Strivec: Sparse Tri-Vector Radiance Fields

Quankai Gao∗1 Qiangeng Xu∗1 Hao Su2 Ulrich Neumann1 Zexiang Xu3
1University of Southern California 2UC San Diego 3Adobe 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 [6], 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 [4] 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 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 [23].

1. Introduction

Representing 3D scenes as radiance fields [22] has enabled photo-realistic rendering quality and emerged as a popular design choice in 3D vision and graphics applications. While many methods [27, 42, 3] (following NeRF [22]) purely use MLPs to represent neural fields, recent works, like TensoRF [6] and Instant-NGP [23], 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).

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-
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 [6], 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 [32]) or multi-view stereo (like Point-NeRF [39]) – 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 [19, 41] or points [39]) 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 PointNeRF, 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 [13, 18], mesh [15, 36, 30], point cloud [28, 1, 35] and implicit function [7, 21, 24, 40]. In recent years, continuous neural field representations stand out in various 3D tasks such as single-view 3D reconstruction [38, 11], surface completion [8, 26], multi-view reconstruction [24] and novel view synthesis [22, 20]. 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 [22] 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, 34] 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 [19, 41, 32, 43]
and points [39]. Low-rank representations such as triplane [5, 10] and tensor decomposition [6, 25] have also been studied. In particular, DVGO [32] and Plenoxels [41] 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 [23] that hashes spatial features into multi-scale hash tables and TensoRF [6] 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 [29] and BlockNeRF [33] 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 [6], 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 \( \mathcal{T} = \{ \tau_n | n = 1, \ldots, N \} \) denote a cloud of tri-vector tensors. Each local tensor \( \tau \) is located at \( p \), covering a local cuboid space \( \omega \) 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 \( \tau \) encodes a local multi-channel feature grid that includes a (single-channel) density grid \( A_\sigma \) and a (multi-channel) appearance grid \( A_\chi \), similar to the tensor grid in TensoRF [6]. In contrast to using a single global tensor in TensoRF [6], we model the volume density and view-dependent colors with multiple local tensors. In particular, for an arbitrary location \( c \in \Omega \), we select \( M \) nearest tensors that cover \( c \). 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 \( \sigma \) is directly obtained after the aggregation and the view-dependent color \( c \) is regressed by a small MLP \( \psi \) along with the viewing direction \( d \). The continuous radiance field can be expressed as:

\[
\sigma, c = A_\sigma(\{G^\sigma(\chi)\}), \psi(A_\chi(\{G^\chi(\chi)\}), d). \tag{2}
\]

#### 3.1. Local tri-vector tensors.

We apply the classic Canonical polyadic (CP) decomposition [4] to model our local tensors with tri-vector components.

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

\[
\tau = \sum_{r=1}^{R} \lambda_r \psi_r^0 \otimes \psi_r^1 \otimes \cdots \otimes \psi_r^M, \tag{3}
\]

where \( \otimes \) denotes outer product; the weighting factor \( \lambda_r \) can be absorbed into vectors \( \{\psi_r^0, \ldots, \psi_r^M\} \).

**Density and appearance tensors.** In our case of modeling a 3D radiance field, we set the geometry grid \( G^\sigma \in \mathbb{R}^{I_1 \times I_2 \times I_3} \) as a 3D tensor. And the multi-channel appearance grid \( G^\chi \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times P} \) 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, \( G^\sigma \) and \( G^\chi \), by CP decomposition:

\[
G^\sigma = \sum_{r=1}^{R_\sigma} A_{\sigma,r} = \sum_{r=1}^{R_\sigma} \psi_{\sigma,r}^0 \otimes \psi_{\sigma,r}^1 \otimes \psi_{\sigma,r}^2, \tag{4}
\]

\[
G^\chi = \sum_{r=1}^{R_\chi} A_{\chi,r} \otimes b_r = \sum_{r=1}^{R_\chi} \psi_{\chi,r}^0 \otimes \psi_{\chi,r}^1 \otimes \psi_{\chi,r}^2 \otimes b_r, \tag{5}
\]
Here $R_\sigma$ and $R_c$ denote numbers of component; $A_{\sigma,r}$ and $A_{c,r}$ are the component tensors that are factorized spatially; $v_{\sigma,r}^X, ..., v_{c,r}^Z$ are the 1D vectors with resolution $I, J, K$, modeling scene geometry and appearance along $X, Y, Z$ axis; $R_\sigma$ and $R_c$ are the component numbers; $b_r$ expresses the feature dimension.

As done in TensoRF [6], we stack all feature-mode vectors $b_r$ as columns together, which ends up a $P \times R_c$ appearance matrix $B$. 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 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 $v_r^X, v_r^Y, v_r^Z$.

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

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

Then, for example, to get $A_{\sigma,r}$ at $(\tilde{x}, \tilde{y}, \tilde{z})$, we can compute and trilinearly interpolate eight $A_{\sigma,r}$ on the corners. As mentioned in [6], applying linear interpolation on each vector first is mathematically equivalent and can reduce the computation cost. Under the rule of outer product, we have $A_{r,i,j,k} = v_{r,i}^X v_{r,j}^Y v_{r,k}^Z$, then the interpolated density features at location $\chi$ are:

$$G^\sigma(\chi) = \sum_r v_{\sigma,r}^X(\tilde{x})v_{\sigma,r}^Y(\tilde{y})v_{\sigma,r}^Z(\tilde{z}) = \sum_r A_{\sigma,r}(\tilde{\chi}), \quad (7)$$

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

Similarly, the interpolated appearance features can be computed as:

$$G^c(\chi) = \sum_r v_{c,r}^X(\tilde{x})v_{c,r}^Y(\tilde{y})v_{c,r}^Z(\tilde{z})b_r \quad (8)$$

$$= \sum_r A_{c,r}(\tilde{\chi})b_r \quad (9)$$

$$= B \cdot (\oplus [A_{c,r}]_r), \quad (10)$$

where “$\oplus$” denotes concatenation, “$\cdot$$” denotes dot product. The appearance feature $G^c(\chi) \in \mathbb{R}^P$ 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 $\chi$. 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

$$w_m = \frac{1}{\|p_m - \chi\|}. \quad (11)$$
With this weight function, we directly obtain the density feature via the weighted sum:

\[ f^\sigma(\chi) = \frac{1}{\sum w_m} \sum_{m=1}^{M} w_m G_m^\sigma(\chi). \] (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:

\[ f^c(\chi) = \frac{1}{\sum w_m} \sum_{m=1}^{M} w_m G_m^c(\chi) \] (13)

\[ = \frac{1}{\sum w_m} \sum_{m=1}^{M} w_m B^c \cdot (\oplus [A_c, r]) \] (14)

\[ = \frac{1}{\sum w_m} B^c \cdot (\sum_{m=1}^{M} w_m (\oplus [A_c, r])). \] (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 \(\chi\), we gather the density and appearance features from a set of tensor clouds that cover \(\chi\), \(\{T_s | 1 \leq s \leq S, \chi \in \Omega_s\}\). Please note that tensor clouds of certain scales might not cover the location, so that \(\|\{T_s\}\| \leq S\). We simply compute the mean features across these scales:

\[ f^\sigma(\chi) = \frac{1}{\|\{T_s\}\|} \sum_s f_s^\sigma(\chi), \] (16)

\[ f^c(\chi) = \frac{1}{\|\{T_s\}\|} \sum_s f_s^c(\chi). \] (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 \(\psi\).

4. Rendering and Reconstruction

Volume Rendering We evaluate each pixel’s color with physically-based volume rendering via differentiable ray marching. Following NeRF [22], we sample \(Q\) shading points at \(\{q | q = 1, ..., Q\}\) along the ray, and accumulate radiance by density:

\[ e = \sum_{q=1}^{Q} T_q (1 - \exp(-\sigma_q d_q)) c_q, \] (18)

\[ T_q = \exp(-\sum_{t=1}^{q-1} \sigma_t d_t). \]

\(\sigma_q\) and \(c_q\) are the density and radiance at shading points; \(d_t\) 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 [32], which finishes in seconds.

To maintain training stability and speed, each tensor \(\tau\)’s position \(p\) and coverage \(\omega\) is fixed once determined. We also initialize the \(3(R_{\sigma} + R_c)\) vectors \(\{v_s, ..., v_s\}\) of each tensor by normal distribution. For each scale \(s\), a \(P \times R_c\) appearance matrix \(B_s^c\) is shared by all tri-vector tensors of that scale. Specifically, “\(B^c(\cdot)\)” in Eqn.15 can be efficiently implemented as a fully-connected neural layer. Therefore, \(B^c\) for each scale and a global appearance MLP \(\psi\) will be implemented as neural networks and initialized by default methods [12].

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 \(L_1\) regularization loss on density feature vectors \(v_{s, \tau}\) to promote geometry sparsity and to avoid
overfitting as done in TensoRF [6]:

\[ \mathcal{L}_r = \| C - \tilde{C} \|_2^2, \quad (19) \]

\[ \mathcal{L}_{L1} = \frac{1}{N} \sum_{r} \| v_{\sigma, r} \|_1, \quad (20) \]

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

\[ \mathcal{L} = \mathcal{L}_r + \alpha \mathcal{L}_{L1}. \quad (21) \]

We set the weight of the sparsity term \( \alpha \) as \( 1 \times 10^{-5} \) by default.

5. Implementation

To obtain coarse scene geometry, we modify the coarse density estimation introduced in [32] 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 Scan-Net [9], or point clouds from multiview stereo. According to the experiments, our method is not very sensitive to the initial geometry. Please refer to our supplementary material for more details. We set the default number of scales to 3. In a scene box of \([-1,1] \), we rasterize the scene geometry (occupied 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 [32], our feature decoding network \( \psi \) is a 2-layer MLP with 128 channels. For each scale, its appearance matrix \( \mathbf{B}^\psi \) is implemented by a single linear layer of 27 channels.

We implement the framework in PyTorch [14] with customized CUDA operations. During optimization, we adopt the coarse to fine strategy in [6], 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 [16] 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 [20] and the quantitative results compared with other methods are reported in Tab.1, including NeRF [22], Plenoxels [41], DVGO [32], Point-NeRF [39], inGP [23], and TensoRF [6]. 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, \( \text{LPIPS}_{V_{gg}} \) and \( \text{LPIPS}_{Alex} \) 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 is 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 compactness 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.
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 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.

Table 2: Comparisons of our method with TensoRF [6] on the chair and lego scenes of Synthetic-NeRF dataset [20] when considering rotation of different angles around z-axis.

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

6.2. Evaluation on the real datasets.

The ScanNet dataset. We evaluate our method on the real dataset, ScanNet [9] with the two scenes selected by NSVF [19], and compare with other methods. We follow the same experiment setting as done in NSVF [19], 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 also obtain the results of Point-NeRF 100k steps from the authors, using the provided mesh for fairness. We find the
Figure 5: Qualitative comparisons on the ScanNet dataset.

### Table 3: Quantitative comparison on two scenes in the ScanNet dataset [9]. Point-NeRF, TensoRF-CP, TensoRF-VM and Ours-48 are optimized for 100k steps.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR</th>
<th>SSIM</th>
<th>LPIPS</th>
<th># Param. (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRN [31]</td>
<td>18.25</td>
<td>0.592</td>
<td>0.586</td>
<td>-</td>
</tr>
<tr>
<td>NeRF [20]</td>
<td>22.99</td>
<td>0.620</td>
<td>0.369</td>
<td>-</td>
</tr>
<tr>
<td>NSVF [19]</td>
<td>25.48</td>
<td>0.688</td>
<td>0.301</td>
<td>-</td>
</tr>
<tr>
<td>Point-NeRF [39]</td>
<td>25.92</td>
<td>0.891</td>
<td>0.273</td>
<td>159.01</td>
</tr>
<tr>
<td>TensoRF-CP [6]</td>
<td>27.54</td>
<td>0.751</td>
<td>0.328</td>
<td>0.97</td>
</tr>
<tr>
<td>TensoRF-VM [6]</td>
<td>28.61</td>
<td>0.787</td>
<td>0.290</td>
<td>49.92</td>
</tr>
<tr>
<td>Ours-48</td>
<td>29.05</td>
<td>0.792</td>
<td>0.243</td>
<td>12.82</td>
</tr>
</tbody>
</table>

Table 3: Quantitative comparison on two scenes in the ScanNet dataset [9]. Point-NeRF, TensoRF-CP, TensoRF-VM and Ours-48 are optimized for 100k steps.

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 its 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 [9] with the 5 object scenes selected by NSVF [19]. We use the very coarse geometries estimated by DVGO[32] to distribute our tensors. We follow the same experiment setting as done in TensoRF [6], 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.

### 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.

<table>
<thead>
<tr>
<th>Scale</th>
<th>PSNR</th>
<th>SSIM</th>
<th># Param. (M)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single(0.6)</td>
<td>32.22</td>
<td>0.957</td>
<td>1.75</td>
<td>18.22m</td>
</tr>
<tr>
<td>Single(0.3)</td>
<td>32.73</td>
<td>0.961</td>
<td>4.15</td>
<td>21.31m</td>
</tr>
<tr>
<td>Single(0.15)</td>
<td>31.96</td>
<td>0.952</td>
<td>10.20</td>
<td>28.55m</td>
</tr>
<tr>
<td>Multi(0.6, 0.3)</td>
<td>33.11</td>
<td>0.963</td>
<td>6.20</td>
<td>30.12m</td>
</tr>
<tr>
<td>Multi(0.6, 0.3, 0.15)</td>
<td>33.55</td>
<td>0.965</td>
<td>13.52</td>
<td>35.70m</td>
</tr>
</tbody>
</table>

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 [23] 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.

<table>
<thead>
<tr>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>LPIPS$_{Alex}$ ↓</th>
<th># Param.(M) ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRF [22]</td>
<td>25.78</td>
<td>0.864</td>
<td>0.198</td>
</tr>
<tr>
<td>NSVF [19]</td>
<td>28.40</td>
<td>0.900</td>
<td>0.153</td>
</tr>
<tr>
<td>TensoRF-CP$_{30k}$ [6]</td>
<td>27.59</td>
<td>0.897</td>
<td>0.144</td>
</tr>
<tr>
<td>TensoRF-VM$_{30k}$ [6]</td>
<td>28.56</td>
<td>0.920</td>
<td>0.125</td>
</tr>
<tr>
<td>Ours-48$_{30k}$</td>
<td><strong>28.70</strong></td>
<td><strong>0.922</strong></td>
<td><strong>0.113</strong></td>
</tr>
</tbody>
</table>

Table 5: Quantitative comparison on scenes in the Tanks and Temples dataset [17] selected in NSVF [19]. 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 [6] 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 the supplementary material.

7. Conclusion

In this work, we have presented a novel approach for high-quality neural scene reconstruction and photorealistic 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
