Dynamic PlenOctree for Adaptive Sampling Refinement in Explicit NeRF

Haotian Bai[1] Yiqi Lin[1] Yize Chen[1,2*] Lin Wang[1,2*]

[1] VLIS LAB, AI Thrust, HKUST(GZ) [2] Dept. of CSE, HKUST

haotianwhite@outlook.com, linyq29@gmail.com, yizechen@ust.hk, linwang@ust.hk

Project homepage: https://vlislab22.github.io/DOT/

Abstract

*The explicit neural radiance field (NeRF) has gained considerable interest for its efficient training and fast inference capabilities, making it a promising direction such as virtual reality and gaming. In particular, PlenOctree (POT) [43], an explicit hierarchical multi-scale octree representation, has emerged as a structural and influential framework. However, POT's fixed structure for direct optimization is sub-optimal as the scene complexity evolves continuously with updates to cached color and density, necessitating refining the sampling distribution to capture signal complexity accordingly. To address this issue, we propose the dynamic PlenOctree (**DOT**), which adaptively refines the sample distribution to adjust to changing scene complexity. Specifically, DOT proposes a concise yet novel hierarchical feature fusion strategy during the iterative rendering process. Firstly, it identifies the regions of interest through training signals to ensure adaptive and efficient refinement. Next, rather than directly filtering out valueless nodes, DOT introduces the sampling and pruning operations for octrees to aggregate features, enabling rapid parameter learning. Compared with POT, our DOT outperforms it by enhancing visual quality, reducing over* $55.15/68.84\%$ *parameters, and providing 1.7/1.9 times FPS for NeRF-synthetic and Tanks & Temples, respectively.*

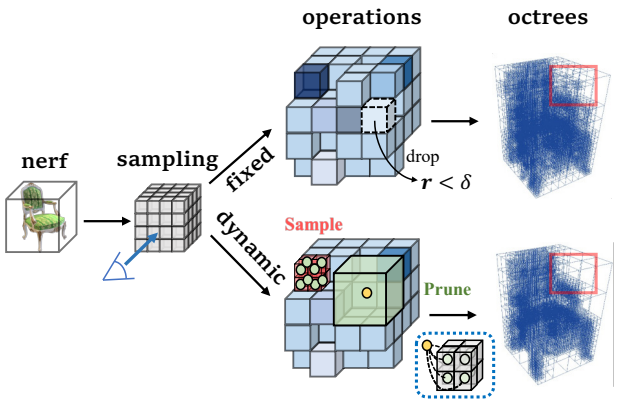

Figure 1: While the POT framework is effective, its fixed octree structure can limit its adaptability to varying scene complexities. We introduce hierarchical feature fusion with sampling/pruning to overcome this limitation, as illustrated by the dashed box below. Varying colors on the grid represent the training signals. Internal nodes are denoted in **orange**, while leaf nodes in **green**. Pruning occurs in regions of weak signal, where cached properties in leaf nodes are aggregated, and the averaged value is propagated to internal nodes, which become the new leaves. Complementary sampling takes place in the red regions. The resulting sampling distribution exhibits significant improvement, as highlighted by the red boxes in our final octree results.

1. Introduction

Rendering photo-realistic scenes and objects is crucial for providing users with an immersive and interactive experience in virtual reality [11, 45] and metaverse [19, 29, 50]. Neural radiance field (NeRF) [26] has emerged as a promising solution for modeling 3D scenes and objects with only calibrated multi-view images. Many subsequent approaches [2, 20, 27, 36, 46] have been proposed to further enhance NeRF's rendering power in terms of the training time, inference speed, and quality. PlenOctree (POT) [43] stands out among these approaches. It employs an explicit octree structure with spherical basis functions to accelerate and enhance the rendering quality. Such method achieves high-quality rendering at over 150 FPS on an NVIDIA V100 GPU, opening up new possibilities for real-time and high-quality rendering, utilizing explicit octrees. Moreover, POT bridges the implicit and explicit NeRFs. Specifically, it demonstrates that POT can transform the implicit NeRFs into the octree representation, further boosting the NeRF training by five orders of magnitudes with the early stop.

Technically, POT's main contributions can be classified

*L. Wang and Y. Chen are the corresponding authors.

into two folds: quality enhancement with non-Lambertian effects and a multi-scale sampling strategy with an octree. Firstly, POT employs spherical harmonics (SH) to model the non-Lambertian view-dependent effects and directly stores them along with the density in POT's leaves for fast training and inference. Secondly, POT employs a multi-scale approach that pre-samples density and SH using a multi-level tabulated volume. The resulting octree structure facilitates capturing intricate features with deeper octree sub-divisions and diversifies the sampling density according to the distribution of signal complexity, *i.e*. the expressiveness of cached color and density through the scene.

However, after the octree construction, POT keeps the division fixed for optimization. We argue such a process is sub-optimal, as the signal complexity can vary during training. Therefore, its initial sample distribution may not provide a sufficient sampling rate, leading to aliasing or oversampling issues. Our proposed dynamic design addresses emerging questions on how to calibrate the spatial distribution and aggregate learned features during its construction. In addition, there has been growing interest in recent research for striking a balance between compactness and expressiveness in NeRF representation, with sampling methods falling into two main categories. The first category, known as importance sampling [6, 26], involves predicting the locations of samples and allocating more resources to complex regions. Although sampling on regions of interest can effectively capture signal complexity, predicting the locations can be computationally expensive, especially when dealing with millions of rays. The second category [21, 44, 7], such as POT, relies on stratified dense sampling, followed by rejecting samples below a certain threshold. While filtering saves computational resources by discarding valueless samples, the heuristic rejection process may accidentally drop valuable samples, potentially lowering performance. Moreover, this process can break the global view consistency as the volume rendering strives to build a consistent 3D representation across all views.

In this paper, we propose a concise yet novel *hierarchical feature fusion* approach that combines the benefits of importance sampling and rejection methods. Specifically, we exploit the evolving signal complexity during training by utilizing the ray weight or density values as training signals through importance sampling, which incurs no additional cost. With these guided signals, we employ a rejection method to prune valueless regions and selectively sample the most promising regions to capture fine details, striking a balance between accuracy and efficiency. We temporarily fix the octree and optimize the cached properties to adapt to recent modification. The entire process is iterative, allowing us to progressively calibrate the octree structure to increase its compactness and capture more details as training progresses.

Notably, *we do not directly drop out voxels but instead, fuse learned features while modifying the octree division*. As depicted in Fig. 1, our novel hierarchical feature fusion approach facilitates adaptive refinement and enables rapid parameter learning through octree sampling and pruning operations. We demonstrate its effectiveness by showing that it can save around 60% parameters while enhancing rendering quality across different scenes.

Our extensive experiments demonstrate the benefits of DOT over POT. Our method enriches rendering views, reduces the number of required parameters by around 60%, and nearly doubles the rendering speed. Furthermore, we offer more control over sampling and pruning strength, which enhances flexibility when working with scenes of varying complexity. Specifically, DOT achieves excellent performance, allowing for rendering an 800x800 image at 452 FPS on an RTX 3090 GPU, achieving 1.8 times the FPS of the POT model.

In summary, this paper makes three major contributions:

- We improve the fixed octree design in POT, allowing for iterative refinement of the octree structure based on training signals iteratively without introducing additional cost.
- We introduce the hierarchical feature fusion strategy to support the adaptive refinement of the octree division, enabling rapid parameter learning by aggregating features via octree sampling/pruning operations.
- Experiments on two benchmark datasets show that DOT can dramatically slim POT and accelerate the rendering speed while improving rendering quality.

2. Related Work

Neural Radiance Fields (NeRF). NeRF [26] uses volume rendering to train coordinate-based MLPs that can directly predict color and opacity based on 3D position and 2D viewing direction. The resulting synthesized views have photo-realistic quality, and the differential volume rendering technique has been widely adopted in various applications, including scene and object relighting [33, 49], unbounded scenes [47, 3], dynamic scenes from videos [10, 30, 38, 35], editable scenes, avatars [23, 41], and object surface reconstruction [1, 37]. Recent advances in NeRF focus on improving training speed [20, 27, 21] and rendering quality [2, 36, 47]. In particular, POT aims to provide a real-time high-fidelity rendering experience, as taking minutes to render a novel view is unsuitable for real-time rendering or 3D interaction. Existing frameworks for boosting rendering speed can be divided into three groups. The first group focuses on designing sampling-friendly models for fast rendering [31, 12, 32, 28, 20, 6]. Secondly, [14, 15, 21, 32] provide instant rendering by baking or caching weights into ex-

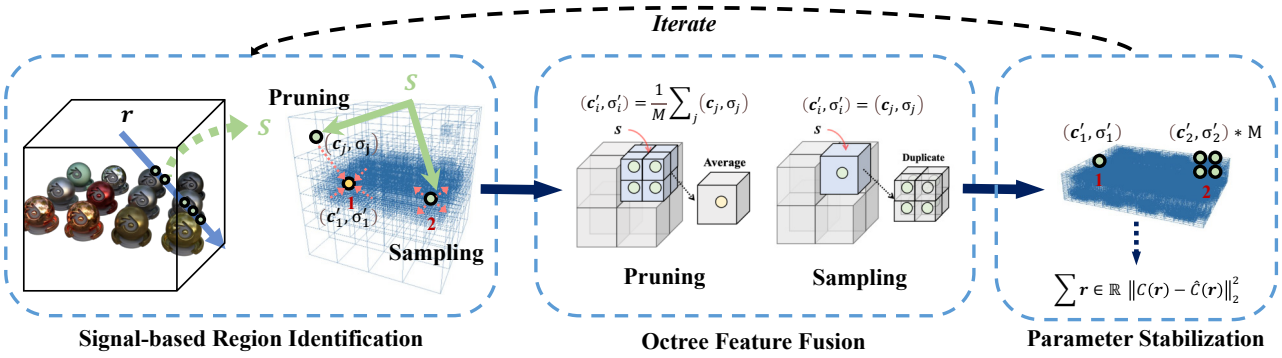

Figure 2: **Overview of DOT pipeline.** The exact grid partitions are **green leaf** nodes, while the **orange internal** nodes are the parent nodes of leaves, only used for octree organization instead of physical space allocation. Guided by the training signal **S**, DOT prunes valueless regions and aggregates their features into the internal nodes "1". It also samples complex areas, such as the leaf nodes "2", by propagating its learned properties into the newly allocated leaf nodes in the next level.

plicit spatial data structures from well-trained NeRF models. The other group of methods [8, 42] represents NeRF using GPU-friendly meshes, boosting the speed by the standard polygon rasterization pipelines and rendering pipelines across different devices.

Sampling Refinement. Dense sampling is usually necessary for pre-sampling the locations along rays to avoid unnecessary computation on sparse regions. Implicit NeRF [26] uses stratified sampling followed by a second network to sample important regions based on previous prediction, while AdaNeRF [6] proposes the dual sampling network to disentangle samples based on sparsity. However, these methods increase inference and training complexity and bring massive costs regarding the millions of rays to render. Recent works optimize this sampling procedure by filtering unnecessary samples with threshold values to avoid additional queries. NSVF [21] rejects points from stratified sampling to accelerate rendering, while [44, 7] filter unnecessary samples based on their rendering contribution after dense sampling. However, the rejection method may accidentally drop valuable samples, potentially lowing performance and breaking the global view consistency as the volume rendering strives to build a consistent 3D representation across all views. Another approach[27, 4, 9] represented by Instant-NGP[27], uses multi-resolution grids that leverage meaningful learning. This method samples the locations that cause significant changes based on the magnitude of gradients in a fixed grid space. However, meaningful learning is not applicable in dynamic scenarios, as the gradients disappear when division is eliminated.

Explicit Spatial Structure in NeRF. Storing learnable features in grid structures is a promising alternative to MLPs for fast rendering because cached values can be accessed directly. Recent research has explored various explicit representations for NeRF, including dense grids [18, 34], sparse 3D voxel grids [21, 7, 43], multiple compact low-rank tensor components [5], 3D point clouds [40], and multi-resolution hash maps [27]. The octree structure has received increasing attention in NeRF due to its multi-scale space division, which allocates more samples to complex regions while quickly skipping sparse regions. ACORN[25] introduced a hybrid implicit-explicit network using octree decomposition and an adaptive resource allocation strategy based on signal complexity. DOT builds on the adaptive sampling idea that uses the training signals to guide the calibration, but we focus more on fusing features in the octree while modifying its structure. This is necessary as features in explicit representations are unable to adapt to the changes as effectively as implicit representations.

3. Method

Overview: DOT introduces a hierarchical feature fusion method designed for octrees. The method comprises three main components: signal-based region identification (see Sec. 3.1), octree feature fusion (see Sec. 3.2), and parameter stabilization (see Sec. 3.3). The DOT pipeline, as shown in Fig. 2, involves an iterative and adaptive modification process based on evolving training signals. Firstly, the method identifies the regions of interest by tracking the instant training signals during the rendering process. Next, guided by the signal, DOT adaptively modifies those regions by pruning and sampling operations, aggregating the features across the octree division. Finally, after stabilizing the learned parameters, DOT iteratively calibrates the octree to capture more details and compact the representation.

Preliminaries: The volume rendering process of POT is fully differentiable, allowing for direct optimization of cached properties such as spherical harmonics (SH) and opacity within the octree structure. Moreover, POT deter-

mines the ray-voxel intersections for each ray in the octree structure, producing a sequence of unevenly distributed sampling segments $\{\delta_i\}_{i=1}^{N_\mathbf{r}}$ that adaptively fit the initial signal complexity by skipping sparse voxels in one step while not missing the small ones. The rendering process, based on the classic work by Kajiya *et al*. [17], infers the color of a ray $\hat{C}(\mathbf{r})$ by integrating $N_\mathbf{r}$ samples along the ray. Specifically, the pixel's predicted color $\hat{C}(\mathbf{r})$ is approximated by accumulating the colors of the samples weighted by Q_i:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N'} T_i Q_i \mathbf{c}_i, \tag{1}$$

$$Q_i = 1 - \exp(-\sigma_i \delta_i), \tag{2}$$

where the light transmitted through a batch of rays $\mathbf{r}$ to sample i is represented by $T_i = \exp(-\sum_{j=0}^{i-1} \sigma_j \delta_j)$; σ_i denotes the opacity of the sample. Q_i denotes the light contribution of sample i, and $\mathbf{c}_i$ is the color vector in the form of SH. $\hat{C}(\mathbf{r})$ is optimized to approximate the ground truth color $C(\mathbf{r})$ by minimizing MSE loss $\sum_{\mathbf{r}\in\mathbb{R}} \|C(\mathbf{r}) - \hat{C}(\mathbf{r})\|_2^2$ over the set of rays $\mathbb{R}$ in the training images.

3.1. Signal-based Region Identification

The accuracy of the predicted color vector $\hat{C}(\mathbf{r})$ in Eq. (1) depends on the quality of $\{\delta_i\}_{i=1}^{N_\mathbf{r}}$, *i.e*. the sampling density to reconstruct $C(\mathbf{r})$ to catch up the changes in scene complexity. Recent studies [39, 7, 15] have highlighted the significance of the signal response such as $\boldsymbol{\sigma}$ and $\boldsymbol{Q}$ in Eq. (2) for regularization when constructing NeRF representations. In the case of POT, the octrees are converted from well-trained implicit NeRFs. Thus resampled values can be taken with high confidence. This allows DOT to leverage these properties directly to guide the latter modifications. In practice, we utilize importance sampling, which prioritizes selecting samples from areas with a high signal response. Additionally, DOT compacts the structure by aggregating features from regions with weak signal values.

Specifically, denote the total number of voxel samples as $N = \sum_\mathbf{r} N_\mathbf{r}$ with each ray interacts with $N_\mathbf{r}$ divisions. The ray weights $\mathbf{Q} = \{\sum_\mathbf{r} Q_{i,\mathbf{r}}\}_{i=1}^{N}$ reflects the overall rendering contribution, and the density $\boldsymbol{\sigma} = \{\sigma_i\}_{i=1}^{N}$. Our prune-only experiment shown in Fig. 3 compares their effectiveness. It demonstrates that choosing $\mathbf{Q}$ as the target training signal eliminates more invisible voxels than $\boldsymbol{\sigma}$, which is crucial for fast ray inference. For instance, the arm of the *lego* model turns out to be more compact with $\mathbf{Q}$. Further quantitative analysis in Tab. 3 proves that the $\mathbf{Q}$ approach is 20% faster than $\boldsymbol{\sigma}$ despite sharing similar memory and fidelity.

3.2. Octree Feature Fusion

After identifying the region to modify based on the training signals, the next step is to determine how to aggregate the cached features during the modification process. A common approach is to discard the voxels with signal values below a fixed threshold. However, simply discarding voxels based on a fixed threshold can lead to the loss of valuable learned features and view consistency in volume rendering, as the volume rendering(Eq. (1)) strives to build a consistent 3D representation across all views. Recent works on explicit NeRFs[7, 21, 44] have shown the benefits of learning properties in a course-to-fine manner, incrementally adding details based on previous estimates. These methods typically begin with dense stratified sampling to learn the feature distribution and context. However, during the transition, learned features are removed, resulting in fewer parameters at the expense of losing globally consistent features.

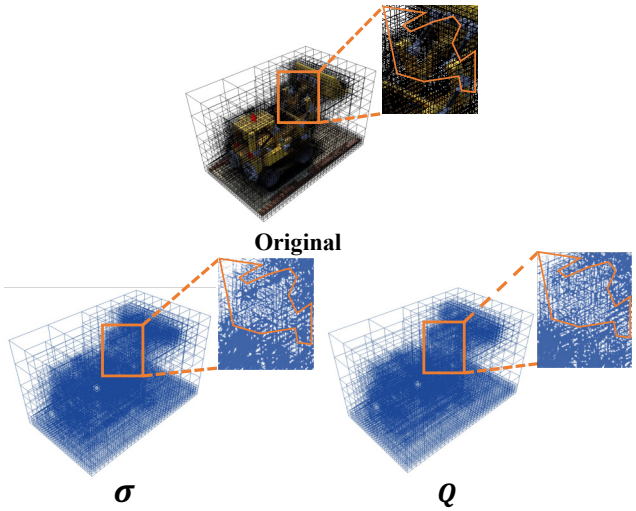

Figure 3: **Comparison on training signals.** We highlight the difference in different choices of training signals, including the opacity $\boldsymbol{\sigma}$ or the ray-weight $\mathbf{Q}$ in the boxed regions, where pruning by $\mathbf{Q}$ eliminates more invisible voxels than $\boldsymbol{\sigma}$. We keep the similar PSNR($\pm$0.05) (Tab. 3) for a fair comparison. Please zoom in to see more details.

Our proposed solution for this issue is to use feature fusion to calibrate the octree structure. We base this approach on the neighboring assumption that features can be propagated across different levels of the octree structure,

$$(\mathbf{c}_i', \sigma_i') = \frac{1}{M} \sum_j (\mathbf{c}_j, \sigma_j), \tag{3}$$

where $M = 8$ is the degree of the octree. Eq. (3) forms the basis of the octree pruning operation. As illustrated in Fig. 2, the internal nodes $\{(\mathbf{c_i}', \sigma_i')\}_i$ collect their leaves' features to represent the larger physical division. Consequently, this approach retains features and their global consistency learned from the volume rendering. Besides, the

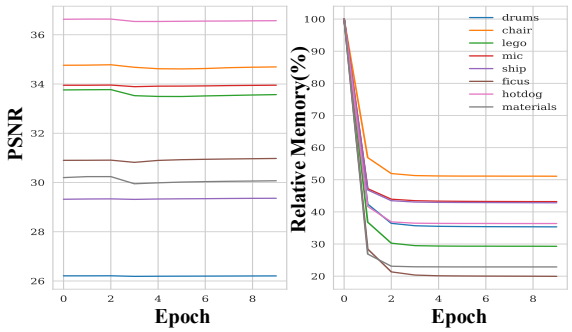

Figure 4: **The training progress with only pruning operation.** Denote the relative memory as the model size DOT/PlenOctree. We train the prune-only model for ten epochs using the signal $\mathbf{Q}$ on the NeRF-synthetic dataset. Surprisingly, DOT's single pruning operation improves the quality in *ficus* while slimming over 80% of memory.

complementary sampling operation can be denoted as,

$$(\mathbf{c}'_j, \sigma'_j) = (\mathbf{c}_i, \sigma_i), \tag{4}$$

which increases the sampling density in the selected partitions $\{(\mathbf{c}'_j, \sigma'_j)\}_j$ and turns $\{(\mathbf{c}_i, \sigma_i)\}_i$ into internal nodes. The effectiveness of our proposed approach is demonstrated in the prune-only experiment(see Fig. 4), where the pruning operation significantly reduces the number of parameters without compromising the quality. This result underscores the importance of adapting the sample distribution in response to changes in signal complexity within the scene. In other words, a fixed sample distribution can result in suboptimal quality and high memory costs.

In addition, we introduce two hyperparameters τ and γ to adjust the strength of pruning and sampling, thereby increasing the flexibility of the signal guidance across different scenes. The selected samples are defined based on signal response with thresholds as $\{i|Q_i \leq \tau\}$ and $\{i|Q_i > \gamma\}$ for a trade-off between memory and performance. We observe that the proposed method provides better control over the level of detail. Specifically, Fig. 6(c) tells the *ship* model pruned with $\tau = 10$ takes surprisingly 3% of memory required by POT while still preserving the necessary details. To achieve more intensive pruning for sparse scenes, we introduce a recursive pruning option that enables marching into the upper part of the octree hierarchy. As shown in Fig. 6(a), the recursive pruning approach is more memory-efficient than the regular one-time pruning while causing a negligible degradation in rendering quality. Specifically, the recursive pruning approach merges nodes whose signal response values are below or equal to the threshold τ in a recursive manner, allowing for more excellent compression of the octree structure. The algorithmic details of the pipeline are presented in *suppl.* materials.

	GPU	Memory↓	PSNR↑	SSIM↑	LPIPS↓	FPS↑
Neural Volumes[24]	Tesla V100	-	23.70	0.834	0.260	1.0
NSVF[22]	Tesla V100	-	31.75	0.953	**0.047**	0.8
AutoInt(8)[20]	Tesla V100	-	25.55	0.911	0.170	0.4
Plenoxels[7]	-	-	31.71	0.958	0.049	∼ 15.0
FastNeRF[12]	RTX 3090	-	29.97	0.941	0.053	238.1
SNeRG[14]	RTX2080Ti	2.71	30.38	0.950	0.050	207.26
MobileNeRF [8]	RTX2080Ti	**0.54**	30.90	0.947	0.062	**744.91**
PlenOctree [43]	RTX3090	1.93	31.71	0.958	0.053	250.1
Ours	RTX3090	0.87	**32.11**	**0.959**	0.053	452.1
Ours(R)	RTX3090	0.80	32.03	0.958	0.054	474.2

Table 1: **Quantitative results on the NeRF-synthetic.** The memory is measured by GB. The memory is measured by GB. The **best** and the second-best results are highlighted. (R) denotes the recursive pruning.

	GPU	Memory↓	PSNR↑	SSIM↑	LPIPS↓	FPS↑
Neural Volumes[24]	Tesla V100	-	23.70	0.834	0.260	1.0
NSVF[22]	Tesla V100	-	28.40	0.900	0.153	0.2
PlenOctree [43]	RTX3090	3.53	28.00	0.917	0.131	74.0
Ours	RTX3090	1.10	28.28	0.922	0.121	186.2
Ours(R)	RTX3090	0.92	28.25	0.922	0.122	216.1

Table 2: **Quantitative results on the Tanks & Temples.**

Methods	Memory(GB)↓	PSNR↑	SSIM↑	LPIPS↓	FPS↑
Sm, Pr	0.815	32.113	0.959	0.053	531.144
Pr_σ	0.611	31.753	0.9578	0.054	419.686
Pr_Q	0.600	31.748	0.9575	0.055	569.288

Table 3: **Ablations on NeRF-synthetic.** We experiment on NVIDIA A100 to evaluate the effectiveness of joint sampling and pruning(with $\boldsymbol{Q}$) (referred to as Sm, Pr) and individual pruning operations targeting different training signals. Specifically, we used Pr_σ to denote pruning based on $\boldsymbol{\sigma}$ and Pr_Q on $\boldsymbol{Q}$.

3.3. Parameter Stabilization

We regularly fix the octree structure for T epochs to allow the model to stabilize the cached properties. Since the structure is revised by pruning and sampling operations, it takes time for the cached values to adapt to the calibrated octree division and update the scene complexity accordingly. To optimize the high-dimensional voxel coefficients, we adopt RMSProp[16] as the optimization algorithm, as the non-convexity of the rendering formula Eq. (1) makes direct optimization challenging. The steady increase in PSNR illustrated in Fig. 6(a) proves the effectiveness.

4. Experiments

4.1. Experiment Setup

Dataset: We used two datasets in our experiments. The first is the NeRF-synthetic dataset, which consists of eight scenes with single objects, including *hotdog*, *materials*, *ficus*, *lego*, *mic*, *drums*, *chair*, and *ship*. The views are at a resolution of 800×800, with 100 views for training and 200

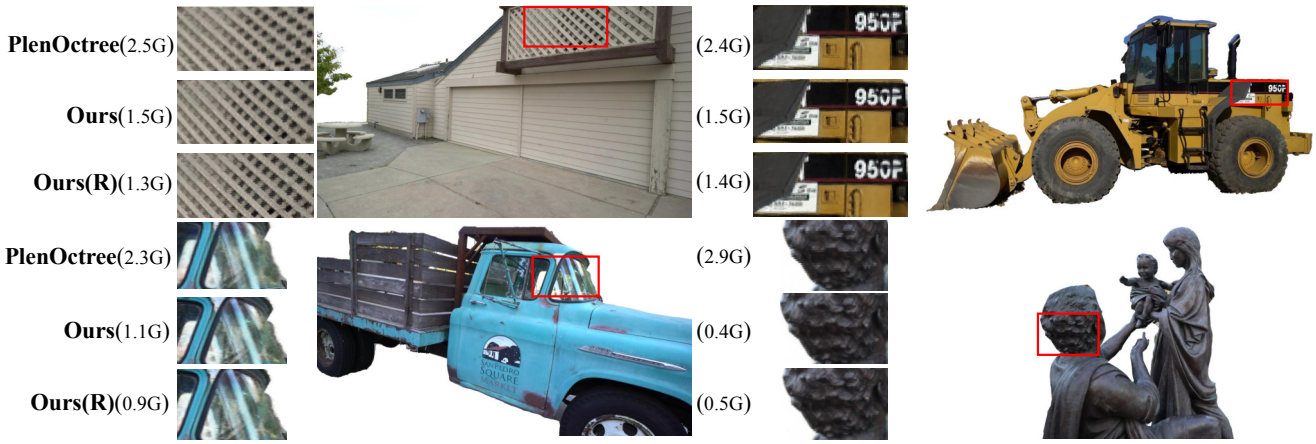

Figure 5: **Tanks** & **Temples qualitative results.** Interestingly, our methods surpass POT in quality with over half parameter size shrinkage, adding more details on the high-frequency regions. Please zoom in to see more details.

for testing per scene. The second dataset is a subset of Tanks & Temples from NSVF [21]. It contains five scenes of real objects captured by an inward-facing camera that circles the scene, and we used foreground masks provided by NSVF. Each scene contains 152-384 images with a resolution of 1920×1080.

Baselines: DOT is built based on the same pretrained NeRF-SH models as POT, where all trained representations store density and SH coefficients converted from NeRF-SH. The grid size is set to 512^3, and the pre-trained models use 16 and 4 SH components for the synthetic and Tanks & Temples datasets, respectively.

Implementation details: We use stochastic image samples to evaluate MSE loss. Specifically, the training pipeline takes 100 epochs, with an interval of $T = 20$. The learning rate is fixed in the main experiments, which is set to 0.1 and 0.01 for σ and $\mathbf{c}$. Furthermore, We apply $\tau = 1, 10$ for the synthetic and the Tanks & Temples datasets, respectively, and γ is set to 0.01 for both.

Compression: To compare the effectiveness of compressed octrees, we use the same median-cut algorithm[13] to quantize the SH coefficients. Our compressed DOT models also support in-browser rendering using WebGL. We directly squeeze the SH-16 POT models for a fair comparison.

4.2. Main Results

We conduct a comprehensive evaluation of DOT compared with POT on both synthetic and real datasets, presented in Tab. 1 and Tab. 2 for the NeRF-synthetic and Tanks & Temples datasets, respectively. Our method achieves significant parameter savings compared to the original POT models, particularly with over 70% memory savings on Tanks & Temples. Moreover, despite having fewer parameters, our method outperforms it in all metrics, including PSNR, SSIM, and LPIPS [48]. The superior performance of DOT can be attributed to its ability to refine the sample distribution to the varied signal complexity dynamically. As demonstrated by Fig. 5, DOT provides more details in complex regions, such as sharper reflections on *windows* and more evident edges on *fences*. Additionally, Fig. 7 shows the more compact structure of DOT, resulting in fewer ray intersections, explaining our significant rendering speed boost. Specifically, in the *materials* scene, the sparse space collapses into a dense box that tightly fits the metal balls while enhancing the density on their surface.

Furthermore, we compare DOT with other state-of-the-art methods, including Neural Volumes[24], NSVF[21], Plenoxels[7], AutoInt[20], FastNeRF[9], SNeRG[14], and MobileNeRF[8]. DOT achieves state-of-the-art rendering quality and stands out regarding memory usage and rendering speed. Compared to MobileNeRF with polygon rasterization, DOT provides more visually appealing rendering results. Furthermore, the gradient-based mesh representations in MobileNeRF may be better suited to certain applications, with the potential crash for inaccurate surfaces or illumination effects. Therefore, it is a trade-off to select between DOT and MobileNeRF for different purposes.

4.3. Ablation Study

In this section, we conduct ablation studies to investigate the design choices of our proposed DOT in detail.

Progressive calibration. We argue that naive rejection by heuristic threshold without further feature fusion leads to suboptimal quality and excessive memory usage. We verify it through the study presented in Fig. 4, which compares the rejection-based POT and feature-fusion-based DOT. The study shows that when only pruning is applied, the PSNR of DOT remains competitive or even outperforms POT after over 60% reduction in memory. Moreover, it also suggests that memory reduction is more intensive in the first

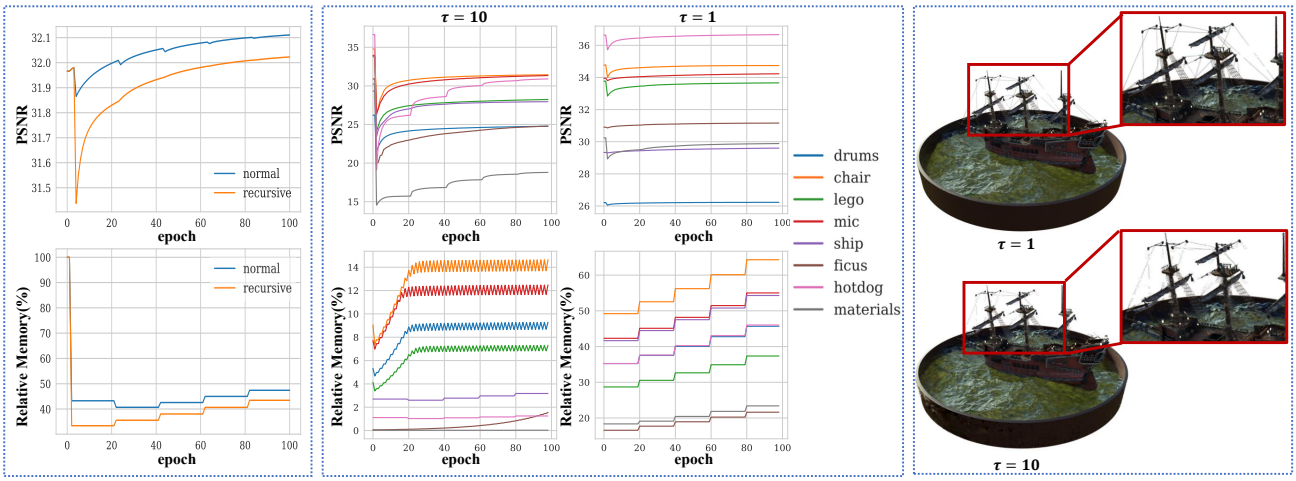

Figure 6: **The ablation studies** Denote the relative memory as the percentage of DOT's memory cost compared to POT, *i.e*. DOT/POT. (a) The comparison between the recursive pruning and the one-time pruning(normal) with the average PSNR and the relative memory usage in NeRF-Synthetic test scenes. We train both models using the same sampling algorithm with different pruning options on the NeRF-Synthetic for 100 epochs. (b) Using a similar experiment setting as (a), we evaluate the effect of pruning strength τ. Specifically, we alter the pruning strength by varying the threshold $\tau = 1$ and $\tau = 10$ to test its effect. (c) The qualitative comparison for (b) on the *ship* scene in the synthetic dataset. Please zoom in to see more details.

few epochs and gradually stabilizes. This could imply that DOT's sample distribution approaches the optimal sampling rate with progressive calibration. We also discover that, the scene like *ficus* can reduce more than 80% parameters while improving rendering quality. Thus, it is not ideal to reject samples heuristically, and DOT makes a difference.

Pruning target. We demonstrated in Fig. 3 that adopting $\boldsymbol{Q}$ as the signal target instead of $\boldsymbol{\sigma}$ yields better results. Here, we provide a quantitative analysis of the outcome. We performed prune-only on both targets for ten epochs, and the training progress for the target $\mathbf{Q}$ is shown in Fig. 4. Table 3 shows that despite similar memory reduction and quality improvement between σ and $\mathbf{Q}$, our results show that the $\boldsymbol{Q}$ approach has a considerable FPS gain of around 20% while maintaining similar quality compared to its counterpart.

Pruning strength. We discuss the option for using recursive pruning to eliminate the unnecessary nodes more thoroughly. As depicted in Fig. 6(a), recursive pruning brings approximately 5-8% more memory reduction but leads to a degradation of about 0.1 PSNR. Therefore, the recursive option is more beneficial for large scenes with many sparse regions, such as *ficus* or *materials*, as shown in Fig. 7. On the other hand, we introduce τ to adjust the strength of pruning to enhance DOT's adaptiveness. To evaluate its effect, we set τ to 1 and 10 and test both memory reduction and PSNR on the NeRF-synthetic dataset. We observe that $\tau = 10$ maintains the major context with only about 3% of POT's parameters, indicating its flexibility to fit different demand levels. However, as shown in Fig. 6(b), higher τ values may set a bottleneck for potential quality enrichment, especially for complex scenes *e.g*., *materials* and *ficus*. For instance, we observe that memory frequently jitters, which may indicate that the key voxels in rendering are dropped, and further sampling operations may struggle to recover the original quality. Therefore, choosing a proper τ is necessary to balance the trade-off between quality and model size.

Necessity of sampling. We propose a complementary sampling operation to enhance high-frequency details and remedy accidental errors introduced from pruning within the high signal response regions. To demonstrate the necessity of this operation, we conduct an experiment comparing the prune-only models and full models, as presented in Table 3. The results demonstrate that sampling improves the PSNR by 0.15dB at the cost of approximately 10% memory that is to be reduced, which is a considerable boost regarding the cost. While pruning alone can improve rendering fidelity by removing both valueless and incorrect samples that do not fit the training signals, its drawback is its inability to increase the sampling rate to keep up with the signal complexity. For instance, *material* scene on the NeRF-synthetic is a challenging case to render due to its highly varied surface reflection, which requires a dense sampling distribution around the metal balls' surface. However, as shown in Figure Fig. 6(b), the joint operation preserves the original quality of the scene as the training progresses, while the prune-only experiment in Fig. 4 shows a decreased PSNR after

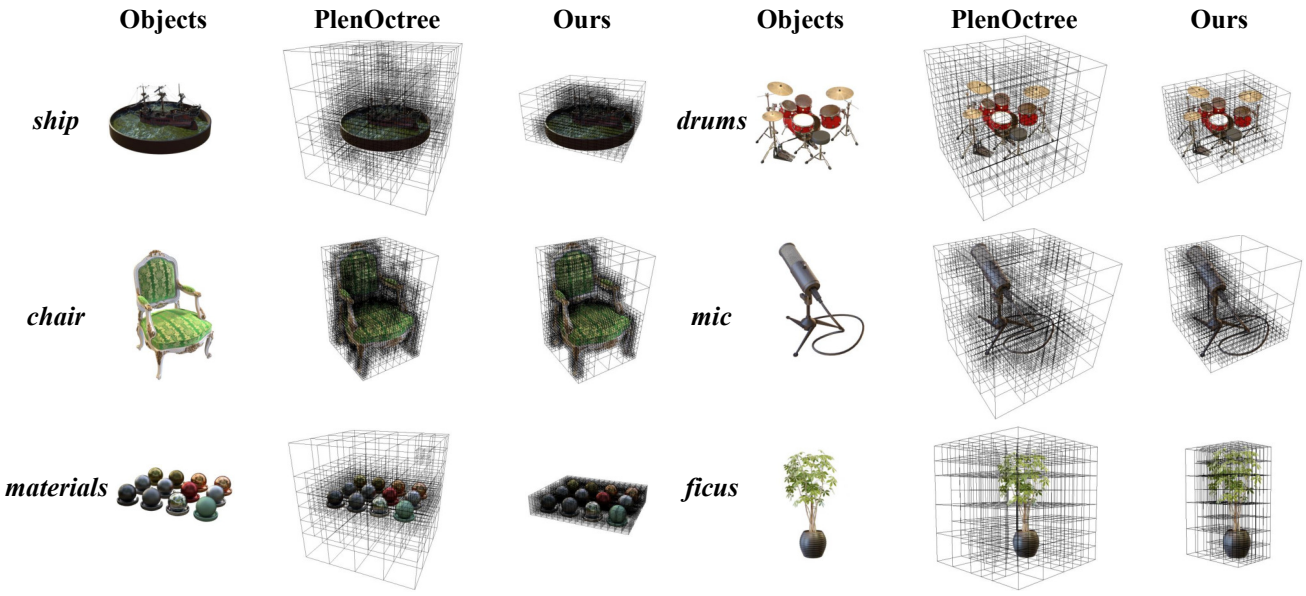

Figure 7: **NeRF-synthetic qualitative results**. We find that our methods produce more compact Octree structures.

tt/syn	Full Models			Compressed Models		
Method	PlenOctree	Ours	Ours(R)	PlenOctree	Ours	Ours(R)
Memory(GB)	2.62/1.94	0.82/1.1	0.79/0.92	0.65/0.39	0.26/0.20	0.23/0.19
FPS (A100)	91/326	177/487	202/518	113/329	183/531	208/565
FPS (RTX3090)	74/250	186/452	216/474	109/289	270/467	239/492
FPS (MX150)	✗/✗	✗/11	✗/12	✗/✗	3/12	3/13

Table 4: **The cross-device test on memory, and FPS on NeRF-Synthetic(syn) and Tanks & Temples scenes(tt).** Denote the abbreviations for NVIDIA devices including A100 PCIE 40GB as A100, GeForce RTX 3090 24GB as RTX3090, and GeForce MX150 2GB as MX150. ✗denotes not available due to overflow, and each item is the performance on tt/syn.

pruning. It tells that the high-frequency features of surface reflections (the parameter stabilization box) in Fig. 2 may not be recovered properly without sampling. Therefore, we assert that sampling is essential for error correction and a further improvement in rendering quality.

4.4. Discussion

In this section, we delve deeper into the aspects of DOT's rendering speed and training time.

Doubled rendering speed. As illustrated in Fig. 7, POT comprises a substantial number of voxel regions distributed in empty spaces and unnecessary divisions, leading to increased computation and memory consumption. In contrast, DOT exhibits more compact spatial divisions than POT, allowing the rays to bypass sparse and redundant areas.

Negligible training time. DOT solely calibrates octrees extracted from the implicit NeRFs, resulting in a fast training process. The signal-based search algorithm scales linearly with the octree size. Therefore, as demonstrated in Fig. 6, it initially reduces more than 50% of parameters, leading to enhanced efficiency due to the smaller size.

4.5. Cross-device Analysis

DOT is adaptable to scenarios with varied computational resources. We evaluate our method on the devices under distinct conditions, including A100-PCIE-40 GB, GeForce-RTX-3090-24GB, and GeForce-MX150-2GB. Specifically, MX150 was announced in mid-2017, tailored for mobile devices such as thin and light laptops. As is shown in Tab. 4, we compare the run-time memory cost and the inference time for three devices with both full and compressed models. Our models successfully accelerate the original models about 1.5 times FPS in the NeRF-Synthetic. For Tanks & Temples, it raises the rendering speed about 2-3 times. Notably, our full models perform at 11 FPS on MX150(only 2GB memory), while PlenOctree fails to launch. After compression, the memory reduction remains effective, bringing our models' access to more challenging datasets like the Tanks & Temples. It proves DOT's ability to be applied more wildly in web rendering, especially for situations with limited resources. *We welcome all readers to refer to the recorded videos, including the real device tests and the vi-*

sual quality comparisons in our suppl. materials.

5. Conclusion and Future Work

We propose DOT, a dynamic structure to address the limitation of the fixed octree design in POT and allow for adaptive adjustment of the octree division for a more memory-efficient representation with higher quality. Compared with the original POT model, our method successfully shrinks over half of the model size, raises the rendering speed about one time, and even enhances the quality. DOT is buttressed by the hierarchical feature fusion strategy during the iterative rendering process, which maintains the globally consistent features instead of dropping them out. Moreover, our model can be applied to more scenarios with limited computational resources with flexible control over the strength of pruning/sampling operations. However, our proposed method cannot reduce the excessive training time for its precursor NeRF-SH, which is required for both POT and DOT, from which they resample and cache the learned properties into the octree leaves for fast inference and optimization. In the future, we plan to explore extensions of our method, enabling the model training from scratch by methods such as reinforcement learning to automatically allocate samples with signal guidance to construct the 3D objects.

Acknowledgement: This paper is supported by the National Natural Science Foundation of China (NSF) under Grant No. NSFC22FYT45 and the Guangzhou City, University and Enterprise Joint Fund under Grant No. SL2022A03J01278.

References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6290–6301, 2022. 2

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5835–5844. IEEE, 2021. 1, 2

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

[4] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, 2022. 3

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

[6] Andreas Kurz et.al. Adanerf: Adaptive sampling for real-time rendering of neural radiance fields. In *ECCV*, 2022. 2, 3

[7] Alex Yu et.al. Plenoxels: Radiance fields without neural networks. *CVPR*, 2022. 2, 3, 4, 5, 6

[8] Zhiqin Chen et.al. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv*, 2022. 3, 5, 6

[9] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. *SIGGRAPH Asia 2022 Conference Papers*, 2022. 3, 6

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

[11] Kyle Gao, Yina Gao, Hongjie He, Denning Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022. 1

[12] Stephan J et.al Garbin. Fastnerf: High-fidelity neural rendering at 200fps. In *ICCV*, 2021. 2, 5

[13] HeckbertPaul. Color image quantization for frame buffer display. *Computer Graphics*, 1982. 6

[14] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021. 2, 5, 6

[15] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul E. Debevec. Baking neural radiance fields for real-time view synthesis. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, 2021. 2, 4

[16] Geoffrey Hinton. Coursera neural networks for machine learning lecture 6, 2018. 5

[17] James T. Kajiya and Brian Von Herzen. Ray tracing volume densities. *international conference on computer graphics and interactive techniques*, 1984. 4

[18] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 3

[19] Lik-Hang Lee, Tristan Braud, Pengyuan Zhou, Lin Wang, Dianlei Xu, Zijun Lin, Abhishek Kumar, Carlos Bermejo, and Pan Hui. All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda. *arXiv preprint arXiv:2110.05352*, 2021. 1

[20] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14556–14565, 2021. 1, 2, 5, 6

[21] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *ArXiv*, abs/2007.11571, 2020. 2, 3, 4, 6

[22] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 5

[23] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM Transactions on Graphics (TOG)*, 40(6):1–16, 2021. 2

[24] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38, 2019. 5, 6

[25] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *ACM Trans. Graph. (SIGGRAPH)*, 40(4), 2021. 3

[26] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, volume 12346, pages 405–421. Springer, 2020. 1, 2, 3

[27] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 1, 2, 3

[28] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In *Computer Graphics Forum*, volume 40, pages 45–59. Wiley Online Library, 2021. 2

[29] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human

bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14314–14323, 2021. 1

[30] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2

[31] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14153–14161, 2021. 2

[32] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021. 2

[33] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021. 2

[34] 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. 3

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

[36] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd E. Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *CVPR*, pages 5481–5490. IEEE, 2022. 1, 2

[37] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 34:27171–27183, 2021. 2

[38] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021. 2

[39] Wenpeng Xing and Jie Chen. Mvsplenoctree: Fast and generic reconstruction of radiance fields in plenoctree from multi-view stereo. *Proceedings of the 30th ACM International Conference on Multimedia*, 2022. 4

[40] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 3

[41] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021. 2

[42] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *ArXiv*, abs/2302.14859, 2023. 3

[43] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 1, 3, 5

[44] Alex et.all Yu. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2, 3, 4

[45] Jiakai Zhang, Liao Wang, Xinhang Liu, Fuqiang Zhao, Minzhang Li, Haizhao Dai, Boyuan Zhang, Wei Yang, Lan Xu, and Jingyi Yu. Neuvv: Neural volumetric videos with immersive rendering and editing. *arXiv preprint arXiv:2202.06088*, 2022. 1

[46] Jian Zhang, Yuanqing Zhang, Huan Fu, Xiaowei Zhou, Bowen Cai, Jinchi Huang, Rongfei Jia, Binqiang Zhao, and Xing Tang. Ray priors through reprojection: Improving neural radiance fields for novel view extrapolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18376–18386, 2022. 1

[47] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv: Computer Vision and Pattern Recognition*, 2020. 2

[48] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[49] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021. 2

[50] Pengyuan Zhou, Jinjing Zhu, Yiting Wang, Yunfan Lu, Zixiang Wei, Haolin Shi, Yuchen Ding, Yu Gao, Qinglong Huang, Yan Shi, et al. Vetaverse: Technologies, applications, and visions toward the intersection of metaverse, vehicles, and transportation systems. *arXiv preprint arXiv:2210.15109*, 2022. 1