $[\![\cdot]\!]$ represents a concatenation across scales.

The space-time planes, denoted as $c_t \in C_t = [xt, yt, zt]$, are initialized and regularized towards 1. This provides some interpretability as it allows the separation of static components where $f_{c_t}(\mathbf{q}) = 1$, representing the multiplicative-1 identity in Equation 1. Still, the case where $f_c(\mathbf{q}) = 0$, representing the multiplicative-0 identity, is not considered. The influence of a zero-valued feature in Equation 1 is globally destructive and is interpreted differently for space-only and space-time planes. zero-valued space-only features indicate empty static space and zero-valued space-time features identify empty static slices of 3-D space at a given time, as illustrated in Figure 2. Hence, our intention is to explore alternative approaches to HP feature fusion that avoid the destructive result of zero-valued features to offer a higher level of interpretability.

3. WavePlanes

In WavePlanes, the representation is stored as a set of 2-D wavelet coefficients, $\mathbf{\Omega}_c^s$ and is encoded using the IDWT into a fine and a coarse feature plane, P_c^0 and P_c^1, as shown in Figure 1 (a). This is accomplished by performing the IDWT twice with the full set and a reduced set of wavelet coefficients, respectively. Note that we use $s \in [0, 1, ..., N]$ to indicate the level of a wavelet plane rather than the scale of a feature plane.

3.1. Feature Plane Reconstruction

We refer to the *father* wavelet coefficients as $\Omega_c^{[s=0]} \in \mathcal{R}^{B \times H/2^N \times W/2^N}$, where B is the feature length, (H, W) is the desired height and width of the reconstructed feature plane and 2^N is a down-sampling factor involved in wavelet decomposition. The *mother* wavelet coefficients are defined as $\Omega_c^{[s>0]} \in \mathcal{R}^{B \times 3 \times H/s' \times W/s'}$, where 3 indicates the number of filters[1] and $s' = 2^{N-s}$ allows for down-sampling where the highest frequency signal will have size $(H/2, W/2)$ and the lowest frequency will have

[1]Containing horizontal, vertical and diagonal filters.

size $(H/2^N, W/2^N)$. Hence, for a wavelet decomposition level of 2, $N=2$ and $s \in [0,1,2]$.

To avoid vanishing gradients for high-frequency coefficients during the IDWT, we apply the frequency-based scaling factor to each plane, denoted by $\mathbf{k} \odot \mathbf{\Omega}_c$ where $\mathbf{k} = [1, \frac{2}{5}, \frac{1}{5}]$ as suggested in [19]. Note that we select smaller values.

To reconstruct the feature planes we use Equation 2. Using a 2-level decomposition, we find that producing two scales of features for $s=1$ and $s=2$ is sufficient.

$$P_c^s(\Omega_c|s) \begin{cases} \text{IDWT}(\mathbf{k} \odot \Omega_c^{[0:s]}), & \text{if } c \in [xy, xz, yz] \\ \text{IDWT}(\mathbf{k} \odot \Omega_c^{[0:s]}) + 1, & \text{otherwise} \end{cases} \tag{2}$$

Where the $+1$ term shifts the space-time features towards 1. This preserves sparsity in the wavelet representation as coefficients are naturally biased towards 0. This biases the feature representation toward 1, providing separability between static and dynamic volumes [10].

3.2. Feature Fusion and Interpretability

Following K-Planes, a 4-D sample, $\mathbf{q}$, is normalized between $[0, S)$, projected onto each plane and bi-linearly interpolated using $f(\mathbf{q})_c^s = \psi(\mathbf{P}_c^s, \pi_c(\mathbf{q}))$, where ψ and π_c were discussed in Section 2. Consequently we can obtain a HP-fused feature using Equation 1.

The feature fusion process allows us to interpret space-only planes as a set of localized features that are transformed by any element-wise operator using features from the space-time planes. This implies that the HP operator is used to constrain the time at which a spatial volume becomes static, i.e. when $f_{c_t}(\mathbf{q}) = 1$. As previously mentioned, the issue when $f_c(\mathbf{q}) = 0$ has been (unknowingly) addressed in K-Planes through the introduction of the sparse transients regularizer, which biases features towards 1. In our case, we relax the weighting for this regularizer and introduce a new factorization scheme that achieves similar benefit by instead conditioning the multiplicative-0 identity in an attempt to increase the rendering quality for occluded regions of empty space - a challenge for the original K-Planes model. We propose two schemes below.

Zero-agreement masked multiplication (ZMM). We first develop the HP fusion scheme, where all zero-valued space-time features for a 4-D volume sample are conditioned to agree on empty dynamic space. If the temporal features of a given sample are not all zero-valued then the zero-valued features are treated as 1-valued features, allowing gradient flow without modifying the fused feature. We express this as a differentiable binary mask, $Q_c(\cdot)$, where 1 indicates zero-valued inputs. We replace all zero-valued features in $f_{c_t}(\mathbf{q})$ with 1 and perform the feature factorization *only* across the space-time planes as shown in Equation 3. Using Equation 4 we subsequently recover an inverted

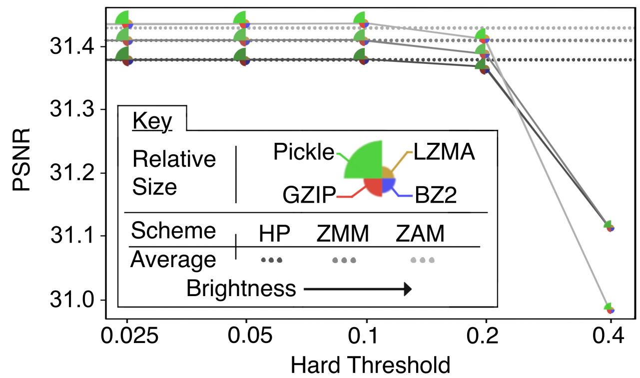


Figure 3. **Evaluating Pickle, GZIP, Burrow-Wheeler (BZ2) and Lempel–Ziv–Markov chain (LZMA) compression algorithms** on the proposed feature fusion schemes, using the D-NeRF T-Rex scene [18]. The model size produced by each compression scheme is represented by the radius of corresponding slice.

binarized mask that is multiplied by the new space-time features as in Equation 5. Finally, the result is multiplied with feature samples from the space-only planes to recover a 4-D feature volume.

$$f'_{c_t}(\mathbf{q}) = f_{c_t}(\mathbf{q}) + Q_{c_t}(f_{c_t}(\mathbf{q})) \tag{3}$$

$$Q^{-1} = |\mathbf{1} - \prod_{c \in C_t} Q_c(f_c(\mathbf{q}))| \tag{4}$$

$$f_{\text{space-time}}(\mathbf{q}) = Q^{-1} \cdot \prod_{c \in C_t} f'_c(\mathbf{q}) \tag{5}$$

Hence, the volume sample becomes empty only when all space-time features for a 4-D sample agree on a zero-value.

Zero-agreement masked addition (ZAM). A natural interpretation of the zero-agreement condition in ZMM is the addition of space-time features only. This is implemented using Equation 6, where a sum of 0 results in the multiplicative-0 identity and the $1/3$ factor ensures that when all space-time features are 1 the sum is also 1, retaining the separability of static and dynamic volumes. Therefore temporally dependant static volumes are still represented by $f_c(\mathbf{q}) = 1$ and empty space is naturally represented by $f(\mathbf{q}) = 0$.

$$f(\mathbf{q}) = [\![\frac{1}{3} \sum_{c \in C_t} f_c^s(\mathbf{q}) \cdot \prod_{c \in C} f_c^s(\mathbf{q})]\!]^{s \in S} \tag{6}$$

3.3. Compressing Wavelets with Hash Maps

A benefit of using wavelets is the ability to compress representations by relying on the tendency of coefficients to be zero. Unlike other plane-based models, this is true for all of our planes and is possible thanks to the condition proposed in Equation 2. We apply compression with Hash Maps after training and demonstrate in Figure 3 the size difference with various lossless compression schemes. Using Hash Maps means our representation can be rapidly reconstructed with $O(1)$ time complexity.

Before compression, we apply hard thresholding to each coefficient on each grid-plane $\mathbf{\Omega}_c$. As we model features and not signals, the significance of non-zero coefficients is ambiguous. Still, the feature landscape is smooth and zero-valued coefficients are not ambiguous, so we can filter near-zero coefficients without deprecation.

We then use a Hash Map to store all non-zero values where each key states the value's location within the set $\mathbf{\Omega}_c$. For most planes this results in (approximately) a 90% reduction of the the number of values to be stored. The only case for which this is not true is for space-only father-wavelets planes, where we observe only a $< 1\%$ reduction, indicating that data is concentrated on these planes.

We select the appropriate threshold value and lossless compression algorithm using Figure 3, where LZMA and a threshold of 0.1 produce the smallest model size without loss. As compression is lossless, the small improvements in PSNR for the ZAM results indicate some denoising capability linked to hard-thresholding. This could be explored in future work for denoising non-zero features distributed around means, potentially indicating material classes.

3.4. Optimization

Regularization. For dynamic scenes, we use three regularizers, namely total variation (TV), spatial smoothness in time (SST) and TS. TV regularization was used for static scenes in Plenoxels [9] and TensoRF [5] and recently for dynamic scenes in K-Planes. This is defined in Equation 7.

$$\mathcal{L}_{TV}(\mathbf{P}) = \frac{1}{|C|n^2} \sum_{c,i,j} (||\mathbf{P}_c^{i,j} - \mathbf{P}_c^{i-1,j}||_2^2 + ||\mathbf{P}_c^{i,j} - \mathbf{P}_c^{i,j-1}||_2^2) \tag{7}$$

The proposed SST regularizer is denoted in Equation 8 and represents the 1-D Laplacian approximation of the second derivatives of the spatial components for each time-plane. Rather than smoothing along the time-axis, as is done in K-Planes, we find that smoothing spatial components produces better results. Please refer to the supplementary materials for an ablation study.

$$\mathcal{L}_{SST} = \frac{1}{|C|n^2} \sum_{c,i,t} \mathbf{P}_c^{i-1,t} - 2\mathbf{P}_c^{i,t} + \mathbf{P}_c^{i+1,t}||_2^2 \tag{8}$$

TS regularization is proposed in Equation 9 and maximizes the number of zero-valued space-time wavelet coefficients. This pairs well with our compression scheme as it directly maximizes the sparsity of $\mathbf{\Omega}_{c_t}$.

$$\mathcal{L}_{TS} = \sum_{c_t \in C_t} ||\mathbf{\Omega}_{c_t}||_1 \tag{9}$$

Feature decoder with learned color basis. We treat fused features as coefficients of spherical harmonics (SH) basis. SH has been used in Plenoctrees [28], Plenoxels and TensoRF for static scenes and was recently used in [24] for modelling scenes as Fourier Plenoctrees. The degree of the SH often leads to limited expressivity. This suggests a trade-off between high computational cost and sensitive convergence for higher levels of expressivity. Instead, NeX [25] uses a per-scene learned color basis, where features are treated as coefficients for a linear decoder. K-Planes demonstrates this for dynamic scenes so we utilize the same approach where a basis is learnt using a tiny fully-fused MLP [17]. This maps the viewing direction $\mathbf{d}$ to red, green and blue basis vectors, $b_i(\mathbf{d})$ for $i \in \{R, G, B\}$. An additional density basis is learnt and left independent of viewing direction. Subsequently, color and density values are recovered by applying the dot product between basis vectors $b_i(\mathbf{d})$ and fused features $f(\mathbf{q})$.

Proposal Sampling Network. Mip-NeRF 360 [2] introduced proposal sampling as an iterable method with n_p stages. At each stage a network roughly predicts spatial density at a given time to infer regions of interest - higher density regions are sampled more frequently. For each stage we have the option to re-use the original proposal network or instantiate additional proposal networks. We chose two stages for the proposal sampling scheme and as it incurs minimal cost we use different proposal networks at each iteration with feature length 8 and 16, respectively. We use a lower resolution version of our model as the proposal network. Finally, the proposal networks are trained using a histogram loss [2].

Importance Sampling based on temporal differences (IST) is proposed in DyNeRF [14] to handle the real dynamic DyNeRF data set. Rays are sampled in relation to their maximum color variance, thus higher sampling is attributed to 4-D volumes that are assumed to be dynamic. In our experiments, we follow the same strategy proposed in K-Planes, whereby we sample rays uniformly for the first half of training and then apply the IST strategy.

3.5. Implementation

Our implementation is a branch off the K-Planes repository. To implement the wavelet functionality we use the *pytorch_wavelets* library [6]. To boost training speed we cache (on the GPU) the resulting feature planes, $\mathbf{P}_c^s$, at the start of every epoch to avoid re-processing the IDWT during regularization. This significantly reduces training time though increases memory consumption, shown in Table 1. The memory footprint for the faster approach is larger than K-Planes yet GPU utilization is lower. This is important as utilization ultimately limits our ability to run K-Planes on larger data sets with our available hardware. y

Method	Mem. ↓	Util. ↓	Time ↓
D-NeRF data set [18]			
K-Planes [10]	7.0 GB	86%	59 mins
Ours	7.7 GB	79%	72 mins
Ours without Caching	7.4 GB	70%	120mins
DyNeRF data set [14]			
K-Planes* [10]	-	-	-
Ours	9.7 GB	89%	510mins
Ours without Caching	9.1 GB	82%	690mins

Table 1. **Computational comparisons of our model with and without caching the wavelet representation.** Further comparison with K-Planes is provided and both models use HP fusion for fair comparison. *Mem.* is the GPU memory. *Util.* is the GPU utilization. *Time* is the training time. *Did not run on our hardware.

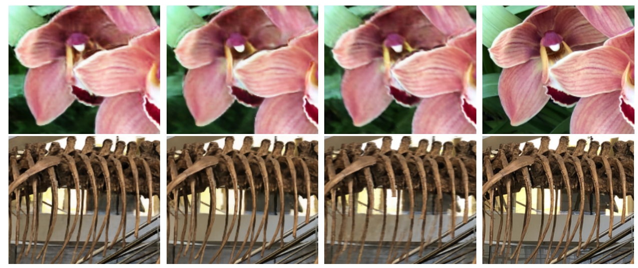
4. Results

We employ a 4k batch size and the same scene bounding boxes used in [10] for fair comparison. For synthetic scenes we select a spatial resolution of $H = W = 256$ for high-complexity scenes and 128 for low-complexity scenes. This reduces memory consumption and training time while producing comparable results, in contrast to other models which usually select resolutions up to 512. Additionally, for all experiments, we evaluate our ZMM-based approach with feature length $B = 64$, using PSNR and SSIM, as shown in Table 2. Quantitative (objective metric) results for the ZAM and HP approaches are provided in the supplementary materials and show similar performance. Note that we did not retrain the existing models; all results were taken from the original papers. In the ablation study, we further separate the PSNR evaluation for foreground and background renders. This reduces ambiguity in image-quality assessment metrics, which can exhibit bias towards large white backgrounds for synthetic scenes. The foreground and background are using the alpha mask of RGBA image data sets. Then morphological dilation is applied to include noisy foreground artifacts in foreground evaluations.

We are limited by access to high-performance compute systems and all tests were carried out on a 24GB NVIDIA RTX 3090 GPU. For LLFF and D-NeRF data sets we used 32GB of RAM whereas DyNeRF required 98GB of RAM. For most data sets this is not an issue. However, for the DyNeRF data set [14] we are unable to pre-generate IST weights with the usual 2× down sampling. Hence, we choose to down sample by 8× during IST weight generation and 2× during training, to demonstrate our model's ability to synthesize real dynamic scenes.

4.1. Real Static Scenes

We assess WavePlanes on the sparse, forward-facing, bounded, multi-view and real static LLFF data set [16]. Each scene contains 20-60 images and is down sampled to 1008×756 pixels. In this experiment WavePlanes is reduced to a tri-plane model, using HP fusion. While it performs satisfactorily over the full data set, WavePlanes demonstrates competitive performance on scenes containing >35 training images, indicated by the partial results in Table 2. Visually, we find that WavePlanes is better at dealing with edges and is capable of modelling complex features, e.g. the rib-cage in Figure 4.



(a) Ours (b) K-Planes (c) TensoRF (d) Ground Truth

Figure 4. **Zoomed visual comparison of real static scene renders**. Demonstrating WavePlanes ability to model static scenes.

4.2. Dynamic Scenes

Real multi-view scenes. The DyNeRF data set [14] is used to test our model's ability to perform novel view synthesis on forward-facing, multi-view (19-21 cameras), real dynamic scenes lasting 10 seconds at 30FPS. The scenes contain different indoor lighting set-ups with a large variation of textures and materials. For this test, we use a temporal resolution of $W = 150$. Quantitative results in Table 2 indicate desirable performance for our method, considering the reduced quality of the sampling strategy. Visually, WavePlanes performs novel view synthesis well and shows sharper edges and less noise on a number of objects in the scene, e.g. the pattern on the uncooked salmon is clearer whilst those of K-Planes are smoothed out, shown in Figure 5. This is notably accomplished with a memory budget at least 4× smaller than other plane-based methods.

Synthetic monocular scenes. The D-NeRF data set [18] was used to perform novel view synthesis on low and high complexity synthetic object animations. These were captured using monocular "teleporting" cameras rotated at 360° and contain 50-200 training frames that are down sampled to 400×400 pixels. For this experiment, the resolution of the time axis for space-time grids is equivalent to the number of training images. WavePlanes performs adequately across the entire data set and particularly well on scenes containing a high amount of occlusion while being compact and small in size. This is indicated by the partial results in Table 2 that comprises the Lego, T-Rex and Hell Warrior scenes. The visual results in Figure 6 shows that WavePlanes achieves cleaner results than K-Planes and produces better detail in high frequency areas than K-Planes and TiNeuVox, e.g. around the T-Rex's mouth. For Hook-

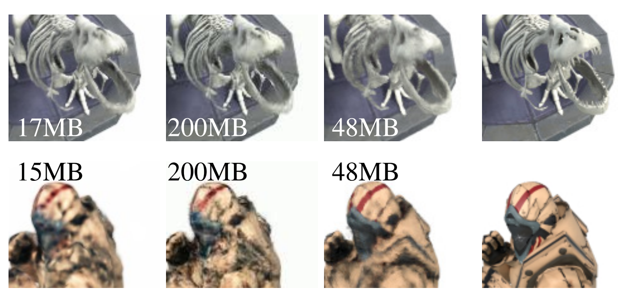
	PSNR ↑		SSIM ↑	# Params ↓	Average Size ↓
	Full	Partial			
Real static LLFF scenes [16]					
Plenoxels [9]	26.29	28.84	0.839	∼500M	-
Masked Wavelets (small) [19]	26.25	29.05	0.839	-	3.2MB
Masked Wavelets (large) [19]	26.54	29.42	0.839	-	7.4MB
K-Planes (explicit) [10]	26.78	29.75	0.847	19M	∼ 100MB
Ours	26.10	29.30	0.825	51M	93MB
Real dynamic DyNeRF scenes [14]					
HexPlanes [3]	31.71	-	-	-	∼ 200MB
K-Planes (explicit) [10]	31.30	-	0.960	51M	250MB
K-Planes (hybrid) [10]	31.92	-	0.964	27M	250MB
Ours	29.75	-	0.922	67M	58MB
Synthetic dynamic D-NeRF assets [18]					
D-NeRF [18]	29.67	26.14	0.95	1-3M	13MB
V4D [11]	33.72	29.06	0.98	275M	377MB
TiNeuVox-S [8]	27.10	28.10	0.96	-	8MB
TiNeuVox-B [8]	28.63	29.08	0.97	-	48MB
HexPlanes [3]	31.04	27.50	-	-	∼ 200MB
K-Planes (explicit) [10]	31.05	27.45	0.97	37M	∼ 200MB
K-Planes (hybrid) [10]	31.61	27.66	0.97	37M	∼ 200MB
Ours	30.50	27.43	0.97	16M	12MB

Table 2. **Average novel view synthesis results**. *Partial* indicates a subset of the data set.

Ours* K-Planes (explicit)* K-Planes (hybrid)* HexPlanes DyNeRF Ground truth

Figure 5. **Qualitative real video results**. WavePlanes can synthesize real objects to an acceptable standard. *Trained on the first 10/40 seconds.

D-NeRF scene, our method smooths high-frequency errors found in the K-Planes making the visual result more comprehensible, e.g., the mouth guard and shoulder of the Hook character.



(a) Ours (b) K-Planes (c) TiNeuVox (d) Ground Truth

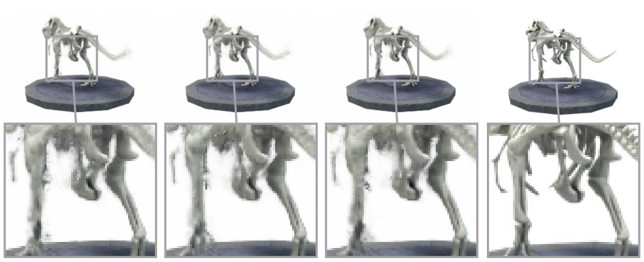
Figure 6. **Visual comparison of the T-Rex and Hook D-NeRF scenes** [18] (Success and Failure case). WavePlanes can produce better results for high occlusion scenes (T-Rex). Even for the failure case WavePlanes is less noisy than K-Planes (Hook).

4.3. Feature Fusion.

Results provided in Table 3 and Figure 7 demonstrate the effectiveness of the proposed feature fusion schemes on the D-NeRF data set. ZMM and ZAM are capable of outperforming other plane-based models for synthetic scenes. Despite the difference in IST down sampling, ZAM provides decent objective results for the real scene. Visually, we notice a slightly smoother render with higher attention to contrast for the ZMM approach.

	PSNR ↑			SSIM ↑	Time ↓
	Whole	Front	Back		
T-Rex (synthetic) D-NeRF [18]					
K-P	31.28	-	-	0.980	58min
HexP.	30.67	-	-	-	11min
HP	31.38	20.71	75.66	0.978	72min
ZMM	31.41	20.74	76.41	0.978	78min
ZAM	31.43	20.77	77.11	0.979	76min
Flame Salmon (real) DyNeRF [14]					
K-P.	28.71	-	-	0.942	222min
HexP.	29.47	-	-	-	720min
HP	27.31	-	-	0.893	510min
ZMM	27.6	-	-	0.893	540min
ZAM	28.25	-	-	0.900	530min

Table 3. **Quantitative results for propose feature fusion schemes**. We compare quality and training time to other plane-based models. K-P and HexP. are the K-Planes (explicit) and HexPlanes+ results, respectively.



HP ZAM ZMM Ground Truth

Figure 7. **Qualitative feature fusion results**. Differences can be seen in the amount of noise and around the left leg and the hip.

4.4. Ablation Study

We perform an ablation study using the D-NeRF synthetic data set [18]. Please refer to the supplementary materials for more results.

Wavelet family. Table 4 compares different wavelet functions with varying degrees of regularity. Regularity indicates the number of continuous derivatives that wavelet function has and is tied to the number of vanishing moments, denoted by the M in "dbM", "coifM", etc. For wavelets with less regularity (e.g. the Haar wavelet) we find our method performs worse on foreground predictions. Wavelets with higher regularity are also negligibly slower. However, we find the coif4 wavelet is an exception as it is performant yet slightly slower (by $\sim$ 5minutes). This is likely because it has a larger filter length, of $6M$, compared with other functions such as dbM with length $2M$. Overall, the results indicate that orthogonality and symmetry have little importance, whereas the function's shape, regularity and filter length impacts performance the most. Similar conclusions were found for low-bit rate wavelet coding tasks [7], which may suggest that this is shared characteristic of low-resolution image tasks, such modelling dynamic scenes using low resolution (400×400 pixel) images.

	PSNR ↑			SSIM ↑
	Whole	Front	Back	
Wavelet families				
Haar	30.85	20.18	75.58	0.977
Coif2	31.06	20.38	77.01	0.977
Coif4	31.10	20.43	**78.45**	0.977
Bior1.3	30.99	20.31	77.98	0.977
Bior4.4	**31.12**	**20.45**	75.41	0.977
Db2	31.00	20.33	77.58	0.977
Db6	31.08	20.42	76.62	0.977
N-Level Wavelet				
2	**31.30**	**20.63**	**76.01**	**0.977**
3	30.46	19.79	74.08	0.975
4	30.22	19.56	69.29	0.974

Table 4. **Wavelet families:** Comparing the quality of different wavelet functions. **N-Level Wavelet:** Comparing varying levels of wavelet decomposition.

Wavelet decomposition level. In Table 4 we compare the number of wavelet levels for the main WavePlane field. More levels of decomposition means more coefficients, which in turn impacts performance and rendering time significantly. The convergence of higher levels of wavelet decomposition is also slower where a 2-level decomposition is $\sim$ 13 minutes faster than a 3-level decomposition.

5. Conclusion and Future Work

Conclusion. We propose WavePlanes, the first wavelet-based approach for modelling dynamic NeRFs. We show that our model can synthesize various scenes without the need for high performance computing resources or custom CUDA code, while producing competitive objective results and perceptually pleasing visual results. By exploiting the characteristics of the wavelet representation, our novel compression scheme reduces the final model size by up to 15× compared to other plane-based models without significantly impacting training time or performance.

Future work. Plane-based approaches are currently limited in their ability to model objects outside of the predefined bounding box [10, 19, 21]. In future work it could prove beneficial to model the out-of-bounds scene with a HDRI-style mapping, similar to what was proposed for NeRF++ [29].

Additionally, using discrete grids with a fixed temporal resolution makes it challenging to model fast motion without introducing some degree of temporal uncertainty (noise). This limitation is shared by current dynamic plane-based models. However, the results from our compression scheme indicate that the high pass filter applied to our wavelet representation may be useful for denoising. Consequently, a more sophisticated approach could be explored to correct regions of space-time affected by high motion noise. For instance, by classifying wavelet features a band pass filter could be used to contract features dispersed around a class means. Filtering non-zero coefficients would also lead to better compression as more values would be repeated. This may alleviate limitations in our compression scheme when a scene contains little empty space, as is the case for numerous LLFF scenes.

References

[1] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16610–16620, 2023. 3

[2] 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. 5

[3] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 2, 7

[4] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970. 2

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. 2, 3, 5

[6] Fergal Cotter. *Uses of complex wavelets in deep convolutional neural networks*. PhD thesis, 2020. 5

[7] Eduardo AB da Silva and Mohammed Ghanbari. On the performance of linear phase wavelet transforms in low bit-rate image coding. *IEEE Transactions on Image Processing*, 5 (5):689–704, 1996. 8

[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*, pages 1–9, 2022. 7

[9] 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. 5, 7

[10] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 2, 4, 6, 7, 9

[11] Wanshui Gan, Hongbin Xu, Yi Huang, Shifeng Chen, and Naoto Yokoya. V4d: Voxel for 4d novel view synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 3, 7

[12] Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970. 2

[13] Mustafa Işık, Martin Rünz, Markos Georgopoulos, Taras Khakhulin, Jonathan Starck, Lourdes Agapito, and Matthias Nießner. Humanrf: High-fidelity neural radiance fields for humans in motion. *arXiv preprint arXiv:2305.06356*, 2023. 2, 3

[14] 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. 2, 5, 6, 7, 8

[15] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4273–4284, 2023. 3

[16] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 6, 7

[17] Thomas Müller. tiny-cuda-nn, 2021. 5

[18] 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. 2, 4, 6, 7, 8

[19] Daniel Rho, Byeonghyeon Lee, Seungtae Nam, Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Masked wavelet representation for compact neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20680–20690, 2023. 2, 3, 4, 7, 9

[20] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. Wire: Wavelet implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18507–18516, 2023. 3

[21] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16632–16642, 2023. 2, 9

[22] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20875–20886, 2023. 2

[23] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023. 2

[24] 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*, pages 13524–13534, 2022. 5

[25] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8534–8543, 2021. 5

[26] Muyu Xu, Fangneng Zhan, Jiahui Zhang, Yingchen Yu, Xiaoqin Zhang, Christian Theobalt, Ling Shao, and Shijian Lu. Wavenerf: Wavelet-based generalizable neural radiance fields. *arXiv preprint arXiv:2308.04826*, 2023. 3

[27] Zhiwen Yan, Chen Li, and Gim Hee Lee. Nerf-ds: Neural radiance fields for dynamic specular objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8285–8295, 2023. 2

[28] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 5

[29] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 9