Strivec: Sparse Tri-Vector Radiance Fields

Quankai Gao*[1] Qiangeng Xu*[1] Hao Su[2] Ulrich Neumann[1] Zexiang Xu[3]
[1]University of Southern California [2]UC San Diego [3]Adobe Research
{quankaig,qiangenx,uneumann}@usc.edu haosu@ucsd.edu zexu@adobe.com

Abstract

We propose Strivec, a novel neural representation that models a 3D scene as a radiance field with sparsely distributed and compactly factorized local tensor feature grids. Our approach leverages tensor decomposition, following the recent work TensoRF [7], to model the tensor grids. In contrast to TensoRF which uses a global tensor and focuses on their vector-matrix decomposition, we propose to utilize a cloud of local tensors and apply the classic CANDECOMP/PARAFAC (CP) decomposition [5] to factorize each tensor into triple vectors that express local feature distributions along spatial axes and compactly encode a local neural field. We also apply multi-scale tensor grids to discover the geometry and appearance commonalities and exploit spatial coherence with the tri-vector factorization at multiple local scales. The final radiance field properties are regressed by aggregating neural features from multiple local tensors across all scales. Our tri-vector tensors are sparsely distributed around the actual scene surface, discovered by a fast coarse reconstruction, leveraging the sparsity of a 3D scene. We demonstrate that our model can achieve better rendering quality while using significantly fewer parameters than previous methods, including TensoRF and Instant-NGP [27].

1. Introduction

Representing 3D scenes as radiance fields [26] has enabled photo-realistic rendering quality and emerged as a popular design choice in 3D vision and graphics applications. While many methods [31, 46, 3] (following NeRF [26]) purely use MLPs to represent neural fields, recent works, like TensoRF [7] and Instant-NGP [27], have demonstrated the advantages of using shared global feature encoding for radiance field modeling, in terms of speed, compactness, and quality. However, these methods share and assign neural features uniformly in a scene (with tensor factors or hash tables), assuming the scene content is equally complex over the entire space, which can be inefficient (requiring high model capacity) to accurately model intricate local scene details (see Fig.1).

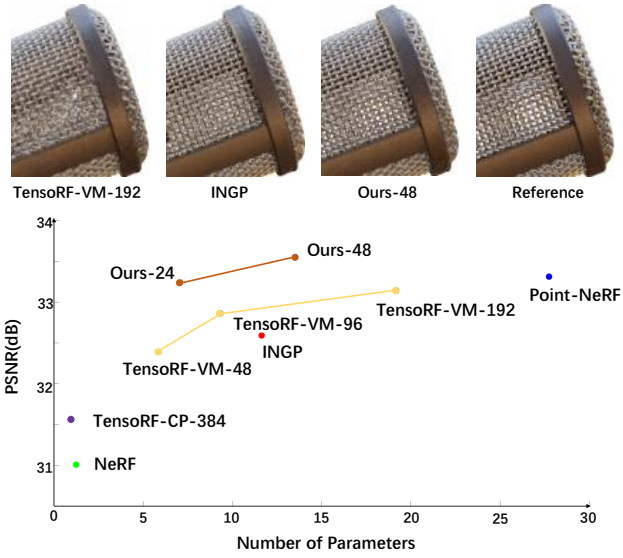

Figure 1: We compare with previous methods in terms of rendering quality (PSNR) and model capacity (number of parameters) on the NeRF Synthetic dataset on the bottom. Our method and TensoRF are shown with different model sizes. Our approach consistently achieve better rendering quality with fewer model parameters than TensoRF, as well as other methods like iNGP. On the top, we show one example of visual comparisons of the mic scene that has challenging fine-grained geometric structures, where our approach captures most of the details and is the closest to the reference. Note that the results of NeRF and Point-NeRF use 200k optimization steps while the rest use only 30k steps.

We aim to accurately and compactly model a 3D scene and reproduce the complex local details. To this end, we propose Strivec, a novel neural scene representation that utilizes *sparsely distributed* and *compactly factorized* local tensor grids to model a volumetric radiance field for high-

*Equal contribution.
Code and results: https://github.com/Zerg-Overmind/Strivec

quality novel view synthesis. As shown in Fig.1, our approach is able to accurately model the complex scene structures that are not recovered well by previous methods. More importantly, our superior rendering quality is achieved with much less model capacity.

In particular, we base our model on TensoRF [7], a recent approach that leverages tensor factorization in radiance field modeling. It is fast, compact, and of high rendering quality. TensoRF applies CP and vector-matrix (VM) decomposition techniques to factorize a field into vectors and matrices and model the entire scene as a global factorized tensor. Instead of a single global tensor, we leverage a sparse set of multiple small local tensors distributed around the scene surface for more efficient scene modeling. Specifically, each of our tensors represents a local radiance field inside its local bounding box and is compactly modeled with factorized triple vectors based on the CP decomposition.

Note that the global CP decomposition in TensoRF has led to a highly compact model but cannot achieve comparable rendering quality to their VM decomposition. This is because a tri-vector CP component is rank-one, while a global feature grid of an entire 3D scene is often complex and of high rank, requiring a large (impractical) number of CP components for high accuracy. TensoRF addresses this by introducing matrix factors in their VM decomposition, essentially increasing the rank of each tensor component. Our model instead consists of multiple small tensor grids, exploiting local spatial commonalities in a scene. Compared to a global tensor, our local tensor is less complex and of much lower rank, thus effectively reducing the required number of CP components (per tensor) and enabling practical high-quality radiance field reconstruction with highly compact tri-vector factors. Our local tri-vector tensors can lead to superior rendering quality and compactness over TensoRF's VM model (see Fig. 1). We also observe that our local tensors are generally more robust than a global tensor against the orientation of spatial axes (which can affect the rank of a tensor and thus affects the quality; see Fig. 2).

Importantly, adopting local tensors (instead of a global one) also brings us the flexibility to allocate neural features according to the actual scene distribution, enabling more efficient scene modeling and better usage of model parameters than a global representation. To do so, we pre-acquire coarse scene geometry – that can be easily achieved via a fast RGBσ volume reconstruction (like DVGO [36]) or multi-view stereo (like Point-NeRF [43]) – to directly distribute local tensors around the actual scene surface, leading to a sparse scene representation that avoids unnecessarily modeling the empty scene space. Note that while previous methods have also leveraged sparse representations (with voxels [22, 45] or points [43]) of radiance fields, their local features are modeled and optimized independently. Our model instead correlates a group of local features inside a local box and compactly express them with triple vectors, uniquely exploiting the local spatial coherence along axes and imposing local low-rank priors in the feature encoding via tensor factorization. Moreover, unlike previous sparse representations that only use a single-scale feature grid or point cloud, we distribute multi-scale local tensors to effectively model the scene geometry and appearance at multiple scales in a hierarchical manner. In particular, for an arbitrary 3D location, we aggregate the neural features from its neighboring tri-vector components at all scales and decode the volume density and view-dependent color from the aggregated features for radiance field rendering.

Our approach takes the best of previous local and global radiance field representations. Compared with global representations like TensoRF and Instant-NGP, our model takes advantage of the sparsity of a scene more directly; compared with local representations like Plenoxels and Point-NeRF, our model makes use of the local smoothness and coherence of scene geometry and appearance. As shown in our experimental results on both synthetic and real datasets, our model is able to achieve state-of-the-art rendering quality on these datasets, outperforming previous methods, including TensoRF and Instant-NGP, while using significantly fewer model parameters, demonstrating the superior representational power of our model.

2. Related Work

Scene representations. To represent a 3D scene, traditional and learning-based methods have studied various representations, such as depth map [16, 21], mesh [18, 40, 34], point cloud [32, 1, 39] and implicit function [10, 25, 28, 44]. In recent years, continuous neural field representations stand out in various 3D tasks such as single-view 3D reconstruction [42, 14], surface completion [11, 30], multi-view reconstruction [28] and novel view synthesis [26, 24]. Compared with traditional discrete representations, a continuous field have no limitation on spatial resolution, e.g., volume resolution or the number of points. It can also naturally be represented by neural networks, such as an MLP, which are known for approximating complex functions well.

Neural field representations. Specifically, NeRF [26] represents a 3D scene as a radiance field with a global coordinate MLP, which models geometry, lighting and texture information jointly, leading to photo-realistic rendering quality in novel view synthesis. Apart from its advantage, purely MLP-based NeRF models [3, 38] in general suffer from inefficiency [2] when modeling highly complex or large-scale scenes, due to limited model capacity, slow optimization speed, and the cost of modeling empty space.

To model radiance fields more efficiently, recent works have explored combining neural fields with various traditional 3D representations, including voxels [22, 45, 36, 48]

and points [43]. Low-rank representations such as tri-plane [6, 13] and tensor decomposition [7, 29] have also been studied. In particular, DVGO [36] and Plenoxels [45] respectively use dense and sparse voxels with neural features for radiance field modeling. While being efficient to optimize, these localized feature grid-based representations lead to a large model size and can face overfitting issues when the features are of very high resolution. Consequently, DVGO can also work with a low-resolution grid and Plenoxels requires additional spatial regularization terms. On the other hand, recent works have adopted global feature encoding to express a high-resolution feature grid, including Instant-NGP [27] that hashes spatial features into multi-scale hash tables and TensoRF [7] that factorizes a feature grid into vector and matrix factors. These global feature encoding methods exploit the spatial correlation across the entire scene space, leading to fast and compact reconstruction and surpassing previous MLP-based or grid-based representations on rendering quality. However, similar to NeRF, such global representation can also be limited by its model capacity when representing highly complex or large-scale content.

Our approach instead combines local and global representations. Our tri-vector fields are sparsely distributed in the scene, similar to local representations (like plenoxels and Point-NeRF); meanwhile, features in each field are represented by tri-vector components shared across the local region as done in TensoRF, exploiting spatial feature commonalities. Our model leverages both spatial sparsity and coherence, leading to much higher compactness and better reconstruction quality than previous local and global representations (see Tab. 1).

Relevant to our work, previous methods, such as KiloNeRF [33] and BlockNeRF [37] have also utilized multiple local MLPs to represent a scene. Specifically, KiloNeRF focuses and speeding up NeRF and their rendering quality is sacrificed; BlockNeRF essentially uses multiple NeRFs to increase the total model capacity. Instead of pure MLPs, our work is built upon tensor factorization-based feature encoding as done in TensoRF [7], and we in fact achieve superior rendering quality while decreasing the model capacity.

3. Sparse Tri-Vector Field Representation

We now present our novel radiance field representation. In essence, our model consists of a cloud of small local tri-vector tensors at multiple scales, designed to leverage both sparsity and multi-scale spatial coherence (see Fig. 2).

Let $\mathcal{T} = \{\tau_n | n = 1, ..., N\}$ denote a cloud of tri-vector tensors. Each local tensor τ is located at p, covering a local cuboid space ω with an edge length of l. This cloud of tri-vector tensors represents a radiance field for the 3D space:

$$\Omega = \bigcup_{n=1}^{N} \omega_n. \tag{1}$$

Here, each tensor τ encodes a local multi-channel feature grid that includes a (single-channel) density grid A_σ and a (multi-channel) appearance grid A_c, similar to the tensor grid in TensoRF [7]. In contrast to using a single global tensor in TensoRF [7], we model the volume density and view-dependent colors with multiple local tensors. In particular, for an arbitrary location $\chi \in \Omega$, we select M nearest tensors that cover χ. Across the selected tensors, we aggregate the extracted density and appearance features recovered by their tri-vector factors for radiance field property regression, where the volume density σ is directly obtained after the aggregation and the view-dependent color c is regressed by a small MLP ψ along with the viewing direction $\mathbf{d}$. The continuous radiance field can be expressed as:

$$\sigma_\chi, c_\chi = A_\sigma(\{\mathcal{G}^\sigma(\chi)\}), \psi(A_c(\{\mathcal{G}^c(\chi)\}), \mathbf{d}). \tag{2}$$

3.1. Local tri-vector tensors.

We apply the classic Canonical polyadic (CP) decomposition [5] to model our local tensors with tri-vector components.

CP decomposition. CP decomposition factorizes a M dimension tensor $\tau \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_M}$ into a linear combination of R rank-1 tensors:

$$\tau = \sum_{r=1}^{R} \lambda_r \mathbf{v}_r^0 \otimes \mathbf{v}_r^1 \otimes ..., \otimes \mathbf{v}_r^M, \tag{3}$$

where $\otimes$ denotes outer product; the weighting factor λ_r can be absorbed into vectors $\{\mathbf{v}_r^0, ..., \mathbf{v}_r^M\}$.

Density and appearance tensors. In our case of modeling a 3D radiance field, we set the geometry grid $\mathcal{G}^\sigma \in \mathbb{R}^{I \times J \times K}$ as a 3D tensor. And the multi-channel appearance grid $\mathcal{G}^c \in \mathbb{R}^{I \times J \times K \times P}$ corresponds to a 4D tensor. The fourth appearance mode is of lower dimension (compared with the spatial modes), representing the final dimension of the features sent to the MLP decoder network.

According to Eqn.3, we factorize each tensor's feature grids, $\mathcal{G}^\sigma$ and $\mathcal{G}^c$, by CP decomposition:

$$\mathcal{G}^\sigma = \sum_{r=1}^{R_\sigma} \mathcal{A}_{\sigma,r} = \sum_{r=1}^{R_\sigma} \mathbf{v}_{\sigma,r}^X \otimes \mathbf{v}_{\sigma,r}^Y \otimes \mathbf{v}_{\sigma,r}^Z, \tag{4}$$

$$\mathcal{G}^c = \sum_{r=1}^{R_c} \mathcal{A}_{c,r} \otimes \mathbf{b}_r = \sum_{r=1}^{R_c} \mathbf{v}_{c,r}^X \otimes \mathbf{v}_{c,r}^Y \otimes \mathbf{v}_{c,r}^Z \otimes \mathbf{b}_r, \tag{5}$$

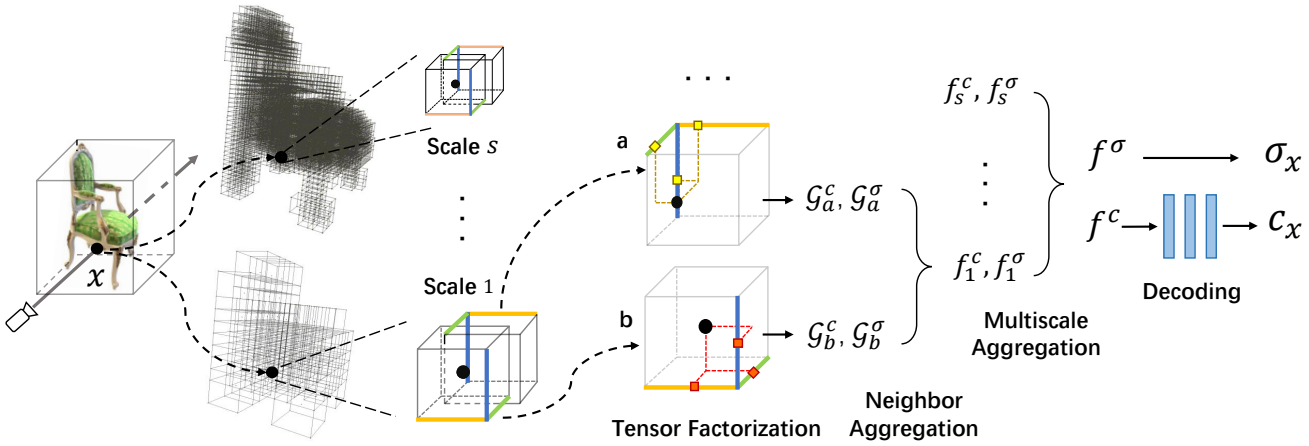

Figure 2: Overview of our Sparse Tri-Vector Radiance Fields. We distribute our local tensors based on a coarse geometry estimated by a fast RGBσ volume reconstruction as done in DVGO [36]. Here, we show our model running under $S = 2$ scales. Each local tensor is factorized as axis-aligned triple based on CP decomposition. For any shading point χ, we extract and evaluate features in each local tensor, according to the factorization (Sec. 3.1). Then, we aggregate these features among nearby tensors (Sec. 3.2) and across different scales (Sec. 3.3). Finally, the density and color are decoded (Sec. 3.4) and used by volume rendering (Sec.4).

Here R_σ and R_c denote numbers of component; $\mathcal{A}_{\sigma,r}$ and $\mathcal{A}_{c,r}$ are the component tensors that are factorized spatially; $\mathbf{v}^X_{\sigma,r}, ..., \mathbf{v}^X_{c,r}, ...$ are the 1D vectors with resolution I, J, K, modeling scene geometry and appearance along X, Y, Z axis; R_σ and R_c are the component numbers; $\mathbf{b}_r$ expresses the feature dimension.

As done in TensoRF [7], we stack all feature-mode vectors $\mathbf{b}_r$ as columns together, which ends up a $P \times R_c$ appearance matrix $\mathbf{B}$. This matrix models the appearance feature variations of the tensor and functions like a appearance dictionary. Note that naively following CP decomposition like TensoRF will assign a different appearance matrix for every local tensor. Instead, we propose to utilize a global appearance matrix $\mathbf{B}^c$ shared across the entire cloud of local tensors, leading to a global appearance dictionary that explains the color correlations across scene. This further improves both the computational efficiency and model compactness of our model.

Therefore, each of our local tensors is represented by their unique local tri-vector factors $\mathbf{v}^X_r$, $\mathbf{v}^Y_r$, $\mathbf{v}^Z_r$.

Feature evaluation. To achieve a continuous field, we consider trilinear interpolation when evaluating the tensor grid features. For a location χ, we first compute its relative position $\tilde{\chi}$ to the selected tensor located at p:

$$\tilde{x}, \tilde{y}, \tilde{z} = x - p_x, y - p_y, z - p_z. \tag{6}$$

Then, for example, to get $\mathcal{A}_{\sigma,r}$ at $(\tilde{x}, \tilde{y}, \tilde{z})$, we can compute and trilinearly interpolate eight $\mathcal{A}_{\sigma,r}$ on the corners. As mentioned in [7], applying linear interpolation on each vector first is mathematically equivalent and can reduce the computation cost. Under the rule of outer product, we have $\mathcal{A}_{r,i,j,k} = \mathbf{v}^X_{r,i}\mathbf{v}^Y_{r,j}\mathbf{v}^Z_{r,k}$, then the interpolated density features at location χ are:

$$\mathcal{G}^\sigma(\chi) = \sum_r \mathbf{v}^X_{\sigma,r}(\tilde{x})\mathbf{v}^Y_{\sigma,r}(\tilde{y})\mathbf{v}^Z_{\sigma,r}(\tilde{z}) = \sum_r \mathcal{A}_{\sigma,r}(\tilde{\chi}), \tag{7}$$

where $\mathbf{v}^X_{\sigma,r}(\tilde{x})$ is $\mathbf{v}^X_{\sigma,r}$'s linearly interpolated value at $(\tilde{x})$ along its X axis. Here, $\mathcal{G}^\sigma(\chi)$ is a scalar.

Similarly, the interpolated appearance features can be computed as:

$$\mathcal{G}^c(\chi) = \sum_r \mathbf{v}^X_{c,r}(\tilde{x})\mathbf{v}^Y_{c,r}(\tilde{y})\mathbf{v}^Z_{c,r}(\tilde{z})\mathbf{b}_r \tag{8}$$

$$= \sum_r \mathcal{A}_{c,r}(\tilde{\chi})\mathbf{b}_r \tag{9}$$

$$= \mathbf{B} \cdot (\oplus[\mathcal{A}_{c,r}]_r), \tag{10}$$

where "$\oplus$" denotes concatenation, "$\cdot$" denotes dot product. The appearance feature $\mathcal{G}^c(\chi) \in \mathbb{R}^P$ is a vector.

3.2. Feature aggregation.

We propose to aggregate the features from M neighboring tensors to jointly model the volume density and appearance for each 3D location χ. In particular, inspired by Point-NeRF, we leverage an inverse distance-based weighting function to directly aggregate the multi-tensor features. Specifically, this weight can be expressed by

$$w_m = \frac{1}{\|p_m - \chi\|}. \tag{11}$$

With this weight function, we directly obtain the density feature via the weighted sum:

$$f^{\sigma}(\chi) = \frac{1}{\sum w_m} \sum_{m=1}^{M} w_m \mathcal{G}_m^{\sigma}(\chi). \tag{12}$$

Similarly, the appearance feature aggregation can be expressed in a similar way, while using the shared appearance matrix (as described in Sec. 3.1) across local tensors:

$$f^{c}(\chi) = \frac{1}{\sum w_m} \sum_{m=1}^{M} w_m \mathcal{G}_m^{c}(\chi) \tag{13}$$

$$= \frac{1}{\sum w_m} \sum_{m=1}^{M} w_m \mathbf{B}^{c} \cdot (\oplus[\mathcal{A}_{c,r}]_r) \tag{14}$$

$$= \frac{1}{\sum w_m} \mathbf{B}^{c} \cdot (\sum_{m=1}^{M} w_m (\oplus[\mathcal{A}_{c,r}]_r)). \tag{15}$$

Note that owing to sharing the appearance matrix across tensors, we reduce the computational complexity from $O(M \cdot P \cdot R_c)$ in Eqn.14, to $O((M+P) \cdot R_c)$ in Eqn.15.

3.3. Multi-scale tri-vector fields.

Complex 3D scenes often contain multi-frequency geometry and appearance details. This motivates us to build multi-scale tensor clouds to discover the local geometry and appearance commonalities at multiple scales. Our final radiance field is modeled by multiple tri-vector tensor clouds at S different scales. Different clouds consist of tensors with different resolutions.

To regress the density and appearance at a location χ, we gather the density and appearance features from a set of tensor clouds that cover χ, $\{\mathcal{T}_s | 1 \leq s \leq S, \chi \in \Omega_s\}$. Please note that tensor clouds of certain scales might not cover the location, so that $\|\{\mathcal{T}_s\}\| \leq S$. We simply compute the mean features across these scales:

$$f^{\sigma}(\chi) = \frac{1}{\|\{\mathcal{T}_s\}\|} \sum_{s} f_s^{\sigma}(\chi), \tag{16}$$

$$f^{c}(\chi) = \frac{1}{\|\{\mathcal{T}_s\}\|} \sum_{s} f_s^{c}(\chi). \tag{17}$$

Note that $f^{\sigma}(\chi)$ and $f^{c}(\chi)$ are the final density and appearance features we aggregate across multiple scales and multiple neighboring tensors.

3.4. Decoding.

We apply softplus activation on the density feature $f^{\sigma}(\chi)$ to obtain the final volume density and regress the view-dependent color by sending the appearance feature $f^{c}(\chi)$ and the viewing direction $\mathbf{d}$ to the MLP decoder ψ.

4. Rendering and Reconstruction

Volume Rendering We evaluates each pixel's color with physically-based volume rendering via differentiable ray marching. Following NeRF [26], we sample Q shading points at $\{\chi_q \mid q = 1, ..., Q\}$ along the ray, and accumulate radiance by density:

$$\begin{aligned} c &= \sum_{q=1}^{Q} T_q (1 - \exp(-\sigma_q \delta_q)) c_q, \\ T_q &= \exp(-\sum_{t=1}^{q-1} \sigma_t \delta_t). \end{aligned} \tag{18}$$

σ_q and c_q are the density and radiance at shading points; δ_t is the marching distance per step; T denotes transmittance.

Distributing local tensors. First of all, to better leverage the sparsity of a scene, we first obtain a geometric prior that roughly covers the scene geometry. The geometric prior can be in any commonly-used form, e.g., point cloud, occupancy grid, octree, or mesh vertices. Then we can uniformly distribute tensors in the spatial neighborhood of the geometry. For a multi-scale model, each of the scale will be distributed independently. For most of our results, we quickly optimize a coarse RGBA volume from the multi-view images and use the optimized occupancy grid as the geometry prior, as done in DVGO [36], which finishes in seconds.

To maintain training stability and speed, each tensor τ's position p and coverage ω is fixed once determined. We also initialize the $3(R_\sigma + R_c)$ vectors $(\mathbf{v}_{\sigma,r}^{X}, ..., \mathbf{v}_{c,r}^{X}, ...)$ of each tensor by normal distribution. For each scale s, a $P \times R_c$ appearance matrix $\mathbf{B}_s^{c}$ is shared by all tri-vector tensors of that scale. Specifically, "$\mathbf{B}^{c} \cdot ()$" in Eqn.15 can be efficiently implemented as a fully-connected neural layer. Therefore, $\mathbf{B}^{c}$ for each scale and a global appearance MLP ψ will be implemented as neural networks and initialized by default methods [15].

Optimization and objectives. Given a set of multi-view RGB images with camera poses, the sparse tri-vector radiance field is per-scene optimized to reconstruct the radiance fields, under the supervision of the ground truth pixel colors. Following the volume rendering equation 18, the L2 rendering loss can be past back to the global MLP and aggregated features, then, all the way to the the appearance matrices and the feature vectors of local tensors.

We apply a rendering loss to supervise the reconstruction and also apply an $L1$ regularization loss on density feature vectors $\mathbf{v}_{\sigma,r}$ to promote geometry sparsity and to avoid

Method	BatchSize	Steps	Time↓	# Param.(M)↓	PSNR↑	SSIM↑	LPIPS$_{Vgg}$ ↓	LPIPS$_{Alex}$ ↓
NeRF[24]	4096	300k	35.0h	1.25 ●	31.01	0.947	0.081	-
Plenoxels[45]	5000	128k	11.4m	194.50	31.71	0.958	0.049	-
DVGO[36]	5000	30k	15.0m	153.00	31.95	0.960	0.053	0.035
Point-NeRF$_{200k}$[43]	4096	200k	5.5h	27.74	33.31 ●	0.962 *	0.049	0.027
InstantNGP[27]	10k-85k	30k	3.9m	11.64	32.59	0.960	-	-
TensoRF-CP[7]	4096	30k	25.2m	0.98 ●	31.56	0.949	0.076	0.041
TensoRF-VM[7]	4096	30k	17.4m	17.95	33.14	0.963 ●	0.047	0.027 ●
Ours-24	4096	30k	34.3m	7.07 ●	33.24 ●	0.963 ●	0.046 ●	0.026 ●
Ours-48	4096	30k	35.7m	13.52	33.55 ●	0.965 ●	0.044 ●	0.025 ●

Table 1: Comparisons of our method with other radiance-based models [24, 41, 22, 43, 7, 27] on the Synthetic-NeRF dataset [24]. Ours-24 is the one with 24 components while Ours-48 is the one with 48 components. We report the corresponding rendering quality (PSNR, SSIM, and LPIPS), model capacity (number of parameters), and training time, batch size and steps. Our model achieves the best rendering quality with a compact model size. We report PointNeRF's updated SSIM.

overfitting as done in TensoRF [7]:

$$\mathcal{L}_r = \|C - \tilde{C}\|_2^2, \tag{19}$$

$$\mathcal{L}_{L1} = \frac{1}{N}\sum_{r}^{R_\sigma} \|\mathbf{v}_{\sigma,r}\|, \tag{20}$$

where $\tilde{C}$ is the ground truth pixel color, $\|\mathbf{v}_{\sigma,r}\|$ is the sum of absolute values of elements on density vectors, and N is the total number of elements. The total loss is:

$$\mathcal{L} = \mathcal{L}_r + \alpha\mathcal{L}_{L1}. \tag{21}$$

We set the weight of the sparsity term α as $1e^{-5}$ by default.

5. Implementation

To obtain coarse scene geometry, we modify the coarse density estimation introduced in [36] and get a 100^3 occupancy volume in 30 seconds. We can skip this step if there exists available geometric data, e.g., the meshes in ScanNet [12], or point clouds from multiview stereo. According to the experiments, our method is not very sensitive to the initial geometry. Please refer to Appendix.B. for more details. We set the default number of scales to 3. In a scene box of [-1,1], we rasterize the scene geometry (occuppied voxels centers or points) onto 3 grids with different voxel sizes, e.g., $0.4^3, 0.2^3, 0.1^3$. For each grid, we distribute tri-vector tensors at the center of its occupied voxels. The tensor spatial coverage of these 3 scales is $0.6^3, 0.3^3, 0.15^3$, respectively. For each scale, we query $M = 4$ nearby tensors. Following [36], our feature decoding network ψ is a 2-layer MLP with 128 channels. For each scale, its appearance matrix $\mathbf{B}^c$ is implemented by a single linear layer of 27 channels.

We implement the framework in PyTorch [17] with customized CUDA operations. During optimization, we adopt the coarse to fine strategy in [7], linearly up-sample the vectors' dimension (I, J, K) from 29 to 121 for scale one, 15 to 61 for scale two, and 7 to 31 for scale three. The up-sampling happens at step 2000, 3000, 4000, 5500, and 7000. We use the Adam optimizer [19] with initial learning rates of 0.02 for vectors and 0.001 for networks. On a single 3090 RTX GPU, we train each model for 30000 steps while each batch contains 4096 rays. Please find more details in the supplemental materials.

6. Experiments

6.1. Evaluation on the NeRF Synthetic Dataset.

We first evaluate our method on the Synthetic-NeRF dataset [24] and the quantitative results compared with other methods are reported in Tab.1, including NeRF [26], Plenoxels [45], DVGO [36], Point-NeRF [43], iNGP [27], and TensoRF [7]. We report our models of two different model sizes with different numbers of components; both settings are with the same 3 scales of local tensors.

Our approach achieves the best averaged PSNR, LPIPS$_{Vgg}$ and LPIPS$_{Alex}$ in all the methods, leading to superior visual quality as shown in Fig. 3 Meanwhile, our high rendering quality is achieved with a compact model size. When compared with local voxel-based representations, such as Plenoxels and DVGO, our approach are significantly better.

On the other hand, global featuring encoding-based methods, like iNGP and TensoRF, are known for their high compactness and can achieve higher rendering quality than local voxel-based methods. Nonetheless, our method can still outperform them. Note that, even our smaller model (Ours-24) leads to better rendering quality than both iNGP and TensoRF that leverage global feature encoding, while our model uses significantly fewer parameters (about 60% and 40% of the size of iNGP and TensoRF). This clearly demonstrates the high visual quality and high compatness of our model with our sparsely distributed tri-vector tensors.

In all the baseline methods, Point-NeRF is able to achieve relatively higher rendering quality than others.

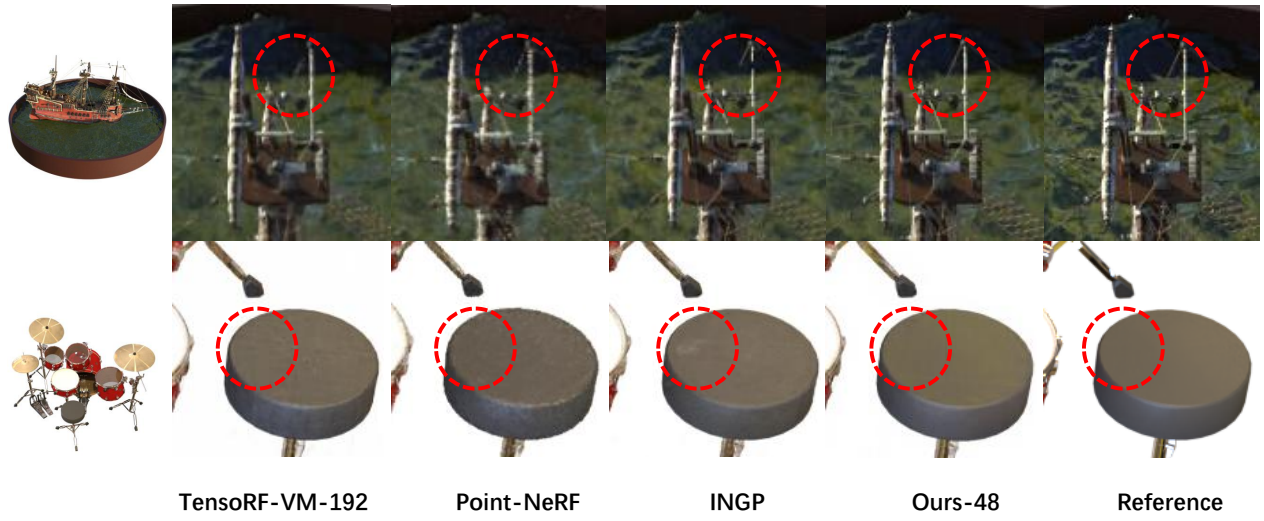

Figure 3: Qualitative comparisons on the NeRF Synthetic dataset [26].

However, this is enabled by optimizing their model for 300k steps with a long period of 5.5 hours. In contrast, our method achieves higher quality with significantly fewer optimization steps (only 30k) and optimization time (about 36 min). As expected, our model is slower to optimize than TensoRF due to the additional costs of multi-tensor aggregation. However, though speed is not our focus in this work, our model can still converge quickly and lead to significantly faster reconstruction than MLP-based methods, such as NeRF, as well as Point-NeRF that is point-based.

Performance w.r.t. rotation. We observe that tensor factorization-based methods can be sensitive to the orientation of the coordinate frames, since axis-aligned features are used; in essence, this is because the rank of a sparse tensor is sensitive to rotation, as shown in Fig. 4. Especially, this can benefit reconstruction on synthetic synthetic scenes where the objects are perfectly aligned with the axes, e.g. the lego scene in the NeRF synthetic data. However, we find that our method based on local tensors are more robust against the orientation of the axes than a global TensoRF. In particular, we compare our method with TensoRF in Tab.2 with different degrees of rotation around the Z axis on two scenes, lego (which is strongly aligned with axes) and chair (which is less aligned and thus less affected) . As shown in the table, while both methods are affected by the rotation, our method has much smaller drops of PSNRs.

6.2. Evaluation on the real datasets.

The ScanNet dataset. We evaluate our method on the real dataset, ScanNet [12] with the two scenes selected by NSVF [22], and compare with other methods. We follow the same experiment setting as done in NSVF [22], using the provided mesh to distribute our tensors, and optimize our model, TensoRF for the same 100k steps for fair comparisons. Please note that Point-NeRF uses all scanned depth images as initial geometry instead of meshes. Therefore, we

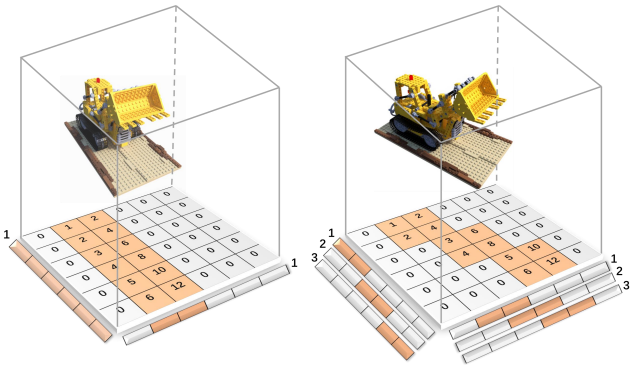

Figure 4: A toy example to illustrate the TensoRF-CP with global decomposition in (left) axis-aligned and (right) non-axis-aligned situations. The bottom shows the grid values. In axis-aligned case, only 1 component is needed to represent the scene (vector bases recover grid values by outer product). In non-axis-aligned case, however, 3 components are needed because the rank of matrix changes from 1 to 3 after scene rotation. While our design with local low-rank tensors can alleviate this issue.

chair / lego	rot 0°	rot 5°	rot 45°
TensoRF-CP	33.60 / 34.50	32.90 / 29.79	32.50 / 28.57
TensoRF-VM	35.76 / 36.46	34.91 /32.53	34.55 / 32.31
Ours-48	35.88 / 36.52	35.72 / 35.37	35.64 / 34.97

Table 2: Comparisons of our method with TensoRF [7] on the chair and lego scenes of Synthetic-NeRF dataset [24] when considering rotation of different angles around z-axis.

Figure 5: Qualitative comparisons on the ScanNet dataset.

	Average over Scene 101 and Scene 241			
	PSNR ↑	SSIM ↑	LPIPS$_{Alex}$ ↓	# Param.(M) ↓
SRN [35]	18.25	0.592	0.586	-
NeRF [24]	22.99	0.620	0.369	-
NSVF [22]	25.48	0.688	0.301	-
Point-NeRF[43]	25.92	0.891	0.273	159.01
TensoRF-CP[7]	27.54	0.751	0.328	0.97
TensoRF-VM[7]	28.61	0.787	0.290	49.92
Ours-48	29.05	0.792	0.243	12.82

Table 3: Quantitative comparison on two scenes in the ScanNet dataset [12]. Point-NeRF, TensoRF-CP, TensoRF-VM and Ours-48 are optimized for 100k steps.

also obtain the results of Point-NeRF 100k steps from the authors, using the provided mesh for fairness. We find the Scannet data has many holes in the provided mesh geometry, while methods, such as NSVF and Point-NeRF, require accurate initial geometry; though Point-NeRF can potentially fix them with it point growing technique as shown in their original paper, it is not able to fully address them in 100k step optimization and lead to holes in their final rendering. Our approach instead does not require very accurate coarse geometry, since our local tensors cover relatively large regions. We show the quantitative results in Tab. 3 and qualitative results in Fig. 5. Note that our method can also perform well on real scenes, achieving the highest performance in terms of PSNR and LPIPS$_{Alex}$, while using the second smallest model size (the smallest one is TensoRF-CP$_{100k}$). Our visual quality is also higher than the comparison methods.

The Tanks and Temples dataset. We also evaluate our method on another real dataset, Tanks and Temples [12] with the 5 object scenes selected by NSVF [22]. We using the very coarse geometries estimated by DVGO[36] to distribute our tensors. We follow the same experiment setting as done in TensoRF [7], optimizing our model for the same 30k steps for fair comparisons. As is shown in Tab. 3, our method outperforms other methods in terms of PSNR, SSIM and LPIPS$_{Alex}$, while using the second smallest model size.

Scale	PSNR ↑	SSIM ↑	# Param.(M) ↓	Time ↓
Single(0.6)	32.22	0.957	1.75	18.22m
Single(0.3)	32.73	0.961	4.15	21.31m
Single(0.15)	31.96	0.952	10.20	28.55m
Multi(0.6, 0.3)	33.11	0.963	6.20	30.12m
Multi(0.6, 0.3, 0.15)	33.55	0.965	13.52	35.70m

Table 4: Ablation under different scale settings on Synthetic-NeRF dataset. We select 3 scales of tensors with cube sizes of 0.6, 0.3, and 0.15.

6.3. Ablation study

We analyze our model in terms of different scales in Table.4, while keeping the number of components the same (here we use 48). The scale here is the size of our local tensors of each axis. We considered 3 different scales, i.e., 0.6, 0.3, and 0.15 respectively as single-scale settings and some of their combinations as multi-scale settings. Note that even with a single scale (0.3), the performance of our method can be comparable with some other methods such as iNGP [27] while ours have less than half of the model size. When increasing the number of scales or decreasing the size of local tensors, our model size will also increase. In general, there is a trade-off of our method between scales and computational consumption (time and size).

Usually, a smaller scale can lead to better performance, though our method with a scale of 0.15 does not strictly follow because we don't have high-quality input geometry to place these local tensors with a very small size. In

fact, according to our per-scene breakdown results on the Synthetic-NeRF dataset (please refer to our supplemental material), single-scale(0.075) can achieve higher performance than single-scale(0.3) and single-scale(0.15) on most scenes, except for ship because it has many thin structures that our coarse reconstruction does not cover.

	PSNR ↑	SSIM ↑	LPIPS$_{Alex}$ ↓	# Param.(M) ↓
NeRF [26]	25.78	0.864	0.198	-
NSVF [22]	28.40	0.900	0.153	-
TensoRF-CP$_{30k}$[7]	27.59	0.897	0.144	**0.97**
TensoRF-VM$_{30k}$[7]	28.56	0.920	0.125	49.92
Ours-48$_{30k}$	**28.70**	**0.922**	**0.113**	14.11

Table 5: Quantitative comparison on scenes in the Tanks and Temples dataset [20] selected in NSVF [22]. TensoRF-CP, TensoRF-VM and Ours-48 are optimized for 30k steps.

We also compare our method with a variant that uses vector-matrix (VM) decomposition [7] in each local tensor instead of CP decomposition. Please refer to Appendix.A. for more details. Also, we can achieve a higher training and inference speed without a significant performance drop, which we refer to Appendix.E.

7. Conclusion

In this work, we have presented a novel approach for high-quality neural scene reconstruction and photo-realistic novel view synthesis. We propose a novel tensor factorization-based scene representation, which leverages CP decomposition to compactly model a 3D scene as a sparse set of multi-scale tri-vector tensors that express local radiance fields. Our representation leverages both sparsity and spatial local coherence, and leads to accurate and efficient modeling of complex scene geometry and appearance. We demonstrate that the sparse tri-vector radiance fields can achieve superior rendering quality than previous state-of-the-art neural representations, including TensoRF and iNGP, while using significantly fewer parameters.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *International Conference on Machine Learning (ICML)*, pages 40–49, 2018. 2

[2] Relja Arandjelović and Andrew Zisserman. Nerf in detail: Learning to sample for view synthesis. *arXiv preprint arXiv:2106.05264*, 2021. 2

[3] 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 (ICCV)*, pages 5855–5864, 2021. 1, 2

[4] 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 (CVPR)*, pages 5470–5479, 2022. 13

[5] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970. 1, 3

[6] 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 (CVPR)*, pages 16123–16133, 2022. 3

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, pages 333–350. Springer, 2022. 1, 2, 3, 4, 6, 7, 8, 9, 12, 13, 14, 15

[8] Anpei Chen, Zexiang Xu, Xinyue Wei, Siyu Tang, Hao Su, and Andreas Geiger. Factor fields: A unified framework for neural fields and beyond, 2023. 13

[9] Anpei Chen, Zexiang Xu, Xinyue Wei, Siyu Tang, Hao Su, and Andreas Geiger. Factor fields: A unified framework for neural fields and beyond. *arXiv preprint arXiv:2302.01226*, 2023. 14

[10] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[11] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6970–6981, 2020. 2

[12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, 2017. 6, 7, 8

[13] 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 (CVPR)*, pages 12479–12488, 2023. 3

[14] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4857–4866, 2020. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1026–1034, 2015. 5

[16] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2821–2830, 2018. 2

[17] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, pages 87–104, 2021. 6

[18] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[20] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 9, 13, 15

[21] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2024–2039, 2016. 2

[22] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *arXiv preprint arXiv:2007.11571*, 2020. 2, 6, 7, 8, 9, 14, 15

[23] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019. 15

[24] 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 (CVPR)*, pages 7210–7219, 2021. 2, 6, 7, 8

[25] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[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 *European Conference on Computer Vision (ECCV)*, pages 405–421. Springer, 2020. 1, 2, 5, 6, 7, 9, 12, 13, 14, 15

[27] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, 2022. 1, 3, 6, 8, 12, 13, 14

[28] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[29] Anton Obukhov, Mikhail Usvyatsov, Christos Sakaridis, Konrad Schindler, and Luc Van Gool. Tt-nf: Tensor train neural fields. *arXiv preprint arXiv:2209.15529*, 2022. 3

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 2

[31] 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 (ICCV)*, pages 5865–5874, 2021. 1

[32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[33] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *arXiv preprint arXiv:2103.13744*, 2021. 3

[34] Jonathan Richard Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 86–95, 1998. 2

[35] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019. 8

[36] 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 (CVPR)*, pages 5459–5469, 2022. 2, 3, 4, 5, 6, 8, 13

[37] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8248–8258, 2022. 3

[38] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022. 2

[39] Jinglu Wang, Bo Sun, and Yan Lu. MVPnet: Multi-view point regression networks for 3D object reconstruction from a single image. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019. 2

[40] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single RGB images. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[41] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul 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 (CVPR)*, 2021. 6

[42] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. 2

[43] 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 (CVPR)*, pages 5438–5448, 2022. 2, 3, 6, 8, 13, 14

[44] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Proc. NeurIPS*, 2020. 2

[45] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*, 2021. 2, 3, 6

[46] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 1

[47] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 14

[48] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5449–5458, 2022. 2

Appendix

A. Ablation Studies on Tensor Factorization Strategies

	# Comp	PSNR ↑	SSIM ↑	# Param.(M) ↓
Multi(0.6, 0.3, 0.15)	24	**33.24**	**0.963**	**7.07**
Single(0.3)	96	33.02	0.963	9.15
VM-Cloud (0.3)	6	32.59	0.959	11.36
VM-Cloud (0.3)	12	32.99	0.962	21.64

Table 6: (a) Comparisons on our method pairing with different factorization strategies, e.g., CP decomposition and vector-matrix (VM) decomposition (row 2 vs 3,4). The local tensors' edge lengths are all set as 0.3. (b) We also compare a single-scale model with a multi-scale model (row 1 vs 2). We evaluate these settings on the NeRF Synthetic dataset [26] and evaluate them with both rendering quality and model capacity (#Param. denotes the number of parameters).

Other than CP decomposition, TensoRF [7] also proposes vector-matrix (VM) decomposition, which factorizes a 3D tensor as the summation of vector-matrix bases. Each basis is the outer product of a matrix along a plane, e.g., the XY plane, and a vector along an orthogonal direction, e.g., the Z axis. For comparison, we also explore to replace our tri-vector representation with the vector-matrix representation for each local tensor. Tab. 6 shows that the single-scale tri-vector cloud can outperform the vector-matrix cloud representation with less model capacity.

It is not a surprise that our tri-vector cloud representation achieves more compactness. It applies more compression by factorizing each component of a 3D tensor, with a space complexity of $O(IJK)$, into three vectors, with a space complexity of $O(I + J + K)$. On the other hand, vector-matrix cloud representation factorizes it into three vectors and three matrices, which have a space complexity of $O(IJ + JK + IK)$. Even if we reduce the number of components, the vector-matrix clouds still require more space than our tri-vector representations.

In terms of quality, since our method exploits the spatial sparsity of natural scenes, we only need to factorize each local space independently instead of the entire scene together. The more compact tri-vector representation can benefit from the appearance coherence in local space and result in better performance. In TensoRF [7], since the entire space is factorized all at once, the radiance information is, in general, less coherent across locations and the CP decomposition will lead to a shortage of rank.

B. Ablation Studies on Multi-scale Models

In Tab.6, we also compare our multi-scale tri-vector radiance fields with the single-scale strategy. In our default model, we have three scales, composed of tensors with lengths 0.15, 0.3, and 0.6, respectively. Similar to the findings in iNGP [27], our multi-scale models provide more smoothness and lead to a better rendering quality than their single-scale counterparts. The multi-scale model with 24 components (row 1) can already outperform the single-scale model (row 2), which has more parameters.

C. Ablation Studies on the Number of Tensor Components

We conduct experiments on the NeRF Synthetic dataset [26] to show the relationship between rendering performance and the number of tensor components. In Tab.7, we compare our multi-scale models with 12, 24, 48, and 96 appearance components, respectively. In general, more tensor components will lead to better performance. We also observe that the benefit of adding more components becomes marginal when the number reaches 48. We speculate that it is harder to learn high-frequency details even though the model's capacity can hold high-rank information. Improvement in this aspect can be a promising future direction.

D. Ablation Studies on Initial Geometry

We emphasize that our superior quality stems from our novel scene representation rather than the initial geometry. The initial geometry is simply acquired from a low-res RGBA volume reconstruction, which is coarse and only used to roughly prune empty space.

We show in Fig. 6 that our approach performs robustly with various choices of these geometry structures and consistently achieves high PSNRs, even with a much worse early-stopped RGBA reconstruction. This showcases the key to our superior quality is our Strivec model itself.
In particular, the self-bootstrap geometry is generated purely from our own model with 8 coarse tri-vectors without existing modules in previous work. Moreover, we can also further prune unoccupied tensors during training but we find this leads to similar quality (0.03db difference) and unnecessary extra (+22%) training time. We instead choose to use one single initial geometry to prune empty space in implementation for its simplicity and efficiency.

	PSNR↑	SSIM↑	LPIPS$_{Vgg}$ ↓	LPIPS$_{Alex}$ ↓	# Param.(M)↓
Ours-12	32.94	0.961	0.049	0.028	**4.87**
Ours-24	33.24	0.963	0.046	0.026	7.07
Ours-48	33.55	0.965	0.044	0.025	13.52
Ours-96	**33.59**	**0.965**	**0.043**	**0.024**	21.01

Table 7: Ablation study on the number of tensor components. We use the same setting as our default model but only change the number of components in each variant. These variants are evaluated on the NeRF Synthetic dataset [26].

Figure 6: Our quality with initial geometry by different methods.

E. Speed v.s. Performance

Though speed is not our focus, here, if we reduce the number of scales from 3 to 2 and TopK from 4 to 2 (i.e., Multi(0.6, 0.3) with TopK=2), and Strivec becomes faster than CP and close to VM, while still having competitive quality (see Ours-48(fast) in Tab.8). The fewer ranks of our tensor and the less number of TopK to be find for each sample point along a ray lead to less computation, and thus, acceleration. To conclude, Strivec is capable to improve quality, training time and compactness all together with proper hyper-parameters.

	Train(s)↓	Inference(s/it)↓	#Params.(M)↓	PSNR↑
TensoRF-CP	1914	2.01	0.98	31.56
TensoRF-VM	915	1.60	17.95	33.14
Ours-48(fast)	959	1.67	6.20	33.09

Table 8: Comparison on NeRF Synthetic dataset [26]. We compare the average training time (s), inference time (s/it), the number of parameters (M) and PSNR.

F. Per-scene Breakdown Results of the NeRF Synthetic Dataset

We show the per-scene detailed quantitative results of the comparisons on the NeRF Synthetic dataset [26] in Tab. 10 and qualitative comparisons in our video. With compact model capacity, our method outperforms state-of-the-art methods [26, 27, 43, 7] and achieves the best PSNRs, and LPIPSs in most of the scenes. We report two versions of iNGP [27]. Specifically, iNGP-dark$_{100k}$ is reported in the original paper. According to issue #745 in iNGP's official repo, the method uses a random color background in training and dark background in testing. The number of iterations, 100k, is referenced to its initial code base release. We also refer to the results reported in [8] as iNGP-white$_{30k}$, since the authors use a white background in both training and testing for 30k iterations, which has the same setting as ours and many other compared methods. Please refer to issue #745 and #1266 in iNGP's official repo for more details.

	garden	room	Model Size(avg)
DVGO	24.32	28.35	5.1GB
Ours-48	24.13	28.11	12.6MB

Table 9: Results on the Mip-NeRF 360 dataset.

G. The Tanks and Temples Dataset

We show the qualitative comparison between our Strivec and TensoRF-VM [7] on the Tanks and Temples dataset [20] in Fig.7. Similar to the procedures on the NeRF Synthetic dataset, we build the coarse scene geometry within 30 seconds to place our local tensors. The quantitative results are reported in Tab.11.

H. Mip-NeRF360 Dataset

We evaluate our method on two scenes (one indoor scene and one outdoor scene) of Mip-NeRF360 dataset [4]. Note that we only use the scene warping scheme the same as DVGO [36] and Mip-NeRF360 [4] and keeping other components (i.e., positional encoding, point sampling, etc.) the same as TensoRF [7]. The qualitative and quantitative results are shown in Fig. 8 and Tab. 9 , respectively. Here, we use only two scales in implementation to show our compactness and scalability.

	NeRF Synthetic							
	Chair	Drums	Lego	Mic	Materials	Ship	Hotdog	Ficus
				PSNR↑				
NeRF [26]	33.00	25.01	32.54	32.91	29.62	28.65	36.18	30.13
NSVF [22]	33.19	25.18	32.54	34.27	**32.68**	27.93	37.14	31.23
Point-NeRF$_{20k}$ [43]	32.50	25.03	32.40	32.31	28.11	28.13	34.53	32.67
Point-NeRF$_{200k}$ [43]	35.40	26.06	35.04	35.95	29.61	30.97	37.30	36.13
iNGP-dark$_{100k}$ [27]	35.00	26.02	36.39	36.22	29.78	31.10	37.40	33.51
iNGP-white$_{30k}$ [27, 9]	35.42	24.24	34.82	35.98	28.99	30.72	37.45	32.09
TensoRF-CP [7]-384$_{30k}$	33.60	25.17	34.05	33.77	30.10	28.84	36.24	30.72
TensoRF-VM [7]-192$_{30k}$	35.76	26.01	36.46	34.61	30.12	30.77	37.41	33.99
Ours-12$_{30k}$	35.21	25.96	35.60	35.29	29.54	30.64	37.03	34.21
Ours-24$_{30k}$	35.60	26.16	36.05	35.81	29.79	30.89	37.24	34.37
Ours-48$_{30k}$	**35.88**	**26.20**	**36.52**	**36.65**	29.90	**31.13**	**37.63**	34.47
				SSIM↑				
NeRF	0.967	0.925	0.961	0.980	0.949	0.856	0.974	0.964
NSVF	0.968	0.931	0.960	0.987	**0.973**	0.854	0.980	0.973
Point-NeRF$_{20k}$	0.981	0.944	0.980	0.986	0.959	0.916	0.983	0.986
Point-NeRF$_{200k}$	**0.991**	**0.954**	**0.988**	**0.994**	0.971	**0.942**	**0.991**	**0.993**
iNGP-white$_{30k}$	0.985	0.924	0.979	0.990	0.945	0.892	0.982	0.977
TensoRF-CP-384$_{30k}$	0.973	0.921	0.971	0.983	0.950	0.857	0.975	0.965
TensoRF-VM-192$_{30k}$	0.985	0.937	0.983	0.988	0.952	0.895	0.982	0.982
Ours-12$_{30k}$	0.983	0.937	0.980	0.989	0.948	0.888	0.981	0.983
Ours-24$_{30k}$	0.984	0.940	0.982	0.990	0.952	0.893	0.982	0.984
Ours-48$_{30k}$	0.985	0.940	0.984	0.992	0.953	0.899	0.983	0.985
				LPIPS$_{Vgg}$ ↓				
NeRF	0.046	0.091	0.050	0.028	0.063	0.206	0.121	0.044
Point-NeRF$_{20k}$	0.051	0.103	0.054	0.039	0.102	0.181	0.074	0.043
Point-NeRF$_{200k}$	0.023	0.078	0.024	0.014	0.072	**0.124**	0.037	0.022
iNGP-white$_{30k}$	0.022	0.092	0.025	0.017	0.069	0.137	0.037	0.026
TensoRF-CP-384$_{30k}$	0.044	0.114	0.038	0.035	0.068	0.196	0.052	0.058
TensoRF-VM-192$_{30k}$	0.022	0.073	0.018	0.015	0.058	0.138	0.032	0.022
Ours-12$_{30k}$	0.025	0.070	0.022	0.015	0.062	0.145	0.033	0.022
Ours-24$_{30k}$	0.022	0.067	0.020	0.013	0.058	0.141	0.031	0.021
Ours-48$_{30k}$	**0.021**	**0.064**	**0.017**	**0.011**	**0.056**	0.138	**0.029**	**0.018**
				LPIPS$_{Alex}$ ↓				
NSVF	0.043	0.069	0.029	0.010	**0.021**	0.162	0.025	0.017
Point-NeRF$_{20k}$	0.027	0.057	0.022	0.024	0.076	0.127	0.044	0.022
Point-NeRF$_{200k}$	0.010	0.055	0.011	0.007	0.041	**0.070**	0.016	**0.009**
iNGP-white$_{30k}$	0.022	0.093	0.025	0.017	0.069	0.140	0.037	0.026
TensoRF-CP-384$_{30k}$	0.022	0.069	0.014	0.018	0.031	0.130	0.024	0.024
TensoRF-VM-192$_{30k}$	0.010	0.051	0.007	0.009	0.026	0.085	0.013	0.012
Ours-12$_{30k}$	0.011	0.051	0.009	0.007	0.027	0.092	0.015	0.013
Ours-24$_{30k}$	0.010	0.049	0.008	0.006	0.024	0.087	0.014	0.012
Ours-48$_{30k}$	**0.009**	**0.048**	**0.007**	**0.005**	0.023	0.086	**0.012**	0.011

Table 10: Detailed breakdown of quantitative metrics on individual scenes in the NeRF Synthetic [26] for our method and baselines. All scores are averaged over the testing images. The subscripts are the number of iterations of the models. NeRF only [26] reports the LPIPS$_{Vgg}$ [47] while NSVF only reports LPIPS$_{Alex}$.

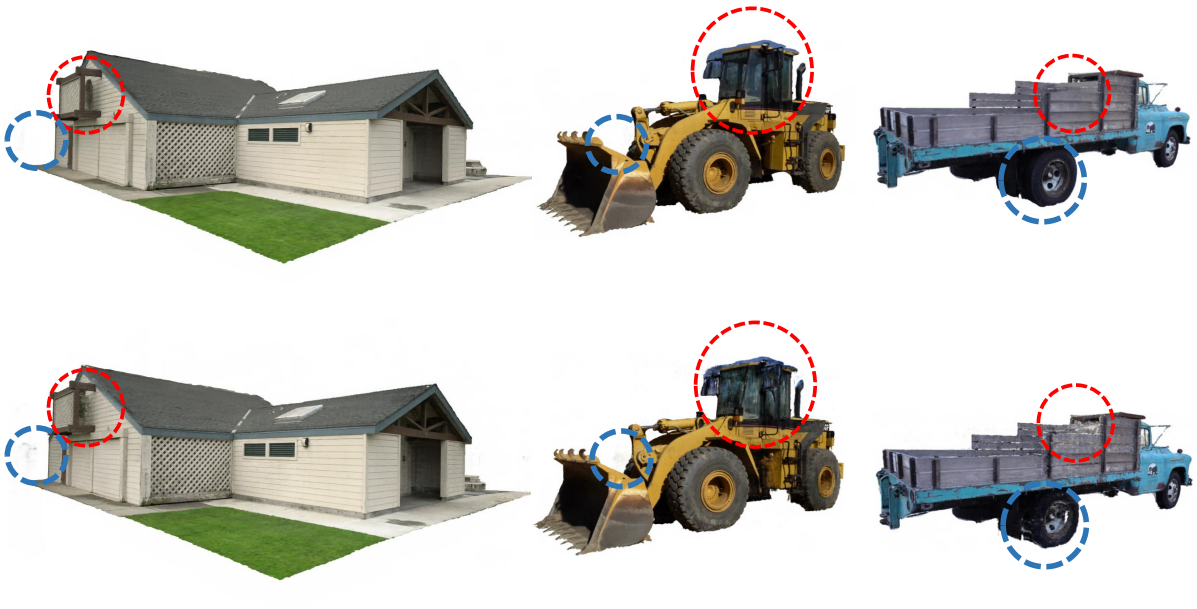

Figure 7: Qualitative comparison on the Tanks and Temples dataset. Top: ours. Bottom: TensoRF-VM.

	Tanks & Tamples					
	Ignatius	Truck	Barn	Caterpillar	Family	Mean
			PSNR ↑			
NV [23]	26.54	21.71	20.82	20.71	28.72	23.70
NeRF [26]	25.43	25.36	24.05	23.75	30.29	25.78
NSVF [22]	27.91	26.92	27.16	26.44	33.58	28.40
TensoRF-CP[7]	27.86	26.25	26.74	24.73	32.39	27.59
TensoRF-VM[7]	28.34	27.14	27.22	26.19	**33.92**	28.56
Ours-48	**28.39**	**27.32**	**28.09**	**26.58**	33.13	**28.70**
			SSIM ↑			
NV [23]	0.992	0.793	0.721	0.819	0.916	0.848
NeRF [26]	0.920	0.860	0.750	0.860	0.932	0.864
NSVF [22]	0.930	0.895	0.823	0.900	0.954	0.900
TensoRF-CP[7]	0.934	0.885	0.839	0.879	0.948	0.897
TensoRF-VM[7]	0.948	0.914	0.864	0.912	**0.965**	0.920
Ours-48	**0.948**	**0.915**	**0.884**	**0.917**	0.957	**0.924**
			LPIPS_{Alex} ↓			
NV [23]	0.117	0.312	0.479	0.280	0.111	0.260
NeRF [26]	0.111	0.192	0.395	0.196	0.098	0.198
NSVF [22]	0.106	0.148	0.307	0.141	0.063	0.153
TensoRF-CP[7]	0.089	0.154	0.237	0.176	0.063	0.144
TensoRF-VM[7]	**0.081**	0.129	0.217	0.139	**0.057**	0.125
Ours-48	0.083	**0.123**	**0.167**	**0.125**	0.065	**0.113**
			LPIPS_{Vgg} ↓			
TensoRF-CP[7]	0.106	0.202	0.283	0.227	0.088	0.181
TensoRF-VM[7]	**0.078**	**0.145**	0.252	0.159	**0.064**	0.140
Ours-48	0.083	0.150	**0.216**	**0.154**	0.078	**0.136**

Table 11: Quantity comparison on five scenes in the Tanks and Temples dataset [20] selected in NSVF [22]. NV, NeRF, and NSVF have not reported their LPIPS_{Vgg}

Figure 8: Qualitative results on Mip-NeRF360 dataset.

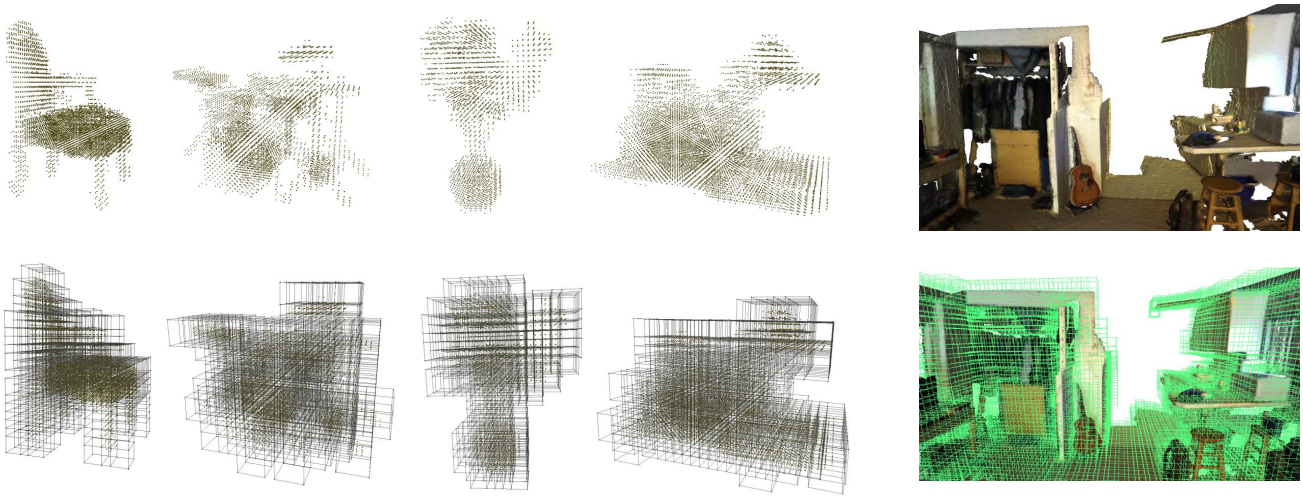

Figure 9: Visualization of local tensors (single scale) on initial geometry.