TensoRF: Tensorial Radiance Fields

Anpei Chen[1⋆] Zexiang Xu[2⋆] Andreas Geiger[3] Jingyi Yu[1] Hao Su[4]

[1]ShanghaiTech University [2]Adobe Research
[3]University of Tübingen and MPI-IS, Tübingen [4]UC San Diego
https://apchenstu.github.io/TensoRF/

Abstract. We present TensoRF, a novel approach to model and reconstruct radiance fields. Unlike NeRF that purely uses MLPs, we model the radiance field of a scene as a 4D tensor, which represents a 3D voxel grid with per-voxel multi-channel features. Our central idea is to factorize the 4D scene tensor into multiple compact low-rank tensor components. We demonstrate that applying traditional CANDECOMP/PARAFAC (CP) decomposition – that factorizes tensors into rank-one components with compact vectors – in our framework leads to improvements over vanilla NeRF. To further boost performance, we introduce a novel vector-matrix (VM) decomposition that relaxes the low-rank constraints for two modes of a tensor and factorizes tensors into compact vector and matrix factors. Beyond superior rendering quality, our models with CP and VM decompositions lead to a significantly lower memory footprint in comparison to previous and concurrent works that directly optimize per-voxel features. Experimentally, we demonstrate that TensoRF with CP decomposition achieves fast reconstruction (< 30 min) with better rendering quality and even a smaller model size (< 4 MB) compared to NeRF. Moreover, TensoRF with VM decomposition further boosts rendering quality and outperforms previous state-of-the-art methods, while reducing the reconstruction time (< 10 min) and retaining a compact model size (< 75 MB).

1 Introduction

Modeling and reconstructing 3D scenes as representations that support high-quality image synthesis is crucial for computer vision and graphics with various applications in visual effects, e-commerce, virtual and augmented reality, and robotics. Recently, NeRF [31] and its many follow-up works [61,26] have shown success on modeling a scene as a radiance field and enabled photo-realistic rendering of scenes with highly complex geometry and view-dependent appearance effects. Despite the fact that (purely MLP-based) NeRF models require small memory, they take a long time (hours or days) to train. In this work, we pursue

⋆ Equal contribution.
Research done when Anpei Chen was in a remote internship with UCSD.

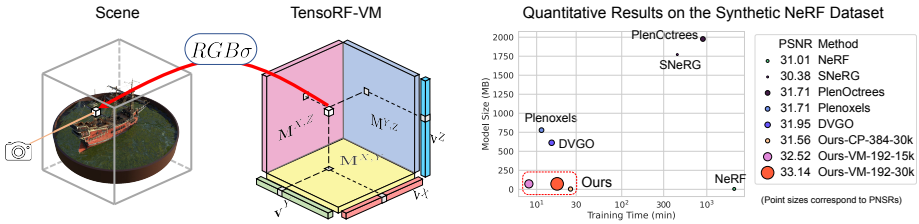

Fig. 1: Left: We model a scene as a tensorial radiance field using a set of vectors ($\mathbf{v}$) and matrices ($\mathbf{M}$) that describe scene appearance and geometry along their corresponding axes. These vector/matrix factors are used to compute volume density σ and view-dependent RGB color via vector-matrix outer products for realistic volume rendering. Right: In comparison with previous and concurrent methods, our TensoRF models can achieve the best rendering quality and are the only methods that can simultaneously achieve fast reconstruction and high compactness. (Our models are denoted with their decomposition techniques, number of components, and training steps.)

a novel approach that is both *efficient in training time* and *compact in memory footprint*, and at the same time achieves *state-of-the-art* rendering quality.

To do so, we propose TensoRF, a novel radiance field representation that is highly compact and also fast to reconstruct, enabling efficient scene reconstruction and modeling. Unlike coordinate-based MLPs used in NeRF, we represent radiance fields as an explicit voxel grid of features. Note that it is *unclear* whether voxel grid representation can benefit the efficiency of reconstruction: While previous work has used feature grids [26,59,17], they require large GPU memory to store the voxels whose size grows cubically with resolution, and some even require pre-computing a NeRF for distillation, leading to very long reconstruction time.

Our work addresses the inefficiency of voxel grid representations in a principled framework, leading to a family of simple yet effective methods. We leverage the fact that a feature grid can naturally be seen as a 4D tensor, where three of its modes correspond to the XYZ axes of the grid and the fourth mode represents the feature channel dimension. This opens the possibility of exploiting classical tensor decomposition techniques – which have been widely applied to high-dimensional data analysis and compression in various fields [23] – for radiance field modeling. We, therefore, propose to factorize the tensor of radiance fields into multiple *low-rank* tensor components, leading to an accurate and compact scene representation. Note that our central idea of tensorizing radiance fields is general and can be potentially adopted to any tensor decomposition technique.

In this work, we first attempt the classic CANDECOMP/PARAFAC (CP) decomposition [6]. We show that TensoRF with CP decomposition can already achieve photo-realistic rendering and lead to a more compact model than NeRF that is purely MLP based (see Fig. 1 and Tab. 1). However, experimentally, to further push reconstruction quality for complex scenes, we have to use more component factors, which undesirably increases training time.

Therefore, we present a novel vector-matrix (VM) decomposition technique that effectively reduces the number of components required for the same expression capacity, leading to faster reconstruction and better rendering. In particular, inspired by the CP and block term decomposition [12], we propose to factorize the full tensor of a radiance field into multiple vector and matrix factors per tensor component. Unlike the sum of outer products of pure vectors in CP decomposition, we consider the sum of vector-matrix outer products (see Fig. 2). In essence, we relax the ranks of two modes of each component by jointly modeling two modes in a matrix factor. While this increases the model size compared to pure vector-based factorization in CP, we enable each component to express more complex tensor data of higher ranks, thus significantly reducing the required number of components in radiance field modeling.

With CP/VM decomposition, our approach compactly encodes spatially varying features in the voxel grid. Volume density and view-dependent color can be decoded from the features, supporting volumetric radiance field rendering. Because a tensor expresses discrete data, we also enable efficient trilinear interpolation for our representation to model a continuous field. Our representation supports various types of per-voxel features with different decoding functions, including neural features – depending on an MLP to regress view-dependent colors from the features – and spherical harmonics (SH) features (coefficients) – allowing for simple color computation from the fixed SH functions and leading to a representation without neural networks.

Our tensorial radiance fields can be effectively reconstructed from multi-view images and enable realistic novel view synthesis. In contrast to previous works that directly reconstruct voxels, our tensor factorization reduces space complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n)$ (with CP) or $\mathcal{O}(n^2)$ (with VM), significantly lowering memory footprint. Note that, although we leverage tensor decomposition, we are not addressing a decomposition/compression problem, but a reconstruction problem based on gradient decent, since the feature grid/tensor is unknown. In essence, our CP/VM decomposition offers low-rank regularization in the optimization, leading to high rendering quality. We present extensive evaluation of our approach with various settings, covering both CP and VM models, different numbers of components and grid resolutions. We demonstrate that all models are able to achieve realistic novel view synthesis results that are on par or better than previous state-of-the-art methods (see Fig. 1 and Tab. 1). More importantly, our approach is of high computation and memory efficiency. All TensoRF models can reconstruct high-quality radiance fields in 30 min; our fastest model with VM decomposition takes less than 10 min, which is significantly faster (about 100x) than NeRF and many other methods, while requiring substantially less memory than previous and concurrent voxel-based methods. Note that, unlike concurrent works [43,32] that require unique data structures and customized CUDA kernels, our model's efficiency gains are obtained using a standard PyTorch implementation. As far as we know, our work is the first that views radiance field modeling from a tensorial perspective and pose the problem of radiance field reconstruction as one of low-rank tensor reconstructions.

2 Related Work

Tensor decomposition. Tensor decomposition [23] has been studied for decades with diverse applications in vision, graphics, machine learning, and other fields [36,20,51,13,1,19]. In general, the most widely used decompositions are Tucker decomposition [50] and CP decomposition [6,16], both of which can be seen as generalizations of the matrix singular value decomposition (SVD). CP decomposition can also be seen as a special Tucker decomposition whose core tensor is diagonal. By combining CP and Tucker decomposition, block term decomposition (BTD) has been proposed with its many variants [12] and used in many vision and learning tasks [2,58,57]. In this work, we leverage tensor decomposition for radiance field modeling. We directly apply CP decomposition and also introduce a new vector-matrix decomposition, which can be seen as a special BTD.

Scene representations and radiance fields. Various scene representations, including meshes [15,52], point clouds [40], volumes [18,41], implicit functions [29,39], have been extensively studied in recent years. Many neural representations [9,62,45,28,4] are proposed for high-quality rendering or natural signal representation [44,48,25]. NeRF [31] introduces radiance fields to address novel view synthesis and achieves photo-realistic quality. This representation has been quickly extended and applied in diverse graphics and vision applications, including generative models [8,33], appearance acquisition [3,5], surface reconstruction [53,35], fast rendering[42,59,17,14], appearance editing [55,27], dynamic capture [24,37] and generative model [34,7]. While leading to realistic rendering and a compact model, NeRF with its pure MLP-based representation has known limitations in slow reconstruction and rendering. Recent methods [59,26,17] have leveraged a voxel grid of features in radiance field modeling, achieving fast rendering. However, these grid-based methods still require long reconstruction time and even lead to high memory costs, sacrificing the compactness of NeRF. Based on feature grids, we present a novel tensorial scene representation, leveraging tensor factorization techniques, leading to fast reconstruction and compact modeling.

Other methods design generalizable network modules trained across scenes to achieve image-dependent radiance field rendering [49,60,54,11] and fast reconstruction [10,56]. Our approach focuses on radiance field representation and only considers per-scene optimization (like NeRF). We show that our representation can already lead to highly efficient radiance field reconstruction without any across-scene generalization. We leave the extensions to generalizable settings as future work.

Concurrent work. The field of radiance field modeling is moving very fast and many concurrent works have appeared on arXiv as preprints over the last three months. DVGO [47] and Plenoxels [43] also optimize voxel grids of (neural or SH) features for fast radiance field reconstruction. However, they still optimize per-voxel features directly like previous voxel-based methods, thus requiring large memory. Our approach instead factorizes the feature grid into compact components and leads to significantly higher memory efficiency. Instant-NGP [32] uses multi-resolution hashing for efficient encoding and also leads to high

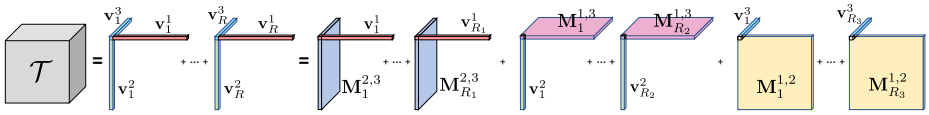

Fig. 2: Tensor factorization. Left: CP decomposition (Eqn. 1), which factorizes a tensor as a sum of vector outer products. Right: our vector-matrix decomposition (Eqn. 3), which factorizes a tensor as a sum of vector-matrix outer products.

compactness. This technique is orthogonal to our factorization-based technique; potentially, each of our vector/matrix factor can be encoded with this hashing technique and we leave such combination as future work.

3 CP and VM Decomposition

We factorize radiance fields into compact components for scene modeling. To do so, we apply both the classic CP decomposition and a new vector-matrix (VM) decomposition; both are illustrated in Fig. 2. We now discuss both decompositions with an example of a 3D (3rd-order) tensor. We will introduce how to apply tensor factorization in radiance field modeling (with a 4D tensor) in Sec. 4.

CP decomposition. Given a 3D tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$, CP decomposition factorizes it into a sum of outer products of vectors (shown in Fig. 2):

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{v}_r^1 \circ \mathbf{v}_r^2 \circ \mathbf{v}_r^3 \tag{1}$$

where $\mathbf{v}_r^1 \circ \mathbf{v}_r^2 \circ \mathbf{v}_r^3$ corresponds to a rank-one tensor component, and $\mathbf{v}_r^1 \in \mathbb{R}^I$, $\mathbf{v}_r^2 \in \mathbb{R}^J$, and $\mathbf{v}_r^3 \in \mathbb{R}^K$ are factorized vectors of the three modes for the rth component. Superscripts denote the modes of each factor; $\circ$ represents the outer product. Hence, each tensor element $\mathcal{T}_{ijk}$ is a sum of scalar products:

$$\mathcal{T}_{ijk} = \sum_{r=1}^{R} \mathbf{v}_{r,i}^1 \mathbf{v}_{r,j}^2 \mathbf{v}_{r,k}^3 \tag{2}$$

where i, j, k denote the indices of the three modes.

CP decomposition factorizes a tensor into multiple vectors, expressing multiple compact rank-one components. CP can be directly applied in our tensorial radiance field modeling and generate high-quality results (see Tab. 1). However, because of too high compactness, CP decomposition can require many components to model complex scenes, leading to high computational costs in radiance field reconstruction. Inspired by block term decomposition (BTD), we present a new VM decomposition, leading to more efficient radiance field reconstruction.

Vector-Matrix (VM) decomposition. Unlike CP decomposition that utilizes pure vector factors, VM decomposition factorizes a tensor into multiple vectors

and matrices as shown in Fig. 2 right. This is expressed by

$$\mathcal{T} = \sum_{r=1}^{R_1} \mathbf{v}_r^1 \circ \mathbf{M}_r^{2,3} + \sum_{r=1}^{R_2} \mathbf{v}_r^2 \circ \mathbf{M}_r^{1,3} + \sum_{r=1}^{R_3} \mathbf{v}_r^3 \circ \mathbf{M}_r^{1,2} \tag{3}$$

where $\mathbf{M}_r^{2,3} \in \mathbb{R}^{J\times K}$, $\mathbf{M}_r^{1,3} \in \mathbb{R}^{I\times K}$, $\mathbf{M}_r^{1,2} \in \mathbb{R}^{I\times J}$ are matrix factors for two (denoted by superscripts) of the three modes. For each component, we relax its two mode ranks to be arbitrarily large, while restricting the third mode to be rank-one; e.g., for component tensor $\mathbf{v}_r^1 \circ \mathbf{M}_r^{2,3}$, its mode-1 rank is 1, and its mode-2 and mode-3 ranks can be arbitrary, depending on the rank of the matrix $\mathbf{M}_r^{2,3}$. In general, instead of using separate vectors in CP, we combine every two modes and represent them by matrices, allowing each mode to be adequately parametrized with a smaller number of components. R_1, R_2, R_3 can be set differently and should be chosen depending on the complexity of each mode. Our VM decomposition can be seen as a special case of general BTD.

Note that, each of our component tensors has more parameters than a component in CP decomposition. While this leads to lower compactness, a VM component tensor can express more complex high-dimensional data than a CP component, thus reducing the required number of components when modeling the same complex function. On the other hand, VM decomposition is still of very high compactness, reducing memory complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$, compared to dense grid representations.

Tensor for scene modeling. In this work, we focus on the task of modeling and reconstructing radiance fields. In this case, the three tensor modes correspond to XYZ axes, and we thus directly denote the modes with XYZ to make it intuitive. Meanwhile, in the context of 3D scene representation, we consider $R_1 = R_2 = R_3 = R$ for most of scenes, reflecting the fact that a scene can distribute and appear equally complex along its three axes. Therefore, Eqn. 3 can be re-written as

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{v}_r^X \circ \mathbf{M}_r^{Y,Z} + \mathbf{v}_r^Y \circ \mathbf{M}_r^{X,Z} + \mathbf{v}_r^Z \circ \mathbf{M}_r^{X,Y} \tag{4}$$

In addition, to simplify notation and the following discussion in later sections, we also denote the three types of component tensors as $\mathcal{A}_r^X = \mathbf{v}_r^X \circ \mathbf{M}_r^{YZ}$, $\mathcal{A}_r^Y = \mathbf{v}_r^Y \circ \mathbf{M}_r^{XZ}$, and $\mathcal{A}_r^Z = \mathbf{v}_r^Z \circ \mathbf{M}_r^{XY}$; here the superscripts XYZ of $\mathcal{A}$ indicate different types of components. With this, a tensor element $\mathcal{T}_{ijk}$ is expressed as

$$\mathcal{T}_{ijk} = \sum_{r=1}^{R} \sum_{m} \mathcal{A}_{r,ijk}^m \tag{5}$$

where $m \in XYZ$, $\mathcal{A}_{r,ijk}^X = \mathbf{v}_{r,i}^X \mathbf{M}_{r,jk}^{YZ}$, $\mathcal{A}_{r,ijk}^Y = \mathbf{v}_{r,j}^Y \mathbf{M}_{r,ik}^{XZ}$, and $\mathcal{A}_{r,ijk}^Z = \mathbf{v}_{r,k}^Z \mathbf{M}_{r,ij}^{XY}$. Similarly, we can also denote a CP component as $\mathcal{A}^\gamma = \mathbf{v}_r^X \circ \mathbf{v}_r^Y \circ \mathbf{v}_r^Z$, and Eqn. 5 can also express CP decomposition by considering $m = \gamma$, where the summation over m can be removed.

4 Tensorial Radiance Field Representation

We now present our Tensorial Radiance Field Representation (TensoRF). For simplicity, we focus on presenting TensoRF with our VM decomposition. CP decomposition is simpler and its decomposition equations can be directly applied with minimal modification (like Eqn. 5).

4.1 Feature grids and radiance field.

Our goal is to model a radiance field, which is essentially a function that maps any 3D location $\mathbf{x}$ and viewing direction d to its volume density σ and view-dependent color c, supporting differentiable ray marching for volume rendering. We leverage a regular 3D grid $\mathcal{G}$ with per-voxel multi-channel features to model such a function. We split it (by feature channels) into a geometry grid $\mathcal{G}_\sigma$ and an appearance grid $\mathcal{G}_c$, separately modelling the volume density σ and view-dependent color c.

Our approach supports various types of appearance features in $\mathcal{G}_c$, depending on a pre-selected function S that coverts an appearance feature vector and a viewing direction d to color c. For example, S can be a small MLP or spherical harmonics (SH) functions, where $\mathcal{G}_c$ contains neural features and SH coefficients respectively. We show that both MLP and SH functions work well with our model (see Tab.1). On the other hand, we consider a single-channel grid $\mathcal{G}_\sigma$, whose values represent volume density directly, without requiring an extra converting function. The continuous grid-based radiance field can be written as

$$\sigma, c = \mathcal{G}_\sigma(\mathbf{x}), S(\mathcal{G}_c(\mathbf{x}), d) \tag{6}$$

where $\mathcal{G}_\sigma(\mathbf{x})$, $\mathcal{G}_c(\mathbf{x})$ represent the trilinearly interpolated features from the two grids at location $\mathbf{x}$. We model $\mathcal{G}_\sigma$ and $\mathcal{G}_c$ as factorized tensors.

4.2 Factorizing radiance fields

While $\mathcal{G}_\sigma \in \mathbb{R}^{I\times J\times K}$ is a 3D tensor, $\mathcal{G}_c \in \mathbb{R}^{I\times J\times K\times P}$ is a 4D tensor. Here I, J, K correspond to the resolutions of the feature grid along the X, Y, Z axes, and P is the number of appearance feature channels.

We factorize these radiance field tensors to compact components. In particular, with the VM decomposition. the 3D geometry tensor $\mathcal{G}_\sigma$ is factorized as

$$\mathcal{G}_\sigma = \sum_{r=1}^{R_\sigma} \mathbf{v}^X_{\sigma,r} \circ \mathbf{M}^{YZ}_{\sigma,r} + \mathbf{v}^Y_{\sigma,r} \circ \mathbf{M}^{XZ}_{\sigma,r} + \mathbf{v}^Z_{\sigma,r} \circ \mathbf{M}^{XY}_{\sigma,r} = \sum_{r=1}^{R_\sigma} \sum_{m\in XYZ} \mathcal{A}^m_{\sigma,r} \tag{7}$$

The appearance tensor $\mathcal{G}_c$ has an additional mode corresponding to the feature channel dimension. Note that, compared to the XYZ modes, this mode is often of lower dimension, leading to a lower rank. Therefore, we do not combine this mode with other modes in matrix factors and instead only use vectors, denoted

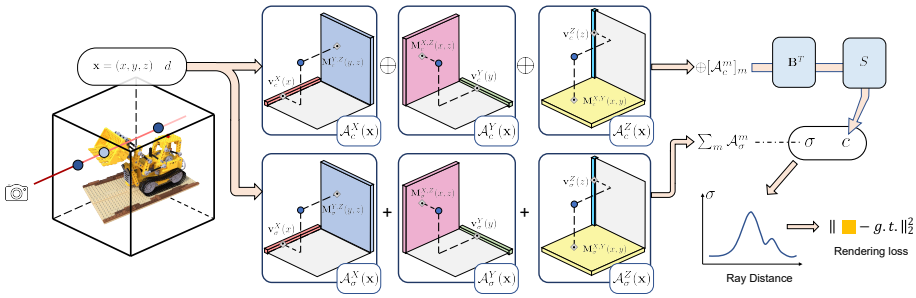

Fig. 3: TensoRF (VM) reconstruction and rendering. We model radiance fields as tensors using a set of vectors ($\mathbf{v}$) and matrices ($\mathbf{M}$), which describe the scene along their corresponding (XYZ) axes and are used for computing volume density σ and view-dependent color c in differentiable ray marching. For each shading location $\mathbf{x} = (x, y, z)$, we use linearly/bilinearly sampled values from the vector/matrix factors to efficiently compute the corresponding trilinearly interpolated values ($\mathcal{A}(\mathbf{x})$) of the tensor components. The density component values ($\mathcal{A}_\sigma(\mathbf{x})$) are summed to get the volume density (σ) directly. The appearance values ($\mathcal{A}_c(\mathbf{x})$) are concatenated into a vector ($\oplus[\mathcal{A}_c^m(x)]_m$) that is then multiplied by an appearance matrix $\mathbf{B}$ and sent to the decoding function S for RGB color (c) regression.

by $\mathbf{b}_r$, for this mode in the factorization. Specifically, $\mathcal{G}_c$ is factorized as

$$\begin{aligned}\mathcal{G}_c &= \sum_{r=1}^{R_c} \mathbf{v}_{c,r}^X \circ \mathbf{M}_{c,r}^{YZ} \circ \mathbf{b}_{3r-2} + \mathbf{v}_{c,r}^Y \circ \mathbf{M}_{c,r}^{XZ} \circ \mathbf{b}_{3r-1} + \mathbf{v}_{c,r}^Z \circ \mathbf{M}_{c,r}^{XY} \circ \mathbf{b}_{3r} \\ &= \sum_{r=1}^{R_c} \mathcal{A}_{c,r}^X \circ \mathbf{b}_{3r-2} + \mathcal{A}_{c,r}^Y \circ \mathbf{b}_{3r-1} + \mathcal{A}_{c,r}^Z \circ \mathbf{b}_{3r} \qquad (8)\end{aligned}$$

Note that, we have $3R_c$ vectors $\mathbf{b}_r$ to match the total number of components.

Overall, we factorize the entire tensorial radiance field into $3R_\sigma + 3R_c$ matrices ($\mathbf{M}_{\sigma,r}^{YZ}$,...,$\mathbf{M}_{c,r}^{YZ}$,...) and $3R_\sigma + 6R_c$ vectors ($\mathbf{v}_{\sigma,r}^X$,...,$\mathbf{v}_{c,r}^X$,...,$\mathbf{b}_r$). In general, we adopt $R_\sigma \ll I, J, K$ and $R_c \ll I, J, K$, leading to a highly compact representation that can encode a high-resolution dense grid. In essence, the XYZ-mode vector and matrix factors, $\mathbf{v}_{\sigma,r}^X$, $\mathbf{M}_{\sigma,r}^{YZ}$, $\mathbf{v}_{c,r}^X$, $\mathbf{M}_{c,r}^{YZ}$, ..., describe the spatial distributions of the scene geometry and appearance along their corresponding axes. On the other hand, the appearance feature-mode vectors $\mathbf{b}_r$ express the global appearance correlations. By stacking all $\mathbf{b}_r$ as columns together, we have a $P \times 3R_c$ matrix $\mathbf{B}$; this matrix $\mathbf{B}$ can also be seen as a global appearance dictionary that abstracts the appearance commonalities across the entire scene.

4.3 Efficient feature evaluation.

Our factorization-based model can compute each voxel's feature vector at low costs, only requiring one value per XYZ-mode vector/matrix factor. We also enable efficient trilinear interpolation for our model, leading to a continuous field.

Direct evaluation. With VM factorization, a density value $\mathcal{G}_{\sigma,ijk}$ of a single voxel at indices ijk can be directly and efficiently evaluated by following Eqn. 5:

$$\mathcal{G}_{\sigma,ijk} = \sum_{r=1}^{R_\sigma} \sum_{m \in XYZ} \mathcal{A}^m_{\sigma,r,ijk} \tag{9}$$

Here, computing each $\mathcal{A}^m_{\sigma,r,ijk}$ only requires indexing and multiplying two values from its corresponding vector and matrix factors.

As for the appearance grid $\mathcal{G}_c$, we always need to compute a full P-channel feature vector, which the shading function S requires as input, corresponding to a 1D slice of $\mathcal{G}_c$ at fixed XYZ indices ijk:

$$\mathcal{G}_{c,ijk} = \sum_{r=1}^{R_c} \mathcal{A}^X_{c,r,ijk}\mathbf{b}_{3r-2} + \mathcal{A}^Y_{c,r,ijk}\mathbf{b}_{3r-1} + \mathcal{A}^Z_{c,r,ijk}\mathbf{b}_{3r} \tag{10}$$

Here, there's no additional indexing for the feature mode, since we compute a full vector. We further simplify Eqn. 10 by re-ordering the computation. For this, we denote $\oplus[\mathcal{A}^m_{c,ijk}]_{m,r}$ as the vector that stacks all $\mathcal{A}^m_{c,r,ijk}$ values for $m = X, Y, Z$ and $r = 1, ..., R_c$, which is a vector of $3R_c$ dimensions; $\oplus$ can also be considered as the concatenation operator that concatenates all scalar values (1-channel vectors) into a $3R_c$-channel vector in practice. Using matrix $\mathbf{B}$ (introduced in Sec. 4.1) that stacks all $\mathbf{b}_r$, Eqn. 10 is equivalent to a matrix vector product:

$$\mathcal{G}_{c,ijk} = \mathbf{B}(\oplus[\mathcal{A}^m_{c,ijk}]_{m,r}) \tag{11}$$

Note that, Eqn. 11 is not only formally simpler but also leads to a simpler implementation in practice. Specifically, when computing a large number of voxels in parallel, we first compute and concatenate $\mathcal{A}^m_{c,r,ijk}$ for all voxels as column vectors in a matrix and then multiply the shared matrix $\mathbf{B}$ once.

Trilinear interpolation. We apply trilinear interpolation to model a continuous field. Naïvely achieving trilinear interpolation is costly, as it requires evaluation of 8 tensor values and interpolating them, increasing computation by a factor of 8 compared to computing a single tensor element. However, we find that trilinearly interpolating a component tensor is naturally equivalent to interpolating its vector/matrix factors linearly/bilinearly for the corresponding modes, thanks to the beauty of linearity of the trilinear interpolation and the outer product.

For example, given a component tensor $\mathcal{A}^X_r = \mathbf{v}^X_r \circ \mathbf{M}^{YZ}_r$ with its each tensor element $\mathcal{A}_{r,ijk} = \mathbf{v}^X_{r,i}\mathbf{M}^{YZ}_{r,jk}$, we can compute its interpolated values as:

$$\mathcal{A}^X_r(\mathbf{x}) = \mathbf{v}^X_r(x)\mathbf{M}^{YZ}_r(y,z) \tag{12}$$

where $\mathcal{A}^X_r(\mathbf{x})$ is $\mathcal{A}_r$'s trilinearly interpolated value at location $\mathbf{x} = (x, y, z)$ in the 3D space, $\mathbf{v}^X_r(x)$ is $\mathbf{v}^X_r$'s linearly interpolated value at x along X axis, and $\mathbf{M}^{YZ}_r(y,z)$ is $\mathbf{M}^{YZ}_r$'s bilinearly interpolated value at (y,z) in the YZ plane. Similarly, we have $\mathcal{A}^Y_r(\mathbf{x}) = \mathbf{v}^Y_r(y)\mathbf{M}^{XZ}_r(x,z)$ and $\mathcal{A}^Z_r(\mathbf{x}) = \mathbf{v}^Z_r(z)\mathbf{M}^{XY}_r(x,y)$

(for CP decomposition $\mathcal{A}_r^\gamma(\mathbf{x}) = \mathbf{v}_r^X(x)\mathbf{v}_r^Y(y)\mathbf{v}_r^Z(z)$ is also valid). Thus, trilinearly interpolating the two grids is expressed as

$$\mathcal{G}_\sigma(\mathbf{x}) = \sum_r \sum_m \mathcal{A}_{\sigma,r}^m(\mathbf{x}) \tag{13}$$

$$\mathcal{G}_c(\mathbf{x}) = \mathbf{B}(\oplus[\mathcal{A}_{c,r}^m(\mathbf{x})]_{m,r}) \tag{14}$$

These equations are very similar to Eqn. 9 and 11, simply replacing the tensor elements with interpolated values. We avoid recovering 8 individual tensor elements for trilinear interpolation and instead directly recover the interpolated value, leading to low computation and memory costs at run time.

4.4 Rendering and reconstruction.

Equations 6, 12–14 describe the core components of our model. By combining Eqn. 6,13,14, our factorized tensorial radiance field can be expressed as

$$\sigma, c = \sum_r \sum_m \mathcal{A}_{\sigma,r}^m(\mathbf{x}) \ , \ S(\mathbf{B}(\oplus[\mathcal{A}_{c,r}^m(\mathbf{x})]_{m,r}), d) \tag{15}$$

i.e., we obtain continuous volume density and view-dependent color given any 3D location and viewing direction. This allows for high-quality radiance field reconstruction and rendering. Note that, this equation is general and describes TensoRF with both CP and VM decomposition. Our full pipeline of radiance field reconstruction and rendering with VM decomposition is illustrated in Fig. 3.

Volume rendering. To render images, we use differentiable volume rendering, following NeRF [31]. Specifically, for each pixel, we march along a ray, sampling Q shading points along the ray and computing the pixel color by

$$C = \sum_{q=1}^{Q} \tau_q(1 - \exp(-\sigma_q \Delta_q))c_q, \ \tau_q = \exp(-\sum_{p=1}^{q-1} \sigma_p \Delta_p) \tag{16}$$

Here, σ_q, c_q are the corresponding density and color computed by our model at their sampled locations $\mathbf{x}_q$; Δ_q is the ray step size and τ_q represents transmittance.

Reconstruction. Given a set of multi-view input images with known camera poses, our tensorial radiance field is optimized per scene via gradient descent, minimizing an L2 rendering loss, using only the ground truth pixel colors as supervision. Our radiance field is explained by tensor factorization and modeled by a set of global vectors and matrices as basis factors that correlate and regularize the entire field in the optimization. However, this can sometimes lead to overfitting and local minima issues in gradient descent, leading to outliers or noises in regions with fewer observations. We utilize standard regularization terms that are commonly used in compressive sensing, including an L1 norm loss and a TV (total variation) loss on our vector and matrix factors, which effectively address these issues. We find that only applying the L1 sparsity loss is adequate

for most datasets. However, for real datasets that have very few input images (like LLFF[30]) or imperfect capture conditions (like Tanks and Temples [22,26] that has varying exposure and inconsistent masks), a TV loss is more efficient than the L1 norm loss.

To further improve quality and avoid local minima, we apply coarse-to-fine reconstruction. Unlike previous coarse-to-fine techniques that require unique subdivisions on their sparse chosen sets of voxels, our coarse-to-fine reconstruction is simply achieved by linearly and bilinearly upsampling our XYZ-mode vector and matrix factors.

5 Implementation details

We briefly discuss our implementation; please refer to the appendix for more details. We implement our TensoRF using PyTorch [38], without customized CUDA kernels. We implement the feature decoding function S as either an MLP or SH function and use $P = 27$ features for both. For SH, this corresponds to 3rd-order SH coefficients with RGB channels. For neural features, we use a small MLP with two FC layers (with 128-channel hidden layers) and ReLU activation.

We use the Adam optimizer [21] with initial learning rates of 0.02 for tensor factors and (when using neural features) 0.001 for the MLP decoder. We optimize our model for T steps with a batch size of 4096 pixel rays on a single Tesla V100 GPU (16GB). We apply a feature grid with a total number of N^3 voxels; the actual resolution of each dimension is computed based on the shape of the bounding box. To achieve coarse-to-fine reconstruction, we start from an initial low-resolution grid with N_0^3 voxels with $N_0 = 128$; we then upsample the vectors and matrices linearly and bilinearly at steps 2000, 3000, 4000, 5500, 7000 with the numbers of voxels interpolated between N_0^3 and N^3 linearly in logarithmic space. Please refer to Sec. 6 for the analysis on different total steps (T), different resolutions (N), and different number of total components ($3R_\sigma + 3R_c$).

6 Experiments

We now present an extensive evaluation of our tensorial radiance fields. We first analyze our decomposition techniques, the number of components, grid resolutions, and optimization steps. We then compare our approach with previous and concurrent works on both 360° objects and forward-facing datasets.

Analysis of different TensoRF models. We evaluate our TensoRF on the Synthetic NeRF dataset [31] using both CP and VM decompositions with different numbers of components and different numbers of grid voxels. Table 2 shows the averaged rendering PSNRs, reconstruction time, and model size for each model. We use the same MLP decoding function (as described in Sec. 5) for all variants and optimize each model for 30k steps with a batch size of 4096.

Note that both TensoRF-CP and TensoRF-VM achieve consistently better rendering quality with more components or higher grid resolutions. TensoRF-CP

achieves super compact modeling; even the largest model with 384 components and 500^3 voxels requires less than 4MB. This CP model also achieves the best rendering quality in all of our CP variants, leading to a high PSNR of 31.56, which even outperforms vanilla NeRF (see Tab. 1).

On the other hand, because it compresses more parameters in each component, TensoRF-VM achieves significantly better rendering quality than TensoRF-CP; even the smallest TensoRF-VM model with only 48 components and 200^3 voxels is able to outperform the best CP model that uses many more components and voxels. Remarkably, the PSNR of 31.81 from this smallest VM model (which only takes 8.6 MB) is already higher than the PSNRs of NeRF and many other previous and concurrent techniques (see Tab. 1). In addition, 192 components are generally adequate for TensoRF-VM; doubling the number to 384 only leads to marginal improvement. TensoRF-VM with 300^3 voxels can already lead to high PSNRs close to or greater than 33, while retaining compact model sizes (<72MB). Increasing the resolution further leads to improved quality, but also larger model size.

Also note that all of our TensoRF models can achieve very fast reconstruction. Except for the largest VM model, all models finish reconstruction in less than 30 min, significantly faster than NeRF and many previous methods (see Tab. 1).

Optimization steps. We further evaluate our approach with different optimization steps for our best CP model and the VM models with 300^3 voxels. PSNRs and reconstruction time are shown in Tab. 3. All of our models consistently achieve better rendering quality with more steps. Our compact CP-384 model (3.9MB) can even achieve a PSNR greater than 32 after 60k steps, higher than the PSNRs of all previous methods in Tab. 1. On the other hand, our VM models can quickly achieve high rendering quality in very few steps. With only 15k steps, many models achieve high PSNRs that are already state-of-the-art.

Comparisons on 360° scenes. We compare our approach with state-of-the-art novel view synthesis methods, including previous works (SRN[46], NeRF[31], NSVF[26]), SNeRG[17], PlenOctrees[59]) and concurrent works (Plenoxels [43], DVGO[47]). In particular, we compare with them using our best CP model and our VM models (300^3 voxels) with 48 and 192 components. We also show a 192-component VM model with spherical harmonics shading function. Table 1 shows the quantitative results (PSNRs and SSIMs) of ours and comparison methods on three challenging datasets, where we also show the corresponding batch size, optimization steps, time, and final output model size for each model, to compare all methods from multiple perspectives. Note that all of our CP and VM models can outperform NeRF on all three datasets while taking substantially less optimization time and fewer steps. Our best VM-192 model can even achieve the best PSNRs and SSIMs on all datasets, significantly outperforming the comparison methods. Our approach can also achieve qualitatively better renderings with more appearance and geometry details and less outliers, as shown in Fig. 4.

Our models are highly efficient, which all require less than 75MB space and can be reconstructed in less than 30 min. This corresponds to more than 70x speed up

			Synthetic-NeRF				NSVF		TanksTemples	
Method	BatchSize	Steps	Time ↓	Size(MB)↓	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
SRN [46]	-	-	>10h	-	22.26	0.846	24.33	0.882	24.10	0.847
NSVF [26]	8192	150k	>48*h	-	31.75	0.953	35.18	0.979	28.48	0.901
NeRF [31]	4096	300k	∼35h	5.00	31.01	0.947	30.81	0.952	25.78	0.864
SNeRG [17]	8192	250k	∼15h	1771.5	30.38	0.950	-	-	-	-
PlenOctrees [59]	1024	200k	∼15h	1976.3	31.71	0.958	-	-	27.99	0.917
Plenoxels [43]	5000	128k	11.4m	778.1	31.71	0.958	-	-	27.43	0.906
DVGO [47]	5000	30k	15.0m	612.1	31.95	0.957	35.08	0.975.	28.41	0.911
Ours-CP-384	4096	30k	25.2m	3.9	31.56	0.949	34.48	0.971	27.59	0.897
Our-VM-192-SH	4096	30k	16.8m	71.9	32.00	0.955	35.30	0.977	27.81	0.907
Ours-VM-48	4096	30k	13.8m	18.9	32.39	0.957	35.34	0.976	28.06	0.909
Ours-VM-192	4096	15k	8.1m	71.8	32.52	0.959	35.59	0.978	28.07	0.913
Ours-VM-192	4096	30k	17.4m	71.8	33.14	0.963	36.52	0.982	28.56	0.920

Table 1: We compare our method with previous and concurrent novel view synthesis methods on three datasets. All scores of the baseline methods are directly taken from their papers whenever available. We also report the averaged reconstruction time and model size for the Synthetic-NeRF dataset. NVSF requires 8 GPUs for optimization (marked by a star), while others run on a single GPU. DVGO's 30k steps correspond to 10k for coarse and 20k for fine reconstruction.

	#Comp	200^3	300^3	500^3
TensoRF-CP	48	27.98/09:29/0.74	28.24/11:45/1.09	28.38/14:20/1.85
	96	28.50/09:57/0.88	28.83/12:12/1.29	29.06/15:27/2.18
	192	29.50/11:09/1.08	29.99/13:41/1.59	30.33/18.03/2.66
	384	30.47/14:41/1.59	31.08/18:09/2.33	31.56/25:11/3.93
TensoRF-VM	48	31.81/11:29/08.6	32.39/13:51/23.5	32.63/18:17/55.8
	96	32.33/11:54/16.5	32.86/14:08/37.3	33.06/20:11/105.
	192	32.63/13:26/32.3	33.14/17:36/76.7	33.31/27.18/204.
	384	32.69/17:24/63.4	33.21/25:14/143.	33.39/43:19/397.

Table 2: We compare the averaged PSNRs / optimization time (mm:ss) / model sizes (MB) of CP and VM TensoRF models on Synthetic NeRF dataset [31] with different numbers of components and grid resolutions, optimized for 30k steps.

	5k	15k	30k	60k	5k	15k	30k	60k
CP-384	28.37	30.80	31.56	32.03	03:03	11:30	25:11	51:47
VM-48	29.28	31.80	32.39	32.68	01:57	06:21	13:51	27:20
VM-96	29.65	32.26	32.86	33.17	02:01	06:41	14:08	28:57
VM-192	29.86	32.52	33.14	33.44	02:16	08:08	17:37	35:50
VM-384	29.95	32.62	33.21	33.52	02:51	11:30	25:14	52:50

Table 3: PSNRs and time of CP and VM models with different training steps on the Synthetic-NeRF dataset [31].

Method	Time ↓	Size	PSNR↑	SSIM↑
NeRF [31]	36h	**5.00M**	26.50	0.811
Plenoxels [43]	24:00m	2.59G	26.29	0.829
Ours-VM-48	**19:44m**	90.4M	26.51	0.832
Ours-VM-96	25.43m	179.7M	**26.73**	**0.839**

Table 4: Quantitative comparisons of our method with NeRF and Plenoxels on forward-facing scenes [30].

compared to NeRF that requires about 1.5 days for optimization. Our CP model is even more compact than NeRF. Moreover, SNeRG and PlenOctrees require pre-training a NeRF-like MLP, requiring long reconstruction time too. DVGO and Plenoxels are concurrent works, which can also achieve fast reconstruction in less than 15 min. However, as both are voxel-based methods and directly optimize voxel values, they lead to huge model sizes similar to previous voxel-based methods like SNeRG and PlenOctrees. In contrast, we factorize feature grids and model them with compact vectors and matrices, leading to substantially

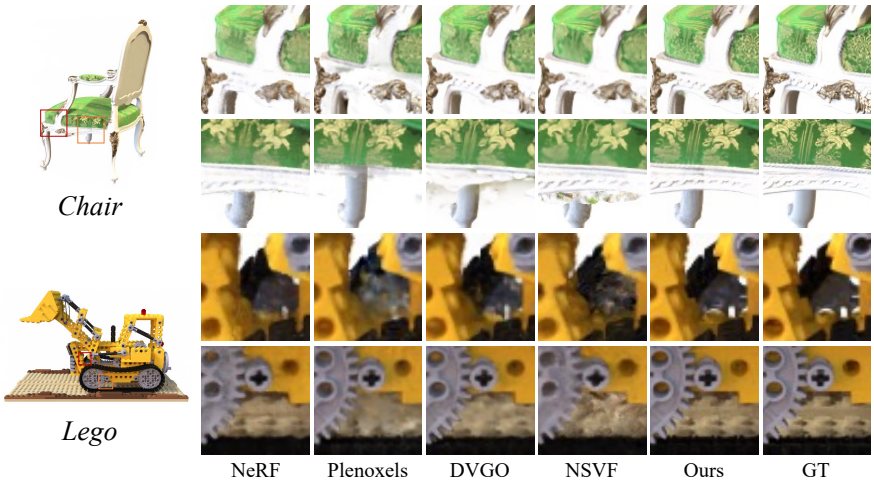

Fig. 4: Qualitative results of our VM-192-30k model and comparison methods (NeRF[31], plenoxels [43], DVGO [47], NSVF [26]) on two Synthetic NeRF scenes.

smaller model sizes. Meanwhile, our VM-192 can even reconstruct faster than DVGO and Plenoxels, taking only 15k steps, and achieving better quality in most cases. In fact, Plenoxels' fast reconstruction relies on quickly optimizing significantly more steps ($>$ 4 times our steps) with their CUDA implementation. Our models are implemented with standard PyTorch modules and already achieve much better rendering quality with fewer steps taking comparable and even less reconstruction time than Plenoxels. Note that our SH model essentially represents the same underlying feature grid as Plenoxels but can still lead to more compact modeling and better quality with fewer steps, showing the advantages of our factorization based modeling. In general, our approach enables fast reconstruction, compact modeling, and photo-realistic rendering simultaneously.

Forward-facing scenes. Our approach can also achieve efficient and high-quality radiance field reconstruction for forward-facing scenes. We show quantitative results of our two VM models on the LLFF dataset [30] and compare with NeRF and Plenoxels in Tab. 4. Our models outperform the previous state-of-the-art NeRF and take significantly less reconstruction time. Compared with Plenoxels [43], our approach leads to comparable or faster reconstruction speed, better quality, and substantially smaller model sizes.

7 Conclusion.

We have presented a novel approach for high-quality scene reconstruction and rendering. We propose a novel scene representation – TensoRF which leverages tensor decomposition techniques to model and reconstruct radiance fields compactly as factorized low-rank tensor components. We hope our findings in tensorized low-rank feature modeling can inspire other modeling and reconstruction tasks.

References

1. Ballester-Ripoll, R., Pajarola, R.: Tensor decomposition methods in visual computing. IEEE Vis. Tutorials **3** (2016)
2. Ben-Younes, H., Cadene, R., Thome, N., Cord, M.: Block: Bilinear superdiagonal fusion for visual question answering and visual relationship detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 8102–8109 (2019)
3. Bi, S., Xu, Z., Srinivasan, P., Mildenhall, B., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Neural reflectance fields for appearance acquisition. arXiv preprint arXiv:2008.03824 (2020)
4. Bi, S., Xu, Z., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. In: Proc. ECCV (2020)
5. Boss, M., Braun, R., Jampani, V., Barron, J.T., Liu, C., Lensch, H.: Nerd: Neural reflectance decomposition from image collections. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12684–12694 (2021)
6. Carroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. Psychometrika **35**(3), 283–319 (1970)
7. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., Mello, S.D., Gallo, O., Guibas, L., Tremblay, J., Khamis, S., Karras, T., Wetzstein, G.: Efficient geometry-aware 3D generative adversarial networks. In: CVPR (2022)
8. Chan, E.R., Monteiro, M., Kellnhofer, P., Wu, J., Wetzstein, G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5799–5809 (2021)
9. Chen, A., Wu, M., Zhang, Y., Li, N., Lu, J., Gao, S., Yu, J.: Deep surface light fields. Proceedings of the ACM on Computer Graphics and Interactive Techniques **1**(1), 1–17 (2018)
10. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14124–14133 (2021)
11. Chibane, J., Bansal, A., Lazova, V., Pons-Moll, G.: Stereo radiance fields (srf): Learning view synthesis from sparse views of novel scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (jun 2021)
12. De Lathauwer, L.: Decompositions of a higher-order tensor in block terms—part ii: Definitions and uniqueness. SIAM Journal on Matrix Analysis and Applications **30**(3), 1033–1066 (2008)
13. Deng, H., Liu, Y., Wang, B., Yang, J., Ma, L., Holzschuch, N., Yan, L.Q.: Constant-cost spatio-angular prefiltering of glinty appearance using tensor decomposition. ACM Transactions on Graphics (TOG) **41**(2), 1–17 (2022)
14. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14346–14355 (2021)
15. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mâché approach to learning 3d surface generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 216–224 (2018)
16. Harshman, R.A., et al.: Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis (1970)

17. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. arXiv preprint arXiv:2103.14645 (2021)
18. Ji, M., Gall, J., Zheng, H., Liu, Y., Fang, L.: SurfaceNet: An end-to-end 3D neural network for multiview stereopsis. In: Proc. ICCV (2017)
19. Ji, Y., Wang, Q., Li, X., Liu, J.: A survey on tensor techniques and applications in machine learning. IEEE Access **7**, 162950–162990 (2019)
20. Kamal, M.H., Heshmat, B., Raskar, R., Vandergheynst, P., Wetzstein, G.: Tensor low-rank and sparse light field photography. Computer Vision and Image Understanding **145**, 172–181 (2016)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
22. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics **36**(4) (2017)
23. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review **51**(3), 455–500 (2009)
24. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6498–6508 (2021)
25. Liang, R., Sun, H., Vijaykumar, N.: Coordx: Accelerating implicit neural representation with a split mlp architecture. arXiv preprint arXiv:2201.12425 (2022)
26. Liu, L., Gu, J., Lin, K.Z., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. NeurIPS (2020)
27. Liu, S., Zhang, X., Zhang, Z., Zhang, R., Zhu, J.Y., Russell, B.: Editing conditional radiance fields. arXiv preprint arXiv:2105.06466 (2021)
28. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. ACM Trans. Graph. (2019)
29. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. Proc. CVPR (2019)
30. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019)
31. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European conference on computer vision. pp. 405–421. Springer (2020)
32. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. **41**(4), 102:1–102:15 (Jul 2022)
33. Niemeyer, M., Geiger, A.: Giraffe: Representing scenes as compositional generative neural feature fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11453–11464 (2021)
34. Niemeyer, M., Geiger, A.: Giraffe: Representing scenes as compositional generative neural feature fields. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
35. Oechsle, M., Peng, S., Geiger, A.: Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: International Conference on Computer Vision (ICCV) (2021)
36. Panagakis, Y., Kossaifi, J., Chrysos, G.G., Oldfield, J., Nicolaou, M.A., Anandkumar, A., Zafeiriou, S.: Tensor methods in computer vision and deep learning. Proceedings of the IEEE **109**(5), 863–890 (2021)

37. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. ACM Trans. Graph. **40**(6) (dec 2021)
38. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
39. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: European Conference on Computer Vision. pp. 523–540. Springer (2020)
40. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proc. CVPR (2017)
41. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3d data. In: Proc. CVPR (2016)
42. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In: International Conference on Computer Vision (ICCV) (2021)
43. Sara Fridovich-Keil and Alex Yu, Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR (2022)
44. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Advances in Neural Information Processing Systems **33**, 7462–7473 (2020)
45. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: Deepvoxels: Learning persistent 3D feature embeddings. In: Proc. CVPR (2019)
46. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In: Advances in Neural Information Processing Systems (2019)
47. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. arXiv preprint arXiv:2111.11215 (2021)
48. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. NeurIPS (2020)
49. Trevithick, A., Yang, B.: Grf: Learning a general radiance field for 3d scene representation and rendering. In: arXiv:2010.04595 (2020)
50. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3), 279–311 (1966)
51. Vasilescu, M.A.O., Terzopoulos, D.: Tensortextures: Multilinear image-based rendering. In: ACM SIGGRAPH 2004 Papers, pp. 336–342 (2004)
52. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single RGB images. In: Proc. ECCV (2018)
53. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. NeurIPS (2021)
54. Wang, Q., Wang, Z., Genova, K., Srinivasan, P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: CVPR (2021)
55. Xiang, F., Xu, Z., Hasan, M., Hold-Geoffroy, Y., Sunkavalli, K., Su, H.: Neutex: Neural texture mapping for volumetric neural rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7119–7128 (2021)

56. Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., Neumann, U.: Point-nerf: Point-based neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5438–5448 (2022)
57. Ye, J., Li, G., Chen, D., Yang, H., Zhe, S., Xu, Z.: Block-term tensor neural networks. Neural Networks **130**, 11–21 (2020)
58. Ye, J., Wang, L., Li, G., Chen, D., Zhe, S., Chu, X., Xu, Z.: Learning compact recurrent neural networks with block-term tensor decomposition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9378–9387 (2018)
59. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. arXiv preprint arXiv:2103.14024 (2021)
60. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: CVPR (2021)
61. Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020)
62. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: learning view synthesis using multiplane images. ACM Transactions on Graphics **37**(4), 1–12 (2018)