Refined Tensorial Radiance Field: Harnessing Coordinate-Based Networks for Novel View Synthesis from Sparse Inputs

Mingyu Kim
KAIST AI
callingu@kaist.ac.kr

Jun-Seong Kim
POSTECH EE
gucka28@postech.ac.kr

Se-Young Yun*
KAIST AI
yunseyoung@kaist.ac.kr

Jin-Hwa Kim*
NAVER AI Lab & SNU AIIS
j1nhwa.kim@navercorp.com

Abstract

The multi-plane encoding approach has been highlighted for its ability to serve as static and dynamic neural radiance fields without sacrificing generality. This approach constructs related features through projection onto learnable planes and interpolating adjacent vertices. This mechanism allows the model to learn fine-grained details rapidly and achieves outstanding performance. However, it has limitations in representing the global context of the scene, such as object shapes and dynamic motion over times when available training poses are sparse. In this work, we propose refined tensorial radiance fields that harness coordinate-based networks known for strong bias toward low-frequency signals. The coordinate-based network is responsible for capturing global context, while the multi-plane network focuses on capturing fine-grained details. We demonstrate that using residual connections effectively preserves their inherent properties. Additionally, the proposed curriculum training scheme accelerates the disentanglement of these two features. We empirically show that the proposed method outperforms others for the task with static and dynamic NeRFs using sparse inputs. In particular, we prove that excessively increasing denoising regularization for multi-plane encoding effectively eliminates artifacts; however, it can lead to artificial details that appear authentic but are not present in the data. On the other hand, we note that the proposed method does not suffer from this issue.

1 Introduction

Neural Radiance Fields (NeRFs) have gained recognition for their ability to create realistic images from various viewpoints using the volume rendering technique [1]. Early studies have demonstrated that multi-layer perceptron (MLP) networks, combined with sinuosidal encoding, can effectively synthesize 3-dimensional novel views [2, 3, 4, 5, 6]. These studies have shown that simple coordinate-based MLP networks exhibit strong low-frequency bias, and incorporating wide-spectrum sinusoidal encoding allows for capturing both low and high-frequency signals. Subsequent works illustrated the importance of appropriate sinusoidal encoding in conjunction with target signals to enhance performance [7, 8, 9]. To expedite the learning process, approaches explicitly parameterizing spatial attributes through multi-plane combinations have been introduced [10, 11]. In contrast to the

*Co-corresponding authors.

Learning-Based Solutions for Inverse Problems Workshop on NeurIPS 2023.

(a) HexPlane (b) Ours

Figure 1: The qualitative results of the `standup` case in dynamic NeRFs using 25 training views (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. [24]. Figure (b) is the rendered image of the proposed method.

aforementioned approaches, these methods dramatically reduce training time and produce cleaner and more realistic images, albeit at the cost of greater memory requirements.

For broader real-world applicability, extensive efforts have focused on reliably constructing radiance fields in cases of sparse input data. After the emergence of dynamic scenes dealing with time sparsity, addressing data sparsity has gained more attention in this field, as NeRF models commonly face overfitting issues due to the lack of consistent data for 3 or 4-dimensional space [12]. One set of solutions tackled this by leveraging a pretrained image encoder to compare rendered scenes against consistent 3D environments [13, 14, 15, 16]. Another approach incorporated additional information, such as depth or color constraints, to maintain 3-dimensional coherence [17, 18, 19, 20]. Methods progressively adjusting the frequency spectrum of position encoding have also proven effective in counteracting overfitting without additional information [21, 22].

However, a notable limitation of prior strategies dealing with sparse inputs is their less-than-ideal visual output. While the recent work reported successful reconstruction of static NeRF using voxel-grid parameterization in the sparse input regime with the assistance of denoising penalties like total variation [23], they often lack in adequately representing global elements like object morphology and dynamic motion, as evident in Figure 1a. Even if some renderings look crisp upon close inspection, the overall quality of the rendered results deteriorates due to the absence of global structures.

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, the coordinate-based features are responsible for capturing global context, while the multiple-plane features are responsible for capturing fine-grained details. Moreover, in contexts with occlusions or time-variant dynamics, we employ a progressive weighting scheme that prevents the model from falling into local minima. This prioritizes low-frequency coordinate-based features to capture the global context first, allowing multiple-plane features to describe fine-grained target signals gradually. As a result, images generated by the proposed method exhibit improved clarity in terms of 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. Particularly, it outperforms baselines in dynamic NeRFs from the sparse inputs.

2 Refined Tensorial Radiance Fields: Harnessing Coordinate-based Networks

We propose a novel method, referred to as "refined tensorial radiance field", that leverages coordinate-based networks. To mitigate the constraints of locality inherent in grid structures, our method capitalizes on a combination of distinct coordinate feature encoding techniques and multi-plane representations, as depicted in Figure 2. Before diving into the proposed method, background and notation can be found in Appendix B.

2.1 Architecture and Loss Function

We describe how our model works in the dynamic NeRF case. Applying this model to a 3-dimensional static NeRF is feasible by simply excluding the t variable. A key aspect of our network architecture is the utilization of coordinate-based networks along with explicit representation. In high-level context,

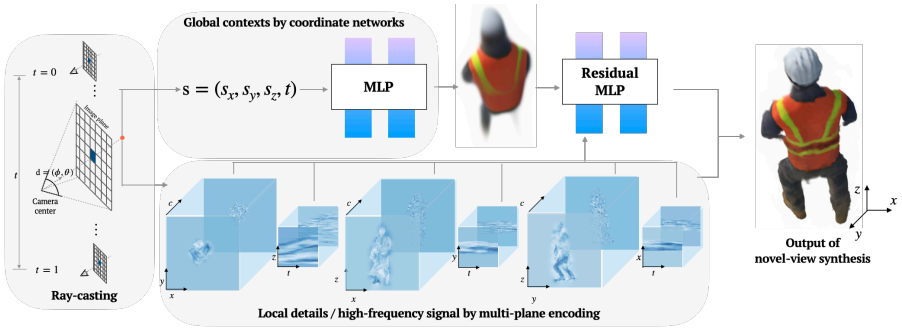


Figure 2: Conceptual illustration of the proposed method utilizing global contexts by coordinate networks and fine-grained details by multi-plane encoding. This method effectively displays two heterogeneous features.

we replace sinusodial encoding with multi-plane encoding while employing the architecture of the origianl NeRF. A coordinate $s := (s_x, s_y, s_z, t)$ is transformed via multi-plane encoding from spatial and temporal plane features $\mathcal{M}$, $\mathcal{V}$ with element-wise multiplication $f = f^{\mathcal{M}} \odot f^{\mathcal{V}} \in \mathbb{R}^{3c}$. These features are then fed into MLPs parameterized by Θ along with their respective coordinates s. Our empirical findings demonstrate that this operation promotes the disentanglement of two features, aligning with our intended purpose.

We introduce a loss function that combines photometric loss and laplacian smoothing across multi-plane features. 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. Laplacian smoothing tends to excessively smooth signals, making them conform to global tendency rather than accurately local finer details [25]. 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 as follows:

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

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 D. While increasing the value of λ_1 allows to removes floating artifacts by over-smoothing the multi-plane features, it creates undesirable deformation that looks authentic but not be present in the training data. Hence, we opt not to utilize excessively high denoising weights. Instead, the coordinate network provides consistent training for multi-plane encoding when capturing high-frequency details. We empirically validate this through our experiments.

2.2 Curriculum Weighting Strategy 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 in accordance with training iterations. 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 the channel dimension of multi-planes. Here, $f = \{f_1, f_2, f_3\}$, and $f_i \in \mathbb{R}^c$ represents the output of multi-plane encoding, and the weighting factor $\gamma(t) = \{\gamma_1(t), \cdots, \gamma_c(t)\} \in \mathbb{R}^c$ is

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

defined as follows:

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

where, $j \in \{1, \cdots, c\}$ is the index of channel dimension and $\alpha(t) = c \cdot \frac{(t-t_s)}{(t_e-t_s)} \in [t_e, t_s]$ 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's worth noting that this weighting function is similar to those used in previous works such as [26, 27, 21, 28].

3 Experiments

In this section, we present our experiments designed to address three pivotal questions: *1) Do existing sinusoidal embedding techniques effectively render clear scenes when given sparse input data? 2) Does the introduction of denoising regularizations enable explicit parameterization methods to consistently capture 3D coherence without artifacts with sparse input data? 3) Does the integration of disparate features, such as multiple planes and coordinates, substantially improve the performance of both static and dynamic NeRF?*

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 compared to forward-facing poses. For performance evaluation, we employ the PSNR metric to gauge the quality of image reconstruction. In addition, SSIM and LPIPS scores are reported to assess the perceptual quality of the rendered images. Further experimental details are described in Appendix E. Numerous experimental results are available in the appendix. Details about the 4-dimensional NeRF can be found in Appendix G, and the ablation studies are covered in Appendix H, Appendix I and Appendix J.

3.1 Three-dimensional 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 fine details without introducing undesirable artifacts under sparse input conditions. Consistent with prior studies such as [16, 21], we trained all models with 8 views. We compare our proposed models with sinusoidal encoding methods; Simplified NeRF, DietNeRF [16], HALO [22] and FreeNeRF [21] and for explicit spatial parameterization methods; DVGO [29], VGOS [23], iNGP [30] and TensoRF [10].

The quantitative results are shown in Table 1. The rendering results including the detailed numerical values can be found in Appendix F. 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 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.

4 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.

Acknowledgement

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.

References

[1] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[2] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[3] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

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

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

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

[7] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: adaptive coordinate networks for neural scene representation. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021.

[8] David B Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. Bacon: Band-limited coordinate networks for multiscale scene representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16252–16262, 2022.

[9] Shayan Shekarforoush, David Lindell, David J Fleet, and Marcus A Brubaker. Residual multiplicative filter networks for multiscale reconstruction. *Advances in Neural Information Processing Systems*, 35:8550–8563, 2022.

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

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

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

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

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

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

[16] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021.

[17] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022.

[18] Yu-Jie Yuan, Yu-Kun Lai, Yi-Hua Huang, Leif Kobbelt, and Lin Gao. Neural radiance fields from sparse rgb-d images for high-quality view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[19] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12892–12901, 2022.

[20] Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. Sparf: Neural radiance fields from sparse and noisy poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4190–4200, 2023.

[21] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8254–8263, 2023.

[22] Liangchen Song, Zhong Li, Xuan Gong, Lele Chen, Zhang Chen, Yi Xu, and Junsong Yuan. Harnessing low-frequency neural fields for few-shot view synthesis. *arXiv preprint arXiv:2303.08370*, 2023.

[23] Jiakai Sun, Zhanjie Zhang, Jiafu Chen, Guangyuan Li, Boyan Ji, Lei Zhao, and Wei Xing. Vgos: Voxel grid optimization for view synthesis from sparse inputs. *arXiv preprint arXiv:2304.13386*, 2023.

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

[25] Veeranjaneyulu Sadhanala, Yu-Xiang Wang, James L Sharpnack, and Ryan J Tibshirani. Higher-order total variation classes on grids: Minimax theory and trend filtering methods. *Advances in Neural Information Processing Systems*, 30, 2017.

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

[27] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5741–5751, 2021.

[28] Hwan Heo, Taekyung Kim, Jiyoung Lee, Jaewon Lee, Soohyun Kim, Hyunwoo J Kim, and Jin-Hwa Kim. Robust camera pose refinement for multi-resolution hash encoding. 2023.

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

[30] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[31] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[32] Gizem Yüce, Guillermo Ortiz-Jiménez, Beril Besbinar, and Pascal Frossard. A structured dictionary perspective on implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19228–19238, 2022.

[33] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In *International Conference on Learning Representations*, 2021.

[34] Sameera Ramasinghe, Lachlan E MacDonald, and Simon Lucey. On the frequency-bias of coordinate-mlps. *Advances in Neural Information Processing Systems*, 35:796–809, 2022.

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

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

[37] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023.

[38] Sihyun Yu, Kihyuk Sohn, Subin Kim, and Jinwoo Shin. Video probabilistic diffusion models in projected latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18456–18466, 2023.

[39] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Pet-neus: Positional encoding tri-planes for neural surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12598–12607, 2023.

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

[41] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

A 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 [1, 4, 5, 6]. However, it was found that a classical coordinate network without this encoding has a bias toward lower frequencies [31, 32]. 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 [2, 3, 33, 34]. Lindell et al. [8] uncovered that improper high-frequency embedding results in artifacts negatively impacting the quality of reconstruction. They addressed this issue using multi-scale bandwidth networks, where each MLP layer has a distinct spectrum of frequency embedding. Subsequent research utilized residual connections to faithfully maintain the designated spectrum without overwhelming high-frequency components [9].

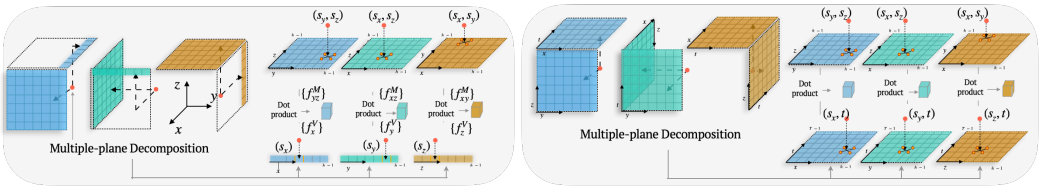
Explicit parameterization Recent developments in explicit representations, such as voxel-grid, hash encoding, and multi-planes, have gained attention due to their fast training, rendering speed, and superior performance compared to positioning encoding-based networks [35, 29, 30, 10, 24, 36]. Sun et al. [29] 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 [30]. 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 [10, 24, 36]. In particular, while the aforementioned approaches were executed on special on-demand GPU computations to boost efficiency, this method achieves comparable speed and performance based on general auto-differential frameworks. As a result, the multiple-plane approach broadens its scope to various tasks, including 3D object generation, video generation, 3D surface reconstruction, and dynamic NeRF [37, 38, 39, 24, 36].

NeRFs in the sparse inputs Early efforts incorporated pre-trained networks trained on large datasets to compensate for the lack of training data [16, 13, 14]. Another alternative approach incorporated additional information, such as depth or color constraints, to ensure the preservation of 3D coherence [17, 18, 19, 20]. 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 [40]. This paper verified their regularization performs well on forward-facing examples like DTU and LLFF datasets. 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 proves effectiveness in mitigating over-fitting without relying on additional information [21, 22]. Compared to explicit representations, those still suffer from unsatisfactory visual quality, characterized by blurry boundaries. We present this problem in the experimental results, both qualitatively and quantitatively. A recent study tried using total variation regularization and direct voxel fields to get rid of artifacts and construct smoother surfaces [23], but it has limitations when applied to higher dimension cases like dynamic NeRFs due to excessive parameter usage.

Another work attempted to use tri-planes with sinusoidal encoding of coordinates to create smoother surfaces [39], but their direction differs from our method since they mainly focus on enriching available features, as well as they did not demonstrate the role of tri-planes and coordinate features. In this paper, our new approach, refined tensorial radiance fields, 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 targets like dynamic NeRFs and extremely limited sparse inputs.

B 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 [10] for the static NeRFs and HexPlane [24] 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 B.1: The schematic of baselines that use the multi-plane encoding. (a) TensoRF employs three planes and lines [10]. (b) HexPlane adopts a total of six multiple planes to include the time axis [24].

B.1 Neural Radiance Fields

Mildenhall et al. [1] 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.

B.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 [35, 29, 30], 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, $i \in \{xy, yz, zs\}$ 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 B.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 $f_i \in \mathbb{R}^c$, 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 C. 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.

B.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 B.1b. Likewise the previous subsection, we denote two index variables, $i \in \{xy, yz, zs\}$ 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:

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

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.

C Multiple-plane Encoding and Concatenating Coordinates

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 [10, 24], 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.

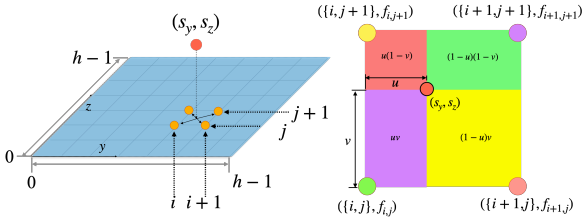


Figure C.2: Blinear interpolation

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:

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

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

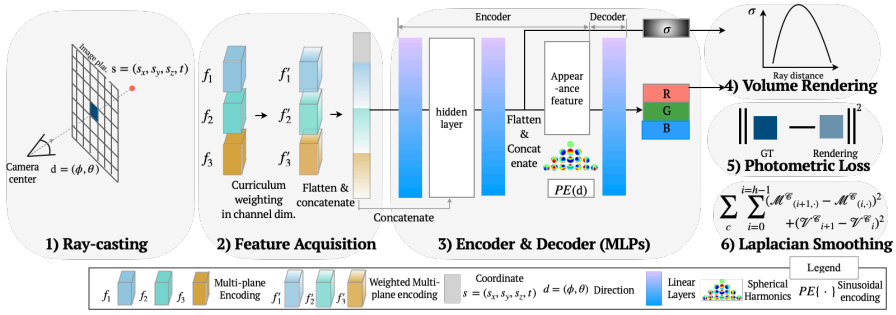


Figure D.3: The schematic of the proposed method.

$s = \{s_x, s_y, s_z, t\} \in \mathbb{R}^4$. 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 $\mathbb{R}^{4+3c}$.

D Implementation Details

We illustrate the schematic of the proposed method in Figure D.3. As shown by Shekarforoush et al. [9], residual networks yield multi-fidelity results by preserving their pre-designated sinusoidal embeddings. In line with this, the proposed method adopts skip connections between acquired features and the hidden layer to serve the same purpose. Our empirical findings demonstrate that this operation promotes the disentanglement of two features, aligning with our intended purpose.

D.1 Hyper-parameters on the Static NeRF

The proposed model consists of multi-plane encoding and MLPs with skip connections. For multi-plane encoding, we use 48 dimensional channels. The resolution of plane features is upsampled up to 8,000,000 (200^3) at the end of training. The weight for Laplacian smoothing(λ_1), curriculum learning schedule, and initial feature resolution differ in each scene as we proceed with hyperparameter tuning for the optimal result. We listed detailed figures of hyperparameters of multi-plane encoding in Table D.1. For decoder MLP layers, we use standard fully connected layers with ReLU activations, 256 channels each. Our encoder contains four fully connected ReLU layers. We include a skip connection after the second layers, which concatenates fused input features. The occupancy is calculated directly from the obtained features and is calculated as the `softplus` of the first channel. Then there is following RGB decoder consists of two layers. From the features obtained by RGB decoder, color values are obtained through `sigmoid` activation.

In our experiments, we trained over 30,000 iterations with batch size of 4,096. We use the Adam optimizer[41] with initial learning rate is set to 0.02 and 0.001 for multi-plane features and for MLPs respectively, and followed learning rate scheduleling following TensorRF[10].

Table D.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

D.2 Hyper-arameters on the Dynamic NeRF

The configuration for the dynamic NeRF case also follows the settings as in the static case. We utilize 48-channel plane features. The initial voxel resolution is set to 4,096 (16^3) and is subsequently

upsampled to 8,000,000 (200^3). For more detailed descriptions, please refer to Table D.2. The structure of the decoder part, initial learning rate, and optimizer configuration remain identical to the static NeRF. Other configurations not specified here are taken directly from HexPlane's method as described in [24].

Table D.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}

E Experiment Details

We conducted the training and evaluation of all models using an NVIDIA A6000 with 48 GB of memory. For the experiments in Table 1, we utilized five different seeds: {0, 700, 19870929, 20220401, 20240507}. However, it's important to note that without explicitly describing the results for all five trials, each experiment was executed once, and the seed 0 was then used as the default. For explanations regarding the dataset and baselines, we provide the following description.

E.1 Datasets

NeRF blender dataset The Blender Dataset [1] 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[21, 16], for training, we used 8 views with IDs of 26, 86, 2, 55, 75, 93, 16, 73 and 8 counting from zeros. For evaluation, we uniformly sampled 25 images from the original test set. Unlike the evaluation settings in the previous works [21, 16], we evaluate all metrics by 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 D-NeRF Dataset [12] is a set of synthetic, bounded, 360 degree, monocular videos for dynamic objects. 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`

E.2 Baselines

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

Diet-NeRF Diet-NeRF [16] is a sinusoidal encoding based model. The model incorporates auxiliary semantic consistency loss which leverages pre-trained CLIP, networks trained on large data-sets 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 which is stated in [16]. For implementation we used the codebase in `https://github.com/ajayjain/DietNeRF`

Free-NeRF Free-NeRF is a sinusoidal encoding based model [21]. This method 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 [29] is a model that uses a three-dimensional dense voxel feature grid. It utilizes independent voxel features for density and color. Shallow MLP follows color encoding. In the first stage, coarse geometry search learning, the initial shape is obtained, which provides the shape prior to the scene and finds 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 The Instant NGP [30] model expresses the voxel feature grid using the Hash function. 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 space sampling. We used the code from https://github.com/kwea123/ngp_pl

VGOS VGOS [23] is the first example of applying the grid-based method to a Few-shot case. 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

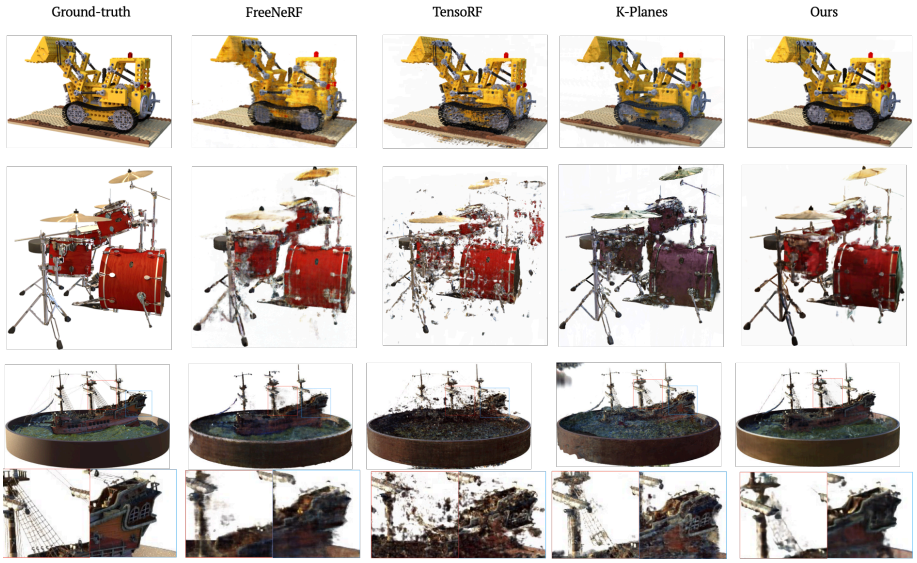


Figure F.4: Rendered images of `ship`, `lego`, `drums` cases in the static NeRF dataset by FreeNeRF, TensoRF, K-Planes and ours.

F The Result Statistics of Three-dimensional Static NeRF Dataset

From Table F.3 to Table F.5, we present the quantitative results for each scene of the synthetic NeRF Dataset. All reported numbers are averages of 5 experiments and corresponding standard deviations. Our model performed better in all metrics compared to all counterpart models. We also summarize the results of the TensorRF model with intense additional laplacian smoothness loss. The $\lambda_1 = 0.1$ value is the optimal value for obtaining the best value. This evidence is provided in Appendix H

TensoRF with strong Laplacian regularization shows comparable performance with the proposed model. The two methods show complement advantages in the novel-view rendering results. To compare, we present a novel-view renderings of `ship` (Figure F.4) and `Mic`. TensorRF with $\lambda_1 = 0.1$ optimizes focusing on reconstruct higher-frequency texture. Therefore, it shows instability in low-frequency information such as a geometry (dec in `ship`), and show high-frequency artifacts on the color side (water regions in `ship`). Proposed method, TensoRefine has more strength in robust optimization, especially in global information. More accurate 3D geometry and view-consistent color reconstruction are possible. On the other hand, there are cases of underfitting in high resolution. Without relying on high denoising regularization, the proposed method nearly achieves the best performance, thanks to the coordinate-based networks responsible for capturing the global context.

Table F.3: The result of average PSNR in the static NeRF. We conduct five trials for each method 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 F.4: The result of average SSIM in the static NeRF. We conduct five trials for each method 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 F.5: The result of average LPIPS in the static NeRF. We conduct five trials for each method 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

G Experiments on Four-dimensional Dynamic Radiance Fields

To demonstrate the robustness of the proposed model on more spare input cases, we conduct our experiences on the dynamic scenarios. We conducted 4-dimensional dynamic NeRF experiences on a D-NeRF data set. This data set comprises monocular cameras of about 50-100 frames duration and different in-ward 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 view was sampled uniformly for each scene. To demonstrate the need for our refined tensorial radiance fields, we compare our method with HexPlane [24] and its variants.

The observations made in subsection 3.1 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 G.6. 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. For instance, the variants of HexPlane do not accurately depict the shape of the blue ball over time, whereas the proposed method successfully does, including the reflection of light on the green ball. In the case of the `jumpingjack` sequence, the proposed method exhibits fewer artifacts and maintains the boundary of the scene better compared to HexPlane.

Table G.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

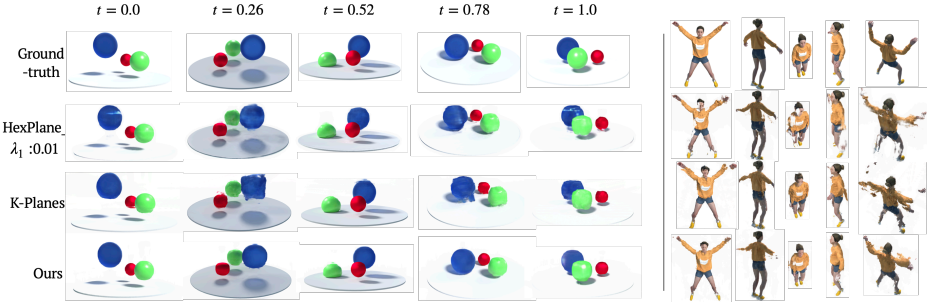


Figure G.5: Rendered images of the `bouncingballs` and `jumpingjacks` in the dynamic NeRF dataset by HexPlane with $\lambda_1 = 0.01$, K-Planes and ours.

H Experiments on Varying Denoising Weights λ_1

We assess the role of Laplacian smoothing regularization within TensoRF, HexPlane, and the proposed method. In this experiment, we incrementally increase the parameter λ_1 from 0.0001 to 1.0, multiplying by a factor of 10. Table H.7 demonstrate that our proposed method outperforms all experiment scenarios in both static and dynamic NeRF, with the sole exception of when $\lambda_1 = 0.001$ in the static NeRF. A notable performance difference was observed in the dynamic NeRF, which presents greater challenges due to time sparsity compared to the static NeRF. In detail, in the static NeRF dataset, our method yielded an average PSNR score between 22.99 and 24.55. In contrast, TensoRF had scores between 24.10 and 24.98 but it does not converge in cases of $\lambda_1 \in \{0.01, 0.1, 1.0\}$. For the dynamic NeRF, HexPlane's scores ranged from 21.95 to 24.15, while ours spanned 24.67 to 25.74. This indicates our method is less dependent on denoising regularization, emphasizing the robust regularization capabilities of coordinate networks for multi-plane encoding. Our observations indicate that the proposed method maintains near-optimal performance across all scenarios once the λ_1 surpasses 0.001. This stability alleviates concerns about searching the regularization value for different scenes, significantly reducing hyperparameter tuning efforts.

Table H.7: 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	ours	HexPlane	ours
0.0001	24.10	23.68	22.83	24.67
0.001	24.98	24.47	23.86	25.38
0.01	-	24.55	24.15	25.74
0.1	-	24.23	23.46	25.84
1.0	-	22.99	21.95	25.42

Furthermore, excessive regularization can introduce undesirable modification, including the introduction of color disturbances as evidenced in the case of `ship` with TensoRF, $\lambda_1 = 0.001$. Unlike the above, our method consistently achieves near-optimal performance without excessive denoising regularization, attributed to the coordinate-based networks capturing global contexts. As depicted

in Figure F.4, our method can restore fine geometries and reproduce accurate colors even under challenging conditions.

Detailed Numerical Evaluation for Different Values of λ_1 in both Static and Dynamic Radiance Fields.

We compare TensorRF for our model according to various degrees of denoising regularization. Table H.8 shows the dependence of the TensorRF, and proposed model on denoising weight. According to the rendering results (figure), for TensoRF denoising does reduce floating artifacts, but, if it is too strong, undesired high-frequency artifacts appear, as described in Appendix F. This requires a searching process for regularization weight for the optimal value. On the other hand, our model is not dominantly affected denoising weights but shows better performance in various λ_1 values. In addition, even when the intensity of denoising increases, undesigned artifacts do not appear. This is possible because our model is a coordinate feature anchoring low-frequency information.

In the case of Dynamic, as sparsity increases, the result shows that denoising regularization alone is not enough for robust reconstruction. TensoRF added with smoothing shows degraded performance compared to the proposed model for all dynamic scenes. In addition, for TensoRF models, the optimal λ_1 value varies depending on the scene's characteristics, which explains that dependence on the smoothed loss becomes severe. In contrast, the proposed model consistently works well regardless of the weighting.

The above results show that our feature-fusion strategy already has sufficient robustness on sparse inputs. In addition, regularization shows synergy with our model design, as it assists in more realistic rendering without producing undesired artifacts, and it does works for extremely sparse input cases.

Table H.8: The comparison of Ours and TensoRF in the static NeRF dataset. We conduct experiments varying the value of λ_1. All models are trained using 8 views. We use seed 0 for reproducibility. 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	-	-	-
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

Table H.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
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

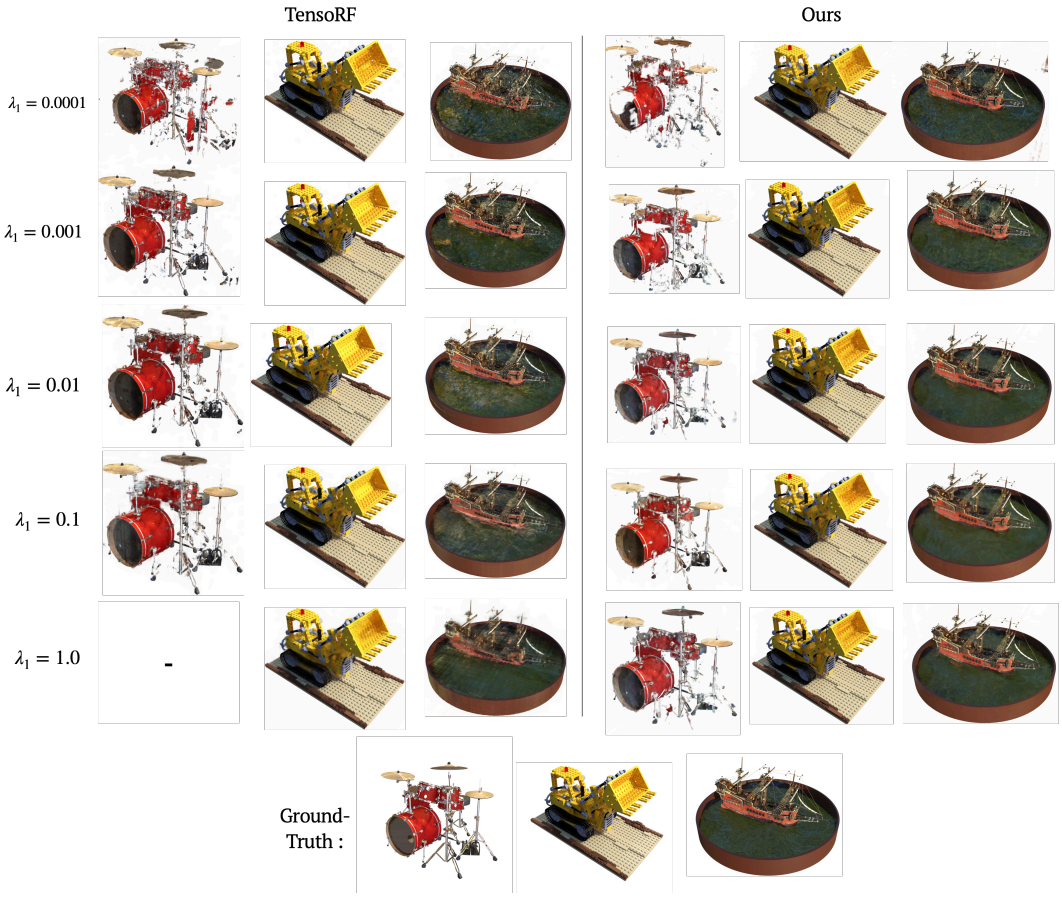


Figure H.6: Rendered images of `ship`, `lego`, `drums` cases in the static NeRF dataset by TensoRF and ours with varying λ_1. We use 61th, 36th and 156th views of `drums`, `lego` and `ship` respectively.

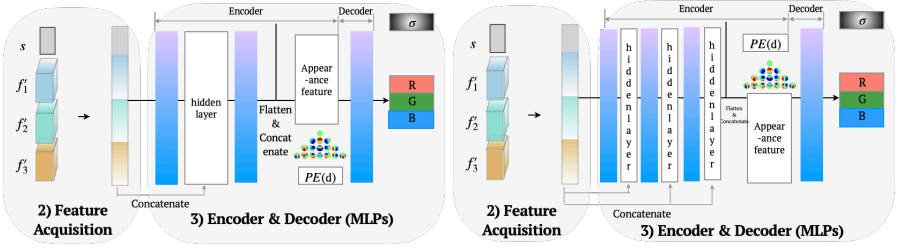
I Ablation Study on Encoding Structures

The encoder of our model applies the skip-connection of fused features. To justify design choice of our model, we compare the results of various encoder structures in static and dynamic cases Table H.8, Table I.11. All possible candidates regarding Encoder structures are listed and their graphical representations are also included in Figure I.8.

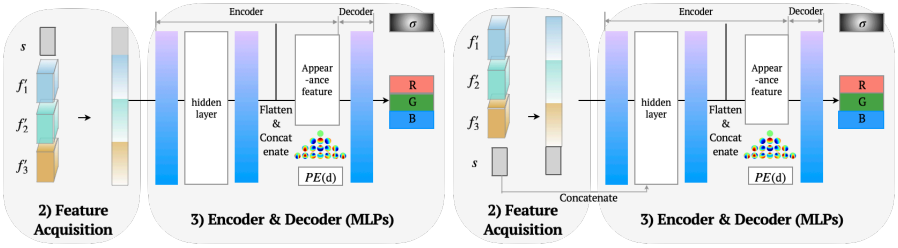
- Type 1 : Skip connection lies on every layer
- Type 2 : No skip connection, and employs fully connected MLPs
- Type 3 : Skip connection, but only coordinate s is concatenated.

In the case of the Dynamic case, smoothness induction in the temporal axis is essential, so the case of Type 3, using only the coordinate feature for skip-connection, shows slightly better performance than ours. Our model design works robustly, considering both static and dynamic cases, which verifies the suitability of model design choices.

Figure H.7: Rendered images of `standup` cases in the dynamic NeRF dataset by HexPlane and ours with varying λ_1. We evaluate {1, 5, 10, 15, 19}th views in the test dataset.



(a) Ours (b) Type 1



(c) Type 2 (d) Type 3

Figure I.8: The graphical representation for encoder structures used in Table I.10 and Table I.11.

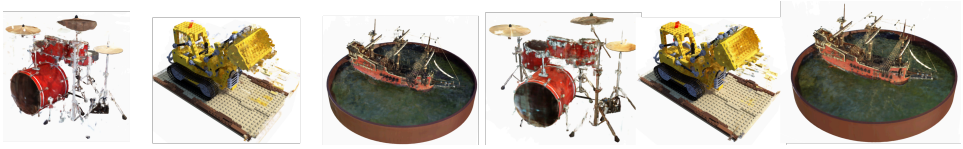
J Disentanglement of Coordinate Networks and Multi-plane Ecoding

Our model learns by separating global shape and detail into coordinate and multi-plane features. Further, we adopt a progressive learning strategy on the channel axis among plane features to induce features to learn coarse-to-fine details. The proposed model shows robust reconstruction performance even when the input is highly sparse, as the proposed model successfully disentangle features into two aspects: (1) between heterogeneous features and (2) among channels in feature planes.

Table I.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 D. 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

(a) Ours (b) Type 1



(c) Type 2 (d) Type 3

Figure I.9: Rendered images are generated by alternating encoder structures. We selected the `drums`, `lego`, and `ship` scenes to follow the settings used in previous experiments.

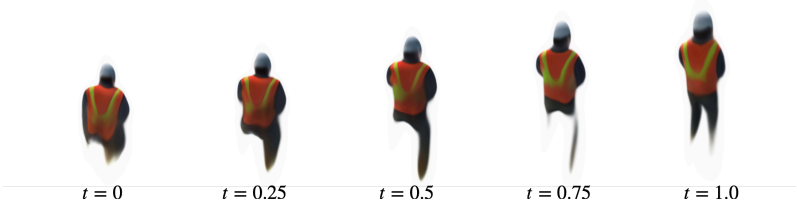
Table I.11: The comparison of encoding structures. We evaluate four types of encoding structures including ours. All models are trained using 25 views in the dynamic NeRF dataset. All hyperparameters are consistent with those described in the original setting included Appendix D. We use seed 0 for reproducibility.

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

First, we analyze the disentanglement between heterogeneous features. We conducted the ablation analysis of the proposed method on the dynamic NeRFs with 25 training views. To qualitatively identify the role that coordinate-based networks in the proposed method, we separately evaluated the model using only the coordinate-based networks, with all multi-plane encodings set to zero. As shown in Figure J.10-(a), the coordinate-based networks are capturing the global context of the scene, including object shapes and large motions as we intended. It is worth noting that the coordinate-based networks can be effectively trained when used in conjunction with multi-plane encoding.

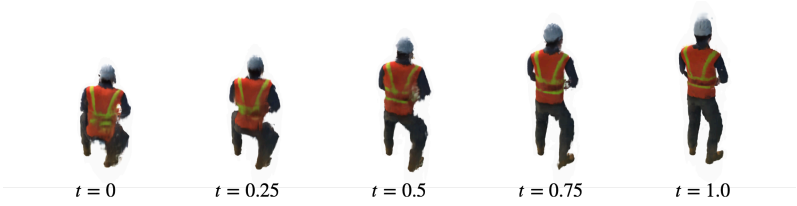
As previous studies have reported that high-frequency features in sinusoidal encoding tend to dominate, we anticipated that multi-plane encoding might overwhelm the coordinate-based networks [8]. However, the proposed architecture successfully disentangles and maintains these two heterogeneous features effectively throughout the training process. This demonstrates the synergy between the coordinate-based networks and multi-plane encoding in the proposed method.

Second, we analyze about channel-wise disentanglement among plane-features. To compare, we visualize multi-plane features of HexPlane and the proposed method, trained on full-view, and 25-views for `standup` scenes (Figure J.11). In `standup`, $z-x$ plane should encode the front shape of the person, and the $z-t$ plane should encode the upward movement.



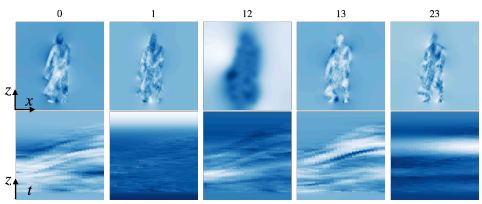
(a) Rendered image by ours with only coordinate networks

(b) Rendered image by ours with curriculum weighting

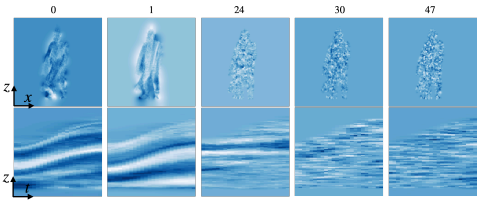


(c) Rendered image with full engangement of multi-plane

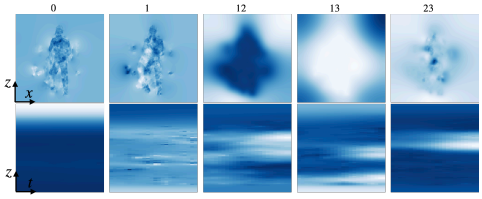
Figure J.10: Rendering results using different feature combinations. We show rendering results from three distinct combination of encoding features, (a) using only coordinates, (b) coordinates with progressively activating multi-plane encoding, and (c) full features. t indicates the timesteps normalized to 1, and we use `standup` scene.



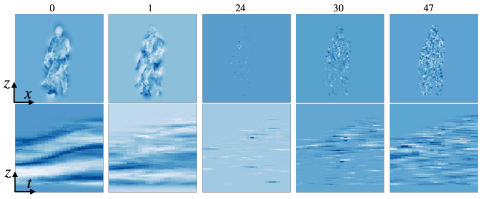
(a) HexPlane($\lambda_1 = 0.001$) trained on full views



(b) Ours trained on full views



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



(d) Ours trained on 25 views

Figure J.11: Visualization of plane encoding features. We visualize 5 representative features from the plane encodings of Hexplane and Ours trained on `standup` scene.

For full-views, in HexPlane plane features (Figure J.11-(a)), global shape information and high-resolution detail are not distinguished and some channels even learn similar information which makes features less expressive. In contrast, the proposed method (Figure J.11-(b)) separates coarse-to-fine information along the channel axis, which effectively increases the expressiveness as each channel encodes information in different areas, In addition, our temporal features show a consistent upward tendency with distinct resolution features, while the hexplane learns similar resolution features with some wrong motions.

This trend is more clearly observable in fewer shots. In Figure J.11-(c), some spatial components result in overfitting or underfitting artifacts, and rightward information is hardly shown on the time axis. In contrast, our model (Figure J.11-(d)) maintains the coarse-to-fine manner. In particular, the trend remains the same with the time axis, confirming how much our progressive learning strategy has in a sparse setting.

From these observations, we verify that our two disentanglement strategies (inter-distinct features and inter-channel) are a way to learn global-to-detail features. This experiment allows it to analyze why the proposed model is more expressive and has robustness in sparse input.