Synergistic Integration of Coordinate Network and Tensorial Feature for Improving Neural Radiance Fields from Sparse Inputs

Mingyu Kim [1] **Jun-Seong Kim** [2] **Se-Young Yun** [1] **Jin-Hwa Kim** [3]

Abstract

The multi-plane representation has been highlighted for its fast training and inference across static and dynamic neural radiance fields. This approach constructs relevant features via projection onto learnable grids and interpolating adjacent vertices. However, it has limitations in capturing low-frequency details and tends to overuse parameters for low-frequency features due to its bias toward fine details, despite its multi-resolution concept. This phenomenon leads to instability and inefficiency when training poses are sparse. In this work, we propose a method that synergistically integrates multi-plane representation with a coordinate-based MLP network known for strong bias toward low-frequency signals. The coordinate-based network is responsible for capturing low-frequency details, while the multi-plane representation focuses on capturing fine-grained details. We demonstrate that using residual connections between them seamlessly preserves their own inherent properties. Additionally, the proposed progressive training scheme accelerates the disentanglement of these two features. We demonstrate empirically that our proposed method not only outperforms baseline models for both static and dynamic NeRFs with sparse inputs, but also achieves comparable results with fewer parameters.

1. Introduction

Neural Radiance Fields (NeRFs) have gained recognition for their ability to create realistic images from various viewpoints using the volume rendering technique (Mildenhall et al., 2021). Early studies have demonstrated that multi-layer perception (MLP) networks, combined with sinusoidal encoding, can effectively synthesize 3-dimensional novel views (Mildenhall et al., 2021; Tancik et al., 2020; Sitzmann et al., 2020; Martel et al., 2021; Lindell et al., 2022). These studies have shown that coordinate-based MLP networks exhibit strong low-frequency bias, and incorporating sinusoidal encoding allows for capturing both low and high-frequency signals.

For broader real-world applicability, extensive efforts have focused on reliably constructing radiance fields in cases of sparse input data (Yu et al., 2021; Wang et al., 2021; Chen et al., 2021; Jain et al., 2021). One set of solutions tackled this by leveraging a pretrained image encoder to compare rendered scenes against consistent 3D environments (Yu et al., 2021; Wang et al., 2021; Chen et al., 2021; Jain et al., 2021). Another approach incorporated additional information, such as depth or color constraints, to maintain 3-dimensional coherence (Deng et al., 2022; Yuan et al., 2022; Roessle et al., 2022; Truong et al., 2023). Methods progressively adjusting the frequency spectrum of position encoding have proven effective in counteracting overfitting without additional information (Yang et al., 2023; Song et al., 2023). However, sinusoidal encoding requires over 5 hours of training time, complicated regularizations, and exhibits a performance gap from explicit representation.

Approaches explicitly parameterizing spatial attributes through voxel-grid, hashgrid, and multi-plane have been introduced (Müller et al., 2022; Chen et al., 2022; Chan et al., 2022). These methods dramatically reduce training time and produce cleaner and more realistic images, meanwhile demanding excessive memory consumption (Lee et al., 2024). The recent works found those representations struggle with low-frequency detail and overfit to high-frequency signals, especially when applying for dynamic scenes, despite using multi-scale representations (Fridovich-Keil et al., 2023; Cao & Johnson, 2023; Peng et al., 2023). While those have marginally had success in the reconstruction of NeRF with the assistance of denoising penalties like total variation (Sun et al., 2023; Fridovich-Keil et al., 2023), they still lack adequate representation of low-frequency spectral features like object shapes and dynamic motion, as shown in Figure 1a.

[1]KAIST AI [2]POSTECH EE [3]NAVER AI Lab & SNU AIIS. Correspondence to: Jin-Hwa Kim and Se-Young Yun <j1nhwa.kim@navercorp.com; yunseyoung@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

(a) HexPlane (b) Ours

Figure 1: The qualitative results of the `standup` case in dynamic NeRFs using 25 training poses (about 17% of the original data). This is challenging due to the limited information available along the time axis. Figure (a) is produced by HexPlane. (Cao & Johnson, 2023). Figure (b) is the rendered image of the proposed method.

To alleviate this issue, we introduce a simple yet powerful approach to fundamentally improve the performance of static and dynamic NeRFs from sparse inputs. In this framework, coordinate-based MLP features handle low-frequency context, while multiple-plane features capture fine-grained details aligning with the spectral bias mentioned earlier. This approach yields three main benefits. First, aligning with the distinct spectral biases of heterogeneous features results in a model less sensitive to hyperparameters and performance variations related to the scene. This is achieved by avoiding both underfitting and overfitting. Second, it allows for stable training through gradual changes in their spectral biases, as discussed in the works by Lin et al. (2021); Yang et al. (2023). Lastly, it facilitates efficient parameter allocation by replacing the need for a low-resolution grid with coordinate-based features.

We achieve this by implementing a residual concatenation of coordinates and multi-plane features across the first two hidden layer blocks, enhancing the efficiency in responding to the coordinates. The images generated by the proposed method exhibit enhanced clarity regarding global contexts and fewer artifacts compared to baselines, as illustrated in Figure 1b. Our extensive experiments show that the proposed method achieves comparable results of multi-plane encoding with high denoising penalties in static NeRFs. Notably, it outperforms baselines in dynamic NeRFs from the sparse inputs. To summarize, we make the following contributions:

- We prove that explicit parameterization has difficulty capturing low-frequency details, even when using multi-resolution grids or coordinate networks without precise integration.
- We validate the proposed method on static and dynamic NeRF tasks, including real-world cases with sparse inputs, while also examining how two features are separated and function independently.
- When we reduce the number of parameters, the proposed method still shows competitive performance. This results from skipping the allocation of a spatial low-resolution grid and replacing it with coordinate-based features.

2. Related Work

Coordinate-based Network and Sinusoidal Encoding In the initial studies of NeRFs, MLP networks with sinusoidal encoding were used to simultaneously describe low and high-frequency details (Mildenhall et al., 2021; Martin-Brualla et al., 2021; Barron et al., 2021; 2022). However, a classical coordinate network without this encoding was found to be biased toward lower frequencies (Rahaman et al., 2019; Yüce et al., 2022). The importance of positioning encoding and sinusoidal activation led to the fundamental exploration of the relationship between rendering performance and the frequency values of target signals (Tancik et al., 2020; Sitzmann et al., 2020; Fathony et al., 2021; Ramasinghe et al., 2022). Lindell et al. (2022) uncovered that improper high-frequency embedding results in artifacts negatively impacting reconstruction quality.

Explicit Parameterization Recent developments in explicit representations, such as voxel-grid, hash encoding, and multi-planes, have gained attention due to their fast training, rendering speed, and superior performance compared to positioning encoding-based networks (Liu et al., 2020; Sun et al., 2022; Müller et al., 2022; Chen et al., 2022; Cao & Johnson, 2023; Fridovich-Keil et al., 2023). Sun et al. (2022) introduced the direct voxel field, using minimal MLP layers to speed up training and rendering. Instant-NGP, based on hash maps, provides multi-resolution spatial features and versatility, extending beyond 3-dimensional spaces to high-resolution 2-dimensional images (Müller et al., 2022). The multi-plane approach has been highlighted for its applicability in expanding to 4-

dimensional without compromising generality, decomposing targets into multiple planes, with each plane responsible for a specific axis (Chen et al., 2022; Cao & Johnson, 2023; Fridovich-Keil et al., 2023). Specifically, while the aforementioned approaches used special on-demand GPU computations for efficiency, this method achieves comparable speed and performance based on general auto-differential frameworks. This widens its applicability to tasks like 3D object generation, video generation, 3D surface reconstruction, and dynamic NeRF (Gupta et al., 2023; Yu et al., 2023; Wang et al., 2023b; Cao & Johnson, 2023; Fridovich-Keil et al., 2023).

NeRFs in the Sparse Inputs Early efforts incorporated pre-trained networks trained on large datasets to compensate for the lack of training data (Jain et al., 2021; Yu et al., 2021; Wang et al., 2021). Another alternative approach incorporated additional information, such as depth or color constraints, to ensure the preservation of 3D coherence (Deng et al., 2022; Yuan et al., 2022; Roessle et al., 2022; Truong et al., 2023). Without the assistance of off-the-shelf models and additional, this line of works devised new regularization to train NeRFs with fewer than ten views. Reg-NeRF incorporates patch-wise geometry and appearance regularization (Niemeyer et al., 2022). This paper verified that their regularization performs well on forward-facing examples like the LLFF dataset. They did not validate object-facing scenes because this assumption demands a high correlation between adjacent views. Recently, progressively manipulating the spectrum of positioning encoding from low to high frequency proved effective in mitigating over-fitting without relying on additional information (Yang et al., 2023; Song et al., 2023). Compared to explicit representations, those still suffer from unsatisfactory visual quality, characterized by blurry boundaries. Recent studies using total variation regularization on explicit representations get rid of artifacts and construct smoother surfaces (Cao & Johnson, 2023; Fridovich-Keil et al., 2023; Sun et al., 2023). However, our findings indicate that this regularization can introduce artificial details that seem real but are not in the data. This can also result in the model failing to converge in certain scenes. We present this problem in the experiments, both qualitatively and quantitatively.

Residual Connection in NeRFs The residual connections aim to enhance the efficiency in responding to input signals (He et al., 2016). In NeRFs, several studies have adopted residual connections to preserve context from earlier stages. Shekarforoush et al. (2022) implemented these connections to accurately maintain a specific spectrum without overpowering high-frequency components. Mihajlovic et al. (2024) utilized residual connections for maintaining temporal coherence from previous frames. While aforementioned works primarily leverage MLP layers, updating Motion-Based explicit representation through residual connections in a spatio-temporal domain is also presented (Wang et al., 2023a).

A few works attempted to use explicit parameterization with the sinusoidal encoding of coordinates, but their direction differs from our method since they mainly focus on enriching available features or fewer parameterization, as well as they did not demonstrate the role of tri-planes and coordinate features (Wang et al., 2023b; Peng et al., 2023; Lee et al., 2024). In this paper, our new approach proposes incorporating two distinct features: coordinate-based and multiple-plane features. We emphasize that the disentanglement of these two heterogeneous features is crucial for reliably constructing NeRFs in sparse inputs. The proposed method performs well even with higher-dimensional cases like dynamic NeRFs and extremely limited sparse inputs.

3. Residual Neural Radiance Fields Spanning Diverse Spectrum for Sparse-Inputs

We propose a novel method that leverages multi-plane spatial features and coordinate-based networks. In sparse input NeRFs, avoiding overfitting to training data is crucial because NeRFs typically use a scheme where one network is tailored to fit a specific scene. Particularly, explicit representations based on local updates of grid structures significantly struggle with capturing global contexts. The proposed method capitalizes on a combination of distinct coordinate feature encoding techniques and multi-plane representations, which follows multi-plane representation by TensoRF and HexPlane in static and dynamic NeRFs (Chen et al., 2022; Cao & Johnson, 2023), as well as ReLU-based coordinate networks. The detailed explanation of these features is included in Appendix A.

Here, we focus on how these two features are integrated to enhance the performance of NeRFs in handling sparse input data. As shown in Figure 2, the proposed method encompasses two distinct contexts; both low and high-frequency information. When the coordinate network is used alone, the output is biased towards low frequency to facilitate global reasoning. However, when all features are engaged, it results in clear and intricate images. We illustrate the main components in the following subsection. In subsection 3.1, we delve into the proposed residual-based architecture to facilitate the disentanglement of two heterogeneous features. Moving on to subsection 3.2, we explain a curriculum weighting strategy for multi-plane features. It ensures that coordinate netework is learned first, followed by channel-wise disentanglement. It aims to provide a more diverse representation without the risk of overfitting where all channels exhibit identical expressions. Lastly, subsection 3.3 explains the loss function, which combines photometric loss and denoising multi-plane representations

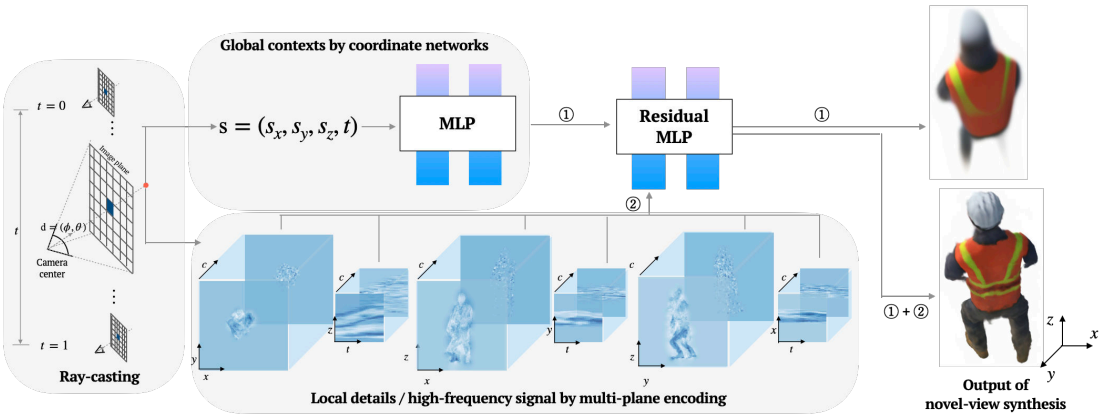


Figure 2: Conceptual illustration of the proposed method utilizing global contexts by coordinate networks and fine-grained details by multi-plane encoding. This method effectively displays two heterogeneous features. The number `1` indicates the use of coordinate network alone, while the symbol `1+2` means the use of both coordinated-based MLP network and multi-plane representation.

like Laplacian smoothing.

Nomenclature This framework considers a camera with origin o and a ray direction d. A ray $\mathbf{r}$, composed of n points, is constructed as $s_k = \mathbf{o} + \tau_k \cdot \mathbf{d}$, where $\tau_k \in \{\tau_1, \cdots, \tau_n\}$. The neural radiance field, parameterized by Θ, predicts the color and density values c^k_Θ, σ^k_Θ by volume rendering. The parameter Θ consists of MLPs including residual networks $\{\phi_l\}_L$ and multi-plane representations $\{\mathcal{M}, \mathcal{V}\}$. The feature corresponding to a ray sample s_k by multi-plane representation is denoted as f_k. For a more detailed explanation of volume rendering and multi-plane representations, please refer to Appendix A.

3.1. Architecture

In high-level context, we replace sinusoidal encoding with multi-plane encoding while employing the architecture of the original NeRF (Mildenhall et al., 2021). The schematic of our architecture is illustrated in Figure 3. A key aspect of our network architecture is the residual concatenation of coordinates value s_k and multi-plane features f_k across the first two blocks. The residual connection accelerates the efficiency in responding to input values, so the network emphasizes the importance of coordinate networks. We employ ReLU activation h to lean toward low-frequency spectral bias (Rahaman et al., 2019; Tancik et al., 2020). More specifically, the residual connection is defined as follows:

$$\begin{aligned} \phi_1(s_k, f_k) &= h(W_1^2 \cdot h(W_1^1 \cdot (s_k \oplus f_k) + b_1^1) + b_1^2) \\ \phi_2(s_k, f_k, \phi_1) &= h(W_2^2 \cdot h(W_2^1 \cdot (s_k \oplus f_k \oplus \phi_1(s_k, f_k)) + b_2^1) + b_2^2 \end{aligned} \tag{1}$$

where, $\{W_l, b_l\}_{l=1}^L$ are the weights and biases of the l-layer MLP. In the residual connection blocks, when $l \leq 2$, the block includes two pairs of weights and biases. For $l > 2$, the subsequent processes contain one pair of weights and biases. $\oplus$ indicates the concatenation of features.

The output layers use different activations, such as the softplus function for density and the sigmoid function for color. The proposed residual connection allows the network to robustly maintain low-frequency spectral bias from coordinate networks without interference from multi-plane features. Our empirical findings demonstrate that this operation promotes the disentanglement of two features, aligning with a spanning diverse spectrum. A detailed analysis of this residual connection is provided in subsection 4.5 along with the performance gain by this architecture.

3.2. Curriculum Weighting for Multi-Plane Encoding

The architecture in the proposed method performs well in scenes with mild occlusion and less dynamic motion. However, it encounters challenges in severe ill-conditioned situations, such as heavy occlusion and rapid motion, as seen in the `drums` in the static NeRF and the `standup` in the dynamic NeRF. To alleviate this issue, we propose a curriculum weighting strategy for multi-plane encoding, aiming to manipulate the engagement of multi-plane features per training step. This approach trains the coordinate-based network first, followed by the subsequent training of multi-plane features. In this subsection, we denote t as the training iteration. Technically, we introduce a weighting factor denoted as $\alpha(t)$ to control the degree of engagement of multi-plane features along multi-plane channel dimensions. Here, $f_{i,k} \in \mathbb{R}^c$ represents the output of i-th plane encoding, and the weighting factor $\gamma(t) = \{\gamma_1(t), \cdots, \gamma_c(t)\} \in \mathbb{R}^c$ is defined as follows:

$$\gamma_j(t) = \begin{cases} 0 & \text{if } \alpha(t) \leq j \\ \frac{1-\cos((\alpha(t)-j)\pi)}{2} & \text{if } 0 < \alpha(t) - j \leq 1 \\ 1 & \text{otherwise,} \end{cases} \tag{2}$$

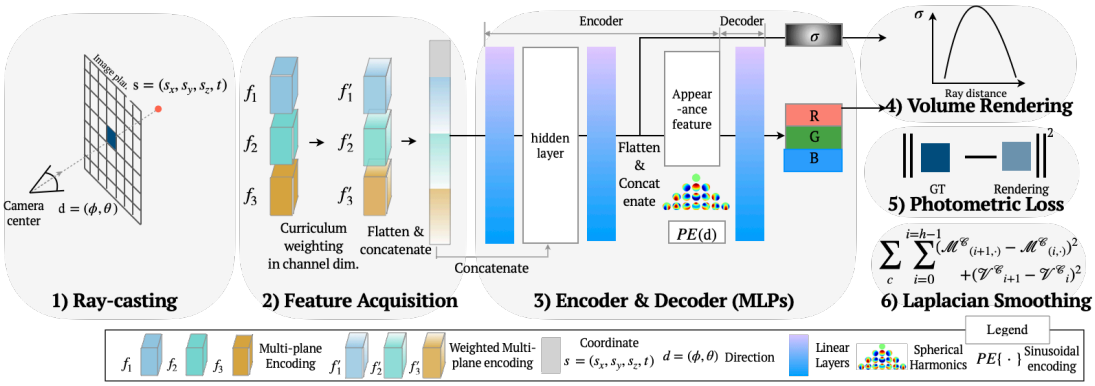


Figure 3: The schematic of the proposed method. The feature acquisition and encoder are discussed in subsection 3.1 and subsection 3.2. The loss function and regularization are described in subsection 3.3.

where, $j \in \{1, \cdots, c\}$ is the index of channel dimension and $\alpha(t) = c \cdot {}^{(t-t_s)}/_{(t_e-t_s)} \in [t_e, t_s]$ is proportional to the number of training iterations t in the scheduling interval $[t_s, t_e]$. The final features f_i' are obtained by $f_i' = f_i \odot \gamma(t)$. Hence, this weighting function is applied to each channel of multi-plane features. After reaching the last time-step of curriculum training, all channels of multi-plane features are fully engaged. It is worth noting that this weighting function is similar to those used in previous works such as (Park et al., 2021; Lin et al., 2021; Yang et al., 2023; Heo et al., 2023). However, the critical difference is a channel-wise weighting function for multi-plane features. This function can be interpreted as gradually increasing the rank of multi-plane features from the perspective of tensor decomposition (Chen et al., 2022). Our experiments find that this strategy effectively prevents all channels of multi-plane features from converging to similar patterns. It even facilitates the flat representation of specific channels when they are redundant. This results in a more diverse spectrum and mitigating overfitting issues.

3.3. Loss Function

We introduce a loss function that combines photometric loss and denoising multi-plane representations like Laplacian smoothing. First, we define the photometric loss $\mathcal{L}_p$ as mean square errors between rendered color $\hat{\mathbf{c}}(\mathbf{r})$ and ground truth pixel color $\mathbf{c}$, $\mathcal{L}_p(\Theta, \mathcal{M}, \mathcal{V}) = \sum_r \|\hat{\mathbf{c}}(\mathbf{r}; \Theta, \mathcal{M}, \mathcal{V}) - \mathbf{c}\|^2$. To tackle the ill-conditioned training problem in NeRFs arising from sparse-input situations, we apply Laplacian smoothing on both feature planes (Cao & Johnson, 2023; Fridovich-Keil et al., 2023). Laplacian smoothing $\mathcal{L}_l$ tends to excessively smooth signals, making them conform to global tendency rather than accurately local finer details (Sadhanala et al., 2017). More information can be found in subsection A.3. Additionally, we regularize each plane feature using the L1 norm for the sparsity of multi-plane features. We use, $\|\mathcal{M}\|_1$ and $\|\mathcal{V}\|_1$ as $\sum_{i=1}^{i=3} \|M_i\|_1$ and $\sum_{i=1}^{i=3} \|V_i\|_1$ respectively. The entire loss function is defined as Equation 3. The only difference in the case of static NeRF comes from the dimension of $\mathcal{V}$. Laplacian loss is not applied to $\mathcal{V}$; the rest of the details are the same as in the 4D case. The hyperparameters and implementation detail can be found in Appendix C.

$$\mathcal{L}(\Theta, \mathcal{M}, \mathcal{V}) = \mathcal{L}_p(\Theta, \mathcal{M}, \mathcal{V}) + \lambda_1 \sum_{i=1}^{3} \left(\mathcal{L}_l(M_i) + \lambda_2 \mathcal{L}_l(V_i)\right) + \lambda_3 \left(\|\mathcal{M}\|_1 + \|\mathcal{V}\|_1\right) \tag{3}$$

While increasing the value of λ_1 allows the removal of floating artifacts by over-smoothing the multi-plane features, it creates undesirable deformation that looks authentic but is not present in the training data. In addition, too high a value for λ_1 can increase learning instability due to excessive penalization. Therefore, finding a feasible weight demands extensive trial and error. However, the proposed method is less sensitive to this issue as the coordinate network itself establishes a bottom line, while the multi-plane encoding compensates for high-frequency details. We empirically validate this through our experiments.

4. Experiments

In this section, we present our experiments designed to address three pivotal questions: *1) Does existing multi-resolution parameterization and its simple integration with coordinate network adequately function low-frequency representations while producing clear scenes on sparse inputs? 2) Does current regularization consistently maintain its effectiveness across various hyper-parameters and scenes, ensuring the capture of 3D coherence on sparse input data? 3) Does the chemical integration of heterogeneous features, such as multiple planes and coordinates, substantially improve the performance of both static and dynamic NeRF?*

To answer those questions, we conduct vast experiments

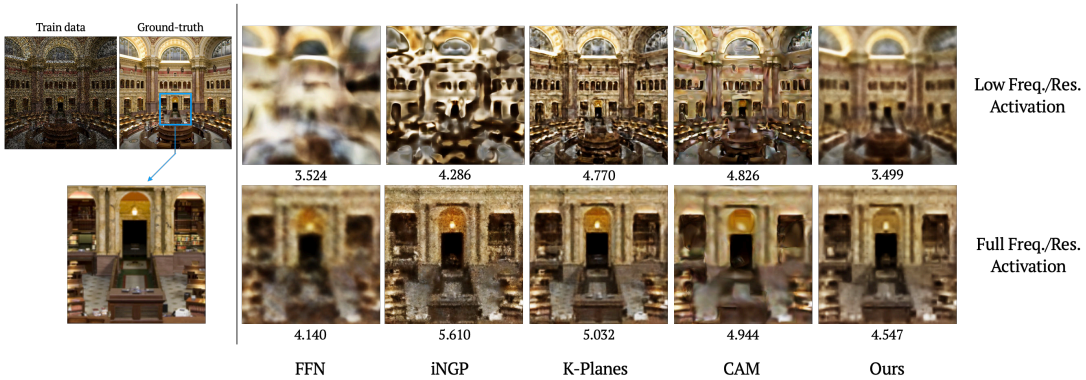


Figure 4: Qualitative results on the image regression trained with a 50% random mask. The first row displays rendered images using only low-frequency or resolution features, while the second row shows images engaged with the full range of features. The numeric value indicates the average magnitude spectrum obtained from the Fourier transform.

over scenarios of two sparse input cases: a few-shot static case and a 4-dimensional dynamic case. To provide a clearer insight into the role of low-frequency representation, we include 2D image regression as an illustrative example. We also include ablation studies to substantiate the rationale behind the architectural choices. We choose the datasets as in-ward-facing object poses, as they are more likely to be occluded by the objects from various viewing locations than forward-facing poses. For performance evaluation, we employ the PSNR metric to gauge image reconstruction quality. In addition, SSIM and LPIPS scores are reported to assess the perceptual quality of the rendered images. Further experimental details are described in Appendix D.

4.1. Motivation Example: Image Regression

We start by demonstrating the diversity in spectrum information the proposed method possesses in $\{512 \times 512\}$ Image Regression tasks. We compare our method with FFN (Tancik et al., 2020) as a sinusoidal embedding and iNGP (Müller et al., 2022), K-Planes (Fridovich-Keil et al., 2023) as explicit parameterization. We also include CAM (Lee et al., 2024), a similar approach that combines coordinate-based networks and grid-based representation for efficient parameterization. Figure 4 shows how baselines handle low and high freqeuncy or resolution features by presenting the average magnitude spectrum by Fourier feature transform. Specifically, in the case of low-frequency features or grids, FFN and CAM, which manipulate their spectrum via sinusoidal encoding, use only the lowest 26 frequencies (10%) of total frequency range. K-Planes, modifed to use four multi-scale tensorial planes, employs only use the lowest resolutional plane with a scale of 32×32. iNGP, with 16 levels of spatial hash-grid, uses only the first two lowest resolutional grids. The proposed method only utilizes the coordinate-based MLP networks with four frequeices $\{2^i | i = 0, 1, 2, 3\}$. On the other hand, as the finest feature, K-Planes, CAM and Ours all adopt a 128×128 plane.

In Figure 4, we observe that all baselines differs the average magnitude spectrum of rendered images between low and full feature engagement, though the extent of manignitude varies. Low feature engagement describes only low-frequency details, resulting in a minimum spectrum magnitude, whereas full feature engagement captures intrigate details, achieving the higest spectrum magnitude. Examining each instance, the sinusoidal method (FFN) faces an underfitting issue as it overly focuses on low frequency details. Explicit representations like iNGP and K-Planes, however, tend to interpret low resolution features focusing on high frequency details despite only low-resolution features are activated. Surprisingly, CAM, despite incoporating various spectral sinusodial embeddings, also struggles to capture low frequency details. This implies explicit representation such as grid or plane cannot effectively handle low frequency details without careful designs.

In contrast, our proposed method balances low and high frequency spectral features, resulting in images that capture both types of details. Remarkably, the rendered images encompass a substantial spectral range, varying from 3.499 to 4.547. This range stands out as the most extensive deviation from the baselines, with the exception of iNGP. While iNGP exhibits the widest spectrum among baselines, the image with low resolution features does not adequately capture global reasoning. Subsequent experiments show that the effectiveness in utilizing low-frequency context for global consistency and then transitioning to high-frequency context to capture the finest details in both static and dynamic NeRF under sparse inputs.

4.2. Static Radiance Fields

We conducted 3-dimensional static NeRF experiments on the NeRF-synthetic dataset to evaluate whether our model adequately captures both the global context of a scene and

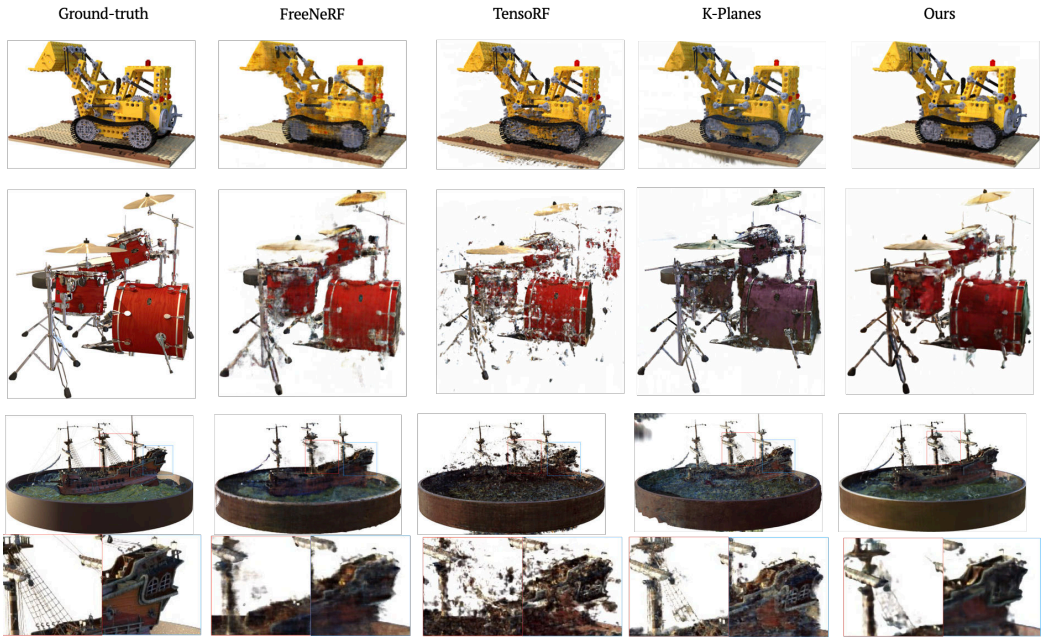


Figure 5: Rendered images of `lego`, `drums` and `ship` cases in the static NeRF dataset by FreeNeRF, TensoRF, K-Planes and ours. The rendered images are $\{83, 129, 95\}$-th in the test set, respectively.

Table 1: Result of evaluation statistics on the static NeRF datasets. We conduct five trials for each scene and report average scores. Average PSNR, SSIM, and LPIPS are calculated across all scenes. We indicates best performance as **bold** and second best as underline

Models	PSNR ↑								Avg. PSNR ↑	Avg. SSIM ↑	Avg. LPIPS ↓
	chair	drums	ficus	hotdog	lego	materials	mic	ship			
Simplified_NeRF	20.35	14.19	21.63	22.57	12.45	18.98	24.95	18.65	19.22	0.827	0.265
DietNeRF	21.32	14.16	13.08	11.64	16.12	12.20	24.70	19.34	16.57	0.746	0.333
HALO	24.77	18.67	21.42	10.22	22.41	21.00	24.94	21.67	20.64	0.844	0.200
FreeNeRF	26.08	19.99	18.43	28.91	24.12	21.74	24.89	23.01	23.40	0.877	0.121
DVGO	22.35	16.54	19.03	24.73	20.85	18.50	24.37	18.17	20.57	0.829	0.145
VGOS	22.10	18.57	19.08	24.74	20.90	18.42	24.18	18.16	20.77	0.838	0.143
iNGP	24.76	14.56	20.68	24.11	22.22	15.16	26.19	17.29	20.62	0.828	0.184
TensoRF	26.23	15.94	21.37	28.47	26.28	20.22	26.39	20.29	23.15	0.864	0.129
K-Planes	27.30	**20.43**	**23.82**	27.58	26.52	19.66	**27.30**	21.34	24.24	**0.897**	**0.085**
Ours	**28.02**	19.55	20.30	**29.25**	**26.73**	**21.93**	26.42	**24.27**	**24.56**	0.896	0.092

fine details without introducing undesirable artifacts under sparse input conditions. Consistent with prior studies such as (Jain et al., 2021; Yang et al., 2023), we trained all models with 8 views. We compare our proposed models with sinusoidal encoding methods; Simplified NeRF, DietNeRF (Jain et al., 2021), HALO (Song et al., 2023) and FreeNeRF (Yang et al., 2023) and for explicit spatial parameterization methods; DVGO (Sun et al., 2022), VGOS (Sun et al., 2023), iNGP (Müller et al., 2022), TensoRF (Chen et al., 2022) and K-Planes (Fridovich-Keil et al., 2023). For all baselines, we applied regularization techniques congruent with their inherent characteristics and configurations.

The quantitative rendering results are shown in Table 1 and Figure 5. More detailed numeric values are contained at Appendix E. First, we observed that the proposed method outperforms the previous state-of-the-art method, FreeNeRF, in terms of both PSNR and perceptual quality. Sinusoidal encoding-based networks fail to capture high-frequency details and are prone to underfit in data with high-resolution structures, (`ficus, lego`). In contrast, grid-based models show robust results in reconstructing high-frequency structures. However, for data with a strong non-Lambertian effect (`drums, ship`), grid-based models tend to miss the global shape and are prone to overfit in

Table 2: Result of evaluation statistics on the D-NeRF datasets. HexPlane employs the weight of denoising regularization as $\lambda_1 = 0.01$ via grid-search. Average PSNR, SSIM, and LPIPS are calculated across all scenes. We indicate the best performance as **bold** for each case.

Training views	Models	Avg. PSNR ↑	Avg. SSIM ↑	Avg. LPIPS ↓
15 poses	HexPlane	21.93	0.921	0.092
	K-Planes	21.50	0.922	**0.086**
	Ours	**22.30**	**0.925**	0.087
20 poses	HexPlane	23.18	0.929	0.082
	K-Planes	22.58	0.931	**0.070**
	Ours	**23.93**	**0.935**	0.072
25 poses	HexPlane	24.15	0.935	0.074
	K-Planes	22.68	0.929	0.107
	Ours	**25.34**	**0.941**	**0.063**

high frequency. Our proposed multi-plane encoding technique can exclusively capture fine-grained details while maintaining global shape learned by coordinate features, leading to more robust novel view synthesis in sparse-input scenarios. This phenomenon consistently occurs in real-world datasets. For more detail, please refer to Appendix J.

4.3. Dynamic Radiance Fields

To demonstrate the robustness of the proposed model on more spare input cases, we conduct our experiences on the dynamic NeRF dataset (Pumarola et al., 2021). This data set comprises monocular cameras of about 50-100 frames in duration and different inward-facing views for each timestep. To verify a harsh situation, we also experimented with fewer frames $\{15, 20, 25\}$ sparse in both views and time aspects. Each pose was sampled uniformly along the time axis for each scene. We compare our method with HexPlane (Cao & Johnson, 2023) and K-Planes (Fridovich-Keil et al., 2023).

The observations made in subsection 4.2 are even more evident in the dynamic NeRFs. The proposed method outperforms every setting of HexPlane in all metrics in the D-NeRFs, as shown in Table 2. HexPlane discretizes the continuous time axis into finite bins, making it less responsive to the time-variant motion of objects when the available training poses are sparse. In contrast, the proposed method can capture the time-variant motion of objects by harnessing the coordinate-based networks first, with multi-plane encoding supplementing the remaining details.

4.4. Stability in Sparse-Input NeRFs

In sparse-input NeRFs, stability is defined as the ability to counteract overfitting. We measure stability by evaluating the minimal performance discrepancy between test viewpoints adjacent to and not adjacent to the training views. Specifically, we examine the variance of PSNR across all test viewpoints in the static NeRF dataset. The total variance of PSNR across all images is calculated using 8,000 images from 8 scenes, each with 200 test viewpoints and five trials. FreeNeRF, which uses MLP and sinusoidal encoding, shows the lowest variance among baselines. Spatially explicit methods like iNGP and TensoRF exhibit significant variances due to their tendency to overfit the training views. While K-Planes reduces instability compared to these methods, its variances still do not match ours. Quantitatively, our method achieves comparable results to FreeNeRF. However, as shown in Table 3, FreeNeRF generally lacks reconstruction performance. Additionally, K-Planes struggles with reconstructing specific scenes, such as ship. In contrast, our method consistently reconstructs all scenes with high quality, avoiding significant degradation. This is also evident in Figure 5. While FreeNeRF exhibits blurry details and K-Planes displays strange color distortion, our method shows the cleanest results without noticeable distortion or artifacts. Considering that our method shows low variances and achieves the highest PSNR, we emphasize the distinction of our approach in terms of both stability and superior capability.

Table 3: Variance of PSNR(↓) on the static NeRF datasets.

FreeNeRF	iNGP	TensoRF	K-Planes	Ours
17.31	23.95	23.22	19.61	18.23

4.5. Ablation Study

To validate the effectiveness of the proposed architecture, we analyze several types of candidates with respect to residual connections. We consider three candidates: Type 1, where skip connection lies at every layer, Type 2, which has no skip connection; and the last one, where only the coordinate value s_k is residual concatenated. The quantitative result is presented in Table 4. We observe that the straightforward implementation of residual connection leads to ineffective training for sprase inputs. However, the proposed method gains remarkble performance

Table 4: Performance evaluation by varying residual connection candidates on the static NeRF dataset with 8 views

Model	Avg. PSNR ↑	Avg. SSIM ↑	Avg. LPIPS ↓
Ours	24.74	0.898	0.089
Type 1	18.77	0.844	0.179
Type 2	19.23	0.848	0.171
Type 3	19.07	0.843	0.175

Table 5: Performance evaluation of the D-NeRF dataset with training steps up to 15,000 and 25 poses. The rendering time is assessed using 20 poses.

Model **Name**	# Params [M]	Avg. PSNR	Avg. Train Time [min]	Avg. Render Time [min]
K-Planes (3*32)	18.6M	23.85	18.93	0.83
K-Planes (3*4)	1.9M	23.41	13.29	0.78
HexPlane (72)	9.7M	24.00	6.78	0.60
HexPlane (6)	0.8M	22.08	6.38	0.68
Ours (48)	3.4M	25.17	12.22	2.14
Ours (12)	1.0M	25.10	8.77	1.73

* Numbers in brackets are the channel dimensions of each multi-plane.

Table 6: Average PSNR across all scenes varying denoising regularization λ_1. The hyphen indicates not converged.

λ_1	Static NeRF (8 views)			D-NeRF (25 views)		
	TensoRF	K-Planes	ours	HexPlane	K-Planes	ours
0.0001	24.10	24.31	23.68	22.83	24.32	24.67
0.001	24.98	24.28	24.47	23.86	24.01	25.38
0.01	-	24.28	24.55	24.15	24.02	25.74
0.1	-	23.64	24.23	23.46	23.55	25.84
1.0	-	22.05	22.99	21.95	22.62	25.42

gap than others, highlighting the necessity of careful desing for handling two heterogeneous features. For more information, please refer to Appendix G. In terms of insensitivity to hyper-parameters, we evaluate explicit parameterization methods on dynamic NeRFs by reducing the channel dimension as shown in Table 5. While baselines exhibit a performance decrease, the proposed method preserves performance even with only 30% of number of parameter used. Moreover, the reduced model with only 1.0M parameters surpasses the other full parameterized baselines. This achievement is attributed to the disentanglement of two heterogeneous representations, as redundant multi-plane for low-resolution features are replaced with the coordinate network. In addition, we explore the sensitivity of regularization in Table 6. It demonstrates that the proposed method maintains near-optimal performance across all hyper parameters. In one case, TensoRF with $\lambda_1 = 0.001$ fortunately performs the best at 24.98, but it fails to converge when λ_1 exceeds 0.01. This indicates its sensitivity to regularization values. K-Plane appears to be more stable, but its overall performance lags behind the proposed method. Moreover, excessive regularization can introduce undesirable modification such as color disturbances. The detailed experimental results are included in Appendix F.

To validate the curriculum weighting, we conduct a comparison between the proposed method and the same architecture that does not utilize progressive training. We choose {`Lego`, `Drums`, `Mic`} from the static NeRF, and {`Hellwarrior`, `Lego`, `Standup`, `Trex`} from the dynamic NeRF where this weighting is applied (See Appendix C). We denote *CL* as the activation of progressive

Table 7: Quantitative results between activation and absence of curriculum weighting for multi-plane encoding. Mean and variance of PSNR are presented for each scene. We conduct three trials using random seeds to measure average PSNR and its variance

	Metric	Static NeRF			D-NeRF			
		Drums	Lego	Mic	Hell Warrior	Lego	Stand up	Trex
CL	Mean(↑)	20.20	26.84	26.61	19.46	24.05	26.22	26.70
	Variance(↓)	2.31	7.20	6.28	2.05	2.93	5.42	3.22
Non-CL	Mean(↑)	19.84	26.54	26.49	19.83	23.94	25.59	26.63
	Variance(↓)	2.55	6.26	9.16	2.80	4.15	4.99	3.40

training and *Non_CL* as its absence. In static NeRFs, we observe that *CL* consistently has a positive impact on performance improvement to Average PSNR, despite the fact that their improvement on reconstruction is minor, ranging from 0.2 to 0.4 in all cases. For variances, {`drums`, `lego`} show no significant difference, but the `Mic` result indicates that *CL* mitigates instability, with variances decreasing by 2.8. This reduction means significantly less discrepancy between images adjacent and not adjacent to training views. The effectiveness of progressive training is more pronounced in dynamic NeRFs. While it does not provide significant improvement in the `hellwarrior` case, it evidently enhances performance in the `standup` case, leading to 0.6 increase in average PSNR. In terms of stability, we observe that most cases are less sensitive to overfitting with progressive training. Although variance slightly increases in the `standup`, it is not significant issue considering average PSNR improvement. In summary, progressive training influences on either performance improvement or mitigation of instability in sparse-input NeRF by gradually engaging multi-plane channels. This allows the initial channels to learn global details, while later channels focus more on finer details. The detailed explanation, including graphics, is provided in Appendix H.

5. Conclusion

In this paper, we introduce refined tensorial radiance fields that seamlessly incorporate coordinate networks. The coordinate network enables the capture of global context, such as object shapes in the static NeRF and dynamic motions in the dynamic NeRF dataset. This property allows multi-plane encoding to focus on describing the finest details. Through extensive experiments, we demonstrate that the proposed method consistently outperforms the baselines and their regularization in the few-shot regime. Notably, the proposed method exhibits strong stability, showing less discrepancy between images adjacent and non-adjancet to training views. Additionally, it preserves performance even with a reduced number of parameters.

Impact Statement

Novel view synthesis is a task to understand the shape and appearance of objects and scenes from a sparse set of images or video. Our model, in particular, can reconstruct fine-detailed 3D shapes with an accurate appearance just from given fewer inputs, both in static and dynamic scenes.

Like previous works, our model can obtain fine reconstruction results only if sufficiently distributed views are given. Recovering high-fidelity 3D shapes and appearances of objects from fewer inputs offers numerous practical applications. However, it also introduces potential drawbacks, such as the leading to the creation of potentially misleading media or potentially facilitating design theft, by duplicating physical objects.

Reproducibility Statement

For reproducibility, our code is available at https://github.com/MingyuKim87/SynergyNeRF. Both training and evaluation codes are included for convenience. Qualitative results can be found on our project page.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [No.2022-0-00641, XVoice: Multi-Modal Voice Meta Learning]. A portion of this work was carried out during an internship at NAVER AI Lab. We also extend our gratitude to ACT-NOVA for providing the computational resources required.

References

Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., and Srinivasan, P. P. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5855–5864, 2021.

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5470–5479, 2022.

Cao, A. and Johnson, J. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 130–141, 2023.

Chan, E. R., Lin, C. Z., Chan, M. A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L. J., Tremblay, J., Khamis, S., et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16123–16133, 2022.

Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., and Su, H. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14124–14133, 2021.

Chen, A., Xu, Z., Geiger, A., Yu, J., and Su, H. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pp. 333–350. Springer, 2022.

Deng, K., Liu, A., Zhu, J.-Y., and Ramanan, D. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12882–12891, 2022.

Fathony, R., Sahu, A. K., Willmott, D., and Kolter, J. Z. Multiplicative filter networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OmtmcPkkhT.

Fridovich-Keil, S., Meanti, G., Warburg, F. R., Recht, B., and Kanazawa, A. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12479–12488, 2023.

Gupta, A., Xiong, W., Nie, Y., Jones, I., and Oğuz, B. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645. Springer, 2016.

Heo, H., Kim, T., Lee, J., Lee, J., Kim, S., Kim, H. J., and Kim, J.-H. Robust camera pose refinement for multi-resolution hash encoding. In *International Conference on Machine Learning*. PMLR, 2023.

Jain, A., Tancik, M., and Abbeel, P. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5885–5894, 2021.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

Knapitsch, A., Park, J., Zhou, Q.-Y., and Koltun, V. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017.

Lee, J. C., Rho, D., Nam, S., Ko, J. H., and Park, E. Coordinate-aware modulation for neural fields. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4UiLqimGm5.

Lin, C.-H., Ma, W.-C., Torralba, A., and Lucey, S. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5741–5751, 2021.

Lindell, D. B., Van Veen, D., Park, J. J., and Wetzstein, G. Bacon: Band-limited coordinate networks for multiscale scene representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16252–16262, 2022.

Liu, L., Gu, J., Zaw Lin, K., Chua, T.-S., and Theobalt, C. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

Martel, J. N., Lindell, D. B., Lin, C. Z., Chan, E. R., Monteiro, M., and Wetzstein, G. Acorn: adaptive coordinate networks for neural scene representation. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021.

Martin-Brualla, R., Radwan, N., Sajjadi, M. S., Barron, J. T., Dosovitskiy, A., and Duckworth, D. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7210–7219, 2021.

Mihajlovic, M., Prokudin, S., Pollefeys, M., and Tang, S. Resfields: Residual neural fields for spatiotemporal signals. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=EHrvRNs2Y0.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Müller, T., Evans, A., Schied, C., and Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4): 1–15, 2022.

Niemeyer, M., Barron, J. T., Mildenhall, B., Sajjadi, M. S., Geiger, A., and Radwan, N. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5480–5490, 2022.

Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., and Martin-Brualla, R. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5865–5874, 2021.

Peng, S., Yan, Y., Shuai, Q., Bao, H., and Zhou, X. Representing volumetric videos as dynamic mlp maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4252–4262, 2023.

Pumarola, A., Corona, E., Pons-Moll, G., and Moreno-Noguer, F. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10318–10327, 2021.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Ramasinghe, S., MacDonald, L. E., and Lucey, S. On the frequency-bias of coordinate-mlps. *Advances in Neural Information Processing Systems*, 35:796–809, 2022.

Roessle, B., Barron, J. T., Mildenhall, B., Srinivasan, P. P., and Nießner, M. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12892–12901, 2022.

Sadhanala, V., Wang, Y.-X., Sharpnack, J. L., and Tibshirani, R. J. Higher-order total variation classes on grids: Minimax theory and trend filtering methods. *Advances in Neural Information Processing Systems*, 30, 2017.

Shekarforoush, S., Lindell, D., Fleet, D. J., and Brubaker, M. A. Residual multiplicative filter networks for multiscale reconstruction. *Advances in Neural Information Processing Systems*, 35:8550–8563, 2022.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Song, L., Li, Z., Gong, X., Chen, L., Chen, Z., Xu, Y., and Yuan, J. Harnessing low-frequency neural fields for few-shot view synthesis. *arXiv preprint arXiv:2303.08370*, 2023.

Sun, C., Sun, M., and Chen, H.-T. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5459–5469, 2022.

Sun, J., Zhang, Z., Chen, J., Li, G., Ji, B., Zhao, L., and Xing, W. Vgos: Voxel grid optimization for view synthesis from sparse inputs. *arXiv preprint arXiv:2304.13386*, 2023.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33: 7537–7547, 2020.

Truong, P., Rakotosaona, M.-J., Manhardt, F., and Tombari, F. Sparf: Neural radiance fields from sparse and noisy poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4190–4200, 2023.

Wang, L., Hu, Q., He, Q., Wang, Z., Yu, J., Tuytelaars, T., Xu, L., and Wu, M. Neural residual radiance fields for streamably free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 76–87, 2023a.

Wang, Q., Wang, Z., Genova, K., Srinivasan, P. P., Zhou, H., Barron, J. T., Martin-Brualla, R., Snavely, N., and Funkhouser, T. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2021.

Wang, Y., Skorokhodov, I., and Wonka, P. Pet-neus: Positional encoding tri-planes for neural surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12598–12607, 2023b.

Yang, J., Pavone, M., and Wang, Y. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8254–8263, 2023.

Yu, A., Ye, V., Tancik, M., and Kanazawa, A. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4578–4587, 2021.

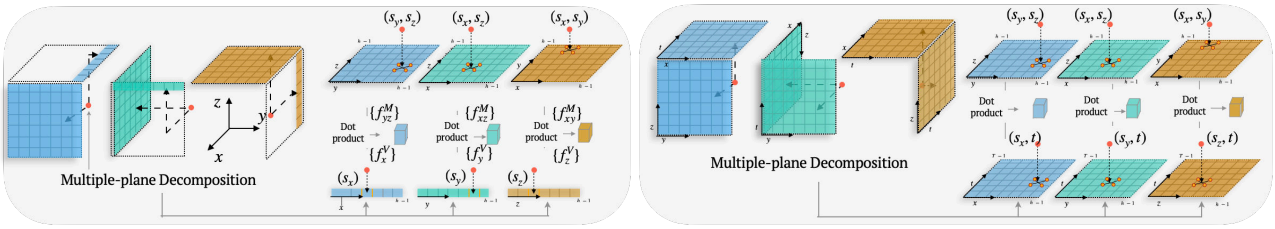
Yu, S., Sohn, K., Kim, S., and Shin, J. Video probabilistic diffusion models in projected latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18456–18466, 2023.

Yuan, Y.-J., Lai, Y.-K., Huang, Y.-H., Kobbelt, L., and Gao, L. Neural radiance fields from sparse rgb-d images for high-quality view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Yüce, G., Ortiz-Jiménez, G., Besbinar, B., and Frossard, P. A structured dictionary perspective on implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19228–19238, 2022.

A. Background

Before delving into the details of the proposed method, we briefly review the fundamentals of the neural radiance fields and multi-plane approach. We describe TensoRF (Chen et al., 2022) for the static NeRFs and HexPlane (Cao & Johnson, 2023) for the dynamic NeRFs. These methods are considered representative works in multi-plane encoding and are serve as main baselines in this paper.



(a) TensoRF (b) HexPlane

Figure A.1: The schematic of baselines that use the multi-plane encoding. (a) TensoRF employs three planes and lines (Chen et al., 2022). (b) HexPlane adopts a total of six multiple planes to include the time axis (Cao & Johnson, 2023).

A.1. Neural radiance fields

Mildenhall et al. (2021) proposed the original NeRF that uses volume rendering to compute predicted color values for novel view synthesis. In this framework, we consider a camera with origin o and a ray direction d. A ray $\mathbf{r}$, composed of n points, is constructed as $o + \tau_k \cdot d$, where $\tau_k \in \{\tau_1, \cdots, \tau_n\}$. The neural radiance field, parameterized by Θ, predicts the color and density values c^k_Θ, σ^k_Θ at each point. Using volume rendering, the predicted color value $\hat{\mathbf{c}}(\mathbf{r})$ are computed as follows; $\hat{\mathbf{c}}(\mathbf{r};\Theta) = \sum_n T_n(1 - \exp(-\sigma^k_\Theta(\tau_{k+1} - \tau_k)))c^k_\Theta$. Here, the accumulated transmittance is computed by $T_n = \exp(-\sum_{k<n} \sigma^k_\Theta(\tau_{k+1} - \tau_k))$. The network parameters Θ are trained by minimizing the phometric loss, comparing $\hat{\mathbf{c}}(\mathbf{r})$ to the ground-truth color $\mathbf{c}$.

However, raw coordinate features alone are insufficient for describing high-frequency details. To resolve this, the paper proposes sinusoidal encoding, which transform coordinates into wide-spectrum frequency components. This encoding enables the description of both low and high-frequency signals, on the other hands, training can be time-consuming since it relies on implicit learning.

A.2. TensoRF: Tensorial Radiance Fields

The tensorial radiance fields provide an explicit parameterization using multiple-plane and fewer MLP layers. Compared to other explicit parameterization (Liu et al., 2020; Sun et al., 2022; Müller et al., 2022), multi-plane parameterization efficiently proves to be efficient for 3-dimensional NeRFs, provided that the plane resolution is sufficiently high. For simplicity, we assume that multi-planes share the same dimension in height, width, and depth denoted as H. This approach employs both plane features denoted as $\mathcal{M} = \{M_{xy}, M_{yz}, M_{zx}\}$ and vector features $\mathcal{V} = \{V_z, V_x, V_y\}$. For convenience, we denote two index variables, $i \in \{xy, yz, zx\}$ for $\mathcal{M}$ and $j \in \{z, x, y\}$ for $\mathcal{V}$. The plane and vector feature is denoted as $M_i \in \mathbb{R}^{c\times H\times H}$, $V_i \in \mathbb{R}^{c\times 1\times H}$. Both plane and vector features have a channel dimensions c to represent diverse information. To calculate the feature value at a given point $s := (s_x, s_y, s_z)$, the point are projected to corresponding planes and lines, and features on the nearest vertices are bilinear interpolated, as illustrated in Figure A.1a. After obtaining the feature values from $\mathcal{M}$ and $\mathcal{V}$, denoted as $f^{\mathcal{M}} = \{f^M_{xy}, f^M_{yz}, f^M_{zx}\}$, and $f^{\mathcal{V}} = \{f^V_z, f^V_x, f^V_y\}$ and each feature $f^{\cdot}_i \in \mathbb{R}^c$, hence $f^{\mathcal{M}}, f^{\mathcal{V}} \in \mathbb{R}^{3c}$. We use element-wise multiplication on $f^{\mathcal{M}}$, $f^{\mathcal{V}}$ to get final feature $f = f^{\mathcal{M}} \odot f^{\mathcal{V}} \in \mathbb{R}^{3c}$. For a more detailed explanation of multi-plane encoding, please refer to Appendix B. TensoRF has independent multi-plane features for density and appearance. TensoRF predicts occupancy by channel-wise summation of final density features across all planes. Conversely, appearance features are concatenated and then fed into MLP layers or spherical harmonics function.

Multiple-plane encoding is mainly designed to emphasize local representation with the nearest vertices. Therefore, TensoRF proposes gradually increasing the resolutions of the learnable planes and vectors during training to address this locality. This intends the model to learn the global context at the coarser resolution and then enhance finer details at the high resolution.

A.3. HexPlane

The following work, HexPlane, extends the multi-plane approach by incorporating the time axis, enabling it to work effectively in dynamic NeRFs. To achieve this, HexPlane builds upon the line features used in TensoRF, extending them into plane features by adding a time axis. This results in six planes, three spatial planes denoted as $\mathcal{M} = \{M_{xy}, M_{yz}, M_{zx}\}$, $M_i \in \mathbb{R}^{c \times H \times H}$ and three temporal planes $\mathcal{V} = \{V_{tz}, V_{tx}, V_{ty}\}$, $V_i \in \mathbb{R}^{c \times T \times H}$ as shown in Figure A.1b. Likewise the previous subsection, we denote two index variables, $i \in \{xy, yz, zx\}$ for $\mathcal{M}$ and $j \in \{tz, tx, ty\}$ for $\mathcal{V}$. Compared to TensoRF, a key difference is that the sample $s := (s_x, s_y, s_z, t)$ includes the time variable. In dynamic NeRFs, dealing with temporal sparsity is a crucial factor for improving performance since the time axis contains relatively sparse information compared to spatial information. HexPlane addresses this challenge by employing denoising regularization, laplacian smoothing, that constrains similarity among adjacent multi-plane features. For an arbitrary plane feature P, Laplacian smoothing function $\mathcal{L}_l$ is defined as below, where h, w refer row and column indices:

$$\mathcal{L}_l(P) = \sum_c \sum_{hw} \left(\left\| P^c_{h+1,w} - P^c_{h,w} \right\|_2^2 + \left\| P^c_{h,w+1} - P^c_{h,w} \right\|_2^2 \right). \tag{A.1}$$

Specifically, HexPlane applies laplacian smoothing on both plane features but give higher priority to temporal planes. This emphasize that time information is significant for capturing dynamic motion accurately. Fundamental operations of HexPlane align with TensoRF, including the direct prediction of density values by multi-plane features and the prediction of color values by concatenating multi-plane features, which are then fed into MLP layers.

B. Multiple-plane Encoding and Concatenating Coordinate

In this subsection, we discuss the use of multiple-plane encoding. Instead of directly predicting the density function using low-rank approximation of voxel grid, as done in previous methods (Chen et al., 2022; Cao & Johnson, 2023), our focus is on creating spatial features with multiple planes. For 3-dimensional data, we denote the plane features as $M_i \in \mathbb{R}^{c \times H \times H}$, and vector features $V_i \in \mathbb{R}^{c \times 1 \times H}$. However, in the case of 4-dimensional data, V changes to plane features. Each plane and vector feature corresponds to an axis in 3-dimensional spaces, such as $\mathcal{M} = \{M_{xy}, M_{yz}, M_{xz}\}$ and $\mathcal{V} = \{V_z, V_x, V_y\}$. In 4-dimensional spaces, the same notation applies to $\mathcal{M}$, but we introduce a time axis in $\mathcal{V}$ represented as $\mathcal{V} = \{V_{zt}, V_{xt}, V_{yt}\}$. The dimensions of $M_{(\cdot)}$ and $V_{(\cdot)}$ are $H \times W$ and D, respectively. We assume that all planes and vectors have the same dimension, i.e., $H = W = D$. We use h as the all grid dimension for plane and vector features for simplicity.

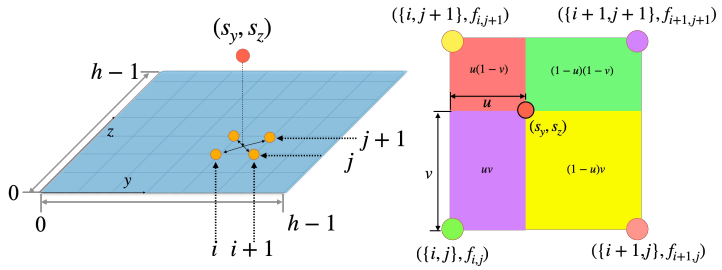


Figure B.2: Blinear interpolation

To compute multiple-plane features, we use bilinear interpolation. In 3-dimensional data, when a data point $s \in \mathbb{R}^3$ is queried, it first drops to the axis for the corresponding dimension, then looks for the nearest vertices. For example, when obtaining plane features on $M_{x,y}$, $s = (s_x, s_y, s_z)$ drops s_z and then looks for corresponding adjacent vertices in M_1. When $(i, j) = \lfloor (s_x, s_y) \rfloor$, the adjacent vertices are defined as $\{(i, j), (i+1, j), (i, j+1), (i+1, j+1)\}$, and their feature values are denoted as $\{f_{(i,j)}, f_{(i+1,j)}, f_{(i,j+1)}, f_{(i+1,j+1)}\}$ at the four nearest grid points. Here, $i, j \in \{0, 1, \cdots, h-1\}$. The component of multiple-plane encoding $f(s_x, s_y)$ is computed by bilinear interpolation as follows:

$$f_{(s_x,s_y)} = (1-u)(1-v) f_{i,j} + u(1-v) f_{i+1,j} + (1-u) v f_{i,j+1} + uv f_{i+1,j+1} \tag{B.2}$$

where, $u = {}^{(s_x - i)}/_{(i+1-i)}$ is the interpolation factor in the x-direction, and $v = {}^{(s_y - j)}/_{(j+1-j)}$ is the interpolation factor in the y-direction. The remaining components $(f_{(s_y,s_z)}, f_{(s_z,s_z)})$ are also computed by simply alternating coordinates. For the vector feature, we use linear interpolation, similar to bilinear interpolation but in 1 dimension. In 3-dimentional data, the features collected are $f^M = \{f_{(s_x,s_y)}, f_{(s_y,s_z)}, f_{(s_z,s_x)}\}$ and $f^V = \{f_{s_z}, f_{s_x}, f_{s_y}\}$, In 4-dimensional data, we can also use bilinear interpolation for $\mathcal{V}$. In this case, the features are $f^M = \{f_{(s_x,s_y)}, f_{(s_y,s_z)}, f_{(s_z,s_x)}\}$ and $f^V =$

$\{f_{(s_z,t)}, f_{(s_x,t)}, f_{(s_y,t)}\}$. Then, we combine them by element-wise producting the two vectors $f = f^M \odot f^V$ to get multiple-plane encoding in $\mathbb{R}^{3c}$.

To reprensete low-frequencies signals apparently, we include the coordinate of a data point $s = \{s_x, s_y, s_z\} \in \mathbb{R}^3$ in 3-dimensional data. In 4-dimensinoal data, these coordinate features become $s = \{s_x, s_y, s_z, t\} \in \mathbb{R}^4$. The final result of encoding is the concatenation of two different features: $\mathbf{f} = \{f, s\}$. For 3-dimensional data, $\mathbf{f}$ is in $\mathbb{R}^{3+3c}$, and in case of 4-dimensional data, $\mathbf{f}$ is in $\mathbb{R}^{4+3c}$.

C. Implementation details

C.1. Hyper-parameters on the Static NeRF

The proposed model incorporates multi-plane encoding and MLPs with skip connections. For the multi-plane encoding, we utilize 48-dimensional channels. The resolution of plane features is upscaled to 8,000,000 (200^3) by the end of training. The hyper-parameters, such as the weight for Laplacian smoothing (λ_1), the curriculum learning schedule, and the initial feature resolution, vary across scenes as part of our hyperparameter tuning to achieve optimal results. However, we use same $\lambda_{2,3}$ across scenes. In this dataset, $\lambda_2 = 1$ is used, and λ_3 is initially set to 0.00008 at the start of training, increasing to 0.00004. This is the same weighting strategy used in the previous method (Chen et al., 2022). Detailed information on the hyperparameters for multi-plane encoding can be found in Table C.1. For the decoder, we employ standard fully connected layers with ReLU activations, each containing 256 channels. The encoder consists of four fully connected ReLU layers, with a skip connection introduced after the second layer, which concatenates the fused input features. Occupancy is directly calculated from the obtained features with `softplus` function applied to the first channel. The RGB decoder, following this, consists of two layers. The color values are obtained from the features processed by the RGB decoder through `sigmoid` activation.

In our experiments, the model was trained over 30,000 iterations with a batch size of 4,096. We utilized the Adam optimizer (Kingma & Ba, 2015) with an initial learning rate of 0.02 for multi-plane features and 0.001 for MLPs, following a learning rate schedule inspired by TensorRF (Chen et al., 2022).

Table C.1: The detailed configuration for the static NeRF experiments. The parameters of curriculum $\{t_e, t_s\}$ are defined in Equation 2. These values are presented as a percentage of the total iteration. The hyphen means that curriculum learning does not apply.

Configs	scenes							
	chair	drums	ficus	hotdog	lego	materials	mic	ship
λ_1	0.001	0.005	0.005	0.009	0.009	0.001	0.009	0.005
curriculum learning	-	{5, 95}	-	-	{10, 50}	-	{0, 50}	-
Initial resolution	16	3	3	24	48	48	48	3

C.2. Hyper-parameters on the Dynamic NeRF

The configuration for the dynamic Neural Radiance Fields (NeRF) case adheres to the same settings as the static case. We utilize plane features with 48 channels. The initial voxel resolution is set at 4,096 (16^3) and is subsequently upscaled to 8,000,000 (200^3). For hyper-parameters, λ_2 is set to 2.5 for all scenes in the dynamic NeRFs to ensure smoother multi-plane features along the time-axis. This approach is proven effective in previous work (Cao & Johnson, 2023). The parameter λ_3 is set to 0.00001 across all dynamic NeRF scenes. For a more detailed description, please refer to Table C.2. The structure of the decoder, initial learning rate, and optimizer configuration remain identical to those used in the static NeRF. Any configurations not specified here follow directly from the HexPlane method as described in (Cao & Johnson, 2023).

D. Experimental Setup

We conducted the training and evaluation of all models using an NVIDIA A6000 with 48 GB of memory. It's important to note that each experiment was executed once using the seed 0 as the default. When an experiment explicitly demanded five

Table C.2: The detailed configuration for the static NeRF experiments. The parameters of curriculum $\{t_e, t_s\}$ are defined in Equation 2. These values are presented as a percentage of the total iteration. The hyphen means that curriculum learning does not apply.

Configs	scenes							
	boundingballs	hellwarrior	hook	jumpingjacks	lego	mutant	standup	trex
λ_1	0.001	0.005	0.001	0.001	0.05	0.001	0.05	0.05
curriculum learning	-	{5, 95}	-	-	{5, 95}	-	{5, 95}	{5, 95}

trials, we utilized five different seeds: {0, 700, 19870929, 20220401, 20240507}. For further details regarding the datasets and the baselines, we provide additional explanations in the following subsection.

D.1. Datasets

NeRF blender dataset The Blender Dataset (Mildenhall et al., 2021) is a set of synthetic, bounded, 360°, in-ward facing multi-view images of static object. Blender Dataset includes eight different scenes. Following the previous method(Yang et al., 2023; Jain et al., 2021), for training, we used 8 views with IDs of 26, 86, 2, 55, 75, 93, 16, 73 and 8 counting from zeros. While previous works uniformly sampled 25 images from the original test set (Yang et al., 2023; Jain et al., 2021), we evaluate all data using full-resolution images (800 × 800 pixels) for both training and testing. We downloaded Blender dataset from `https://www.matthewtancik.com/nerf`

D-NeRF dataset The D-NeRF dataset is a set of synthetic, bounded, 360 degree, monocular videos for dynamic objects (Pumarola et al., 2021). The D-NeRF dataset includes eight different scenes of varying duration, from 50 frames to 200 frames. To train the baseline under severe sparsity settings, we sub-sample the number of training views from the original D-NeRF dataset. For instance, in the case of `bouncingballs` that originally contains 150 views in the training set, we select a total of 25 views, evenly spaced apart, by starting from 0 and increasing by 6 at each step. For other scenes and varying number of views, we apply the same sampling method. We downloaded D-NeRF dataset from `https://github.com/albertpumarola/D-NeRF`

Tank and Temples The Tank and Temples dataset includes real-world scenes and corresponding multi-view images of static objects (Knapitsch et al., 2017). In this study, we select four scenes, `Family`, `Barn`, `Truck` and `Caterpillar`. Each scene exhibits variations in the number of training poses and images, reflecting different camera distribution. For instance, some poses are placed close to the object, while others are farther away, creating varying levels of difficult. Among scenes, the `Family` is relatively similar to the original in-ward case. The dataset is obtained from the following URL: `https://dl.fbaipublicfiles.com/nsvf/dataset/TanksAndTemple.zip`

D.2. Baselines

In this chapter, we briefly explain the method we compared as a baseline in our experiments. Regarding TensorRF and Hexplane, we described in detail in Appendix A.

Diet-NeRF Diet-NeRF is a sinusoidal encoding based model (Jain et al., 2021). The model incorporates auxiliary semantic consistency loss which leverages the pre-trained CLIP network trained on large datasets to compensate for the lack of training data. Auxiliary semantic consistency loss regularize semantic similarity between rendered view and given input images. We also compare the simplified NeRF (Jain et al., 2021). For implementation we used the codebase in `https://github.com/ajayjain/DietNeRF`

Free-NeRF Free-NeRF is a model based on sinusoidal encoding (Yang et al., 2023). Yang et al. (2023) employed progressive activation of positioning embedding within a single model. It initially establishes global contextual shape and subsequentially describes fine-grained details. To reduce floating artifacts, it penalize near-camera density values, following the prior knowledge of object is located in a distance to the camera. For implementation we used the code from `https://github.com/Jiawei-Yang/FreeNeRF/tree/main`

DVGO DVGO is a model that uses a three-dimensional dense voxel feature grid (Sun et al., 2022). It utilizes independent voxel features for density and color. Shallow MLP follows color encoding. In the first stage, coarse geometry explores learning the shape prior of the scene and finding empty voxels. Subsequently, in the fine reconstruction stage, they upsample the grid to a higher resolution and apply free-space skipping to optimize the occupied section densely. We used the code from https://github.com/sunset1995/DirectVoxGO

Instant-NGP Instant NGP model expresses the voxel feature grid using the Hash function (Müller et al., 2022). It allocates features corresponding to each voxel to the hash table, reducing the memory required while allowing collisions. Instant NGP utilizes the multi-resolution feature grid and uses features of resolution that log-scale uniformly increase from 16 to 1024-4096. It maintains a fast speed by inferring empty spaces through occlusion values such as TensorRF and DVGO and avoiding sampling from void regions. We used the code from https://github.com/kwea123/ngp_pl

VGOS VGOS is the first example of applying the grid-based method to a few-shot case (Sun et al., 2023). The method induces smoothness by adding total variation regularization to the dense grid feature, feature, depth, and color. In addition, progressive voxel sampling is introduced to prevent floating artifacts under the assumption that there will be a lot in the middle of the occlusion. We follows the code from https://github.com/SJoJoK/VGOS

K-Planes K-planes utilizes the Hadamard product of multi-resolution tri-planes to represent voxel features (Fridovich-Keil et al., 2023). This approach extends from static three-dimensional scenes to dynamic four-dimensional NeRFs like Hex-plane (Cao & Johnson, 2023). K-Planes incorporates TV Loss and employs various regularization methods including distortion loss to reduce floating point artifacts. Furthermore, it adopts the proposal network method suggested in MipNeRF 360 as a sampling approach (Barron et al., 2022). We follow the code from https://github.com/sarafridov/K-Planes

CAM CAM, coordinate-award modulation, is an approach to create parameter efficient neural fields (Lee et al., 2024). It combines explicit representation, such as hash-grid, with a coordinate network. Unlike previous explicit representation that had a large number of parameters due to thier channel dimension. CAM utilizes only two channel explicit representation. Each channel is integrated using an affine transformation with the feature from coodinate network. Additionally, a fourier feature network is used to reduce the number of overall parameters while preserving performance under full poses. However, CAM does not specifically address spectral bias for each feature and sparse input situation. We follow the code from https://github.com/maincold2/CAM

E. The Evaluation Statistics of Static and Dynamic NeRF Datasets

Static NeRFs In Table E.3 through Table E.5, we present the quantitative results for each scene of the synthetic NeRF Dataset. All reported numbers are averages of five experiments, along with their corresponding standard deviations. Our model consistently outperformed all counterpart models across all metrics. We also analyze the performance of the TensorRF model, which incorporates intense additional Laplacian smoothness loss. The optimal λ_1 value of 0.001 was identified for achieving the best results, as detailed in Appendix F. TensorRF with strong Laplacian regularization shows performance comparable to our proposed model. Both two methods exhibit complementary advantages in novel-view rendering results. For qualitative comparison, we showcase novel-view renderings of `ship` (Figure 5). While TensorRF with $\lambda_1 = 0.001$ focuses on reconstructing higher-frequency textures, it shows instability in low-frequency information, such as geometry (deck in `ship`), and displays high-frequency artifacts in color (water regions in `ship`). Conversely, the proposed method excels in robust optimization, particularly in capturing global information. It enables more accurate 3D geometry and consistent color reconstruction across views. However, it might underfits in scenarios that require intricate details. Despite this, without relying heavily on denoising regularization, the proposed method nearly achieves the best performance, primarily attributed to the coordinate-based networks responsible for capturing the global context.

Dynamic NeRFs In the evaluation of Dynamic Neural Radiance Fields (D-NeRF), the experimental results demonstrate a significant performance improvement for the proposed method over baseline approaches. While baselines perform comparably when full poses are available, the proposed method particularly excels as the number of available poses diminishes. This is evident when testing all methods with 25 poses, where a notable performance gap is observed. This gap narrows with a decrease in pose availability, highlighting the challenges of capturing object movement and synthesizing novel views in dynamic scenes with limited data, especially with only {15, 20} frames. Specifically, in scenes with significant move-

Table E.3: The result of average PSNR in the static NeRF. We conduct five trials and use 8 views for training.

Models	PSNR ↑							
	chair	drums	ficus	hotdog	lego	materials	mic	ship
Simplified_NeRF	20.354 ±0.648	14.188 ±2.596	21.629 ±0.171	22.565 ±1.055	12.453 ±3.103	18.976 ±2.306	24.950 ±0.202	18.648 ±0.446
DietNeRF	21.323 ±2.478	14.156 ±5.143	13.082 ±3.892	11.644 ±6.753	16.120 ±7.121	12.200 ±7.343	24.701 ±1.222	19.342 ±4.033
HALO	24.765 ±0.280	18.674 ±0.226	21.424 ±0.204	10.220 ±0.388	22.407 ±1.997	20.996 ±0.032	24.937 ±0.078	21.665 ±0.229
FreeNeRF	26.079 ±0.545	19.992 ±0.050	18.427 ±2.819	28.911 ±0.232	24.121 ±0.633	21.738 ±0.085	24.890 ±1.733	23.011 ±0.148
DVGO	22.347 ±0.253	16.538 ±0.081	19.032 ±0.071	24.725 ±0.241	20.845 ±0.129	18.497 ±0.077	24.373 ±0.252	18.170 ±0.148
VGOS	22.100 ±0.036	18.568 ±0.112	19.084 ±0.061	24.736 ±0.073	20.895 ±0.073	18.418 ±0.036	24.180 ±0.148	18.155 ±0.060
iNGP	24.762 ±0.169	14.561 ±0.082	20.678 ±0.415	24.105 ±0.308	22.222 ±0.076	15.159 ±0.075	26.186 ±0.159	17.288 ±0.135
TensoRF	26.234 ±0.062	15.940 ±0.369	21.373 ±0.152	28.465 ±0.387	26.279 ±0.279	20.221 ±0.109	26.392 ±0.320	20.294 ±0.359
TensorRF($\lambda_1 = 0.001$)	28.527 ±0.208	19.626 ±0.134	21.963 ±0.217	29.373 ±0.218	29.441 ±0.270	21.911 ±0.087	26.998 ±0.325	22.837 ±0.717
K-Planes	27.300 ±0.192	20.427 ±0.153	23.820 ±0.215	27.576 ±0.254	26.520 ±0.262	19.661 ±0.178	27.297 ±0.144	21.337 ±0.240
Ours	28.021 ±0.143	19.550 ±0.587	20.301 ±0.258	29.247 ±0.656	26.725 ±0.565	21.927 ±0.114	26.416 ±0.199	24.266 ±0.163

Table E.4: The result of average SSIM in the static NeRF. We conduct five trials and use 8 views for training.

Models	SSIM ↑							
	chair	drums	ficus	hotdog	lego	materials	mic	ship
Simplified_NeRF	0.852 ±0.003	0.773 ±0.017	0.871 ±0.002	0.891 ±0.004	0.738 ±0.031	0.827 ±0.019	0.931 ±0.001	0.736 ±0.005
DietNeRF	0.857 ±0.025	0.716 ±0.133	0.653 ±0.123	0.705 ±0.111	0.709 ±0.148	0.662 ±0.166	0.933 ±0.011	0.731 ±0.043
HALO	0.883 ±0.001	0.822 ±0.003	0.877 ±0.002	0.806 ±0.064	0.827 ±0.032	0.847 ±0.003	0.931 ±0.000	0.763 ±0.001
FreeNeRF	0.908 ±0.003	0.852 ±0.001	0.866 ±0.008	0.942 ±0.002	0.871 ±0.003	0.862 ±0.001	0.935 ±0.010	0.778 ±0.003
DVGO	0.860 ±0.003	0.761 ±0.002	0.857 ±0.001	0.904 ±0.002	0.820 ±0.001	0.804 ±0.002	0.933 ±0.001	0.689 ±0.003
VGOS	0.857 ±0.001	0.834 ±0.001	0.859 ±0.000	0.905 ±0.000	0.824 ±0.000	0.804 ±0.001	0.932 ±0.001	0.686 ±0.001
iNGP	0.899 ±0.002	0.730 ±0.002	0.886 ±0.004	0.904 ±0.001	0.841 ±0.001	0.748 ±0.002	0.946 ±0.001	0.672 ±0.002
TensoRF	0.919 ±0.001	0.753 ±0.007	0.882 ±0.002	0.938 ±0.002	0.909 ±0.003	0.843 ±0.003	0.947 ±0.002	0.719 ±0.006
TensorRF($\lambda_1 = 0.001$)	0.943 ±0.001	0.856 ±0.004	0.901 ±0.001	0.945 ±0.001	0.941 ±0.002	0.873 ±0.001	0.955 ±0.002	0.772 ±0.006
K-Planes	0.935 ±0.001	0.869 ±0.002	0.925 ±0.001	0.949 ±0.001	0.921 ±0.002	0.850 ±0.001	0.958 ±0.001	0.767 ±0.003
Ours	0.931 ±0.001	0.860 ±0.011	0.881 ±0.002	0.948 ±0.003	0.914 ±0.005	0.879 ±0.001	0.949 ±0.001	0.802 ±0.002

Table E.5: The result of average LPIPS in the static NeRF. We conduct five trials and use 8 views for training.

Models	LPIPS ↓							
	chair	drums	ficus	hotdog	lego	materials	mic	ship
Simplified_NeRF	0.247 ±0.010	0.388 ±0.083	0.153 ±0.007	0.239 ±0.009	0.408 ±0.091	0.205 ±0.042	0.100 ±0.001	0.375 ±0.005
DietNeRF	0.177 ±0.051	0.382 ±0.253	0.447 ±0.201	0.539 ±0.225	0.339 ±0.254	0.426 ±0.282	0.079 ±0.021	0.278 ±0.069
HALO	0.134 ±0.003	0.234 ±0.012	0.109 ±0.012	0.417 ±0.113	0.149 ±0.066	0.167 ±0.012	0.098 ±0.004	0.290 ±0.007
FreeNeRF	0.101 ±0.005	0.142 ±0.003	0.138 ±0.068	0.069 ±0.001	0.092 ±0.003	0.107 ±0.002	0.094 ±0.029	0.228 ±0.003
DVGO	0.120 ±0.004	0.218 ±0.003	0.102 ±0.001	0.106 ±0.003	0.125 ±0.001	0.149 ±0.001	0.062 ±0.001	0.276 ±0.004
VGOS	0.124 ±0.001	0.201 ±0.002	0.100 ±0.001	0.104 ±0.001	0.123 ±0.000	0.148 ±0.001	0.063 ±0.001	0.278 ±0.001
iNGP	0.098 ±0.004	0.345 ±0.005	0.099 ±0.006	0.144 ±0.003	0.127 ±0.002	0.292 ±0.003	0.058 ±0.002	0.312 ±0.003
TensoRF	0.074 ±0.002	0.312 ±0.011	0.105 ±0.003	0.072 ±0.005	0.059 ±0.002	0.129 ±0.004	0.047 ±0.002	0.237 ±0.010
TensorRF($\lambda_1 = 0.001$)	0.047 ±0.001	0.132 ±0.009	0.066 ±0.001	0.050 ±0.001	0.037 ±0.002	0.069 ±0.001	0.037 ±0.001	0.186 ±0.007
K-Planes	0.052 ±0.002	0.107 ±0.005	0.061 ±0.002	0.054 ±0.001	0.051 ±0.002	0.116 ±0.003	0.036 ±0.001	0.199 ±0.005
Ours	0.078 ±0.001	0.139 ±0.022	0.082 ±0.003	0.064 ±0.005	0.057 ±0.005	0.067 ±0.002	0.059 ±0.001	0.191 ±0.004

ment, such as `bouncingballs` and `standup`, the proposed method significantly outperforms others. For example, as depicted in Figure E.3, while variants of HexPlane and K-Planes struggle to accurately render the shape of the blue ball over time, the proposed method successfully captures this detail, including the reflection on the green ball. In the `jumpingjack` sequence, the proposed method also shows fewer artifacts and maintains scene boundaries more effectively compared to HexPlane. Overall, as indicated in Table E.6, the dynamic NeRF dataset demands a model capable of handling time in a continuous manner. Traditional grid-type explicit representations fall short as they rely on discretizing each feature, including time. In contrast, the proposed method leverages a coordinate network that consists of continuous maps, enhanced by multi-plane representations, enabling superior performance on the D-NeRF dataset compared to other baselines.

Variance of PSNR on the static NeRF datasets. We elaborate on the variance of PSNR for each instance in the static NeRF dataset in Table E.7. Specifically, we examine the variance of PSNR across all test viewpoints in the static NeRF dataset. The total variance of PSNR across all images is calculated using 8,000 images from 8 scenes, each with 200 test

Table E.6: Result of evaluation statistics on the D-NeRF datasets. HexPlane employs the weight of denoising regularization as $\lambda_1 = 0.01$ via grid-search. Average PSNR, SSIM, and LPIPS are calculated across all scenes. We indicates best performance as **bold** for each cases

Training views	Models	PSNR ↑								Avg. PSNR ↑	Avg. SSIM ↑	Avg. LPIPS ↓
		bouncingballs	hellwarrior	hook	jumpingjacks	lego	mutant	standup	trex			
15 views	HexPlane	26.56	15.91	**21.03**	20.35	**23.64**	**23.40**	21.48	23.05	21.93	0.921	0.092
	K-Planes	24.10	15.88	19.59	20.97	23.55	22.21	20.63	**25.08**	21.50	0.922	**0.086**
	Ours	**28.09**	**16.48**	20.90	**21.51**	23.54	23.38	**21.87**	24.88	**22.30**	**0.925**	0.087
20 views	HexPlane	28.45	16.85	22.30	20.87	**23.73**	25.02	**23.73**	24.45	23.18	0.929	0.082
	K-Planes	25.43	17.25	21.07	21.40	23.12	25.01	21.01	25.84	22.58	0.931	**0.070**
	Ours	**31.15**	**17.99**	**22.67**	**22.58**	23.49	**25.86**	23.55	**26.04**	**23.93**	**0.935**	0.072
25 views	HexPlane	30.49	17.61	23.10	22.85	24.29	25.81	23.74	25.30	24.15	0.935	0.074
	K-Planes	28.29	9.18*	22.01	22.49	**24.33**	26.02	22.77	**26.37**	22.68	0.929	0.107
	Ours	**34.61**	**19.21**	**23.82**	**24.46**	23.78	**26.75**	**26.07**	26.29	**25.34**	**0.941**	**0.063**
Full views	HexPlane	39.21	23.92	27.97	30.53	24.74	32.19	**33.09**	30.02	30.15	0.964	0.039
	K-Planes	39.76	24.57	28.10	31.07	**25.13**	**32.42**	32.99	**30.25**	**30.54**	**0.967**	**0.033**
	Ours	**40.25**	**24.63**	**28.50**	**31.70**	25.09	31.19	31.45	29.76	30.20	0.960	0.049

* indicates the model does not converge

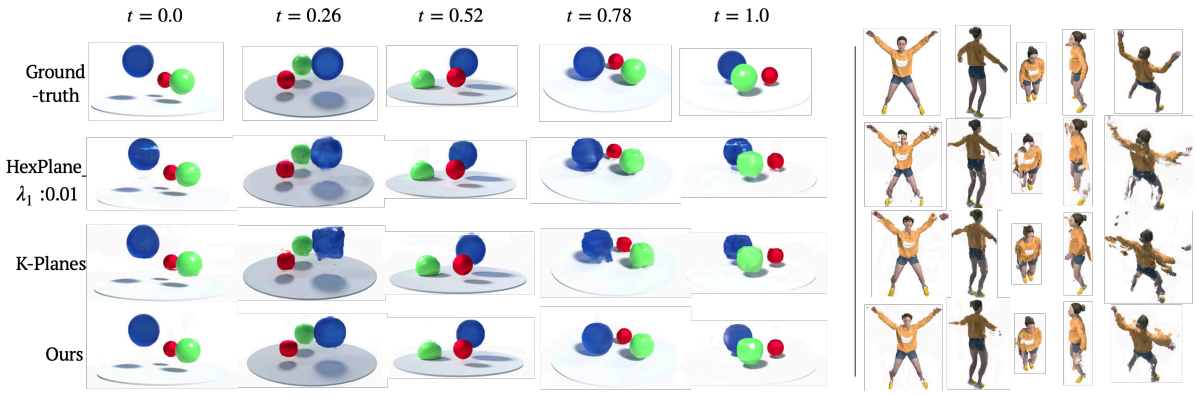


Figure E.3: Rendered images of the `bouncingballs` and `jumpingjacks` in the dynamic NeRF dataset by HexPlane with $\lambda_1 = 0.01$, K-Planes and ours. All models are trained using 25 views

viewpoints and five trials.

Examining Table E.7, while FreeNeRF records the lowest variance across nearly all scenes except for Ficus and Mic, it consistently lags behind the top score across all scenes. This observation highlights FreeNeRF tends to underfit. Notably, in scenes like Ficus and Mic where caputring intricate and delicate signals such as thin leaves or mesh structures is crucial, FreeNeRF struggles. This suggest that FreeNeRF is stable, but it faces difficulties in effectively rendering complex details effectively. On the other hand, methods like iNGP and TensoRF exhibit poor average PSNR scores such as Drums and Ship, and achieve low variances. This pattern indicates a failure to learn effectively, resulting in meaningless output and negligible difference between training and test views due to overfitting. Although these methods sometimes perform well in scenes with simple geometry like Hotdog and Lego, they often lack stability and underperform in scenarios requiring sparse input handling for NeRF applications.

However, as stated in subsection 4.4, our method consistently shows low variances in most scenes—except for hotdog and materials, where we achieve the highest PSNR—we emphasize the distinction of our approach in terms of both stability and superior capability compared to K-Planes.

Table E.7: Variance of PSNR(↓) on the static NeRF dataset.

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Total
FreeNeRF	5.07	1.72	8.72	11.16	6.42	2.13	17.28	6.48	17.31
iNGP	8.43	3.13	1.37	12.39	7.78	2.96	9.02	6.03	23.95
TensoRF	10.88	2.86	2.17	13.11	10.27	2.82	8.06	5.71	23.22
K-Planes	10.74	2.55	2.83	27.19	10.76	2.99	9.23	11.48	19.61
Ours	3.82	2.82	2.00	16.38	8.72	4.09	9.68	6.01	18.23

F. Experiments on Varying Denoising Hyper-parameter λ_1

We evaluated the role of Total Variation or Laplacian smoothing regularization for TensoRF, HexPlane, and the proposed method by incrementally increasing the regularization parameter λ_1 from 0.0001 to 1.0, each step multiplying by a factor of 10. The results, displayed in Table F.8, show that our proposed method outperforms all scenarios in both static and dynamic NeRF datasets, except at $\lambda_1 = 0.001$ in the static dataset. At this value, TensoRF achieved the highest PSNR of 24.98, but it struggled to converge at higher λ_1 values, highlighting its sensitivity and difficulties in training robustly with varying regularization strengths. For dynamic NeRF datasets, where time sparsity presents additional challenges, HexPlane's performance varied between PSNR scores of 21.95 to 24.15, whereas our method ranged from 24.67 to 25.74, indicating a lesser dependency on denoising regularization. This suggests that the coordinate networks used in our method provide robust regularization for multi-plane encoding, reducing the necessity for intensive hyperparameter tuning across different scenes. Furthermore, excessive regularization led to undesirable modifications such as color disturbances in TensoRF's rendering of the `ship` scene with $\lambda_1 = 0.001$, as evidenced by our results. Our method, on the other hand, maintained near-optimal performance across a broad range of λ_1 values without necessitating excessive denoising regularization, thanks to its ability to capture global contexts through coordinate-based networks.

To provide deeper analysis, Figure F.4 qualitatively illustrates how the dependency of TensoRF, K-Planes and the proposed model on the denoising weight affects performance. While TensoRF could reduce floating artifacts with appropriate denoising levels, excessive regularization led to unwanted color distortions, necessitating a delicate balance in regularization weight tuning. In contrast, our model displayed consistent performance across various λ_1 settings without introducing undesigned artifacts, even as denoising intensity increased. This capability stems from the model's reliance on coordinate features that anchor low-frequency information, providing a stable base for robust reconstructions. In dynamic scenes, as input sparsity increases, Table F.9 and Figure F.5 affirm that denoising regularization alone is insufficient in both HexPlane and K-Planes. For example, they exhibit degraded performance compared to our model and the optimal λ_1 values for each model fluctuate across scenes. This underscores a heavy reliance on regularization. Conversely, our proposed model demonstrated high adaptability and robustness regardless of the regularization intensity, underscoring that our feature-fusion strategy is inherently robust against sparse inputs. Consequently, regularization synergizes with our model design, aiding in more realistic rendering without producing undesired artifacts and effectively handling sparse input cases.

Table F.8: The comparison of Ours, K-Planes and TensoRF in the static NeRF dataset. We conduct experiments varying the value of λ_1. All models are trained using 8 views. The hyphen means that the model is not converged.

Models	PSNR ↑								Avg. PSNR ↑	Avg. SSIM ↑	Avg. LPIPS ↓
	chair	drums	ficus	hotdog	lego	materials	mic	ship			
TensoRF ($\lambda_1 = 0.0001$)	27.15	16.85	21.84	29.35	28.03	21.41	26.99	21.17	24.10	0.880	0.103
TensoRF ($\lambda_1 = 0.001$)	28.24	19.94	21.94	29.46	29.04	22.03	26.62	22.58	24.98	0.898	0.078
TensoRF ($\lambda_1 = 0.01$)	27.97	20.04	-	29.22	28.93	21.98	-	23.24	-	-	-
TensoRF ($\lambda_1 = 0.1$)	-	19.80	-	28.12	27.11	21.37	-	21.93	-	-	-
TensoRF ($\lambda_1 = 1.0$)	-	-	-	25.97	24.55	19.36	-	22.24	-	-	-
K-Planes ($\lambda_1 = 0.0001$)	27.16	20.50	23.82	27.75	26.29	19.87	27.46	21.68	24.31	0.897	0.083
K-Planes ($\lambda_1 = 0.001$)	27.08	20.20	23.26	27.94	27.06	20.02	26.76	21.94	24.28	0.900	0.081
K-Planes ($\lambda_1 = 0.01$)	27.10	20.27	22.62	27.64	26.48	20.59	27.08	22.46	24.28	0.899	0.082
K-Planes ($\lambda_1 = 1.0$)	23.54	17.53	22.31	27.08	26.01	19.74	26.46	21.93	23.64	0.893	0.090
K-Planes ($\lambda_1 = 0.1$)	25.98	19.60	20.72	26.11	24.15	19.09	24.56	20.73	22.05	0.876	0.112
Ours ($\lambda_1 = 0.0001$)	27.79	17.67	19.30	28.62	24.81	21.49	26.16	23.57	23.68	0.884	0.111
Ours ($\lambda_1 = 0.001$)	27.94	19.04	20.07	29.13	27.26	21.85	26.93	23.55	24.47	0.893	0.091
Ours ($\lambda_1 = 0.01$)	27.61	19.21	20.17	29.51	27.31	21.55	26.74	24.27	24.55	0.895	0.098
Ours ($\lambda_1 = 0.1$)	27.07	19.60	20.55	29.09	25.43	22.50	26.13	23.56	24.23	0.889	0.108
Ours ($\lambda_1 = 1.0$)	25.12	17.99	19.89	27.64	22.74	21.98	25.55	23.05	22.99	0.876	0.136

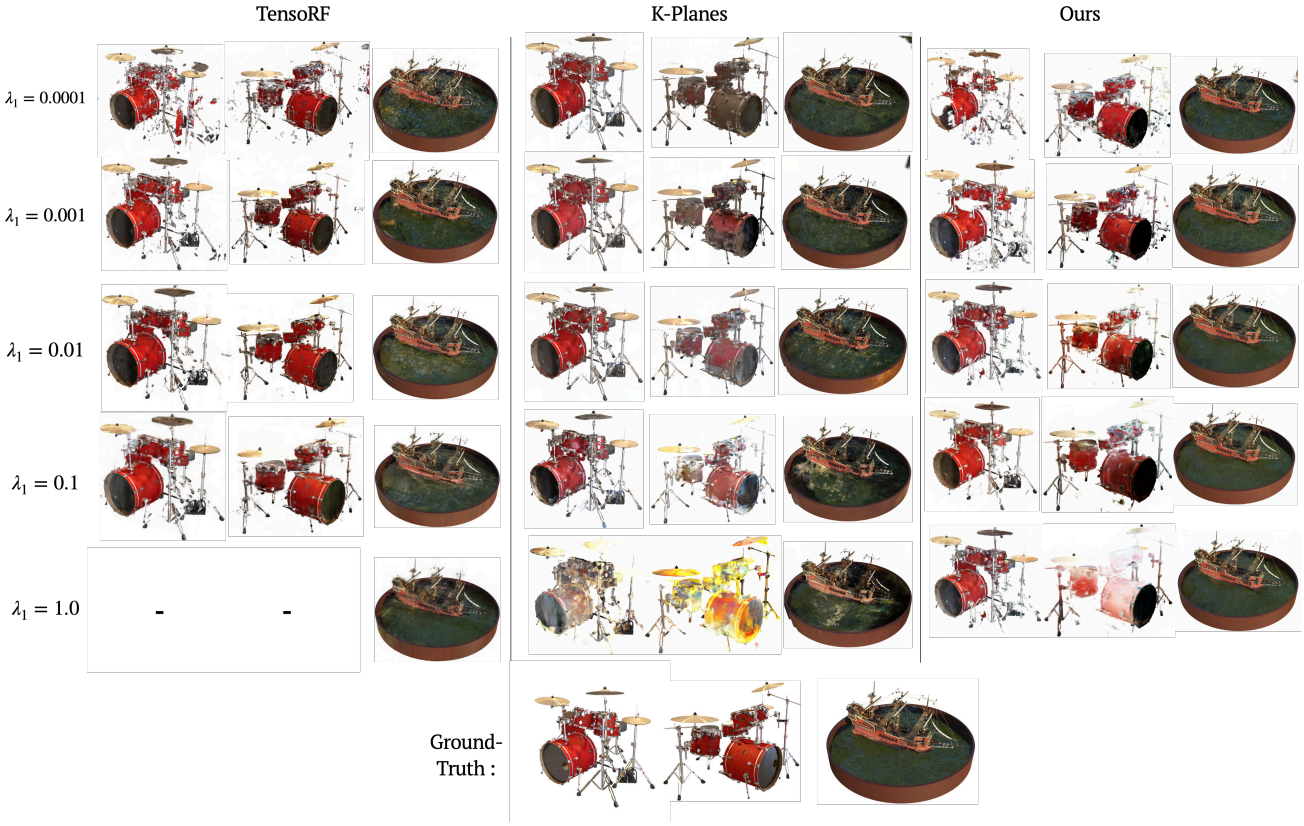


Figure F.4: Rendered images of `drums` and `ship` cases in the static NeRF dataset by TensoRF, K-Planes and ours with varying λ_1. We select the 61st and 36th images for the `drums` scene and the 156th images for the `ship`.

Table F.9: The comparison of Ours and HexPlane in the dynamic NeRF dataset. We conduct experiments varying the value of λ_1. All models are trained using 25 views. We use seed 0 for reproducibility.

Models	PSNR ↑								Avg. PSNR ↑	Avg. SSIM ↑	Avg. LPIPS ↓
	bouncingballs	hellwarrior	hook	jumpingjacks	lego	mutant	standup	trex			
HexPlane ($\lambda_1 = 0.0001$)	28.80	16.32	21.44	21.98	23.81	24.67	21.30	24.34	22.83	0.926	0.082
HexPlane ($\lambda_1 = 0.001$)	30.25	16.86	22.61	22.70	24.21	26.03	23.07	25.19	23.86	0.934	0.070
HexPlane ($\lambda_1 = 0.01$)	30.49	17.61	23.10	22.86	24.29	25.81	23.74	25.30	24.15	0.935	0.074
HexPlane ($\lambda_1 = 0.1$)	29.64	18.24	22.13	21.75	23.72	24.63	23.08	24.53	23.46	0.928	0.090
HexPlane ($\lambda_1 = 1.0$)	26.60	17.79	21.05	19.73	23.53	22.75	19.88	24.30	21.95	0.917	0.117
K-Planes ($\lambda_1 = 0.0001$)	29.39	16.72	22.69	23.98	24.03	26.42	24.47	26.88	24.32	0.937	0.074
K-Planes ($\lambda_1 = 0.001$)	29.22	17.92	22.29	22.73	24.12	26.20	23.22	26.35	24.01	0.939	0.061
K-Planes ($\lambda_1 = 0.01$)	29.38	18.29	22.33	22.78	23.82	26.18	23.02	26.33	24.02	0.938	0.062
K-Planes ($\lambda_1 = 0.1$)	28.85	17.53	21.52	22.52	24.02	26.00	22.74	25.25	23.55	0.931	0.074
K-Planes ($\lambda_1 = 1.0$)	25.29	17.90	20.99	21.63	23.61	25.06	21.73	24.73	22.62	0.928	0.087
Ours ($\lambda_1 = 0.0001$)	32.80	18.34	23.39	23.18	23.79	26.33	23.77	25.77	24.67	0.936	0.071
Ours ($\lambda_1 = 0.001$)	34.13	19.01	23.90	24.72	23.92	26.86	24.26	26.22	25.38	0.942	0.062
Ours ($\lambda_1 = 0.01$)	33.71	19.69	23.83	24.77	24.20	26.89	25.96	26.86	25.74	0.943	0.064
Ours ($\lambda_1 = 0.1$)	32.91	19.80	24.08	24.63	24.36	26.85	27.69	26.40	25.84	0.941	0.074
Ours ($\lambda_1 = 1.0$)	32.21	19.52	24.33	24.36	23.51	26.23	27.18	26.05	25.42	0.937	0.088

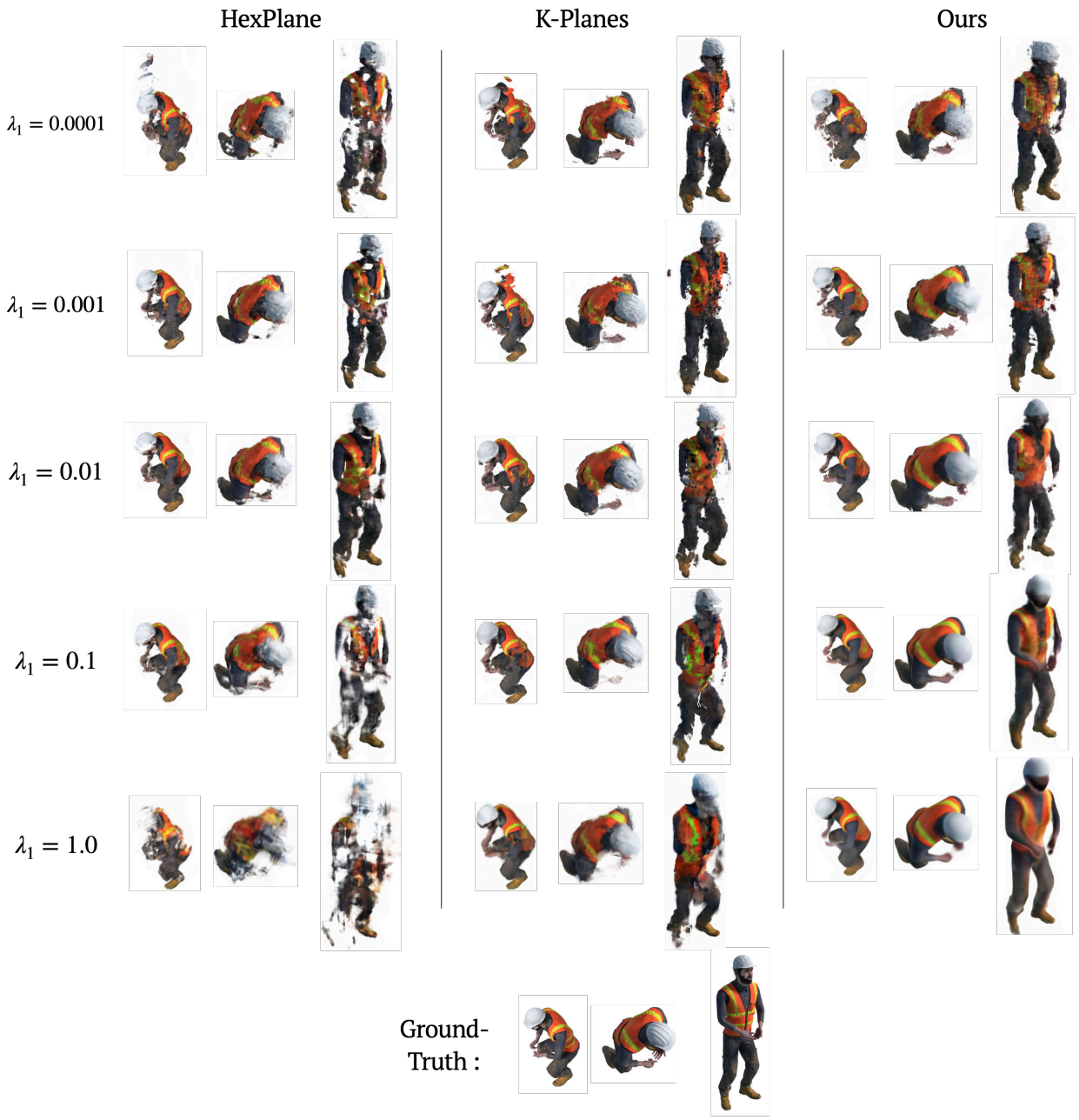


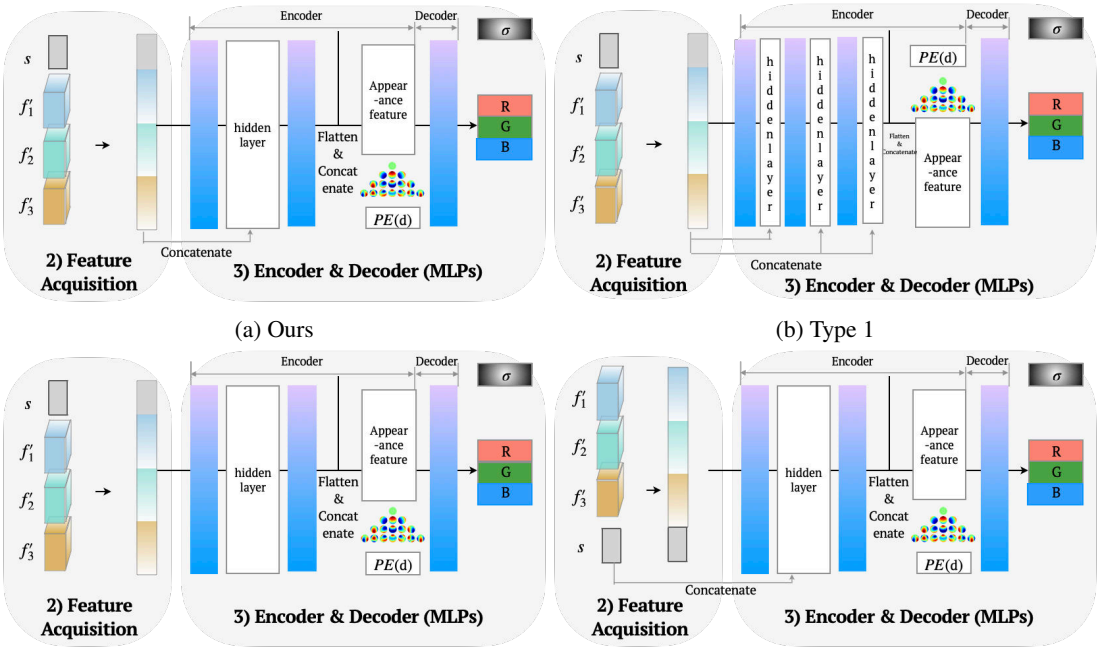
Figure F.5: Rendered images of `standup` cases in the dynamic NeRF dataset by HexPlane and ours with varying λ_1. We evaluate $\{0, 10, 19\}$th views in the test dataset.

G. Ablation Study on Residual Neural Radiance Fields

The encoder of our model utilizes a skip-connection of fused features. To justify the design choice of our model, we compare the results of various encoder structures in static and dynamic cases. All possible candidates for encoder structures are listed, and their graphical representations are also included in Figure G.6.

- Type 1 : Skip connection lies on every layer
- Type 2 : No skip connection, and employs fully connected MLPs
- Type 3 : Skip connection, but only coordinate s is concatenated.

Through Table G.10 and Table G.11, we determine that the proposed model represents the optimal architecture. Their visual outcomes are illustrated in Figure G.7. Partiularly, in the case of dynamic NeRFs, inducing smoothness in the temporal axis is crucial. In Type 3, where only the coordinate feature is used for skip-connection, it shows slightly better performance than our model. However, our model design demonstrates robustness across both static and dynamic cases, confirming the suitability of our model design choices.



(a) Ours (b) Type 1

(c) Type 2 (d) Type 3

Figure G.6: The graphical representation for encoder structures used in Table G.10 and Table G.11.

Table G.10: The comparison of encoding structures. We evaluate four types of encoding structures including ours. All hyperparameters are consistent with those described in the original setting included Appendix C. All models are trained using 8 views in the static NeRF dataset. We use seed 0 for reproducibility.

Models	PSNR ↑								Avg. PSNR ↑	Avg. SSIM ↑	Avg. LPIPS ↓
	chair	drums	ficus	hotdog	lego	materials	mic	ship			
Ours	28.15	20.09	20.04	29.43	27.58	22.06	26.41	24.18	24.74	0.898	0.089
Type 1	23.83	17.85	19.14	18.45	20.54	12.97	14.61	22.78	18.77	0.844	0.179
Type 2	26.15	18.02	19.53	17.78	19.73	11.72	18.06	22.87	19.23	0.848	0.171
Type 3	25.16	19.40	19.33	17.94	20.88	11.85	14.62	23.35	19.07	0.843	0.175

Table G.11: The comparison of encoding structures. We evaluate four types of encoding structures including ours. All models are trained using 25 views in the dynamic NeRF dataset. All hyperparameters are consistent with those described in the original setting included Appendix C. We use seed 0 for reproducibility.

Models	PSNR ↑								Avg. PSNR ↑	Avg. SSIM ↑	Avg. LPIPS ↓
	bouncingballs	hellwarrior	hook	jumpingjacks	lego	mutant	standup	trex			
Ours	33.83	18.93	23.54	24.24	23.69	26.59	26.06	26.05	25.37	0.942	0.063
Type 1	33.99	18.01	24.01	24.26	23.91	26.95	24.55	26.56	25.28	0.941	0.064
Type 2	33.35	18.08	23.82	24.58	24.08	26.85	24.46	26.84	25.26	0.941	0.063
Type 3	32.74	18.64	24.24	24.83	23.99	27.08	25.17	26.81	25.44	0.942	0.062

(a) Ours (b) Type 1

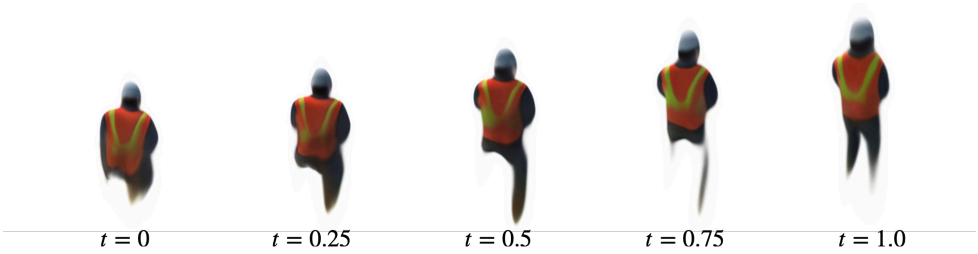
(c) Type 2 (d) Type 3

Figure G.7: Rendered images are generated by alternating encoder structures. We selected the `drums`, `lego`, and `ship` scenes to follow the settings used in previous experiments.

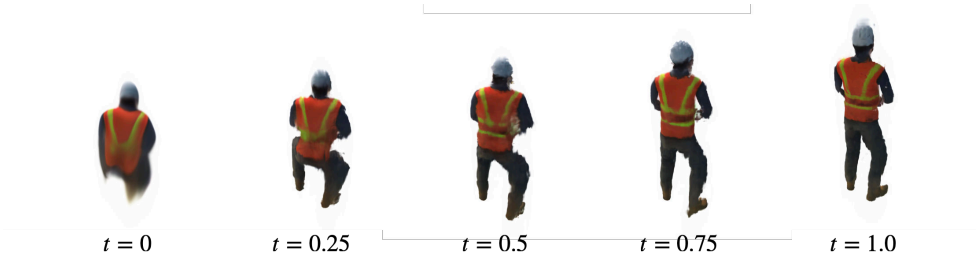
H. Visualization of Disentangling Coordinate Network and Tensorial Feature

Our model employs a disentanglement strategy that separates global shape and detail into coordinate networks and multi-plane features respectively. Additionally, we implement a progressive learning approach on the channel axis within plane features, enhancing the model's ability to cover details from global to local scales. We demonstrate disentangling features into: (1) heterogeneous two features and (2) channel-wise distinct features. First, we explore disentanglement between heterogeneous features. An ablation study on dynamic NeRFs with 25 training views helped us understand the role of coordinate-based networks in our method. Testing the model solely with coordinate networks, as seen in Figure H.8a, revealed that they capture the scene's global context, such as object shapes and significant motions. Although Lindell et al. (2022) indicated a potential dominance of high-frequency features, our model maintains a balance, demonstrating the synergistic function of coordinate network and multi-plane feature.

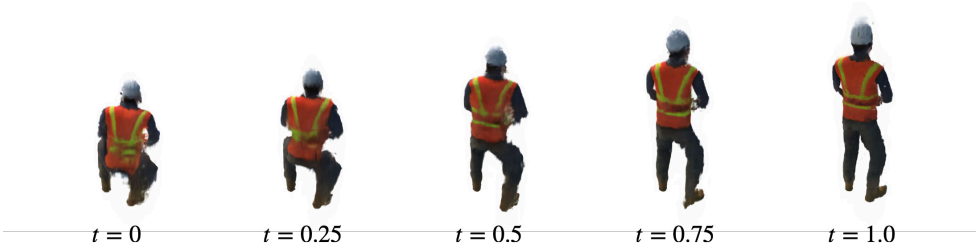
Second, we assess channel-wise disentanglement among plane features. We compared multi-plane features of HexPlane and our method, trained on both full views and 25 views of the `standup` scenes, as depicted in Figure H.9. In the `standup` scenario, the $z - x$ plane should represent the front shape of the person, and the $z - t$ plane should depict upward movement. HexPlane features for full views, shown in Figure H.9a, do not differentiate well between global shape information and intricate local details, with some channels overlapping in learned information. In contrast, our method distinguishes between multi-plane features along the channel axis, enhancing expressiveness as shown in Figure H.9b. Our method maintains this differentiation even with fewer views, as observed in Figure H.9c. Moreover, HexPlane exhibits floating artifacts and lacks visibility of upward movement on the time axis, whereas our model shows consistency in both full and reduced view scenarios, as seen in Figure H.9d. Particularly, our model can selectively learn channel-wise, indicating minimal impact through flat representations. These findings confirm that our dual disentanglement strategies for distinct features and channel-wise distinctions effectively enable learning from global to detailed features, enhancing expressiveness and robustness in handling sparse inputs.



(a) Rendered image by ours with only coordinate networks

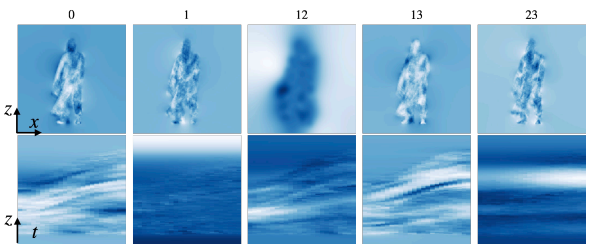


(b) Rendered image by ours with curriculum weighting

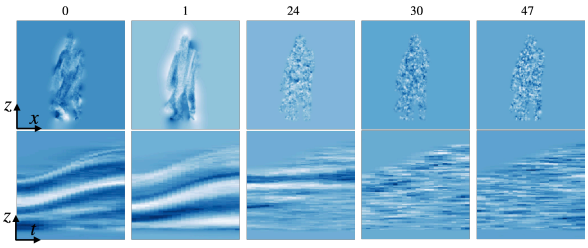


(c) Rendered image with full engangement of multi-plane

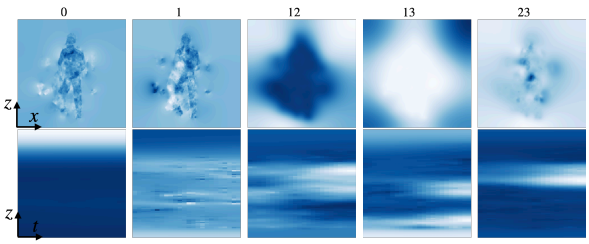
Figure H.8: Rendering results using different feature combinations. We show rendering results from three distinct combination of encoding features, (a) using only coordinates, (b) coordinates with progressively activating multi-plane encoding, and (c) full features. t indicates the timesteps normalized to 1, and we use `standup` scene.



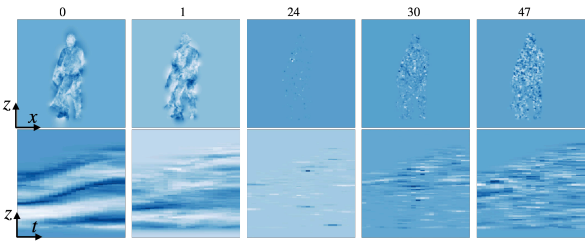
(a) HexPlane($\lambda_1 = 0.001$) trained on full views



(b) Ours trained on full views



(c) HexPlane($\lambda_1 = 0.001$) trained on 25 views



(d) Ours trained on 25 views

Figure H.9: Visualization of plane encoding features. We visualize 5 representative features from the plane encodings of Hexplane and Ours trained on `standup` scene.

I. Comparison of the Number of Parameters and Analysis of Training/Rendering Times

We evaluated our model against iNGP, TensoRF, K-Planes, and other methods using static and dynamic NeRF datasets, limiting the training steps to 15,000 and using only 8 views for training. Rendering time was measured over 200 frames for static NeRFs and 20 frames for dynamic scenarios. We demonstrate the model parameters in Table I.12 and Table I.13, where bracketed numbers indicate channel counts in multi-plane features. K-Planes model, featuring multi-resolution multi-plane characteristics, calculates total channels as the product of resolutions and channel dimension per resolution.

Table I.12 shows that our method achieves comparable performance to TensoRF with the optimized λ_1, yet uses significantly fewer parameters. TensoRF, despite optimal performance at 64 channels, faces instability during training and rendering. Reducing channels to 20 causes convergence issues in scenes like {chair, ficus, mic}, highlighting its limitations with sparse inputs. Although K-Planes appears stable, it underperforms and demands more parameters. Our method, while slower in rendering compared to TensoRF and K-Planes, excels in stability during both training and rendering, ensuring consistent performance even with fewer channels, which is ideal for sparse inputs. In the case of dynamic NeRFs, our method outperforms all baselines even with reduced parameters (approximately 1 million) as shown in Table I.13. While slightly slower in rendering, the performance difference is minimal. This is attributed to the low frequency detail handled by coordinate network, eliminating the need to apply multi-plane features for low frequency detail. Consequently, we accomplish minimal use of parameters because multi-plane features are tasked solely with high-frequency details.

Despite its slower rendering speed, especially in complex scenes like `ficus` and `drums`, we argue that our model focuses on refining NeRF architecture, ensuring compatibility with fast training NeRFs frameworks without compromising on stability during sparse input training. Future efforts could explore using sequential MLPs like `tinycudann` to enhance rendering speed, although this might introduce instability given our current focus on maintaining robustness in sparse scenarios. Overall, our experiment highlights the robustness of our method in maintaining stable training and consistent rendering quality, proving crucial in conditions with sparse inputs where reliability across various settings is essential.

Table I.12: Comparison of the number of parameters and analysis of training and rendering time in static NeRFs. † indicates the optimized hyper-parameter $\lambda_1 = 0.001$ used.

Model Name	# Params [M]	Avg. PSNR	Avg. Training Time [min]	Avg. Rendering Time [min]
iNGP (T=19)	11.7M	19.26	7.60	0.82
iNGP (T=18)	6.4M	19.99	6.40	0.91
K-Planes (3*16)	17M	23.95	17.61	6.83
K-Planes (2*16)	4.4M	23.16	13.72	6.51
TensoRF† (64)	17.3M	25.23	7.72	7.82
TensoRF† (20)	6.1M	-	-	-
Ours (48)	6.0M	24.36	31.16	46.02
Ours (24)	3.0M	23.74	24.06	40.76

Table I.13: Comparison of the number of parameters and analysis of training and rendering time in dynamic NeRFs.

Model Name	# Params [M]	Avg. PSNR	Avg. Training Time [min]	Avg. Rendering Time [min]
K-Planes (3*32)	18.6M	23.85	18.93	0.83
K-Planes (3*4)	1.9M	23.41	13.29	0.78
HexPlane (72)	9.7M	24.00	6.78	0.60
HexPlane (6)	0.8M	22.08	6.38	0.68
Ours (48)	3.4M	25.17	12.22	2.14
Ours (12)	1.0M	25.10	8.77	1.73

J. Experimental Result of Real-world Dataset : Tanks and Temples

The proposed method is also evaluated on the real-world Tanks and Temples dataset (Knapitsch et al., 2017), where it was compared with the baseline TensoRF models, including the optimized setting ($\lambda_1 = 0.001$). We focus on how each method handles the preservation of global context in scenes. As shown in the Figure J.10, the proposed method consistently produces better rendered images than the baselines by preserving the global context. This is crucial when dealing with sparse input situations where maintaining the overall structure and shape of objects is essential even in real-world situations. Despite TensoRF's focus on local details leading to partial but incomplete reconstructions seen in the case of `family`, our method excels in capturing the overall scene composition. This ability ensures that the larger structure and form of objects in the scene are accurately reconstructed, even at the cost of some finer details. Therefore, we demonstrate that the proficiency of our method becomes more apparent under conditions of sparse input data, making it particularly suitable for real-world applications where input data might be limited or incomplete.

Quantitatively, the proposed method shows its strength, especially in SSIM scores. While PSNR is a valuable metric for image quality, it can be biased in this context due to the lack of mask information and the inclusion of full-resolution white backgrounds. On the other hand, SSIM focuses on the perceived quality of structural information in the images. As shown in Figure J.11b, the proposed method consistently achieves higher SSIM scores across all scenes, indicating its superior capability in preserving the structural integrity and overall composition of scenes.

To sum up, the proposed method distinguishes itself from the baselines through its robust ability to preserve the global context of scenes, handle sparse input data effectively, and render images that are both structurally sound and visually realistic. These inherent properties highlight its potential for broader application in real-world scenarios, where input data is often sparse and incomplete.

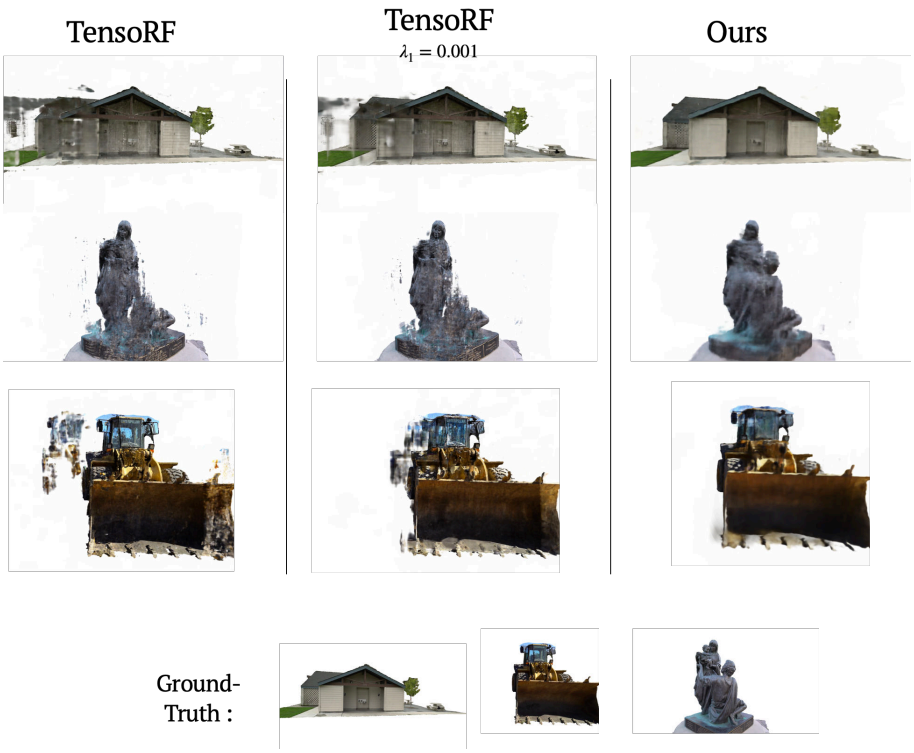
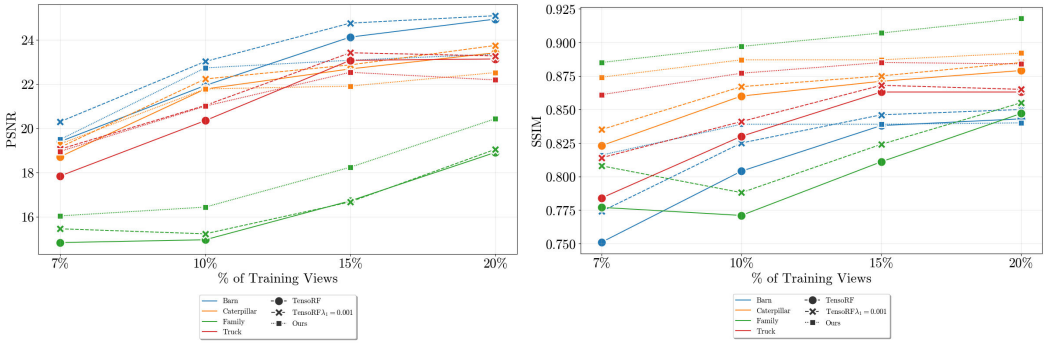


Figure J.10: The qualitative results of baselines and the proposed method on the Tanks and Temples dataset. We specifically show $\{47, 11, 12\}$-th images of `Barn`, `Family` and `Caterpillar` from the test dataset. We use $\{7, 10, 15\}$ percentiles of the training views for the `Caterpillar` `Barn` and `Family` scenes, respectively.



(a) PSNR (b) SSIM

Figure J.11: The line plots of PSNR and SSIM on the Tanks and Temples dataset varying the number of training views.