Geometric Algebra Planes: Convex Implicit Neural Volumes

Irmak Sivgin*, Sara Fridovich-Keil*[*] Gordon Wetzstein, & Mert Pilanci
Department of Electrical Engineering, Stanford University
{isivgin, sarafk, gordonwz, pilanci}@stanford.edu
* denotes equal contribution

Volume parameterizations abound in recent literature, from the classic voxel grid to the implicit neural representation and everything in between. While implicit representations have shown impressive capacity and better memory efficiency compared to voxel grids, to date they require training via nonconvex optimization. This nonconvex training process can be slow to converge and sensitive to initialization and hyperparameter choices that affect the final converged result. We introduce a family of models, GA-Planes, that is the first class of implicit neural volume representations that can be trained by *convex* optimization. GA-Planes models include any combination of features stored in tensor basis elements, followed by a neural feature decoder. They generalize many existing representations and can be adapted for convex, semiconvex, or nonconvex training as needed for different inverse problems. In the 2D setting, we prove that GA-Planes is equivalent to a low-rank plus low-resolution matrix factorization; we show that this approximation outperforms the classic low-rank plus sparse decomposition for fitting a natural image. In 3D, we demonstrate GA-Planes' competitive performance in terms of expressiveness, model size, and optimizability across three volume fitting tasks: radiance field reconstruction, 3D segmentation, and video segmentation. Code is available at https://github.com/sivginirmak/Geometric-Algebra-Planes.

1. Introduction

Volumes are everywhere—from the world we live in to the videos we watch to the organs and tissues inside our bodies. In recent years tremendous progress has been made in modeling these volumes using measurements and computation [1], to make them accessible for downstream tasks in applications including manufacturing [2, 3], robotic navigation [4, 5], entertainment and culture [6, 7], and medicine [8–11]. All methods that seek to model a volume face a three-way tradeoff between *model size*, which determines hardware memory requirements, *expressiveness*, which determines how faithfully the model can represent the underlying volume, and *optimizability*, which captures how quickly and reliably the model can learn the volume from measurements. Certain applications place stricter requirements on model size (e.g. for deployment on mobile or edge devices), expressiveness (e.g. resolution required for medical diagnosis or safe robotic navigation), or optimizability (e.g. for interactive applications), but all stand to benefit from improvements to this three-way pareto frontier.

Many existing strategies have been successfully applied at different points along this pareto frontier; some representative examples from computer vision are summarized in Appendix A.1. Our goal is to maintain or surpass the existing pareto frontier of model size and expressiveness while improving optimization stability through convex optimization.

Our approach introduces *convex* and *semiconvex* reformulations of the volume modeling optimization process that apply to a broad class of volume models we call *Geometric Algebra Planes*, or GA-Planes for short. We adopt the term *semiconvex* for Burer-Monteiro (BM) factorizations of a convex objective, as introduced in [12], within the context of convex neural networks. BM factorized problems have the property that every local minimum is globally optimal [12].

[*] Incoming faculty at the School of Electrical & Computer Engineering, Georgia Institute of Technology (sfk@gatech.edu).

Submitted to Second Conference on Parsimony and Learning (CPAL 2025). Do not distribute.

GA-Planes is a mixture-of-primitives model that generalizes several existing volume models including voxels and tensor factorizations. Most importantly, most models in this family can be formulated for optimization by a convex program, as long as the objective function (to fit measurements of the volume) is convex. At the same time, *any* GA-Planes model can also be formulated for nonconvex optimization towards *any* objective, matching the range of applicability enjoyed by common models. While only our convex and semiconvex models come with guarantees of convergence to global optimality, all the models we introduce extend the pareto frontier of model size, expressiveness, and optimizability on diverse tasks.

Concretely, we make the following contributions:

- We introduce GA-Planes, a mixture-of-primitives volume parameterization inspired by geometric algebra basis elements. GA-Planes combines any subset of line, plane, and volume features at different resolutions, with an MLP decoder. This GA-Planes family of parameterizations generalizes many existing volume and radiance field models.
- We derive convex and semiconvex reformulations of the GA-Planes training process for certain tasks and a large subset of the GA-Planes model family, to ensure our model optimizes globally regardless of initialization.
- We analyze GA-Planes in the 2D setting and show equivalence to a low-rank plus low-resolution matrix approximation whose expressiveness can be directly controlled by design choices. We demonstrate that this matrix decomposition is expressive for natural images, outperforming the classic low-rank plus sparse approximation.
- We demonstrate convex, semiconvex, and nonconvex GA-Planes' high performance in terms of memory, expressiveness, and optimizability across three volume-fitting tasks: 3D radiance field reconstruction, 3D segmentation, and video segmentation.

2. Related Work

Volume parameterization. Many volume parameterizations have been proposed and enjoy widespread use across diverse applications. Here we give an overview of representative methods used in computer vision, focusing on methods that parameterize an entire volume (rather than e.g. a surface). These parameterizations achieve different tradeoffs between memory usage, representation quality, and ease of optimization; richer descriptions are provided in Appendix A.1.

Coordinate MLPs like NeRF [13] and Scene Representation Networks [14] are representative of Implicit Neural Representations (INRs), which excel at reducing model size (with decent expressiveness) but suffer from slow optimization. At the opposite end of the spectrum, explicit voxel grid representations like Plenoxels [15] and Direct Voxel Grid Optimization [16] can optimize quickly but require large model size to achieve good expressiveness (resolution). Many other methods [17–22] find their niche somewhere in between, achieving tractable model size, good expressiveness, and reasonably fast optimization time in exchange for some increased sensitivity (to initialization, randomness, and prior knowledge) in the optimization process. GA-Planes matches or exceeds the performance of strong baselines [17, 18, 23] in terms of model size and expressiveness, while introducing the option to train by convex or semiconvex optimization with guaranteed convergence to global optimality.

Radiance field modeling. Most of the works described above are designed for the task of modeling a radiance field, in which the training measurements consist of color photographs from known camera poses. The goal is then to faithfully model the optical density and view-dependent color of light inside a volume so that unseen views can by rendered accurately. This task is also referred to as *novel view synthesis* [13, 14]. Although we do demonstrate superior performance of GA-Planes in this setting, we note that the volumetric rendering formula used in radiance field modeling [13, 24, 25] yields a nonconvex photometric loss function, regardless of model parameterization.

3D segmentation. We test our convex and semiconvex GA-Planes parameterizations on fully convex objectives, namely volume (xyz) segmentation with either indirect 2D tomographic supervision or direct supervision, as well as video (xyt) segmentation with direct 3D supervision. This 3D (xyz) segmentation task has also been studied in recent work [26, 27], though these methods require additional inputs such as a pretrained radiance field model or monocular depth estimator. Our setup is most similar to [28], which uses an implicit neural representation trained with cross-entropy loss and direct 3D supervision of the occupancy function. Instead of having direct access to this 3D training data, we infer 3D supervision labels via Space Carving [29] from 2D image masks obtained by image segmentation (via [30]).

Convex neural networks. Recent work has exposed an equivalence between training a shallow [31] or deep [32] neural network and solving a convex program whose structure is defined by the architecture and parameter dimensions of the corresponding neural network. The key idea behind this convexification procedure is to enumerate (or randomly sample from) the possible activation paths through the neural network, and then treat these paths as a fixed dictionary whose coefficients may be optimized according to a convex program. Given a data matrix $X \in \mathbb{R}^{n\times d}$ and labels $y \in \mathbb{R}^n$, a 2-layer nonconvex ReLU MLP approximates y as

$$y \approx \sum_{j=1}^{m} (XU_j)_+ \alpha_j, \tag{1}$$

where m is the number of hidden neurons and U and α are the first and second linear layer weights, respectively. [31] proposed to instead approximate y as

$$y \approx \sum_{i=1}^{P} D_i X (v_i - w_i), \tag{2}$$

subject to $(2D_i - I_n)Xv_i \geq 0$ and $(2D_i - I_n)Xw_i \geq 0$ for all i. The parameters v and w in eq. (2) replace the first and second layer weights U and α from the nonconvex formulation in eq. (1) (optimal values of U and α can be recovered from optimal values of v and w). Here D_i represent different possible activation patterns of the hidden neurons as $\{D_i\}_{i=1}^P := \{\text{Diag}(\mathbb{1}[Xu \geq 0]) : u \in \mathbb{R}^d\}$, which is the finite set of hyperplane arrangement patterns obtained for all possible $u \in \mathbb{R}^d$. We can sample different u's to find all distinct activation patterns $\{D_i\}_{i=1}^P$, where P is the number of regions in the partitioned input space. Enumerating all such patterns would yield an exact equivalence with the global minimizer of the nonconvex ReLU MLP in eq. (1), but may be complicated or intractable due to memory limitations. Subsampling $\tilde{P}$ patterns results in a convex program with tractable size, whose solution is one of the stationary points of the original non-convex problem [31]. We apply this idea to create convex and semiconvex GA-Planes models by convexifying the feature decoder MLP according to this procedure.

Geometric (Clifford) algebra. Geometric algebra (GA) is a powerful framework for modeling geometric primitives and interactions between them [33]. The fundamental entity in GA is the multivector, which is a sum of vectors, bivectors, trivectors, etc. In 3D GA, an example is the multivector $\mathbf{e}_1\mathbf{e}_2 + \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$, representing the sum of a bivector (a plane) and a trivector (a volume). The geometric product in GA allows us to derive a volume element by multiplying a plane and a line, e.g. $(\mathbf{e}_1\mathbf{e}_2)\mathbf{e}_3 = \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$. We use the shorthand $\mathbf{e}_{123} = \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$, and similarly for other multivector components. Inspired by this framework, we define the GA-Planes model family to include any volume parameterization that combines any subset (including the complete subset and the empty subset) of the linear geometric primitives $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, planar geometric primitives $\{\mathbf{e}_{12}, \mathbf{e}_{13}, \mathbf{e}_{23}\}$, and/or volumetric primitive $\{\mathbf{e}_{123}\}$ with a (potentially convexified) MLP feature decoder. We leverage geometric algebra to combine these primitives into a desired trivector (volume). To our knowledge, this work is the first to use geometric algebra in neural volume models.

3. Model

3.1. The GA-Planes Model Family

A GA-Planes model represents a volume using a combination of geometric algebra features $\mathbf{e}_c$ derived by interpolating the following parameter grids:

- Line (1-dimensional) feature grids $\{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$, where each grid has shape $[r_1, d_1]$ with spatial resolution r_1 and feature dimension d_1.
- Plane (2-dimensional) feature grids $\{\mathbf{g}_{12}, \mathbf{g}_{13}, \mathbf{g}_{23}\}$, where each grid has shape $[r_2, r_2, d_2]$ with spatial resolution r_2 and feature dimension d_2.

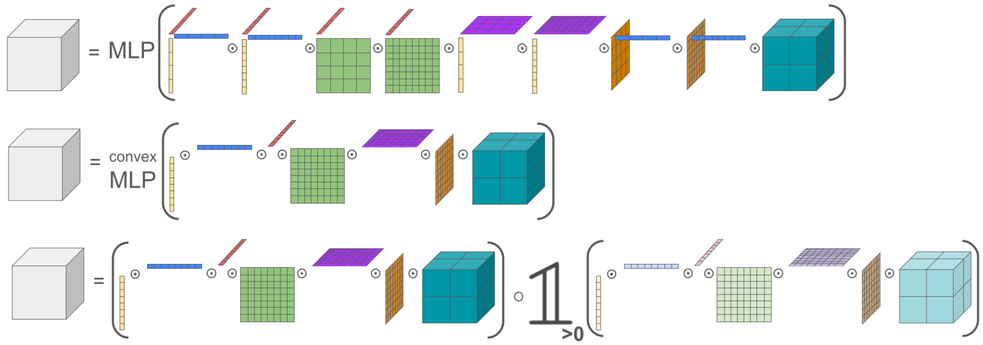


Figure 1: Overview of the GA-Planes models we use in our experiments. Our nonconvex model (top) uses a standard MLP decoder and multiplication of features when the result yields a volume under geometric algebra; it also concatenates features across mult-resolution grids. Our semiconvex (middle) and convex (bottom) models use a single resolution for each feature grid, and avoid multiplication of features since that would induce nonconvexity. The pastel-colored grids inside the indicator function of the convex model are frozen at initialization and used as fixed ReLU gating patterns. $\odot$ denotes concatenation and $\circ$ denotes elementwise multiplication.

- A single volume feature grid $\{\mathbf{g}_{123}\}$ with shape $[r_3, r_3, r_3, d_3]$.

A GA-Planes model may include multiple copies of a given basis element with different resolution and feature dimensions, to effectively capture multi-resolution signal content. The x, y, and z spatial resolutions of each grid may differ in practice; for simplicity of notation we use isotropic resolutions.

Let $q = (x, y, z)$ be a coordinate of interest in $\mathbb{R}^3$. We first extract features corresponding to q from each of our line, plane, and volume feature grids $\mathbf{g}_c$ by linear, bilinear, and trilinear interpolation, respectively:

$$\mathbf{e}_c := \psi\big(\mathbf{g}_c, \pi_c(q)\big), \tag{3}$$

where π_c projects q onto the coordinates of the c'th feature grid $\mathbf{g}_c$ and ψ denotes (bi/tri)linear interpolation of a point into a regularly spaced grid. The resulting feature $\mathbf{e}_c$ is a vector of length d_1 if $c \in \{1, 2, 3\}$, length d_2 if $c \in \{12, 13, 23\}$ or length d_3 if $c = 123$. We repeat this projection and interpolation procedure over each of our line, plane, and volume feature grids, and combine the resulting feature vectors by any combination of elementwise multiplication ($\circ$), addition ($+$), and concatenation ($\odot$) along the feature dimension. Note that multiplication and addition require $d_1 = d_2$. Finally, the spatially-localized combined feature vector is decoded using an MLP-based decoder D. The decoder can take as input both the feature vector arising from the feature grids as well as possible auxiliary inputs, such as (positionally encoded) viewing direction or 3D coordinates, depending on the task.

We consider any model that fits the above description to fall into the GA-Planes family. The specific models we use for nonconvex, semiconvex, and convex optimization are illustrated in Figure 1. Any GA-Planes model can be trained by nonconvex optimization; any GA-Planes model that avoids elementwise multiplication of features can be made semiconvex or fully convex as long as its training objective is also convex. For example, minimizing mean squared error for a linear inverse problem (like MRI, CT, additive denoising, super-resolution, inpainting, or segmentation) is a convex objective. However, the photometric loss used in radiance field modeling cannot be readily convexified because of the nonlinear accumulation of light along rays due to occlusion—so for our radiance field experiments we use a nonconvex GA-Planes model.

Our experiments focus primarily on two specific GA-Planes models that exemplify some of the strongest convex and nonconvex representations in the GA-Planes family. For our experiments including convex optimization, namely 3D segmentation with 2D or 3D supervision, and video segmentation, we use the following GA-Planes model (illustrated in the second and third rows of Figure 1) which can be trained by either convex, semi-convex, or nonconvex optimization as described in the following subsections:

$$\mathrm{D}(\mathbf{e}_1 \odot \mathbf{e}_2 \odot \mathbf{e}_3 \odot \mathbf{e}_{12} \odot \mathbf{e}_{13} \odot \mathbf{e}_{23} \odot \mathbf{e}_{123}). \tag{4}$$

Here we use $\odot$ to denote concatenation of features. For our radiance field experiments, we use the following nonconvex member of the GA-Planes family (illustrated with multiresolution feature grids in the first row of Figure 1):

$$\mathrm{D}((\mathbf{e}_1 \circ \mathbf{e}_2 \circ \mathbf{e}_3) \odot (\mathbf{e}_1 \circ \mathbf{e}_{23}) \odot (\mathbf{e}_2 \circ \mathbf{e}_{13}) \odot (\mathbf{e}_3 \circ \mathbf{e}_{12}) \odot \mathbf{e}_{123}), \tag{5}$$

which leverages geometric algebra to multiply ($\circ$) lower-dimensional (vector and bivector) features together into 3D volume (trivector) features, but cannot be convexified because of this multiplication. We use multiresolution copies of the line and plane feature grids $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_{12}, \mathbf{g}_{13}, \mathbf{g}_{23}$, but only a single resolution for the volume grid $\mathbf{g}_{123}$ since it is already at lower resolution. Note that the precise architecture of the decoder D may vary depending on the specific modeling task, such as whether the quantity of interest is view-dependent. We use the notation D to denote any feature decoder that uses a combination of linear layers and ReLU nonlinearities. For our nonconvex experiments the decoder is a standard fully-connected ReLU neural network; decoder details for our semiconvex and convex models are presented in the following subsections.

3.2. Semiconvex GA-Planes

For our segmentation experiments (with volumes and videos), we use the GA-Planes architecture in eq. (4), with concatenation instead of multiplication of features; we denote this concatenated feature vector as $f(q)$, the input to the decoder. Our semiconvex formulation of this model uses a convex MLP as the decoder:

$$\tilde{y}(q) = \sum_{i=1}^{h} ({W_i}^\top f(q)) \mathbb{1}[{\overline{W}_i}^\top f(q) \geq 0]. \tag{6}$$

Here W denotes the trainable hidden layer MLP weights, and $\overline{W}$ denotes the same weights frozen at initialization inside the indicator function. The indicator function, denoted as $\mathbb{1}(*)$, returns 1 if the argument is true, and 0 otherwise. Although this MLP decoder is fully convex, we refer to this model as semiconvex (in particular biconvex; see [12]) because the combined grid features $f(q)$ are multiplied by the trainable MLP hidden layer weights W, though the objective is separately convex in each of these parameters.

3.3. Convex GA-Planes

For our segmentation experiments (with volumes and videos), we also present a fully convex GA-Planes model that is similar to the semiconvex model described above, except that we fuse the learnable weights of the MLP decoder with the weights of the feature mapping, to remove the product of parameters (which is semiconvex but not convex). Our convex model is:

$$\tilde{y}(q) = \sum_{c \in \{1,2,3,12,13,23,123\}} \mathbb{1}_{d(c)}^\top (\mathbf{e}_c \circ \mathbb{1}\left[\bar{\mathbf{e}}_c \geq 0\right]), \tag{7}$$

where the features $\mathbf{e}_c$ are interpolated from optimizable parameter grids with feature dimension $d(c) \in \{d_1, d_2, d_3\}$, whereas the gating variables $\bar{\mathbf{e}}_c$ inside the indicator function are derived from the same grids frozen at their initialization values to preserve convexity. Here $\circ$ denotes elementwise (Hadamard) product of vectors. These indicator functions take the same role as the ReLU in a nonconvex MLP, using a sampling of random activation patterns based on the grid values at initialization.

4. Theory

4.1. Equivalence to Matrix Completion in 2D

In three dimensions, the complete set of geometric algebra feature grids are those that we include in the GA-Planes family: $\{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_{12}, \mathbf{g}_{13}, \mathbf{g}_{23}, \mathbf{g}_{123}\}$. In two dimensions, the complete set of geometric algebra feature grids is: $\{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_{12}\}$. In this two-dimensional setting, we can analyze different members of our GA-Planes family, and show equivalence to various formulations of the classic matrix completion problem.

Notation. As usual, we use $\circ$ to denote elementwise multiplication and $\odot$ to denote concatenation. We use $\mathbb{1}_{a \times b}$ to denote the all-ones matrix of size $a \times b$ and $\mathbb{1}[\cdot]$ to denote the indicator function, which evaluates to 1 when its argument is positive and 0 otherwise. Our theorem statements consider equivalence to a matrix completion problem in which $M \in \mathbb{R}^{m \times n}$ is the target matrix and $U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{n \times k}$ are the low-rank components to be learned. We include theoretical results for 2D GA-Planes models combine features by addition ($+$), multiplication ($\circ$), or concatenation ($\odot$) and decode features using a linear decoder (as a warmup), a convex MLP, or a nonconvex MLP. The most illuminating results are presented in the theorem statements that follow; the rest (and all proofs) are deferred to Appendix A.2.

Assumptions. Our theorem statements assume that the line feature grids have the same spatial resolution as the target matrix, and thus do not specify the type of interpolation. However, the results hold exactly even if the dimensions do not match, and nearest neighbor interpolation is used; the empirical performance is similar or even slightly improved in practice by using (bi)linear interpolation of features (see Appendix A.3 for a comparison and Appendix A.2 for a generalization to other interpolation methods). The theorems assume that the optimization objective is to minimize the Frobenius norm of the error relative to the target matrix; this is equivalent to minimizing mean squared error measured directly in the representation space. In particular, this objective function is the one we use for our convex experiments (video and volume segmentation fitting), where we have access to direct supervision; our radiance field experiments instead use indirect measurements (along rays) that are not exactly equivalent to the setting of the theorems.

Theorem 1. *The two-dimensional representation $D(\mathbf{e}_1 + \mathbf{e}_2)$ with linear decoder $D(f(q)) = \alpha^T f(q)$ is equivalent to a low-rank matrix completion model with the following structure:*

$$\min_{U,V} \|M - (U\mathbb{1}_{k \times n} + \mathbb{1}_{m \times k}V^T)\|_F^2. \tag{8}$$

These two models are equivalent in the sense that $U^* = \mathbf{g}_1^* diag(\alpha^*)$ and $V^* = \mathbf{g}_2^* diag(\alpha^*)$ where U^*, V^* is the optimal solution to the low-rank matrix completion problem in eq. (8) *and $\mathbf{g}_1^*, \mathbf{g}_2^*, \alpha^*$ are the optimal grid features and linear decoder for the $D(\mathbf{e}_1 + \mathbf{e}_2)$ model.*

The two-dimensional representation $D(\mathbf{e}_1 \circ \mathbf{e}_2)$ with the same linear decoder is equivalent to the standard low-rank matrix completion model:

$$\min_{U,V} \|M - UV^T\|_F^2. \tag{9}$$

These two models are equivalent in the same sense as above, except that $V^* = \mathbf{g}_2^*$.

Remark. Using a linear decoder reveals a dramatic difference in representation capacity between feature addition (or concatenation) and multiplication. Using addition, the maximum rank of the matrix approximation is 2 regardless of the feature dimension k. Using multiplication, the maximum rank of the approximation is k. With feature multiplication, the optimal values of the feature grids are identical to the rank-thresholded singular value decomposition (SVD) of M, where the feature grids $\mathbf{g}_1$ and $\mathbf{g}_2$ recover the left and right singular vectors and the decoder α learns the singular values of M. This is the optimal rank k approximation of a matrix M.

Theorem 2. *The two-dimensional representation $D(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_{12})$ with linear decoder $D(f(q)) = \alpha^T f(q)$ is equivalent to a low-rank plus low-resolution matrix completion model with the following structure:*

$$\min_{U,V,L} \|M - (U\mathbb{1}_{k \times n} + \mathbb{1}_{m \times k}V^T + \varphi(L))\|_F^2, \tag{10}$$

where $L \in \mathbb{R}^{m_l \times n_l}$ is the low-resolution component to be learned, with upsampling (interpolation) function φ. These two models are equivalent in the sense that $U^* = \mathbf{g}_1^* diag(\alpha^*)$, $V^* = \mathbf{g}_2^* diag(\alpha^*)$, and $L^* = \mathbf{g}_{12}^* \alpha^*$, where U^*, V^*, L^* is the optimal solution to the low-rank plus low-resolution matrix completion problem in eq. (10) *and $\mathbf{g}_1^*, \mathbf{g}_2^*, \mathbf{g}_{12}^*, \alpha^*$ are the optimal grid features and linear decoder for the $D(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_{12})$ model.*

The two-dimensional representation $D(\mathbf{e}_1 \circ \mathbf{e}_2 + \mathbf{e}_{12})$ with the same linear decoder is equivalent to a low-rank plus low-resolution matrix completion model:

$$\min_{U,V,L} \|M - (UV^T + \varphi(L))\|_F^2. \tag{11}$$

These two models are equivalent in the same sense as above, except that $V^* = \mathbf{g}_2^*$.

Remark. Theorem 2 describes the behavior of a 2D, linear-decoder version of our GA-Planes model, both the version with addition/concatenation of features (eq. (4)) and the version with multiplication of features (eq. (5)). Extending the same idea to 3D, we can interpret GA-Planes as a low-rank plus low-resolution approximation of a 3D tensor (volume). We can understand this model as first fitting a low-resolution volume and then finding a low-rank approximation to the high-frequency residual volume. When we use multiplication of features, the low-rank residual approximation is optimal and analogous to the rank-thresholded SVD.

In Theorem 3 and Theorem 4 we extend the analysis of this two-dimensional GA-Planes model with line features to the version with a convex and nonconvex MLP decoder, respectively. We again derive analogies to matrix completion, but find that the introduction of an MLP decoder, whether convex or nonconvex, fundamentally alters the rank constraints faced by the model.

Theorem 3. *The two-dimensional representation $D(\mathbf{e}_1 \circ \mathbf{e}_2)$ with a two-layer convex MLP decoder $D(f(q)) = \sum_{i=1}^{h} (W_i{}^\top f(q))\mathbb{1}[\overline{W}_i{}^\top f(q) \geq 0]$ is equivalent to a masked low-rank matrix completion model:*

$$\min_{U,V,W} \left\| M - \sum_{i,j} W_{i,j} U_j V_j^\top \circ B_i \right\|_F^2, \tag{12}$$

where $W \in \mathbb{R}^{h \times k}$ contains the trainable weights of the convex MLP decoder, with indices $j = 1, \ldots, k$ for the input dimension and $i = 1, \ldots, h$ for the hidden layer dimension. $B_i \in \mathbb{R}^{m \times n}$ denotes the binary masking matrix formed by random, fixed gates of the convex MLP decoder; $B_i = \mathbb{1}[\sum_j \overline{W}_{i,j} U_j V_j^\top \geq 0]$, where $\overline{W}$ denotes the weight matrix W with values fixed at random initialization.

This matrix completion model and our GA-Planes model $D(\mathbf{e}_1 \circ \mathbf{e}_2)$ with convex MLP decoder are equivalent in the sense that $U^* = \mathbf{g}_1^*$, $V^* = \mathbf{g}_2^*$, and $W^* = W^*$, where U^*, V^*, W^* is the optimal solution to the masked low-rank matrix completion problem eq. (12) and $\mathbf{g}_1^*, \mathbf{g}_2^*, W^*$ are the optimal grid features and convex MLP decoder weights for the $D(\mathbf{e}_1 \circ \mathbf{e}_2)$ model. The optimal mask matrices B_i^* are defined by the fixed random weight initialization $\overline{W}$ and the optimal singular vector matrices U^*, V^*.

Remark. We can interpret the matrix completion model of Equation (12) as a sum of h different low-rank approximations, where the matrices within each of the h groups are constrained to share the same singular vectors U_j, V_j. The binary masks B_i effectively allow each of these h low-rank approximations to attend to (or complete) a different part of the matrix M before being linearly combined through the weights (singular values) $W_{i,j}$. The upper limit of the rank of this matrix approximation is thus $\min(n, m)$, because the mask matrices can arbitrarily increase the rank beyond the constraint faced by models with a linear decoder. Note that if the feature grids $\mathbf{g}_1$ and $\mathbf{g}_2$ have spatial resolution r_1 less than $\min(n, m)$, the maximum rank will be r_1.

Theorem 4. *The two-dimensional representation $D(\mathbf{e}_1 \circ \mathbf{e}_2)$ with a standard two-layer MLP decoder $D(f(q)) = \alpha^T (W f(q))_+$ is equivalent to a low-rank matrix completion model with the following structure:*

$$\min_{U,V,W,\alpha} \left\| M - \sum_{i=1}^{h} \alpha_i \Big(\sum_{j=1}^{k} W_{i,j} U_j V_j^\top \Big)_+ \right\|_F^2, \tag{13}$$

where $W \in \mathbb{R}^{h \times k}$ is the weight matrix for the MLP decoder's hidden layer (with width h) and $\alpha \in \mathbb{R}^h$ is the weight vector of the MLP decoder's output layer.

This matrix completion model and our GA-Planes model $D(\mathbf{e}_1 \circ \mathbf{e}_2)$ with nonconvex MLP decoder are equivalent in the sense that $U^* = \mathbf{g}_1^*$, $V^* = \mathbf{g}_2^*$, $W^* = W^*$, and $\alpha^* = \alpha^*$, where U^*, V^*, W^*, α^* is the optimal solution to the masked low-rank matrix completion problem eq. (13) and $\mathbf{g}_1^*, \mathbf{g}_2^*, W^*, \alpha^*$ are the optimal grid features and MLP decoder for the $D(\mathbf{e}_1 \circ \mathbf{e}_2)$ model.

Remark. The upper limit of the rank of this matrix approximation is $\min(n, m)$, the same as with a convex MLP decoder. Again, note that if the feature grids $\mathbf{g}_1$ and $\mathbf{g}_2$ have spatial resolution r_1 less than $\min(n, m)$, the maximum rank will be r_1.

We summarize the maximum attainable ranks of different 2D models in Table 1 (see Appendix A.2.3 and Appendix A.2.4 for matrix representations of $\mathrm{D}(\mathbf{e}_1 + \mathbf{e}_2)$ and $\mathrm{D}(\mathbf{e}_1 \odot \mathbf{e}_2)$ with convex and nonconvex MLP

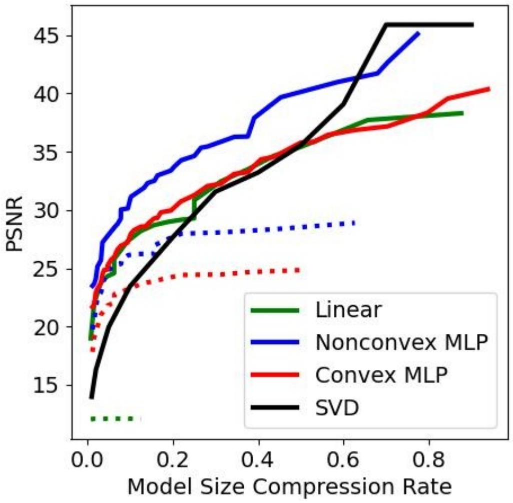


Figure 2: 2D image fitting experiments with the *astronaut* image from SciPy, validating matrix completion analysis summarized in Table 1. We compare 2D GA-Planes models of the form $\mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2)$ (solid colorful lines) and $\mathrm{D}(\mathbf{e}_1 + \mathbf{e}_2)$ (dotted colorful lines) with the optimal low-rank approximation provided by singular value decomposition (solid black line).

decoders). Experimental validation of these theoretical results on the task of 2D image fitting (compression) is provided in Figure 2, with a comparison of interpolation schemes in Figure 8 in the appendix. As expected, we find that a linear decoder model with multiplication dramatically outperforms its additive counterpart, which does not improve with increasing model size. We also find that 2D GA-Planes models with MLP decoders can match or exceed the compression performance of the optimal low-rank representation found by singular value decomposition (SVD), especially when using a nonconvex MLP. This is a testament to the capacity of an MLP decoder to increase representation rank using fewer parameters than a traditional low-rank decomposition, as well as to the resolution compressibility of natural images. These experimental results also provide complementary information the one-sided bounds on representation rank and fitting error in our theoretical analysis.

Model	Linear decoder	convex MLP decoder	MLP decoder
$\mathrm{D}(\mathbf{e}_1 + \mathbf{e}_2)$	2	r_1	r_1
$\mathrm{D}(\mathbf{e}_1 \odot \mathbf{e}_2)$	2	r_1	r_1
$\mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2)$	k	r_1	r_1

Table 1: Maximum attainable ranks of different 2D GA-Planes models, using only line features. Here k is the feature dimension and r_1 is the spatial dimension of the features, which need never exceed $\min(m, n)$. Replacing a linear decoder with a convex or nonconvex MLP can dramatically increase the rank of the representation.

4.2. Lower Bounds

Based on the matrix completion theorems and their summary in Table 1, we present lower bounds on the Frobenius norm errors of each 2D GA-Planes model. We denote the optimal fitting error of the linear and MLP decoder models by $E_{linear}(\mathrm{D}(f(q)))$ and $E_{MLP}(\mathrm{D}(f(q)))$ for different feature combinations $f(q)$. For models with a linear decoder,

$$E_{linear}(\mathrm{D}(\mathbf{e}_1 + \mathbf{e}_2)) \geq \sigma_2(M) \tag{14}$$

$$E_{linear}(\mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2)) \geq \sigma_k(M) \tag{15}$$

$$E_{linear}(\mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2 + \mathbf{e}_{12})) \geq \sigma_k(M - \varphi(L^*)), \tag{16}$$

where L^* is a downsampled version of the target M, at the same resolution as the feature grid $\mathbf{g}_{12}$.

For models with convex or nonconvex MLP decoders,

$$E_{MLP}(\mathrm{D}(\mathbf{e}_1 + \mathbf{e}_2)) \geq \sigma_{r_1}(M) \tag{17}$$

$$E_{MLP}(\mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2)) \geq \sigma_{r_1}(M) \tag{18}$$

$$E_{MLP}(\mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2 + \mathbf{e}_{12})) \geq \sigma_{r_1}(M - \varphi(L^*)). \tag{19}$$

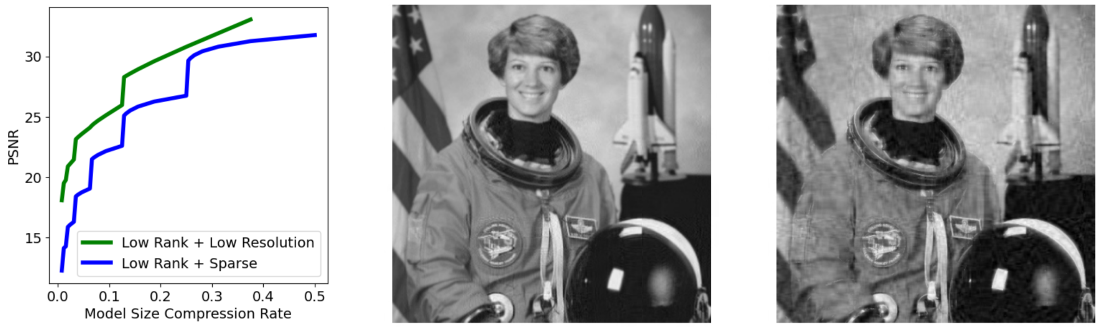
We can see from these bounds that the approximation error of a model can be reduced dramatically by the introduction of a convex or nonconvex MLP decoder, depending on the singular value decay of the target image M.

4.3. Interpretation: Low Rank + Low Resolution

Combining multiple parameterization strategies with complementary representation capacities is a time-honored strategy in signal processing. A classic example is the combination of sparse and low-rank models used to represent matrices in the compressive sensing literature [34]. This decomposition tends to work well because the residual error of a low-rank matrix approximation is often itself well-approximated by a sparse set of entries.

As shown in Theorem 2, we can view the GA-Planes family as following a similar strategy with a combination of low-rank and low-resolution approximations. In this framing, the line and plane features combine to form a low-rank but high-resolution approximation, while the volume grid is a full-rank but low-resolution approximation. This parameterization is generally easier to train because sparse models must either store large numbers of empty values (high memory) or store the locations of nonzero entries and suffer from poorly-conditioned spatial gradients (difficult optimization). Indeed there are volume parameterizations that utilize sparsity, such as point clouds, surface meshes, surfels, and Gaussian splats, but these tend to be more challenging to optimize (e.g. requiring high memory [15] or heuristic updates and good initialization [20]).

We illustrate this low-rank plus low-resolution interpretation in Figure 3 with a simple experiment, in which we approximate a grayscale image (the *astronaut* image from SciPy) using either a sum of low-rank and low-resolution components (similar to the structure of GA-Planes) or the classic sum of low-rank and sparse components. In this experiment we compute the optimal low-rank components of each model type using the SVD, which corresponds to multiplication of features.



(a) Pareto Frontiers for Image Compression

(b) Low Rank + Low Res PSNR 29.60

(c) Low Rank + Sparse PSNR 26.26

Figure 3: For a natural image, approximation as a sum of low rank and low resolution components (green points and subfigure b) achieves higher fidelity compared to the classic matrix decomposition as a sum of low rank and sparse components (blue points and subfigure c), with the same parameter budget (18.75% of the original image size, for subfigures b and c). The GA-Planes model family generalizes the idea of a low rank plus low resolution approximation to three dimensions.

5. Experiments

5.1. Radiance Field Modeling

Our experiments for the radiance field reconstruction task are built on the NeRFStudio framework [35] and use all scenes from NeRF-Blender [13]. For this task, we train each volume representation based on the photometric loss that is standard in the NeRF literature [13, 24, 25]. This loss is the mean squared error at the pixel color level, but is inherently nonconvex because the forward model for volume rendering is nonlinear.

Our results are summarized in Figure 4, which reports PSNR, SSIM [36], and LPIPS [37] for each model as a function of its size. We provide example renderings in Figure 5 and Figure 6; per-scene pareto-optimal curves and additional renderings are in Appendix A.5 and Appendix A.6.

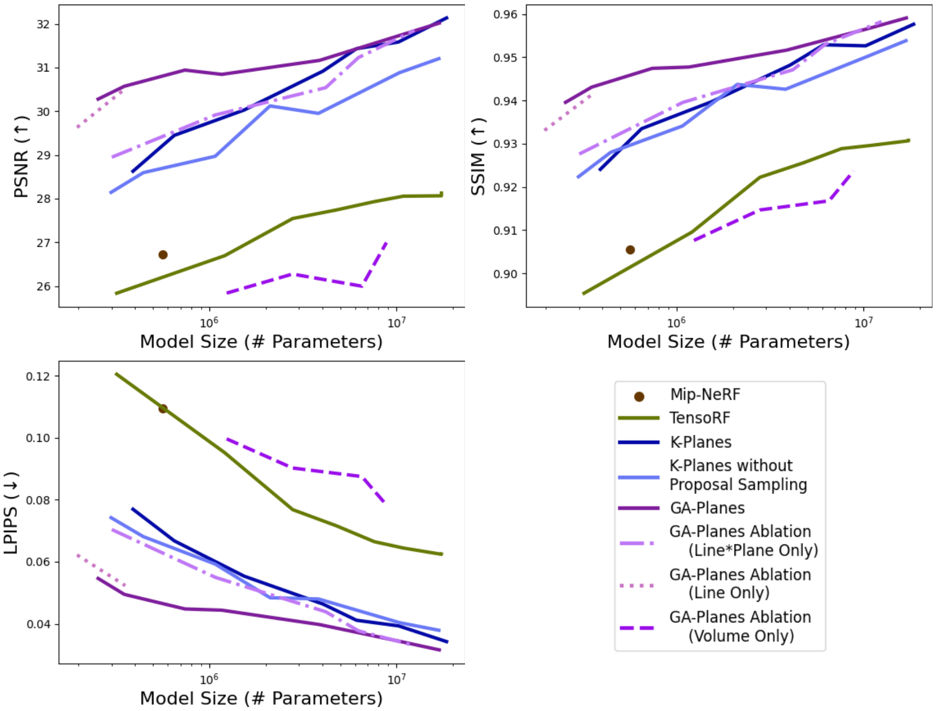


Figure 4: Results on radiance field reconstruction. Nonconvex GA-Planes (with feature multiplication) offers the most efficient representation: when the model is large it performs comparably to the state of the art models, but when model size is reduced it retains higher performance than other models. Here all models are trained for the same number of epochs on all 8 scenes from the Blender dataset, and the average results are shown.

Because the loss function is inherently nonconvex for this task, we focus only on nonconvex models and use our most expressive GA-Planes parameterization as defined in eq. (5). We compare this GA-Planes model with several popular models as implemented in NeRFStudio: Mip-NeRF [23], TensoRF [17], and K-Planes [18]. For K-Planes, we include versions with and without proposal sampling, a strategy for efficiently allocating ray samples during training and rendering. Proposal sampling is the default for K-Planes, but we include a version without proposal sampling because (1) all other models in this experiment do not use proposal sampling, and (2) we find that proposal sampling boosts the PSNR for K-Planes but hurts its SSIM and LPIPS. We also include several ablations of our GA-Planes model: one with only the volume features (similar to a multiresolution version DVGO [16] or Plenoxels [15]), one with only the line features (similar to a multiresolution TensoRF-CP), and one with only the line-plane products (similar to a multiresolution TensoRF-VM). At large model sizes (∼10 million parameters) most models including GA-Planes perform comparably well. However, as model size shrinks we find that only GA-Planes and its line-only ablation reach comparable metrics as the larger models.

Indeed, we find that because GA-Planes contains features with different dimensionalities (line, plane, and volume), it experiences little loss in quality over a wide range of model sizes. For small model sizes, we allocate most of the model memory to the line features, since their spatial resolution grows linearly with parameter count. As the parameter budget grows, we allocate more parameters to the plane features, whereas the performance of the line-only model stagnates with increasing size. Similarly, as model size grows even further we allocate more parameters to the volume features, whose memory footprint grows cubically with spatial resolution.

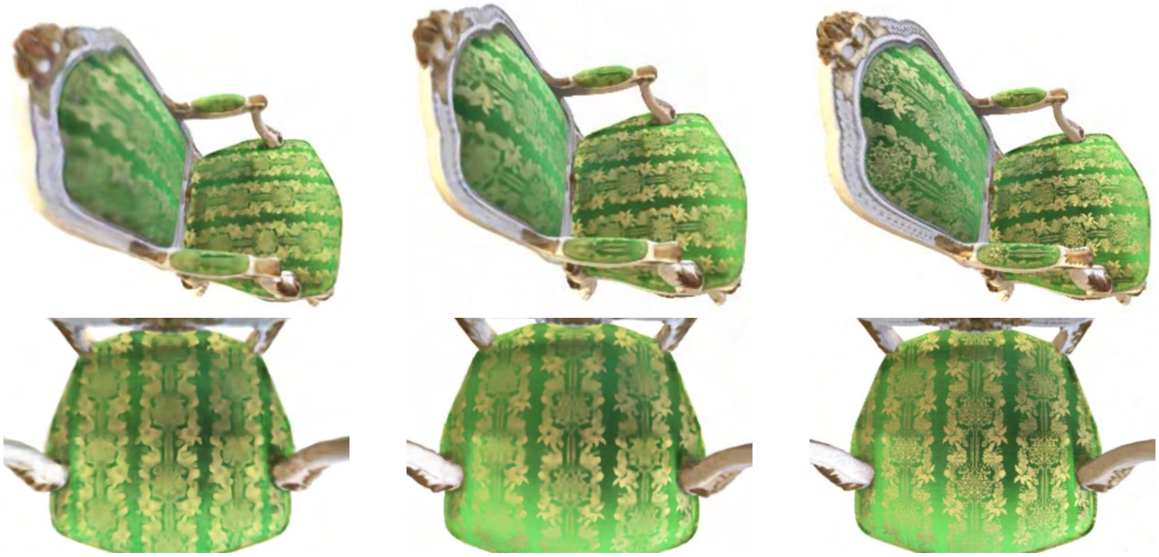


Figure 5: Rendering comparison for the *chair* scene: TensoRF on the left (0.32 M parameters), K-Planes in the middle (0.39 M parameters), GA-Planes on the right (0.25 M parameters).

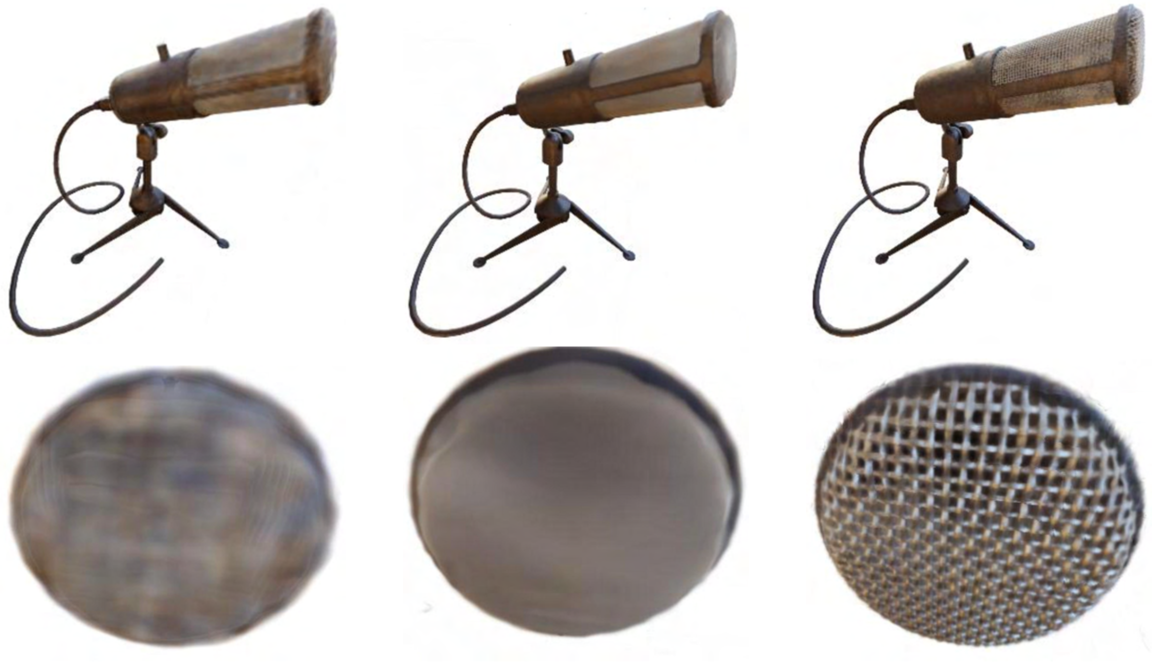


Figure 6: Rendering comparison for the *mic* scene: TensoRF on the left (0.32 M parameters), K-Planes in the middle (0.39 M parameters), GA-Planes on the right (0.25 M parameters).

5.2. 3D Segmentation

Our experiments for the 3D segmentation task use the opacity masks from the NeRF-Blender *lego* scene [13]. The task is to “lift” 2D segmentation masks to 3D, rather than generating 3D segmentations directly from raw photographs. We compare the GA-Planes architecture with concatenation of features (described by eq. (4)) and the simpler Tri-Plane representation proposed in [38], in which the three plane features are added together and decoded without use of line or volume features ($\mathrm{D}(\mathbf{e}_{12} + \mathbf{e}_{13} + \mathbf{e}_{23})$). We train three versions of each model: convex, semiconvex, and nonconvex, to validate that the GA-Planes architecture in eq. (4)

is more robust than the Tri-Plane model across these different formulations. We also compare GA-Planes with multiplication of features, in eq. (5), to the Tri-Plane model using multiplication of plane features, like K-Planes ($\mathrm{D}(\mathbf{e}_{12} \circ \mathbf{e}_{13} \circ \mathbf{e}_{23})$). These only have nonconvex formulations.

2D Supervision. One set of 3D segmentation experiments relies on 2D supervision, in which we minimize the mean squared error between the ground truth object segmentation masks and the average ray density at each viewpoint. This form of 2D supervision is essentially tomography in the presence of occlusion. After training, we render the projections of our reconstructed volume model at each of the test angles, and threshold it to produce a binary object mask. We then compute the intersection over union (IOU) metric to compare our predicted segmentation masks with the ground truth masks. Results are reported in Table 2.

	Convex	Semiconvex	Nonconvex
GA-Planes (with $\circ$)	-	-	0.877
GA-Planes (with $\odot$)	0.875	0.883	0.880
Tri-Planes (planes with $\circ$, like K-Planes)	-	-	0.877
Tri-Planes (planes with $+$, like [38])	0.681	0.868	0.863

Table 2: Intersection over union (IOU) for recovering novel view object segmentation masks from segmentation mask training with 2D tomographic supervision.

We find that the GA-Planes architecture with concatenation outperforms the Tri-Planes architecture using addition when trained with equal total number of parameters (see Appendix A.7 for specific feature resolution and dimensions), regardless of whether training is convex, semiconvex, or nonconvex. Further, we see that GA-Planes retains similar performance regardless of optimization strategy, whereas the Tri-Plane model learns poorly via convex optimization. When the features are combined through multiplication, the models achieve the same IOU score.

3D Supervision. Our second set of 3D segmentation experiments leverages direct 3D supervision via Space Carving [29]. Space Carving supervision operates on the principle that if any ray passing through a given 3D coordinate is transparent, the density at that 3D coordinate must be zero. This method recovers the visual hull of the object by "carving out" empty space around it. Our results with 3D supervision, in Table 3, parallel those with 2D supervision: GA-Planes performs well regardless of whether it is optimized in a convex, semiconvex, or nonconvex formulation, whereas the Tri-Plane model performs decently under nonconvex optimization but much worse with convex or semiconvex formulation.

	Convex	Semiconvex	Nonconvex
GA-Planes (with $\circ$)	-	-	0.926
GA-Planes (with $\odot$)	0.932	0.957	0.964
Tri-Planes (planes with $\circ$, like K-Planes)	-	-	0.881
Tri-Planes (planes with $+$, like [38])	0.642	0.636	0.941

Table 3: Intersection over union (IOU) for recovering novel view object segmentation masks from segmentation mask training with 3D Space Carving supervision.

5.3. Video Segmentation

Our experiments for the video segmentation task use a variety of nonconvex, semiconvex, and convex models from the GA-Planes family. We treat the video segmentation task as similar to the volume segmentation task with 3D supervision described above: now the volume dimensions are x, y, t rather than x, y, z, and the supervision is performed directly in 3D using segmentation masks for a subset of the video frames (every third frame is held out for testing). Note that although we refer to this task as video segmentation, the models are essentially tasked with temporal superresolution of object masks in a video rather than predicting masks from a raw video of a moving object. Our dataset preparation pipeline uses the skateboarding video and preprocessing steps described at [39], which involves first extracting a bounding box with YOLOv8 [40] and then segmenting the skateboarder with SAM [30].

Our results are summarized in Figure 7 and Table 4. Similar to the volume segmentation setting, here we again find that GA-Planes performs well across convex, semiconvex, and nonconvex training, though its performance is slightly reduced under fully convex training, perhaps because the convex model size is slightly reduced due to fusing the decoder parameters into the feature grids. In contrast, the simpler Tri-Plane models perform poorly on this task regardless of training strategy: they fail to learn the temporal sequence of the video, producing masks that focus only on the skateboarder's less-mobile core. In this experiment, we also compare nonconvex models of each type (Tri-Plane and full GA-Planes) in which the features are combined in a linear way (by concatenation or addition) versus a nonlinear way (by multiplication). The similar performance between addition/concatenation and multiplication of features observed here is in line with results reported in a similar ablation study in K-Planes [18], in which the benefits of feature multiplication were only evident when using a linear decoder rather than a nonlinear MLP decoder as is used here.

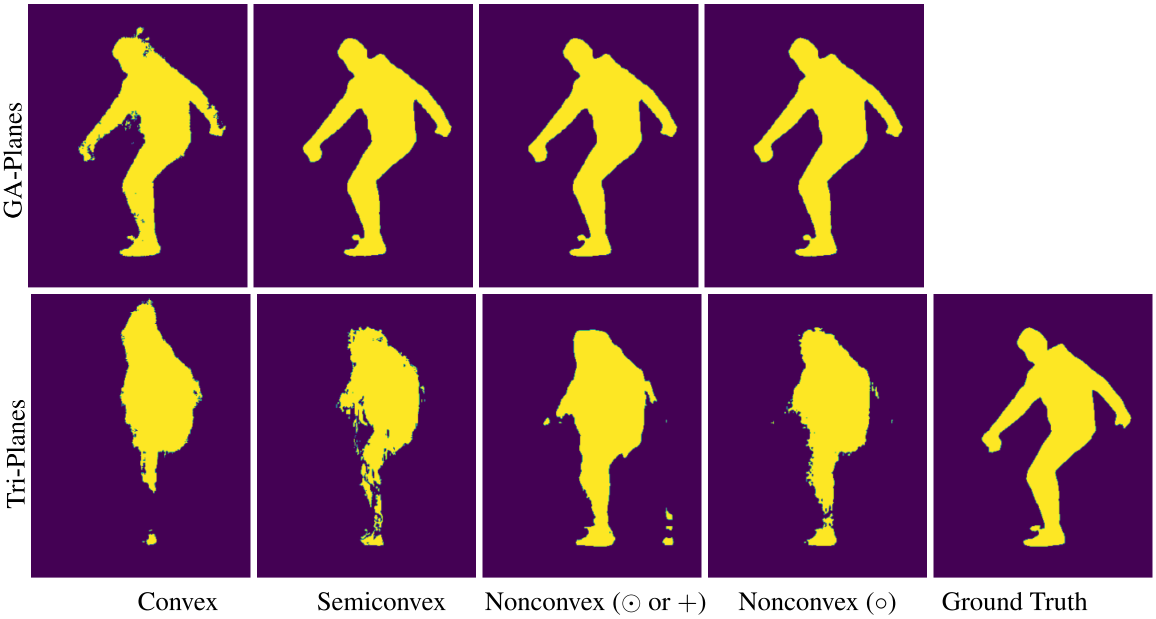


Figure 7: Intersection over union (IOU) for predicting segmentation masks for unseen frames within a video of a segmented skateboarder.

	Convex	Semiconvex	Nonconvex
GA-Planes (with $\circ$)	-	-	0.974
GA-Planes (with $\odot$)	0.913	0.975	0.981
Tri-Planes (planes with $\circ$, like K-Planes)	-	-	0.727
Tri-Planes (planes with $+$, like [38])	0.557	0.647	0.732

Table 4: Intersection over union (IOU) for temporal superresolution of segmentation masks in a video, computed on held-out test frames. Models that involve multiplication of features can only be trained by nonconvex optimization.

6. Discussion

In this work we introduce GA-Planes, a family of volume parameterizations that generalizes many existing representations (see Appendix A.1). We specifically focus on two members of the GA-Planes family (with concatenation versus multiplication of features), and offer both theoretical interpretation and empirical evaluation of these models.

Theoretically, we show that these models form a low-rank plus low-resolution tensor factorization. In two dimensions, the GA-Planes model with feature multiplication yields the optimal low-rank plus low-resolution matrix approximation, whereas the model with feature concatenation is likewise low-rank plus low-resolution but may not be optimally parameter-efficient. In other words, 2D GA-Planes with feature multiplication is equivalent to first taking a low-resolution matrix approximation (by filtering and downsampling) and then finding the optimal low-rank approximation to the residual (by thresholding the singular values in the SVD).

For the GA-Planes model with feature concatenation (or addition, but not multiplication), we derive semiconvex and fully convex formulations based on convexifying the MLP decoder and fusing weights as needed. We empirically demonstrate convex, semiconvex, and nonconvex GA-Planes models are effective on three tasks: radiance field reconstruction, 3D object segmentation (with 2D or 3D supervision), and video segmentation (temporal superresolution of object masks) (see Appendix A.7 for additional details concerning model configurations for each task). We find that GA-Planes matches or exceeds the performance of strong baselines across a range of model sizes, and that GA-Planes without multiplication performs similarly well regardless of whether its formulation (with decoder) is nonconvex, semiconvex, or fully convex.

Limitations. In this work we focus on 3D (or smaller) representations, rather than higher dimensions (e.g. dynamic volumes), and we demonstrate GA-Planes for reconstruction tasks rather than also generation tasks. Both of these extensions are promising avenues for extending GA-Planes. In our experiments, we use the same first-order optimization algorithm for all models. However, our convex GA-Planes formulation is designed to be compatible with any convex solver (e.g. cvxpy), and we expect its performance may improve by leveraging these efficient convex optimization algorithms. We demonstrate preliminary benefits of convexity and semiconvexity in terms of training stability in Appendix A.4.

References

[1] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, volume 41, pages 703–735. Wiley Online Library, 2022.

[2] Aditya M Intwala and Atul Magikar. A review on process of 3d model reconstruction. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 2851–2855. IEEE, 2016.

[3] Eugen Šlapak, Enric Pardo, Matúš Dopiriak, Taras Maksymyuk, and Juraj Gazda. Neural radiance fields in the industrial and robotics domain: applications, research opportunities and use cases. *Robotics and Computer-Integrated Manufacturing*, 90:102810, 2024.

[4] Yuhang Ming, Xingrui Yang, Weihan Wang, Zheng Chen, Jinglun Feng, Yifan Xing, and Guofeng Zhang. Benchmarking neural radiance fields for autonomous robots: An overview. *arXiv preprint arXiv:2405.05526*, 2024.

[5] Liyana Wijayathunga, Alexander Rassau, and Douglas Chai. Challenges and solutions for autonomous ground robot scene understanding and navigation in unstructured outdoor environments: A review. *Applied Sciences*, 13(17):9877, 2023.

[6] Jian Liu, Xiaoshui Huang, Tianyu Huang, Lu Chen, Yuenan Hou, Shixiang Tang, Ziwei Liu, Wanli Ouyang, Wangmeng Zuo, Junjun Jiang, et al. A comprehensive survey on 3d content generation. *arXiv preprint arXiv:2402.01166*, 2024.

[7] V Croce, G Caroti, L De Luca, A Piemonte, and P Véron. Neural radiance fields (nerf): Review and potential applications to digital cultural heritage. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48:453–460, 2023.

[8] Jayaram K Udupa and Gabor T Herman. *3D imaging in medicine*. CRC press, 1999.

[9] Valentín Masero, JUAN M LEÓN-ROJAS, and José Moreno. Volume reconstruction for health care: a survey of computational methods. *Annals of the New York Academy of Sciences*, 980(1):198–211, 2002.

[10] Alexander Richter, Till Steinmann, Jean-Claude Rosenthal, and Stefan J Rupitsch. Advances in real-time 3d reconstruction for medical endoscopy. *Journal of Imaging*, 10(5):120, 2024.

[11] Mengya Xu, Ziqi Guo, An Wang, Long Bai, and Hongliang Ren. A review of 3d reconstruction techniques for deformable tissues in robotic surgery. *arXiv preprint arXiv:2408.04426*, 2024.

[12] Arda Sahiner, Tolga Ergen, Batu Ozturkler, John M Pauly, Morteza Mardani, and Mert Pilanci. Scaling convex neural networks with burer-monteiro factorization. In *ICLR*, 2024.

[13] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[14] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019.

[15] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

[16] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022.

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

[18] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023.

[19] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. doi: 10.1145/3528223.3530127. URL `https://doi.org/10.1145/3528223.3530127`.

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/`.

[21] Christian Reiser, Richard Szeliski, Dor Verbin, Pratul P. Srinivasan, Ben Mildenhall, Andreas Geiger, Jonathan T. Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *SIGGRAPH*, 2023.

[22] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459863. URL `https://doi.org/10.1145/3450626.3459863`.

[23] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5835–5844. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00580. URL `https://doi.org/10.1109/ICCV48922.2021.00580`.

[24] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[25] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[26] Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Chen Yang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, and Qi Tian. Segment anything in 3d with nerfs. In *NeurIPS*, 2023.

[27] Mikaela Angelina Uy, Ricardo Martin-Brualla, Leonidas Guibas, and Ke Li. Scade: Nerfs from space carving with ambiguity-aware depth estimates. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[28] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

[29] K.N. Kutulakos and S.M. Seitz. A theory of shape by space carving. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 307–314 vol.1, 1999. doi: 10.1109/ICCV.1999.791235.

[30] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. URL `https://arxiv.org/abs/2304.02643`.

[31] Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7695–7705. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/pilanci20a.html`.

[32] Tolga Ergen and Mert Pilanci. Path regularization: A convexity and sparsity inducing regularization for parallel relu networks. *Advances in Neural Information Processing Systems*, 36, 2024.

[33] Leo Dorst, Daniel Fontijne, and Stephen Mann. *Geometric algebra for computer science (revised edition): An object-oriented approach to geometry*. Morgan Kaufmann, 2009.

[34] Venkat Chandrasekaran, Sujay Sanghavi, Pablo A Parrilo, and Alan S Willsky. Sparse and low-rank matrix decompositions. *IFAC Proceedings Volumes*, 42(10):1493–1498, 2009.

[35] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.

[36] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[37] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[38] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *CVPR*, 2022.

[39] Labelbox.com. Using meta's segment anything (sam) model on video with labelbox's model-assisted labeling. `https://labelbox.com/guides/using-metas-segment-anything-sam-model-on-video-with-labelbox-model-assisted-labeling/`. Accessed: 2024-10-01.

[40] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, January 2023. URL `https://github.com/ultralytics/ultralytics`.

[41] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[42] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

[43] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. Wire: Wavelet implicit neural representations. In *Conf. Computer Vision and Pattern Recognition*, 2023.

[44] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[45] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[46] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[47] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

A. Appendix

A.1. Context for GA-Planes

Table 5 summarizes some representative volume models popular in computer vision, and how they relate to GA-Planes along the three-way pareto frontier of model size, expressiveness, and optimizability.

Implicit Neural Representations (INRs) or Coordinate Neural Networks [13, 14, 41–43] parameterize the volume implicitly through the weights of a neural network, typically a multilayer perceptron (MLP) with some modification to overcome spectral bias and represent high frequency content. These models tend to provide decent expressiveness with very small model size; their main drawback is slow optimization.

Voxel grids [15, 16, 29, 44] are perhaps the most traditional parameterization of a volume, where each parameter denotes the function value (density, color, a latent feature, etc.) at a specific grid cell location within the volume. These voxel values can then be combined into a continuous function over the 3D space by some form of interpolation, following the standard Nyquist sampling and reconstruction paradigm of digital signal processing [45]. Voxels offer direct control over expressivity (via resolution) and are easily optimized; their main drawback is memory usage because the number of parameters grows cubically with the spatial resolution.

	Model Size	Expressiveness	Optimizability
Coordinate MLP (NeRF, SRN)	✓	∼	✗
Voxels (Space Carving, Plenoxels, DVGO)	✗	✓	✓
Tensor Factorization (TensoRF, K-Planes)	∼	∼	∼
Hash Embedding (Instant-NGP)	∼	✓	∼
Point Cloud / Splat (3D Gaussian Splatting)	∼	✓	∼
Mixture of Primitives (MVP, MERF)	∼	✓	∼
GA-Planes (Nonconvex)	∼	✓	∼
GA-Planes (Convex)	∼	✓	✓
GA-Planes (Semi-Convex)	∼	✓	✓

Table 5: Context. All volume models face a tradeoff between memory efficiency, expressiveness, and optimizability. The qualitative categorizations here are based on the tradeoffs achieved by representative example methods listed in each category. *Model Size* denotes memory usage during training; other methods exist to compress trained models, e.g. for rendering on mobile hardware. *Optimizability* denotes both speed and stability of optimization/training. For example, Coordinate MLPs tend to train slowly, while Splats train quickly but are sensitive to initialization.

Tensor factorizations [17, 18, 38] parameterize a 3D volume as a combination of lower-dimension objects, namely vectors and matrices (lines and planes). Tensor factorizations tend to balance the three attributes somewhat evenly, offering decent expressiveness and optimizability while using more memory than an INR but less than a high resolution voxel grid.

Hash embeddings [19, 35] are similar to voxels, but replace the explicit voxel grid in 3D with a multiresolution 3D hash function followed by a small MLP decoder to disambiguate hash collisions. They can optimize very quickly and with better memory efficiency compared to voxels; quality is mixed with good high-resolution details but also some high-frequency noise likely arising from unresolved hash collisions or sensitivity to random initialization.

Point clouds / splats [20, 46, 47] represent a volume as a collection of 3D points or blobs, where the points need not be arranged on a regular grid. They are highly expressive and less memory-intensive than voxels (but still more so than some other methods). They can optimize very quickly but often require heuristic or discrete optimization strategies that result in sensitivity to initialization.

Mixture of primitives [21, 22] models combine multiple of the above representation strategies to balance their strengths and weaknesses. For example, combining low resolution voxels with a high resolution tensor factorization is an effective strategy to improve on the expressiveness of tensor factorizations without resorting to the cubic memory requirement of a high resolution voxel grid; this strategy underlies both MERF [21] and GA-Planes.

We emphasize that all of these existing methods (except perhaps voxels) require nonconvex optimization, often for a feature decoder MLP, and thus risk getting stuck in suboptimal local minima depending on the randomness of initialization and the trajectory of stochastic gradients. In practice, as described above, some of the prior methods exhibit greater optimization stability than others, though none (except voxels in limited settings) come with guarantees of convergence to global optimality. In contrast, both the convex and semiconvex GA-Planes formulations come with guarantees that all local optima are also global [12].

Relation to Prior Models. Without any convexity restrictions, the GA-Planes family includes many previously proposed models as special cases:

- NeRF [13]: D
- Plenoxels [15], DVGO [16]: $\text{D}(\mathbf{e}_{123})$
- TensoRF [17]: $\text{D}((\mathbf{e}_1 \circ \mathbf{e}_{23}) \odot (\mathbf{e}_2 \circ \mathbf{e}_{13}) \odot (\mathbf{e}_3 \circ \mathbf{e}_{12}))$
- Tri-Planes [38]: $\text{D}(\mathbf{e}_{12} + \mathbf{e}_{13} + \mathbf{e}_{23})$
- K-Planes [18]: $\text{D}(\mathbf{e}_{12} \circ \mathbf{e}_{13} \circ \mathbf{e}_{23})$
- MERF [21]: $\text{D}(\mathbf{e}_{12} + \mathbf{e}_{13} + \mathbf{e}_{23} + \mathbf{e}_{123})$

Of these, all except for TensoRF and K-Planes are compatible with convex optimization towards any convex objective. Note that different models may use different decoder architectures for D, including both linear and MLP decoders and additional decoder inputs such as encoded viewing direction and/or positionally-encoded coordinates.

A.2. Proof of Theorems

A general note on proofs. In order to represent a matrix $M \in \mathbb{R}^{m \times n}$ with an implicit model, we compute $\text{D}(f(q))$ for $q = (k, l),\ \forall k \in \{1, \dots, m\},\ \forall l \in \{1, \dots, n\}$. Considering line feature grids with resolutions matching m, n; the features will become $\mathbf{e}_1 = (\mathbf{g}_1)_k$, $\mathbf{e}_2 = (\mathbf{g}_2)_l$ for $q = (k, l)$ otherwise they will be $\mathbf{e}_1 = \varphi(\mathbf{g}_1)_k$, $\mathbf{e}_2 = \varphi(\mathbf{g}_2)_l$ where $\varphi(\mathbf{g}_1)$, $\varphi(\mathbf{g}_2)$ now have resolutions m, n after interpolation through φ. Here φ can be any interpolation scheme with linear weighting of inputs, e.g. nearest neighbor, (bi)linear, (bi)cubic, spline, Gaussian, sinc, etc. For the simplicity of notation, we omit φ in line feature grids, and only apply it to the plane feature grid $\mathbf{g}_{12}$, which has lower resolution by design. The proofs consider $\mathbf{g}_1, \mathbf{g}_2 \in \mathbb{R}^{r_1 \times d_1}$ and $\mathbf{g}_{12} \in \mathbb{R}^{r_2 \times r_2 \times d_1}$ (equal feature dimensions, different resolutions), resulting in the matrix representation $\hat{M} \in \mathbb{R}^{r_1 \times r_1}$ (the case where $m = n = r_1$). Note that if the interpolation is done by a method other than nearest neighbor, this may allow a (convex or nonconvex) MLP decoder to increase the rank beyond r_1. We derive expressions for $\hat{M}$ implied by different GA-Planes variations in the parts that follow. The coordinate-wise optimization objective (in the case of a directly supervised mean-square-error loss) corresponds to minimizing the Frobenius norm of the ground truth matrix M and its approximation $\hat{M}$.

A.2.1. Proof of theorem 1

The forward mapping of the model $\text{D}(\mathbf{e}_1 + \mathbf{e}_2)$ is:

$$\tilde{y}(q) = \text{D}(\mathbf{e}_1 + \mathbf{e}_2) = \alpha^\top(\mathbf{e}_1 + \mathbf{e}_2) = \alpha^\top((\mathbf{g}_1)_k + (\mathbf{g}_2)_l) = \sum_{i=1}^{d_1} \alpha_i((\mathbf{g}_1)_{k,i} + (\mathbf{g}_2)_{l,i}), \tag{20}$$

where $(\mathbf{g}_1)_k, (\mathbf{g}_1)_l \in \mathbb{R}^{d_1 \times 1}$.

In matrix form,

$$\hat{M} = \sum_{i=1}^{d_1} \alpha_i((\mathbf{g}_1)_i \mathbb{1}^\top + \mathbb{1}(\mathbf{g}_2)_i^\top) = \sum_{i=1}^{d_1} \alpha_i(\mathbf{g}_1)_i \mathbb{1}^\top + \sum_{i=1}^{d_1} \alpha_i \mathbb{1}(\mathbf{g}_2)_i^\top. \tag{21}$$

Defining $U := \mathbf{g}_1 \text{diag}(\alpha), U \in \mathbb{R}^{r_1 \times d_1}$ and $V := \mathbf{g}_2 \text{diag}(\alpha), V \in \mathbb{R}^{r_1 \times d_1}$, this can be expressed as

$$\hat{M} = U \mathbb{1}_{r_1 \times d_1}^\top + \mathbb{1}_{r_1 \times d_1} V^\top. \tag{22}$$

Note that the resulting matrix $\hat{M} \in \mathbb{R}^{r_1 \times r_1}$ has rank at most 2—very limited expressivity—regardless of the resolution r_1. This is because the all-ones matrix is rank 1, and a product of matrices cannot have higher rank than either of its factors.

Similarly for the multiplicative representation $\mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2)$, the mapping is

$$\tilde{y}(q) = \mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2) = \alpha^\top(\mathbf{e}_1 \circ \mathbf{e}_2) = \alpha^\top((\mathbf{g}_1)_k \circ (\mathbf{g}_2)_l) = \sum_{i=1}^{d_1} \alpha_i (\mathbf{g}_1)_{k,i} (\mathbf{g}_2)_{l,i}, \tag{23}$$

where $(\mathbf{g}_1)_k, (\mathbf{g}_1)_l \in \mathbb{R}^{d_1 \times 1}$. In matrix form,

$$\hat{M} = \sum_{i=1}^{d_1} \alpha_i (\mathbf{g}_1)_i (\mathbf{g}_2)_i^\top = \mathbf{g}_1 \mathrm{diag}(\alpha) \mathbf{g}_2^\top. \tag{24}$$

Defining $U := \mathbf{g}_1 \mathrm{diag}(\alpha), U \in \mathbb{R}^{r_1 \times d_1}$ and $V := \mathbf{g}_2, V \in \mathbb{R}^{r_1 \times d_1}$, this can be expressed as

$$\hat{M} = UV^\top, \tag{25}$$

which is the optimal rank-d_1 decomposition.

A.2.2. Proof of theorem 2

The forward mapping of the model $\mathrm{D}(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_{12})$ becomes:

$$\tilde{y}(q) = \mathrm{D}(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_{12}) = \alpha^\top(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_{12}) = \alpha^\top((\mathbf{g}_1)_k + (\mathbf{g}_2)_l + \varphi(\mathbf{g}_{12})_{k,l}) \tag{26}$$

$$= \sum_{i=1}^{d_1} \alpha_i ((\mathbf{g}_1)_{k,i} + (\mathbf{g}_2)_{l,i}) + \sum_{i=1}^{d_1} \alpha_i \varphi(\mathbf{g}_{12})_{k,l,i}, \tag{27}$$

where $(\mathbf{g}_1)_k, (\mathbf{g}_1)_l, \varphi(\mathbf{g}_{12})_{k,l} \in \mathbb{R}^{d_1 \times 1}$. In matrix form,

$$\hat{M} = \sum_{i=1}^{d_1} \alpha_i ((\mathbf{g}_1)_i \mathbb{1}^\top + \mathbb{1}(\mathbf{g}_2)_i^\top) + \sum_{i=1}^{d_1} \alpha_i \varphi(\mathbf{g}_{12})_i. \tag{28}$$

Noting that the first term is the same as in eq. (21) and defining $L := \mathbf{g}_{12}\alpha, L \in \mathbb{R}^{r_2 \times r_2}$, we reach the expression

$$\hat{M} = U\mathbb{1}_{d_1 \times r_1} + \mathbb{1}_{r_1 \times d_1} V^\top + \varphi(L), \tag{29}$$

since $\sum_{i=1}^{d_1} \alpha_i \varphi(\mathbf{g}_{12})_i = \varphi(\sum_{i=1}^{d_1} \alpha_i (\mathbf{g}_{12})_i) = \varphi(\mathbf{g}_{12}\alpha)$, following the linearity of the upsampling function φ. Note that in the definition of L there is a tensor-vector product that effectively takes a dot product along the last (feature) dimension.

Similarly for the multiplicative representation $\mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2 + \mathbf{e}_{12})$, the mapping is

$$\tilde{y}(q) = \mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2 + \mathbf{e}_{12}) = \alpha^\top(\mathbf{e}_1 \circ \mathbf{e}_2 + \mathbf{e}_{12}) \tag{30}$$

$$= \alpha^\top((\mathbf{g}_1)_k \circ (\mathbf{g}_2)_l + \varphi(\mathbf{g}_{12})_{k,l}) = \sum_{i=1}^{d_1} \alpha_i (\mathbf{g}_1)_{k,i} (\mathbf{g}_2)_{l,i} + \sum_{i=1}^{d_1} \alpha_i \varphi(\mathbf{g}_{12})_{k,l,i}. \tag{31}$$

In matrix notation, we have

$$\hat{M} = \sum_{i=1}^{d_1} \alpha_i ((\mathbf{g}_1)_i (\mathbf{g}_2)_i^\top) + \sum_{i=1}^{d_1} \alpha_i \varphi(\mathbf{g}_{12})_i. \tag{32}$$

Following eq. (24) and using the same definition of L, the final expression becomes

$$\hat{M} = UV^\top + \varphi(L). \tag{33}$$

A.2.3. Proof of theorem 3

For a 2-layer convex MLP with hidden size h, denote the trainable first layer weights as $W \in \mathbb{R}^{h \times d_1}$ and the gating weights as $\overline{W} \in \mathbb{R}^{h \times d_1}$ (which are fixed at random initialization). We will handle three different cases for merging the interpolated features: multiplication ($\circ$), addition ($+$), and concatenation ($\odot$).

The forward mapping of the multiplicative model using a convex MLP, D$(\mathbf{e}_1 \circ \mathbf{e}_2)$ at $q = (k, l)$ is

$$\begin{aligned} \tilde{y}(q) &= \mathbb{1}_h^\top \left((W((\mathbf{g}_1)_k \circ (\mathbf{g}_2)_l)) \circ \mathbb{1}\left[\overline{W}((\mathbf{g}_1)_k \circ (\mathbf{g}_2)_l) \geq 0\right]\right) && (34) \\ &= \sum_{i=1}^{h} \left(\sum_{j=1}^{d_1} W_{i,j}(\mathbf{g}_1)_{k,j}(\mathbf{g}_2)_{l,j}\right) \mathbb{1}\left[\sum_{j=1}^{d_1} \overline{W}_{i,j}(\mathbf{g}_1)_{k,j}(\mathbf{g}_2)_{l,j} \geq 0\right], && (35) \end{aligned}$$

where $\circ$ denotes elementwise multiplication (Hadamard product). The resulting matrix decomposition can then be written as

$$\hat{M} = \sum_{i=1}^{h} \left(\sum_{j=1}^{d_1} W_{i,j}(\mathbf{g}_1)_j(\mathbf{g}_2)_j^\top\right) \circ \mathbb{1}\left[\sum_{j=1}^{d_1} \overline{W}_{i,j}(\mathbf{g}_1)_j(\mathbf{g}_2)_j^\top \geq 0\right]. \tag{36}$$

Now, we define the masking matrix $B_i = \mathbb{1}\left[\sum_{j=1}^{d_1} \overline{W}_{i,j}(\mathbf{g}_1)_j(\mathbf{g}_2)_j^\top \geq 0\right]$ and the eigenvectors $U_j = (\mathbf{g}_1)_j$, $V_j = (\mathbf{g}_2)_j$ to reach the expression from the theorem statement:

$$\hat{M} = \sum_{i,j} W_{i,j} U_j V_j^\top \circ B_i. \tag{37}$$

When the model uses additive features as in D$(\mathbf{e}_1 + \mathbf{e}_2)$, and D is a convex MLP, the prediction is

$$\begin{aligned} \tilde{y}(q) &= \mathbb{1}_h^\top \left((W((\mathbf{g}_1)_k + (\mathbf{g}_2)_l)) \circ \mathbb{1}\left[\overline{W}((\mathbf{g}_1)_k + (\mathbf{g}_2)_l) \geq 0\right]\right) && (38) \\ &= \sum_{i=1}^{h} \left(\sum_{j=1}^{d_1} W_{i,j}((\mathbf{g}_1)_{k,j} + (\mathbf{g}_2)_{l,j})\right) \mathbb{1}\left[\sum_{j=1}^{d_1} \overline{W}_{i,j}((\mathbf{g}_1)_{k,j} + (\mathbf{g}_2)_{l,j}) \geq 0\right]. && (39) \end{aligned}$$

The resulting matrix decomposition can then be written as

$$\hat{M} = \sum_{i=1}^{h} \left(\sum_{j=1}^{d_1} W_{i,j}((\mathbf{g}_1)_j \mathbb{1}_{r_1}^\top + \mathbb{1}_{r_1}(\mathbf{g}_2)_j^\top)\right) \mathbb{1}\left[\sum_{j=1}^{d_1} \overline{W}_{i,j}((\mathbf{g}_1)_j \mathbb{1}_{r_1}^\top + \mathbb{1}_{r_1}(\mathbf{g}_2)_j^\top) \geq 0\right]. \tag{40}$$

Defining $B_i = \mathbb{1}\left[\sum_{j=1}^{d_1} \overline{W}_{i,j}((\mathbf{g}_1)_j \mathbb{1}_{r_1}^\top + \mathbb{1}_{r_1}(\mathbf{g}_2)_j^\top) \geq 0\right]$, $U_j = (\mathbf{g}_1)_j$, $V_j = (\mathbf{g}_2)_j$, we reach the final expression:

$$\hat{M} = \sum_{i,j} W_{i,j}\left(U_j \mathbb{1}_{r_1}^\top + \mathbb{1}_{r_1} V_j^\top\right) \circ B_i. \tag{41}$$

Finally, we show that concatenation of features results in a very similar expression to eq. (41).

When the model uses concatenated features as in D$(\mathbf{e}_1 \odot \mathbf{e}_2)$, and D is a convex MLP (with trainable weights $W \in \mathbb{R}^{h \times 2d_1}$ and fixed gates $\overline{W} \in \mathbb{R}^{h \times 2d_1}$), the prediction at a point q is

$$\tilde{y}(q) = \mathbb{1}_h^\top \left((W((\mathbf{g}_1)_k \odot (\mathbf{g}_2)_l)) \circ \mathbb{1}\left[\overline{W}((\mathbf{g}_1)_k \odot (\mathbf{g}_2)_l) \geq 0\right]\right). \tag{42}$$

Denoting the weights and gates each as a concatenation of 2 matrices, $W = (W_1 \odot W_2)$, $\overline{W} = (\overline{W}_1 \odot \overline{W}_2)$, where $W_1, W_2, \overline{W}_1, \overline{W}_2 \in \mathbb{R}^{h \times d_1}$, we have the following expression:

$$\tilde{y}(q) = \sum_{i=1}^{h} \left(\sum_{j=1}^{d_1} W_{1i,j}(\mathbf{g}_1)_{k,j} + W_{2i,j}(\mathbf{g}_2)_{l,j}\right) \mathbb{1}\left[\sum_{j=1}^{d_1} \overline{W}_{1i,j}(\mathbf{g}_1)_{k,j} + \overline{W}_{2i,j}(\mathbf{g}_2)_{l,j} \geq 0\right]. \tag{43}$$

Following similar steps as for the additive case, we express the matrix decomposition as

$$\hat{M} = \sum_{i,j} (W_{1i,j} U_j \mathbb{1}_{r_1}^\top + W_{2i,j} \mathbb{1}_{r_1} V_j^\top) \circ B_i, \tag{44}$$

where $B_i = \mathbb{1}\left[\sum_{j=1}^{d_1} \overline{W}_{1i,j}(\mathbf{g}_1)_j \mathbb{1}_{r_1}^\top + \overline{W}_{2i,j} \mathbb{1}_{r_1}(\mathbf{g}_2)_j^\top \geq 0\right]$, $U_j = (\mathbf{g}_1)_j$, $V_j = (\mathbf{g}_2)_j$.

In all these representations, a low-rank matrix is multiplied elementwise with a binary mask B_i, which makes the maximum attainable rank r_1. Thus, with a convex MLP decoder, rank of $\hat{M}$ is limited by the resolution of the feature grids.

A.2.4. Proof of theorem 4

For a standard 2-layer ReLU MLP with hidden size h, denote the trainable first and second layer weights as $W \in \mathbb{R}^{h \times d_1}$, $\alpha \in \mathbb{R}^{h \times 1}$. We will handle three different cases for merging the interpolated features: multiplication ($\circ$), addition ($+$), and concatenation ($\odot$).

The forward mapping of the multiplicative model using a standard nonconvex MLP, $\mathrm{D}(\mathbf{e}_1 \circ \mathbf{e}_2)$ is:

$$\tilde{y}(q) = \alpha^\top \left[W((\mathbf{g}_1)_k \circ (\mathbf{g}_2)_l)\right]_+ \tag{45}$$

$$= \sum_{i=1}^{h} \alpha_i \left[\sum_{j=1}^{d_1} W_{i,j} (\mathbf{g}_1)_{k,j} (\mathbf{g}_2)_{l,j}\right]_+ . \tag{46}$$

The resulting matrix decomposition can then be written as

$$\hat{M} = \sum_{i=1}^{h} \alpha_i \left(\sum_{j=1}^{d_1} W_{i,j} (\mathbf{g}_1)_j (\mathbf{g}_2)_j^\top\right)_+ = \sum_{i=1}^{h} \alpha_i \left(\sum_{j=1}^{d_1} W_{i,j} U_j V_j^\top\right)_+ , \tag{47}$$

with $U_j = (\mathbf{g}_1)_j$, $V_j = (\mathbf{g}_2)_j$.

When the model uses additive features as in $\mathrm{D}(\mathbf{e}_1 + \mathbf{e}_2)$, the prediction is

$$\tilde{y}(q) = \alpha^\top \left[W((\mathbf{g}_1)_k + (\mathbf{g}_2)_l)\right]_+ \tag{48}$$

$$= \sum_{i=1}^{h} \alpha_i \left[\sum_{j=1}^{d_1} W_{i,j} ((\mathbf{g}_1)_{k,j} + (\mathbf{g}_2)_{l,j})\right]_+ . \tag{49}$$

The resulting matrix decomposition can then be written as

$$\hat{M} = \sum_{i=1}^{h} \alpha_i \left(\sum_{j=1}^{d_1} W_{i,j} ((\mathbf{g}_1)_j \mathbb{1}_{r_1}^\top + \mathbb{1}_{r_1} (\mathbf{g}_2)_j^\top)\right)_+ = \sum_{i=1}^{h} \alpha_i \left(\sum_{j=1}^{d_1} W_{i,j} (U_j \mathbb{1}_{r_1}^\top + \mathbb{1}_{r_1} V_j^\top)\right)_+ , \tag{50}$$

again with $U_j = (\mathbf{g}_1)_j$, $V_j = (\mathbf{g}_2)_j$.

Finally, we show that concatenation of features results in a very similar expression.

When the model uses concatenated features as in $\mathrm{D}(\mathbf{e}_1 \odot \mathbf{e}_2)$ and D is a standard nonconvex MLP (with trainable weights $W \in \mathbb{R}^{h \times 2d_1}$ and $\alpha \in \mathbb{R}^{h \times 1}$), the prediction at a point q is

$$\tilde{y}(q) = \alpha^\top \left[(W((\mathbf{g}_1)_k \odot (\mathbf{g}_2)_l))_+\right] . \tag{51}$$

Denoting the hidden layer weights as a concatenation of 2 matrices, $W = (W_1 \odot W_2)$, where $W_1, W_2 \in \mathbb{R}^{h \times d_1}$, we have the following expression:

$$\tilde{y}(q) = \sum_{i=1}^{h} \alpha_i \left(\sum_{j=1}^{d_1} W_{1i,j} (\mathbf{g}_1)_{k,j} + W_{2i,j} (\mathbf{g}_2)_{l,j}\right)_+ . \tag{52}$$

Following similar steps, we express the matrix decomposition as

$$\hat{M} = \sum_{i=1}^{h} \alpha_i \left(\sum_{j=1}^{d_1} W_{1i,j} (\mathbf{g}_1)_j \mathbb{1}_{r_1}^\top + W_{2i,j} \mathbb{1}_{r_1} (\mathbf{g}_2)_j^\top\right)_+ \tag{53}$$

$$= \sum_{i=1}^{h} \alpha_i \left(\sum_{j=1}^{d_1} W_{1i,j} U_j \mathbb{1}_{r_1}^\top + W_{2i,j} \mathbb{1}_{r_1} V_j^\top\right)_+ , \tag{54}$$

where $U_j = (\mathbf{g}_1)_j$, $V_j = (\mathbf{g}_2)_j$.

By a similar argument to Appendix A.2.3, the maximum attainable rank of all three representations derived here is limited by r_1.

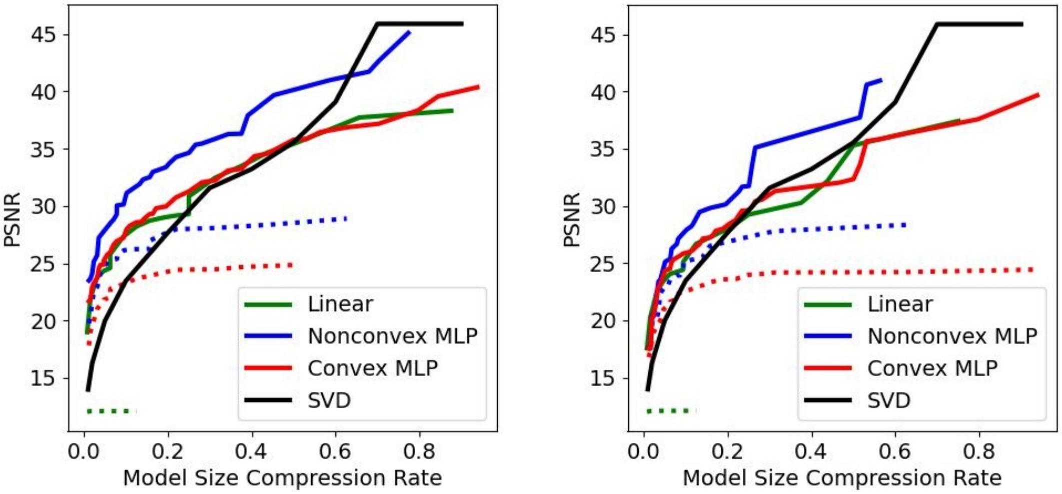


Figure 8: 2D image fitting experiments matching the setting of our theoretical results, with a GA-Planes version using only vector (line) features and a decoder as specified in the legend. Left: linear interpolation of features; Right: nearest neighbor interpolation of features. For the SVD baseline we use low-rank factors whose resolution matches the target image, so no interpolation is needed (this, and the use of a nonlinear decoder, is why in some cases 2D GA-Planes can outperform SVD).

A.3. Interpolation Comparison

Figure 2 shows experiments fitting the SciPy *astronaut* image using the various models considered in our 2D theoretical results. In Figure 8 we compare the same experiment when we use linear interpolation into the vector (line) features (left, same as Figure 2) versus nearest neighbor interpolation (right, same as theorems). In this experiment we find qualitatively similar results regardless of the type of interpolation, with slightly better performance using linear interpolation; in our 3D experiments we use (bi/tri)linear interpolation.

A.4. Benefits of Convexity

In most of our experiments, all models (convex, semiconvex, and nonconvex) are large enough that they are able to optimize well. However, we highlight a benefit of our convex and semiconvex models that they enjoy more stable optimization even with very small model sizes. In Figure 9 we compare test intersection-over-union (IoU) curves for very small models for our video fitting task (hidden dimension 4 in the decoder MLP, and feature dimensions $[d_1, d_2, d_3] = [4, 4, 2]$ and resolutions $[r_1, r_2, r_3] = [32, 32, 16]$ for line, plane, and volume features, respectively). We repeat optimization with 3 different random seeds used to initialize the optimizable parameters (gating weights for the convex and semiconvex models are fixed). While the convex and semiconvex models enjoy stable training curves across random seeds, we find that the nonconvex model experiences much more volatile training behavior (completely failing to optimize with one of the 3 random seeds).

A.5. Results for all Nerfstudio-Blender Scenes

A.6. Example renderings

In this section, we provide qualitative rendering comparisons on various scenes from the Blender dataset. We highlight the superior performance of GA-Planes with limited number of parameters by comparing the smallest K-Planes, TensoRF and GA-Planes models.

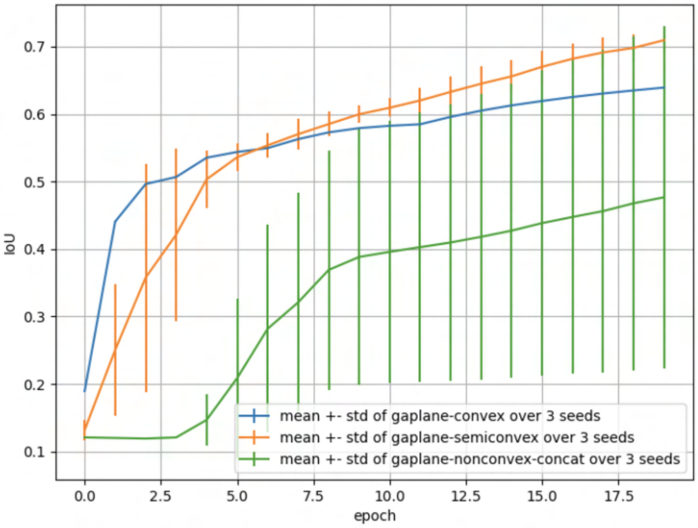


Figure 9: Test performance throughout training on our video fitting task, for a very small GA-Planes model across 3 random seeds. We find that the favorable optimization landscape of our convex and semiconvex models enables reliable training across seeds, whereas the nonconvex model fails to fit any test frame with one of the seeds.

A.7. Model configurations used for experiments

A.7.1. Radiance Field Modeling

The original K-planes model uses 2 proposal networks with different resolutions (as noted in Table 6) and a fixed channel dimension of 8 for both. The resolutions and channel dimensions for either K-planes model (with vs. without proposal sampling) refer to r_2 and d_2, respectively. TensoRF model resolutions and channel dimensions can be interpreted in a similar way, since their feature combination dictates that $d_1 = d_2$ and they initialize the line and plane grids with the same resolution. The only nuance is that TensoRF constructs separate features for color and density decoding. Hence, the channel dimensions for density and color features are listed. Instead of a multiresolution scheme, TensoRF starts from the base resolution of $r_1 = r_2 = 128$ and upsamples the grids to reach the final resolutions on Table 6. Resolutions listed under GA-Planes should be interpreted as $[r_1, r_2, r_3]$; channel dimensions as $[d_1, d_2, d_3]$. For all models that use the multiresolution scheme, the base resolutions (i.e. $[r_1, r_2, r_3]$) are multiplied with the upsampling factors. For instance, a base resolution $[r_1, r_2, r_3]$ with channel dimensions $[d_1, d_2, d_3]$ and multiresolution copies $[m_1, m_2, m_3]$ will generate the grids of GA-Planes as follows: Linear feature grids $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3$ will have the shapes $\{[m_1 r_1, d_1], [m_2 r_1, d_1], [m_3 r_1, d_1]\}$, plane grids $\mathbf{g}_{12}, \mathbf{g}_{23}, \mathbf{g}_{13}$ will have the shapes $\{[m_1 r_2, m_1 r_2, d_2], [m_2 r_2, m_2 r_2, d_2], [m_3 r_2, m_3 r_2, d_2]\}$, and the volume grid $\mathbf{g}_{123}$ will have the shape $[r_3, r_3, r_3, d_3]$. Although we don't use multiresolution copies for the volume grid in GA-Planes, we do use multiresolution for the volume-only GA-Planes ablation. The resolution for that model refers to r_3, and the channel dimensions are also allowed to vary for each resolution (unlike other variants with multiresolution, where the feature dimension is fixed across resolutions). If we denote these varying feature dimensions as $[d_{3a}, d_{3b}, d_{3c}]$, the multiresolution copies of the volume grids will have the shapes $\{[m_1 r_3, m_1 r_3, m_1 r_3, d_{3a}], [m_2 r_3, m_2 r_3, m_2 r_3, d_{3b}], [m_3 r_3, m_3 r_3, m_3 r_3, d_{3c}]\}$.

A.7.2. 3D Segmentation

GA-Planes model uses feature dimensions $[d_1, d_2, d_3] = [36, 24, 8]$ (with $\odot$) or $[d_1, d_2, d_3] = [25, 25, 8]$ (with $\circ$) and resolutions $[r_1, r_2, r_3] = [128, 32, 24]$. Multiresolution grids are not used for this task since density prediction can be achieved by a simpler architecture. The model size is 0.22 M. Tri-Planes model has the feature dimension $d_2 = 4$, and resolution $r_2 = 128$ resulting in a total number of parameters of 0.2 M.

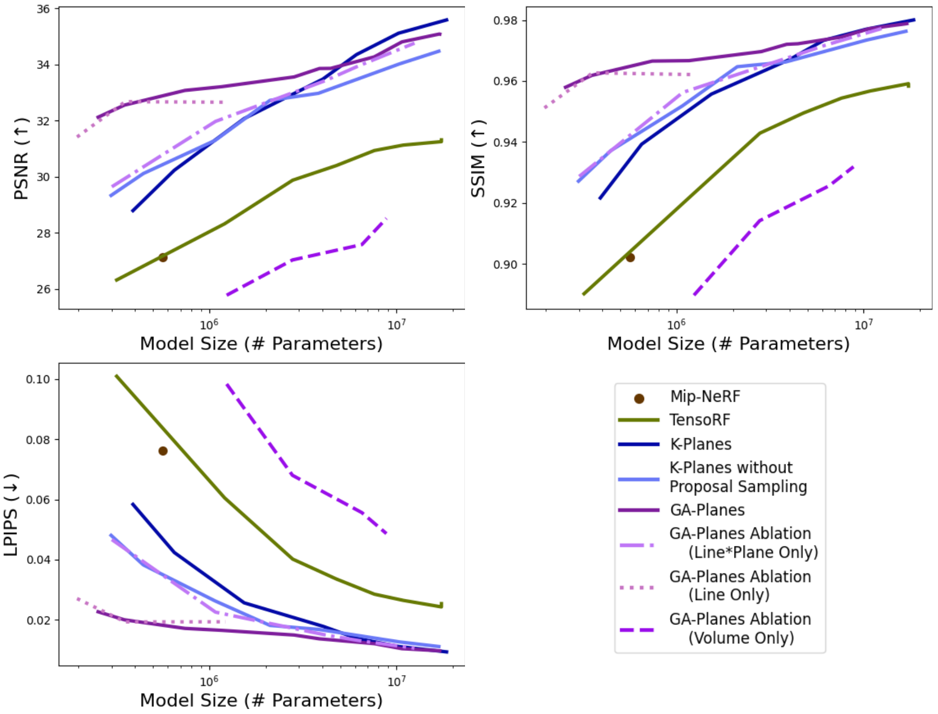


Figure 10: Results on radiance field reconstruction. Nonconvex GA-Planes (with feature multiplication) offers the most efficient representation: when the model is large it performs comparably to the state of the art models, but when model size is reduced it retains higher performance than other models. Here all models are trained for the same number of epochs on the *lego* scene.

Note that we fix these sizes across (non/semi)convex formulations, which causes slight variations in the size of the decoder, however, the grids constitute the most number of parameters, making this effect negligible.

A.7.3. Video Segmentation

GA-Planes model uses feature dimensions $[d_1, d_2, d_3] = [32, 16, 8]$ and resolutions $[r_1, r_2, r_3] = [128, 128, 64]$. When the features are combined by multiplication in the nonconvex model, $d_1 = d_2 = 16$. Multiresolution grids are not used for this task. The model size is 2.9 M. Tri-Planes model has the feature dimension $d_2 = 59$, and resolution $r_2 = 128$ resulting in a total number of parameters of 2.9 M.

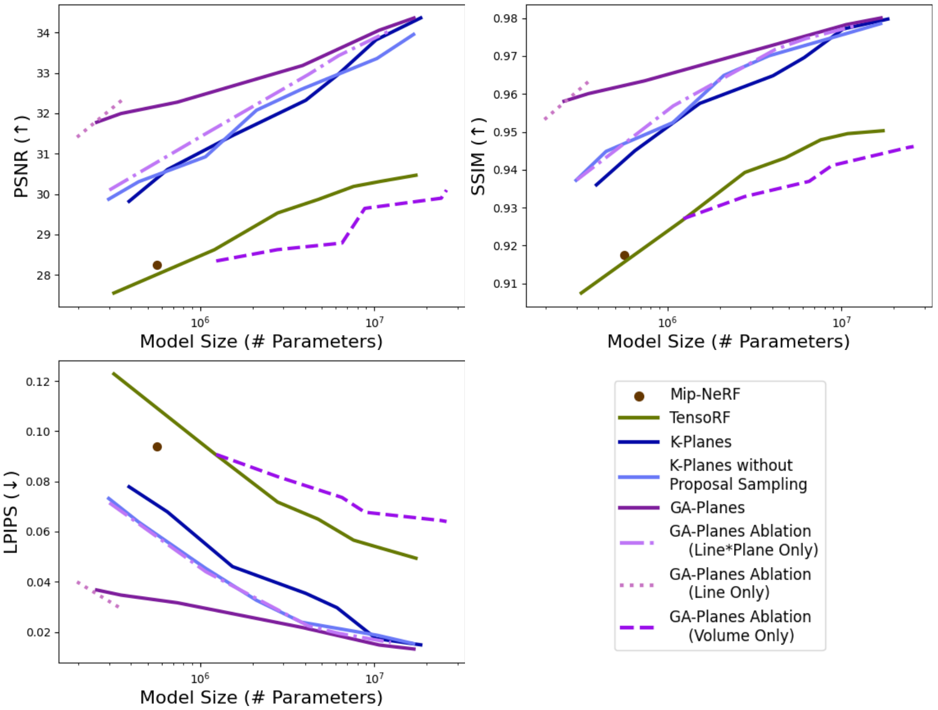


Figure 11: Results on radiance field reconstruction. Nonconvex GA-Planes (with feature multiplication) offers the most efficient representation: when the model is large it performs comparably to the state of the art models, but when model size is reduced it retains higher performance than other models. Here all models are trained for the same number of epochs on the *chair* scene.

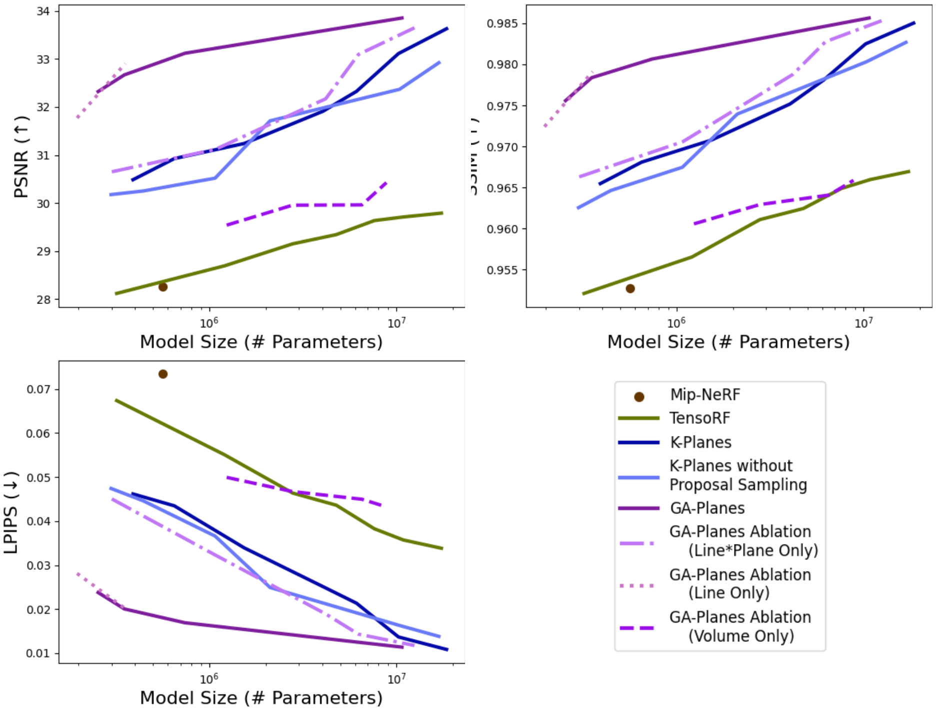


Figure 12: Results on radiance field reconstruction. Nonconvex GA-Planes (with feature multiplication) offers the most efficient representation: when the model is large it performs comparably to the state of the art models, but when model size is reduced it retains higher performance than other models. Here all models are trained for the same number of epochs on the *mic* scene.

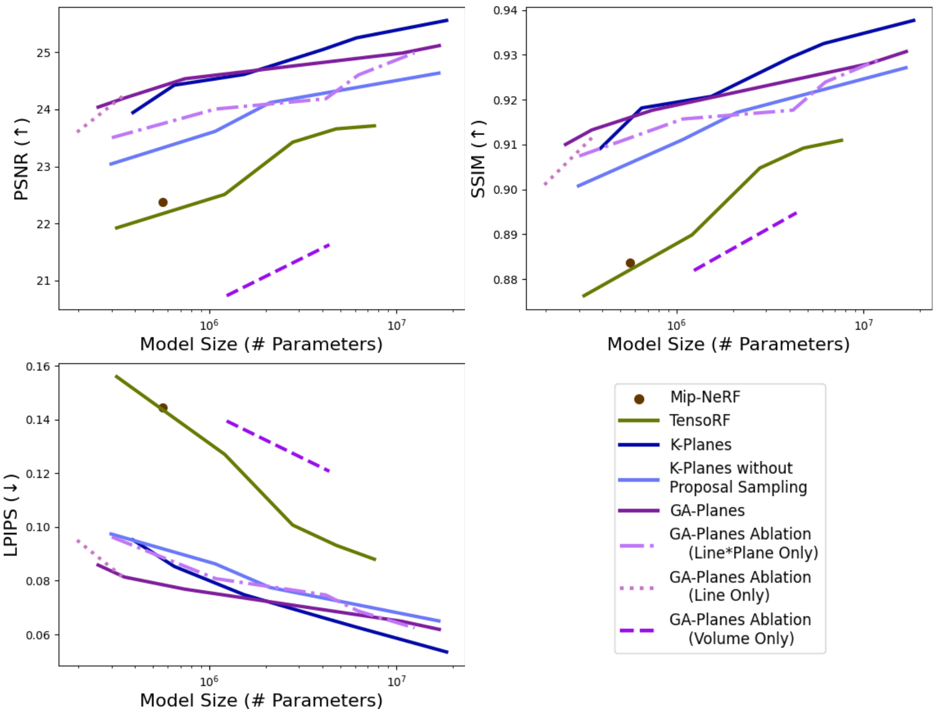


Figure 13: Results on radiance field reconstruction. Nonconvex GA-Planes (with feature multiplication) offers the most efficient representation: when the model is large it performs comparably to the state of the art models, but when model size is reduced it retains higher performance than other models. Here all models are trained for the same number of epochs on the *drums* scene.

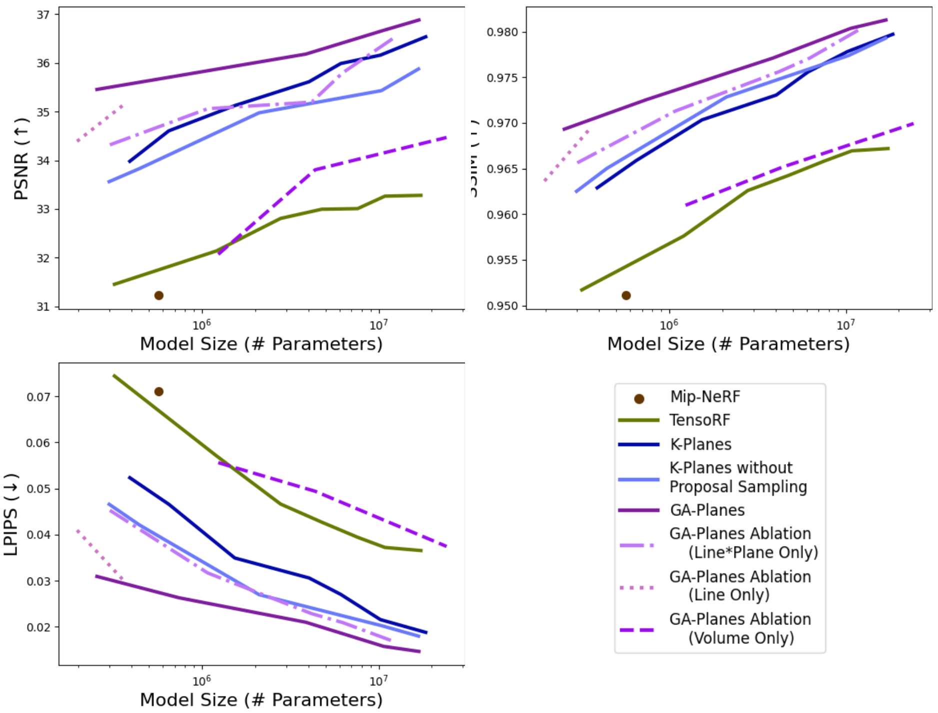


Figure 14: Results on radiance field reconstruction. Nonconvex GA-Planes (with feature multiplication) offers the most efficient representation: when the model is large it performs comparably to the state of the art models, but when model size is reduced it retains higher performance than other models. Here all models are trained for the same number of epochs on the *hotdog* scene.

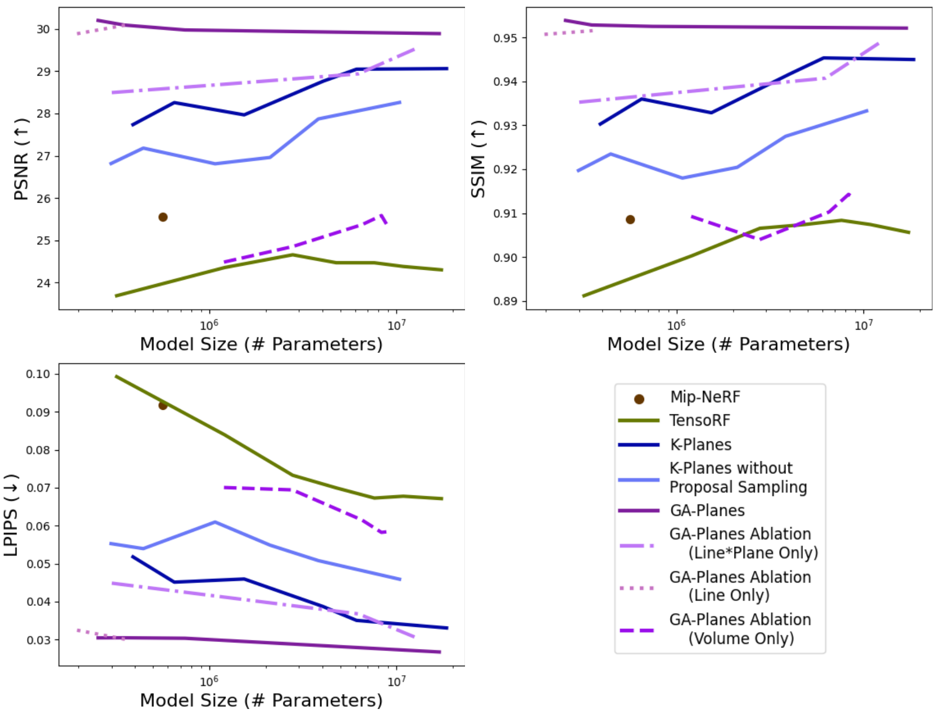


Figure 15: Results on radiance field reconstruction. Nonconvex GA-Planes (with feature multiplication) offers the most efficient representation: when the model is large it performs comparably to the state of the art models, but when model size is reduced it retains higher performance than other models. Here all models are trained for the same number of epochs on the *materials* scene.

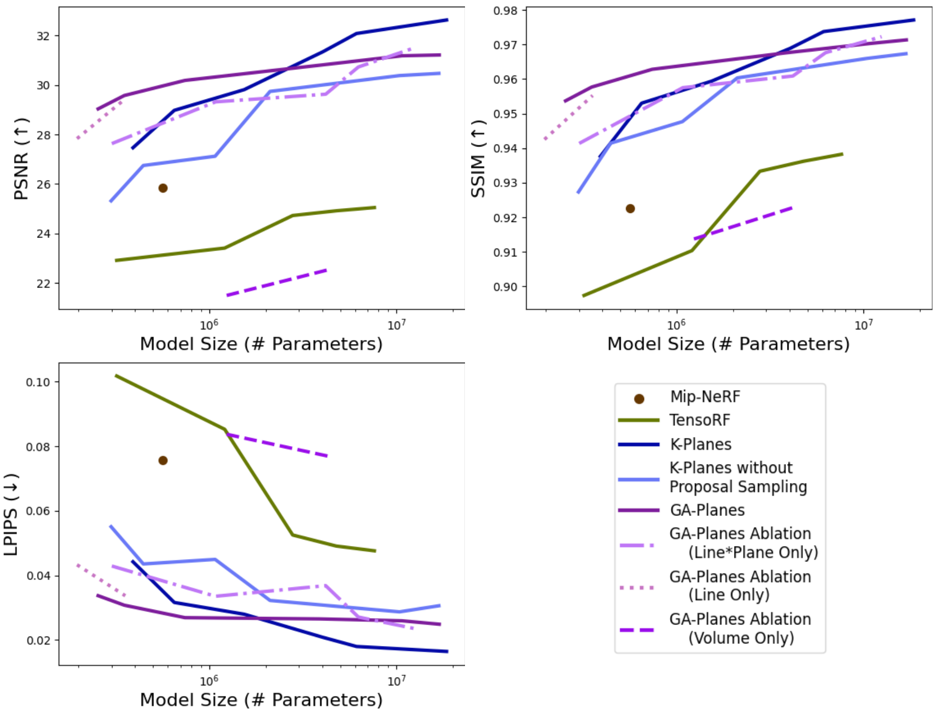


Figure 16: Results on radiance field reconstruction. Nonconvex GA-Planes (with feature multiplication) offers the most efficient representation: when the model is large it performs comparably to the state of the art models, but when model size is reduced it retains higher performance than other models. Here all models are trained for the same number of epochs on the *ficus* scene.

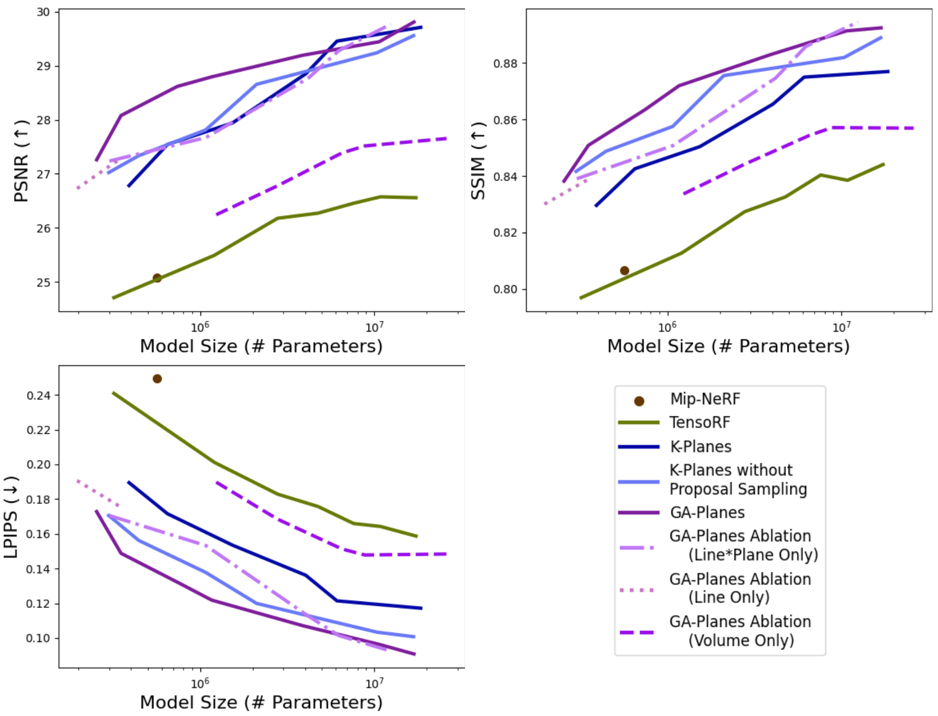


Figure 17: Results on radiance field reconstruction. Nonconvex GA-Planes (with feature multiplication) offers the most efficient representation: when the model is large it performs comparably to the state of the art models, but when model size is reduced it retains higher performance than other models. Here all models are trained for the same number of epochs on the *ship* scene.

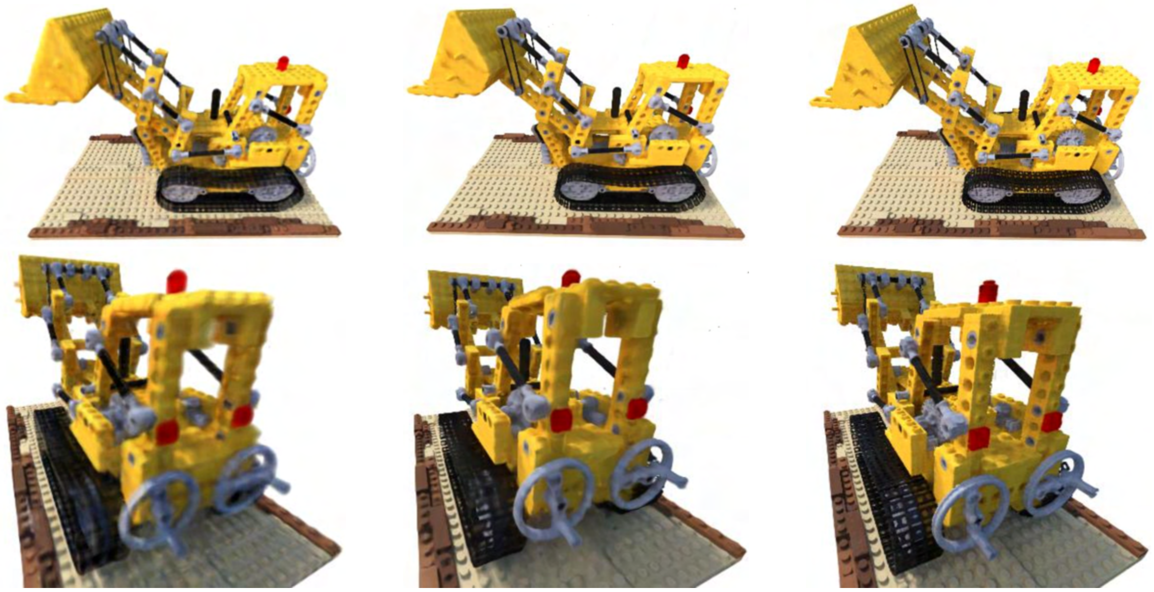


Figure 18: Rendering comparison for the *lego* scene: TensoRF on the left (0.32 M parameters), K-Planes in the middle (0.39 M parameters), GA-Planes on the right (0.25 M parameters).

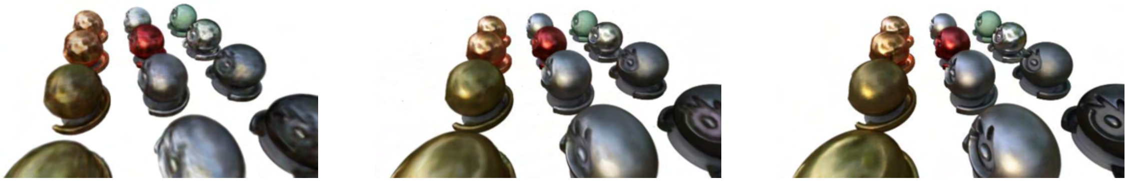


Figure 19: Rendering comparison for the *materials* scene: TensoRF on the left (0.32 M parameters), K-Planes in the middle (0.39 M parameters), GA-Planes on the right (0.25 M parameters).

Model	Resolutions	Channel Dimensions	Multiresolution	Proposal Network Resolutions	Number of model parameters (M)
K-plane	32	4	[1, 2, 4]	[32, 64]	0.390
	32	8	[1, 2, 4]	[32, 64]	0.649
	64	4	[1, 2, 4]	[64, 128]	1.533
	64	8	[1, 2, 4]	[128, 256]	4.041
	64	16	[1, 2, 4]	[128, 256]	6.107
	128	8	[1, 2, 4]	[128, 256]	10.234
	128	16	[1, 2, 4]	[128, 256]	18.493
K-plane without proposal sampling	32	4	[1, 2, 4]	-	0.298
	40	4	[1, 2, 4]	-	0.444
	64	4	[1, 2, 4]	-	1.073
	64	8	[1, 2, 4]	-	2.108
	100	6	[1, 2, 4]	-	3.822
	128	10	[1, 2, 4]	-	10.367
	129	16	[1, 2, 4]	-	16.824
TensoRF	128	[2, 4]	-	-	0.320
	256	[2, 4]	-	-	1.207
	256	[6, 8]	-	-	2.786
	256	[12, 12]	-	-	4.760
	300	[12, 16]	-	-	7.609
	300	[16, 24]	-	-	10.860
	300	[32, 32]	-	-	17.362
	300	[16, 48]	-	-	17.364
GA-plane	[200, 4, 4]	[32, 32, 4]	[1, 2, 4]	-	0.254
	[200, 8, 4]	[32, 32, 4]	[1, 2, 4]	-	0.351
	[200, 16, 8]	[32, 32, 4]	[1, 2, 4]	-	0.740
	[200, 32, 8]	[16, 16, 4]	[1, 2, 4]	-	1.164
	[100, 100, 16]	[6, 6, 8]	[1, 2, 4]	-	3.874
	[200, 128, 32]	[10, 10, 8]	[1, 2, 4]	-	10.681
	[200, 128, 32]	[16, 16, 8]	[1, 2, 4]	-	16.908
GA-plane ablation-VM	32	4	[1, 2, 4]	-	0.301
	64	4	[1, 2, 4]	-	1.078
	128	4	[1, 2, 4]	-	4.180
	128	6	[1, 2, 4]	-	6.251
	128	12	[1, 2, 4]	-	12.465
GA-plane ablation-CP	200	32	[1, 2, 4]	-	0.196
	200	64	[1, 2, 4]	-	0.355
GA-plane ablation-volume	18	[3, 5, 6]	[1, 2, 3]	-	1.236
	24	[4, 4, 6]	[1, 2, 3]	-	2.778
	32	[4, 4, 6]	[1, 2, 3]	-	6.529
	32	[4, 6, 8]	[1, 2, 3]	-	8.824

Table 6: Model configurations used for the radiance field modeling task on the Blender dataset.