Vision, Modeling, and Visualization (2024)
L. Linsen and J. Thies (Eds.)

Adaptive Grids for Neural Scene Representation

Barbod Pajoum , Gereon Fox , Mohamed Elgharib , Marc Habermann , Christian Theobalt

Max Planck Institute for Informatics

Abstract

We introduce a novel versatile approach to enhance the quality of grid-based neural scene representations. Grid-based scene representations model a scene by storing density and color features at discrete 3D points, which offers faster training and rendering than purely implicit methods such as NeRF. However, they require high-resolution grids to achieve high-quality outputs, leading to a significant increase in memory usage. Common standard grids with uniform voxel sizes do not account for the varying complexity of different regions within a scene. This is particularly evident when a highly detailed region or object is present, while the rest of the scene is less significant or simply empty. To address this we introduce a novel approach based on frequency domain transformations for finding the key regions in the scene and then utilize a 2-level hierarchy of grids to allocate more resources to more detailed regions. We also created a more efficient version of this concept, that adapts to a compact grid representation, namely TensoRF, which also works for very few training samples.

CCS Concepts

• ***Computing methodologies*** → ***Volumetric models;*** *Rendering; Reconstruction;*

1. Introduction

Novel view synthesis (NVS) is a significant challenge in computer vision and graphics. Given some input images of a scene with known camera poses, the task is to render images of the scene from unseen camera viewpoints. A popular approach for NVS is Neural Radiance Fields (NeRFs) [MST*21], which can produce photo-realistic images of scenes with intricate geometry and view-dependent appearance effects. NeRF and many of its variants [BMT*21, BMV*22, RPLG21] use a Multilayer Perceptron (MLP) to implicitly represent a 3D scene. The MLP takes the positional encoding of a 3D point along with the viewing direction as input and outputs the corresponding density and color. Although employing an MLP produces great results for novel view synthesis, it also introduces a computational bottleneck. This is because the MLP has to be evaluated for hundreds of points along each ray during both training and rendering. Multiple papers [CXG*22,FKYT*22,GKJ*21,HSM*21,LGZL*20,MESK22, RPLG21, SSC22, YLT*21] have investigated ideas to reduce this computational cost. One idea is to avoid querying a large MLP for every 3D point and to instead store density and colors explicitly in a 3D voxel grid. For example, DVGO [SSC22] and Plenoxels [FKYT*22] train a 3D grid to store density and color features at discrete 3D points. By applying trilinear interpolation to the grid, features can be obtained at any continuous 3D point. Then, with the help of volume rendering the final color of the ray is computed. Learning features explicitly in a grid rather than implicitly by an MLP increases training and rendering speed significantly. However, this comes at the price of considerable memory consumption, as high-resolution grids are needed to achieve high-quality results.

DVGO *Ours*

Figure 1: *Our method allocates more resources to detailed regions of the scene, which results in a higher quality for those regions. In this example, our result reveals significantly more details of the foreground objects compared to the DVGO result on the left. The examined scene is taken from the LF dataset [YSHWSH16].*

In this paper, we present a novel approach for enhancing grid-based neural scene representations. Our approach utilizes the fact that natural scenes usually include regions of varying complexity and detail, which is not taken into account when using uniform grids with a fixed resolution. Instead, we propose a two-step solution to handle this problem: In the first step we identifiy the regions

DOI: 10.2312/vmv.20241205

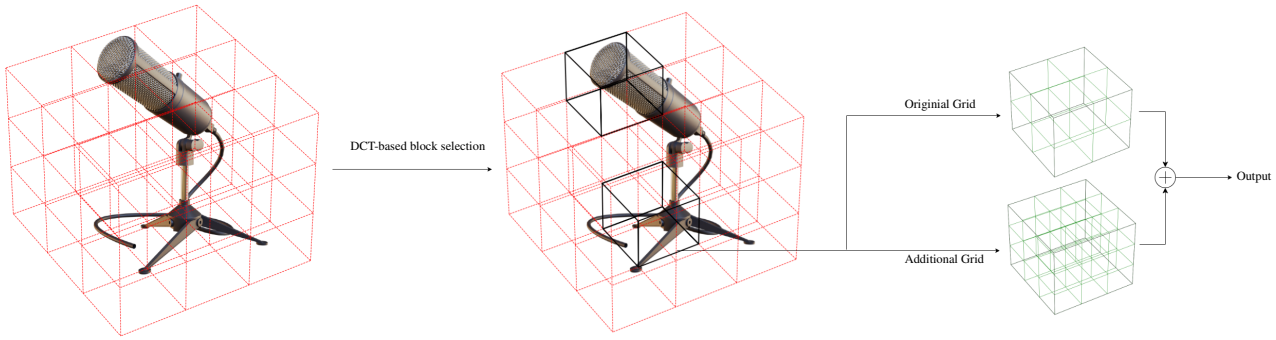

Figure 2: *We illustrate our method using a* $6 \times 6 \times 6$ *voxel-grid and a* $2 \times 2 \times 2$ *block size, resulting in* $N_B = 27$ *blocks. The left diagram shows the blocks in red and for better visual clarity, the* $2 \times 2 \times 2$ *green grids inside each block are not shown. This figure only shows the density grid, as the principle is the same for the multi-channel color grid. We apply DCT to compute the importance measure P for each block, and then select the top* $N_I = 2$ *blocks as the important ones, shown in black in the middle diagram. As expected these blocks have the most details, in this example the microphone's head and stand. The right diagram shows the process for the important blocks: Each block, red or black, has an original 2×2×2 grid. For the important blocks (black), we add a finer grid to capture more details, as shown in the bottom part of the right diagram. We sum the densities from both grids to get the final density at any point within an important block.*

that contain more complex structures than others. To this end, we partition the voxel grid into blocks and then introduce a novel numerical measure for each block that reflects the complexity of its content. Our measure applies Discrete Cosine Transform (DCT) to each block and then uses a weighted mean of higher frequency DCT coefficients as the final importance criterion. By examining a function in the frequency domain, we can obtain useful information about its degree of variation, which is the motivation behind using DCT. In the second step of our method, the blocks are sorted according to their complexity/importance, and then additional higher resolution grids are allocated to the more complex blocks in a hierarchical manner.

As Fig. 1 demonstrates, this method improves the reconstruction quality on complex scenes, especially in detailed regions. We apply our method to two state-of-the-art grid-based methods, DVGO and TensoRF, and show that we can achieve better results, with minimal memory cost and time overhead. To this end, we demonstrate that our method accurately identifies regions of high importance and that the allocation of additional higher-resolution grids improves quality.

In summary, we make three main contributions:

- To the best of our knowledge, our method is the first that uses 3D DCT to find the intricate structures in a 3D scene. We propose a novel measure based on frequency coefficients to find the important blocks.
- Our method uses hierarchical grids for representing blocks of higher complexity. Our optimized implementation of these two-layer grid leads to significant visual quality gain with minimal memory and time overhead.
- We demonstrate that our method can be applied to existing grid-based 3D scene representations, by showing both visual and numerical improvements over DVGO [SSC22] and TensorRF [CXG*22].

2. Related Work

Numerous approaches have been proposed for generating images from novel viewpoints using a set of training images. One approach is to capture the underlying 3D structure of the scene using different forms of representations. This includes meshes [WMG14, WZL*18], volumetric representations [SD99, KS00, LSS*19], or implicit functions, with NeRF [MST*21] being the most prominent example. NeRF can produce high-quality results as the 3D consistency induced by its representation prevents the model from overfitting to the training set. There are various papers trying to extend NeRF to more complicated [MBRS*21] or 360 scenes [BMV*22, ZRSK20] or to enhance its quality [BMT*21]. NeRF has also been adapted for applications other than Novel View Synthesis (NVS). This includes combining generative models with NeRF [CMK*21, NG21], enabling image generation from multiple viewpoints. Furthermore, several works use NeRF to render the scene under novel illuminations [BXS*20, SDZ*21, RES*22], while Block-NeRF [TCY*22] scales NeRF to large-scale urban environments. However, inferring 3D representation for NeRF from a set of 2D images can still introduce complexity and computational overhead.

NeRF is slow to train and render because the underlying MLP needs to be queried multiple times along each ray to obtain its final color through volume rendering. Many methods have attempted to improve NeRF's efficiency, among them one important group of methods [SSC22, FKYT*22, LGZL*20, CXG*22] concentrate on using an explicit learnable voxel grid rather than an implicit MLP. DVGO [SSC22] employs two separate grids for density and color at discrete 3D points. These color features, together with the viewing direction, are then provided to a much shallower MLP than the one used in NeRF. This MLP produces the final color for the given 3D point. Voxel grids and DVGO in particular, can greatly speed up training and rendering, reducing the time from days to minutes. However, a major challenge of their methods is

TensoRF *Ours*

Figure 3: *We use the T-Rex scene from LLFF [MSOC*19] to compare our result on the right with TensoRF on the left. Our method preserves thin structures such as the bones more accurately than TensoRF, while also reducing the memory consumption by more than 20%.*

the excessive memory consumption, which results from the huge size of their 3D grids that mostly contain empty space. This is because scenes frequently have considerable void regions. Plenoxels [FKYT*22] addresses this issue by using a sparse voxel grid, which only stores the density and color features for the voxels that exceed a density threshold. They also replaced neural features with spherical harmonics, which makes their approach completely explicit without any neural networks. Instant-NGP [MESK22] uses a multi-resolution hash grid to store the features, which, combined with their custom cuda kernels, results in fast rendering and high-quality outputs. TensoRF [CXG*22] significantly reduced memory requirements by factorizing the 3D grid into low-rank 1D and 2D vectors through matrix-vector decomposition. This approach allows for much lower memory consumption compared to previous methods, enabling the use of larger grids and achieving improved quality. The more recent version [HX23] further enhances these capabilities. A common limitation of these methods is their disregard for the varying complexity across different regions, leading to a uniform distribution of resources across all non-empty areas of the scene. Instant-NGP, however, is an exception which utilizes a hash grid structure that inherently allocates more resources to regions with higher levels of detail. Some approaches seek to combine implicit and explicit methods to harness the advantages of both. These methods typically train an implicit representation, such as NeRF, and then bake it into an explicit format to enable faster rendering. Noteworthy contributions in this area include [RSV*23, HSM*21].

An alternative line of research aims to eliminate the need for 3D representations and apply a neural network to directly estimate the final color of any ray cast into the scene which is known as light fields [LH96, DLD12, AHZ*22, WRH*22]. Light field is a straightforward and fast method, but it is limited to scenes with a dense capture and a small camera baseline [LH96, DLD12]. To overcome this limitation, some recent works have proposed to use subdivided light fields [AHZ*22] or to distill a NeRF into a light field model to get both 3D consistency and fast rendering speed [WRH*22]. Kerbl et al. [KKLD23] recently proposed a novel representation for NVS called 3D Gaussian Splatting. While their method boasts impressive quality and rendering speed, it has limitations, including poor geometry and issues with sparse training data.

3. Method

We now describe how to use adaptive grids for learning the density and color features of a 3D scene from a collection of input images. We start by reviewing the standard formulation of grid-based methods and then introduce two grid-based methods, DVGO [SSC22] and TensoRF [CXG*22], which we use as baselines for adaptive grids in the next sections. We then explain how to utilize adaptive grids and finally we demonstrate an optimized version of our approach, that integrates adaptive grids into the TensoRF compact representation and achieves state-of-the-art results.

3.1. Grid Based Representations

Grid-based geometry reconstruction methods usually employ two distinct 3D voxel grids for storing density and color features. The density voxel grid has a size of $N_x \times N_y \times N_z \times 1$, where N_x, N_y, and N_z are dataset-dependent hyper-parameters. The color voxel grid has the same dimensions, but with C channels: $N_x \times N_y \times N_z \times C$, where C is a hyper-parameter that controls the number of color features. The density and color features of any 3D point in the grid are obtainable by performing trilinear interpolation on each channel of the relevant grid. Based on this information, images can be obtained using volume rendering, similar to how it is used in NeRF [MST*21]: 1. For a given camera view, rays are projected into the scene. 2. Along each ray, multiple points are sampled and their color features and densities are obtained by applying trilinear interpolation on the grid values. 3. In case the color grid stores neural features, a shallow multilayer perceptron (MLP) takes the features together with a viewing direction and outputs the final color of the desired point. If spherical harmonics are used for the color features, as in Plenoxels [FKYT*22], then the final color can be obtained by evaluating the spherical harmonics at the appropriate viewing direction without using any neural networks. 4. After obtaining the colors and densities of the points along a ray, they are aggregated to obtain the final color $\hat{C}(\mathbf{r})$ of the ray, as in NeRF [MST*21]:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1-\exp(-\sigma_i\delta_i))\mathbf{c}_i \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right) \tag{1}$$

where σ_i and c_i are the density and color of the point i and δ_i is the distance to the next sample on the ray.

The density and color grids, as well as the shallow MLP are jointly learned, as the process of obtaining features from the grid, followed by volume rendering, is differentiable. Compared to NeRF [MST*21], the MLP for color synthesis in grid-based methods is much shallower, increasing the training and rendering speed.

3.2. Baseline Methods

To show the effectiveness of our approach we will apply our idea to DVGO and TensoRF. DVGO is an example of a representation with explicit 3D grids and TensoRF offers a compact representation of 3D grids with reduced memory footprint and improved quality. We summarise these baseline methods briefly:

DVGO [SSC22] starts with a lower resolution grid capturing the coarse geometry of the scene and then continues by training the final fine-grained grid. With the coarse shape information, points are only sampled in the occupied regions, enhancing efficiency and assisting in defining the final grid to tightly enclose the scene. In the coming sections, we will demonstrate how our adaptive grid strategy can be incorporated into DVGO, to enhance its quality by assigning additional resources to the more detailed regions.

TensoRF [CXG*22] represents grids more compactly: Instead of storing a 3D grid with cubically increasing memory (with respect to resolution), TensorRF uses a matrix-vector decomposition (VM), estimating a 3D grid with multiple vectors and matrices that are one- or two-dimensional. This decreases memory usage significantly and also allows higher resolution grids that would not be feasible in standard 3D grid representations such as DVGO. VM decomposition of the 3D grid G can be expressed as follows: For vectors v_i^x, v_j^y and v_k^z and matrices $M_k^{x,y}$, $M_j^{x,z}$, $M_i^{y,z}$, with $1 \le i \le R_1$, $1 \le j \le R_2$ and $1 \le k \le R_3$, the grid tensor G is defined as

$$G = \sum_{i=1}^{R_1} v_i^x \circ M_i^{y,z} + \sum_{j=1}^{R_2} v_j^y \circ M_j^{x,z} + \sum_{k=1}^{R_3} v_k^z \circ M_k^{x,y} \tag{2}$$

where $\circ$ denotes the vector-matrix outer product. Thus, instead of storing grid parameters at every grid point explicitly, TensoRF computes them from the vectors and matrices, which are trainable parameters of the model. The numbers of vectors and matrices to be used, collectively referred to as "components", are governed by hyper-parameters R_1, R_2, and R_3. More components lead to better estimation but also require more memory. For the color grid, an additional vector is added to Eq. (2), to adapt it to multiple channels. This results in a significantly more compact representation. For more details, we refer to [CXG*22]. In the following sections, we describe how to incorporate our adaptive grids idea into this representation and how this can improve results, by using fewer components for approximating the smooth and simple regions and using more components for representing the complex and fine details of the scene.

3.3. Identifying Important Blocks

We now describe how important blocks, i.e. regions of the scene with high-frequency details, can be identified. Then we describe a more elaborate version of our idea, designed for TensoRF.

3.3.1. Standard grid representations:

The grid consists of $N_x \times N_y \times N_z \times F$ cells, where $F = 1$ for the density grid and $F = C$ (number of color channels) for the color grid. We partition the set of grid cells into *blocks*, each comprising $B_x \times B_y \times B_z \times F$ cells. Let N_B be the total number of blocks.

Our goal is to identify "important" blocks, i.e. blocks that contain high-frequency details. To this end, we introduce a novel measure of block complexity based on the analysis of their frequency spectrum. We only explain the process for the density grid, as the one for the color grid is very similar:

The density grid inside block i is denoted by D_i. We assume that the blocks are cubic, i.e., $B_x = B_y = B_z = B$. We apply Discrete Cosine Transform (DCT) to transform D_i to the frequency domain and obtain the DCT coefficients $DCT(D_i) = T_i \in \mathbb{R}^{B\times B\times B}$. The formula for DCT of the ith block where $0 \le i < N_B$ is given by [MRIdVFB23]:

$$T_i[u,v,w] := \alpha(u)\alpha(v)\alpha(w)\sum_{m,n,p=0}^{B-1} D_i[m,n,p]\gamma(m,u)\gamma(n,v)\gamma(p,w) \tag{3}$$

where

$$\alpha(x) := \begin{cases} \frac{1}{\sqrt{N}} & : x = 0 \\ \sqrt{\frac{2}{N}} & : \text{otherwise} \end{cases} \tag{4}$$

and

$$\gamma(a,b) := \cos\left(\frac{\pi}{2N}(2a+1)b\right) \tag{5}$$

The Inverse Discrete Cosine Transform (IDCT) is given as follows [MRIdVFB23] where $0 \le i < N_B$:

$$D_i[m,n,p] = \sum_{u,v,w=0}^{B-1} T_i[u,v,w]\alpha(u)\alpha(v)\alpha(w)\gamma(m,u)\gamma(n,v)\gamma(p,w) \tag{6}$$

By inspecting the IDCT formula, we can see that higher values of u, v and w correspond to higher frequencies in the cosine functions. Therefore, T_i having larger values for large u,v and w values indicates that block i contains high-frequency details. Conversely, if we have a smooth uniform region, then T_i would be nearly zero for all large u, v and w values. We use this intuition to define our novel importance measure P_i for block i:

$$P_i := \sum_{u,v,w=\lfloor\frac{B}{2}\rfloor}^{B-1} W[u,v,w] \times T_i[u,v,w] \;\forall\, 0 \le i < N_B \tag{7}$$

where $W[u,v,w] = (u-\lfloor\frac{B}{2}\rfloor)^2 + (v-\lfloor\frac{B}{2}\rfloor)^2 + (w-\lfloor\frac{B}{2}\rfloor)^2$. Based on Eq. (7), P_i is a weighted mean of the higher frequency coefficients

Color variation *Density variation*

Figure 4: *Most important blocks according to Eq. (7), are highlighted: For the chair, blocks with the greatest color variations are green, while for the ship, blocks with the highest density variations are white. Scenes taken from the Synthetic-NeRF dataset [MST*21]. Complex textures in the left and thin structures in the right have the highest P values.*

in T_i and W is a weight matrix that gives more importance to the larger values of u, v and w. We can rank all N_B blocks according to their importance measure P_i and select the top N_I ones to allocate additional finer resolution grids to them. N_I is a hyper-parameter that depends on the scene and balances between quality gain and memory cost. We can define the importance measure P on both density and color grids: On the density grid it puts more weight on the intricate geometry and structures, while on the color grid it emphasizes complex color variations in the scene. We demonstrate the effectiveness of the proposed measure by showing the blocks with highest P values in two scenes from the synthetic-NeRF dataset [MST*21] in Fig. 4.

3.3.2. TensoRF

The process in Sec. 3.3.1 can also be applied to TensoRF, but slows training down due to two factors: First, TensoRF does not store the 3D grid explicitly, and hence it requires some computation to transform the vectors and matrices into the 3D grid. Second, TensoRF also uses higher grid resolutions than DVGO, which is why iterating over the blocks is more time-consuming. Hence we propose a more efficient approach for identifying the blocks with the highest details: Our intuition is to analyze the frequency spectra of the matrices $M_i^{y,z}$, $M_j^{x,z}$ and $M_k^{x,y}$ in Eq. (2) to estimate the variation in a block. Applying DCT to a matrix M with dimensions $d \times d$ gives us a set of coefficients, denoted by $DCT(M)$. Then we define $Q(M)$ as the importance measure of M:

$$Q(M) := \sum_{i,j=\lfloor \frac{d}{2} \rfloor}^{d} W'[i,j] \times T[i,j] \tag{8}$$

where $W'[i,j] = (i - \lfloor \frac{d}{2} \rfloor)^2 + (j - \lfloor \frac{d}{2} \rfloor)^2$. Eq. (8), like Eq. (7), takes a weighted average of high-frequency components, but for only two dimensions. Based on Eq. (8) we estimate the importance P_i for each block $1 \leq i \leq N_B$ in the grid G as follows. Block i is a subgrid of G with size $B \times B \times B$, starting from the voxels x_i, y_i and z_i. We estimate the total variation for block i by summing up the variations of the matrices from Eq. (2), which gives us P_i:

$$\begin{aligned} P_i := & \sum_{i=1}^{R_1} Q(M_i^{y,z}[y_i : y_i + B, z_i : z_i + B]) \\ + & \sum_{j=1}^{R_2} Q(M_j^{x,z}[x_i : x_i + B, z_i : z_i + B]) \\ + & \sum_{k=1}^{R_3} Q(M_k^{x,y}[x_i : x_i + B, y_i : y_i + B]) \end{aligned} \tag{9}$$

Once we have the importance values for all the blocks, we can follow the method in 3.3.1 and select the N_I blocks with the highest P_I values as the most important blocks in the scene.

3.4. Higher resolution grid for Important Blocks

3.4.1. Standard grid representations

Given the N_I blocks of interest from the previous section, we allocate two supplementary grids of higher resolution to each of these blocks: one grid for the density with a single channel, and another grid for the color with C channels. While the original density block has a size of $B \times B \times B \times 1$, we allocate a finer density grid with a block size of $B_I \times B_I \times B_I \times 1$, where $B_I > B$. During training and rendering, we sum the density values from both grids to obtain the final density at any point within the important blocks. Thus, the finer grid serves as a correction term to the original grid, and enables the representation of finer details that the original grid cannot capture. The same process applies to the color grid, except that it has C channels. An overview of our approach can be seen in Fig. 2. This methodology can be applied to any standard voxel grid representation such as DVGO. However, TensoRF employs a distinctive approach by estimating the 3D grid using vector-matrix decomposition.

3.4.2. Extending TensoRF

TensoRF estimates a 3D grid using vector-matrix decomposition as specified in Eq. (2). Increasing the numbers of components (R_1, R_2, and R_3) leads to improved estimation accuracy at the expense of a larger memory footprint. We propose a selective approach that only increases the numbers of components for those blocks that are deemed important in the scene, rather than naively adding more components for the entire scene. Using a higher resolution grid is not very effective in this case, since TensoRF already employs very fine grids. Instead we add more components for estimating blocks with high frequency details: Any point within an important block is given *two* terms for its density and color. The first term is common to all the points in the scene, and is derived from Eq. (2). The second term is based on the additional $R_1' + R_2' + R_3'$ components but is only defined for important blocks instead of the whole grid.

This method enables us to optimize resource utilization and achieve higher or similar quality with lower memory consumption.

Methods	Synthetic-NeRF			LF			BlendedMVS		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DVGO [SSC22]	31.95	0.957	0.053	25.01	0.80	**0.32**	28.02	0.922	0.075
Ours	**32.50**	**0.961**	**0.051**	**25.8**	**0.831**	**0.32**	**28.5**	**0.934**	**0.069**

Table 1: *Quantitative comparison of our approach to the baseline DVGO on three datasets. Scores are averaged over all scenes in each dataset. Our method consistently improves quality on both synthetic and real-world datasets.*

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
Gaussian Splatting [KKLD23]	23.03	16.2	22.24	22.79	23.81	18.03	23.22	15.98	21.89
Ours	**28.07**	**20.4**	**22.96**	**31.16**	**30.01**	**21.88**	**27.02**	**24.01**	**25.69**

Table 2: *PSNR scores across different scenes from NeRF-Synthetic dataset with only 10 training images. Gaussian Splatting fails to work on such sparse training data while our approach can produce significantly better results.*

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean	Size
Plenoxels [FKYT*22]	33.98	25.35	31.83	36.43	34.1	29.14	33.26	29.62	31.71	780M
NSVF [LGZL*20]	33.19	25.18	31.23	37.14	32.29	32.68	34.27	27.93	31.75	-
Instant-NGP [MESK22]	35.00	26.02	33.51	37.40	36.39	29.78	36.22	**31.10**	33.176	-
DVGO [SSC22]	34.06	25.4	32.58	36.77	34.65	29.58	33.15	29.02	31.90	610Mb
Large DVGO	34.71	25.59	33.07	36.96	35.31	29.58	33.65	29.56	32.30	1.2Gb
Ours-DVGO	35.08	25.67	33.21	37.14	35.35	29.57	34.30	29.81	32.50	850Mb
TensoRF [CXG*22]	35.76	26.01	33.99	**37.41**	36.46	30.12	34.61	**30.77**	33.14	71.8M
Ours-TensoRF	**35.94**	**26.03**	**34.17**	37.39	**36.81**	**30.53**	34.91	30.67	**33.31**	66M

Table 3: *We compare our PSNR scores to state-of-the-art explicit approaches on the Synthetic-Nerf dataset [MST*21]. We observe that all those approaches are outperformed by Ours-TensoRF on almost all scenes, even though our model uses less memory than TensoRF. Our method also can improve DVGO with a moderate increase in memory and outperform the "Large DVGO", which has 1.25 grid size and uses twice the memory of the original version.*

Figure 5: *We compare our method (right) to DVGO (left) with grid size* 200^3*, on the microphone scene from Synthetic-NeRF. Our approach improves the resolution of fine details in the microphone head while using 29.2% less memory(850Mb vs 1.2Gb).*

4. Experimental Results

4.1. Datasets

We evaluate our approach on both synthetic and real-world datasets. We use Synthetic-Nerf [MST*21], a widely used dataset for novel view synthesis methods, which consists of 8 different scenes with complex view-dependent effects and intricate structures. We also test on four objects from the BlendedMVS [YLL*20] dataset, following the protocol provided in [LGZL*20]. For real-world datasets, we use LLFF [MSOC*19], a challenging dataset with 8 different scenes that have very sparse training samples, and LF [YSHWSH16], a dataset with 5 different realistic scenes that have a foreground object in a cluttered background.

4.2. Implementation Details

Our method based on DVGO: We integrated our idea with the DVGO model to show the effectiveness of our approach on standard grid representations. After training DVGO for 5000 steps, when it has learned the overall structure and texture of the scene, we identify the important blocks: First we partition the entire grid into $16 \times 16 \times 16$ blocks and apply DCT to the color grid, yielding the importance measure P_i for each block $0 \leq i < N_B$. Then we sort all the N_B blocks in descending order based on P_i and select the top $N_I = 64$ blocks. In the next step, we allocate an additional $32 \times 32 \times 32$ grid to each of these 64 important blocks and continue training the main grids and the additional higher-resolution grids for 20k steps. For a point in a normal block, the process is the same as the baseline approach, but for a point in an important block, we add the density and color values from both the original and the higher-resolution grid to obtain a fine-grained density and color value. The remaining steps follow the DVGO [SSC22] paper.

Our method based on TensoRF: We adapted our idea to TensoRF, resulting in a compact, state-of-the-art model with adaptive grids. The process is similar to DVGO, but we use larger blocks of

size 64×64×64 due to the larger grid in TensoRF. Additionally, we identify the important blocks after training TensoRF for 7000 steps.

4.3. Comparisons

Speed is a key advantage of grid-based representations, so we only consider methods that have training and rendering speeds comparable to ours. We leave out implicit methods such as [MST*21, BMT*21, BMV*22], that require hours/days of training, while our model can be trained in less than 30 minutes, or even less than 10 minutes with the DVGO baseline. Gaussian Splatting, one of the recent state-of-the-art representation for NVS, achieves a PSNR comparable to ours on the Nerf-Synthetic dataset (33.36dB vs. 33.35dB) and performs better in unbounded scenes. However, it has the drawback of not functioning well in sparse settings with few training images, as shown in Tab. 2. We note that our method is implemented entirely in PyTorch code, unlike approaches such as Instant-NGP, Plenoxels and Gaussian Splatting, which rely on custom CUDA kernels.

Our method based on DVGO: Table 1 compares the quality of our method to that of the bare DVGO by averaging PSNR, SSIM [WBSS04] and LPIPS [ZIE*18] scores over three different datasets: The added precision of our additional blocks leads to a small, but consistent improvement in all scores. The time overhead of our approach is also minimal, as the baseline DVGO training takes 5 minutes and ours takes 6 minutes with a single NVIDIA 3090 GPU. Table 3 compares methods on the Synthetic-NeRF dataset: "Ours-DVGO" consistently outperforms the Large DVGO model while using 30% less memory. This highlights the effectiveness allocating resources where they are most needed. We note that our method is designed for scenes that have certain areas of interest, while other regions are less important. For these scenes, such as the microphone (Fig. 5), we obtain the largest quality improvement over DVGO, as the microphone head has many fine details, while the microphone stand is smooth. Therefore, our method achieves a significant improvement both visually (Fig. 5) and in terms of PSNR (Tab. 3).

Methods	LLFF			
	PSNR↑	SSIM↑	LPIPS↓	Size↓
Plenoxels	26.29	0.839	0.210	2.54Gb
TensoRF-VM-48	26.51	0.832,	0.217	**90Mb**
TensoRF-VM-96	26.73	0.839	**0.204**	180Mb
Ours-TensoRF	**26.75**	**0.848**	0.209	**95Mb**

Table 4: *Quantitative comparison of our method with other grid-based approaches on the LLFF [MSOC* 19] dataset shows that integrating our idea into the TensoRF model yields superior quality compared to TensoRF with 96 components, while utilizing only half as much memory.*

Our method based on TensoRF: Also in Tab. 3, we evaluate our method on the basis of TensoRF. Despite reducing the memory consumption to half, compared to standard TensoRF, we still outperform it in terms of PSNR in almost all the scenes of the Synthetic-NeRF dataset. Our approach reconstructs the intricate details and thin structures better while using less memory as shown in Fig. 3. Our method based on TensoRF also can handle training data as sparse as 10 images for simple scenes. The recent 3D Gaussian splatting paper [KKLD23], however, fails to produce satisfactory results under such sparse settings, see Tab. 2. For more qualitative results please see our supplemental video.

5. Limitations

While our approach clearly uses memory more efficiently and enhances the quality of grid-based methods, it cannot compete with implicit methods such as Mip-NeRF [BMV*22] or Gaussian Splatting [KKLD23] in more complex, realistic and unbounded scenes. This limitation primarily stems from the inherent shortcomings of grid-based approaches, which make them less suitable for such scenarios. Adapting hierarchical representations to implicit methods with the aid of DCT scores could potentially improve these implicit approaches, which we leave for future work.

6. Conclusion

We have presented a method that uses adaptive grids for neural scene representations. Our method has the benefit of being applicable to various grid-based neural scene representations. We have tested our method using two main state of the art representations, showing that it improves their tradeoff between model size and output quality by assigning more resources to regions with complex structures. We also compared against the most recent method of 3D Gaussian Splatting, demonstrating that our method can achieve better results in a few-shot setting. Future work could investigate the idea of hierarchical representations based on frequency spectra for other scene representations, such as Gaussian Splatting [KKLD23] or Mip-NeRF [BMV*22]. We also believe that the concept of hierarchical grids can be useful in interactive applications, when the user requests more accuracy in a chosen part of the scene.

Acknowledgments. This work was supported by ERC Consolidator Grant 4DReply (770784).

References

[AHZ*22] ATTAL B., HUANG J.-B., ZOLLHÖFER M., KOPF J., KIM C.: Learning neural light fields with ray-space embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 19819–19829. 3

[BMT*21] BARRON J. T., MILDENHALL B., TANCIK M., HEDMAN P., MARTIN-BRUALLA R., SRINIVASAN P. P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 5855–5864. 1, 2, 7

[BMV*22] BARRON J. T., MILDENHALL B., VERBIN D., SRINIVASAN P. P., HEDMAN P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 5470–5479. 1, 2, 7

[BXS*20] BI S., XU Z., SRINIVASAN P., MILDENHALL B., SUNKAVALLI K., HAŠAN M., HOLD-GEOFFROY Y., KRIEGMAN D., RAMAMOORTHI R.: Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824* (2020). 2

[CMK*21] CHAN E. R., MONTEIRO M., KELLNHOFER P., WU J., WETZSTEIN G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 5799–5809. 2

[CXG*22] CHEN A., XU Z., GEIGER A., YU J., SU H.: Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision* (2022), Springer, pp. 333–350. 1, 2, 3, 4, 6

[DLD12] DAVIS A., LEVOY M., DURAND F.: Unstructured light fields. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 305–314. 3

[FKYT*22] FRIDOVICH-KEIL S., YU A., TANCIK M., CHEN Q., RECHT B., KANAZAWA A.: Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 5501–5510. 1, 2, 3, 6

[GKJ*21] GARBIN S. J., KOWALSKI M., JOHNSON M., SHOTTON J., VALENTIN J.: Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 14346–14355. 1

[HSM*21] HEDMAN P., SRINIVASAN P. P., MILDENHALL B., BARRON J. T., DEBEVEC P.: Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 5875–5884. 1, 3

[HX23] HAN K., XIANG W.: Multiscale tensor decomposition and rendering equation encoding for view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 4232–4241. 3

[KKLD23] KERBL B., KOPANAS G., LEIMKÜHLER T., DRETTAKIS G.: 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics 42*, 4 (2023). 3, 6, 7

[KS00] KUTULAKOS K. N., SEITZ S. M.: A theory of shape by space carving. *International journal of computer vision 38* (2000), 199–218. 2

[LGZL*20] LIU L., GU J., ZAW LIN K., CHUA T.-S., THEOBALT C.: Neural sparse voxel fields. *Advances in Neural Information Processing Systems 33* (2020), 15651–15663. 1, 2, 6

[LH96] LEVO M., HANRAHAN P.: Light field rendering. 3

[LSS*19] LOMBARDI S., SIMON T., SARAGIH J., SCHWARTZ G., LEHRMANN A., SHEIKH Y.: Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751* (2019). 2

[MBRS*21] MARTIN-BRUALLA R., RADWAN N., SAJJADI M. S., BARRON J. T., DOSOVITSKIY A., DUCKWORTH D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 7210–7219. 2

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG) 41*, 4 (2022), 1–15. 1, 3, 6

[MRIdVFB23] MARTIN-RODRIGUEZ F., ISASI-DE VICENTE F., FERNANDEZ-BARCIELA M.: The 3d-dct transform: didactic experiment and possible useful tool. *arXiv preprint arXiv:2304.00299* (2023). 4

[MSOC*19] MILDENHALL B., SRINIVASAN P. P., ORTIZ-CAYON R., KALANTARI N. K., RAMAMOORTHI R., NG R., KAR A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–14. 3, 6, 7

[MST*21] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM 65*, 1 (2021), 99–106. 1, 2, 3, 4, 5, 6, 7

[NG21] NIEMEYER M., GEIGER A.: Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 11453–11464. 2

[RES*22] RUDNEV V., ELGHARIB M., SMITH W., LIU L., GOLYANIK V., THEOBALT C.: Nerf for outdoor scene relighting. In *European Conference on Computer Vision* (2022), Springer, pp. 615–631. 2

[RPLG21] REISER C., PENG S., LIAO Y., GEIGER A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 14335–14345. 1

[RSV*23] REISER C., SZELISKI R., VERBIN D., SRINIVASAN P., MILDENHALL B., GEIGER A., BARRON J., HEDMAN P.: Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG) 42*, 4 (2023), 1–12. 3

[SD99] SEITZ S. M., DYER C. R.: Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision 35* (1999), 151–173. 2

[SDZ*21] SRINIVASAN P. P., DENG B., ZHANG X., TANCIK M., MILDENHALL B., BARRON J. T.: Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 7495–7504. 2

[SSC22] SUN C., SUN M., CHEN H.-T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 5459–5469. 1, 2, 3, 4, 6

[TCY*22] TANCIK M., CASSER V., YAN X., PRADHAN S., MILDENHALL B., SRINIVASAN P. P., BARRON J. T., KRETZSCHMAR H.: Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 8248–8258. 2

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing 13*, 4 (2004), 600–612. 7

[WMG14] WAECHTER M., MOEHRLE N., GOESELE M.: Let there be color! large-scale texturing of 3d reconstructions. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13* (2014), Springer, pp. 836–850. 2

[WRH*22] WANG H., REN J., HUANG Z., OLSZEWSKI K., CHAI M., FU Y., TULYAKOV S.: R2l: Distilling neural radiance field to neural light field for efficient novel view synthesis. In *European Conference on Computer Vision* (2022), Springer, pp. 612–629. 3

[WZL*18] WANG N., ZHANG Y., LI Z., FU Y., LIU W., JIANG Y.-G.: Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)* (2018), pp. 52–67. 2

[YLL*20] YAO Y., LUO Z., LI S., ZHANG J., REN Y., ZHOU L., FANG T., QUAN L.: Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 1790–1799. 6

[YLT*21] YU A., LI R., TANCIK M., LI H., NG R., KANAZAWA A.: Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 5752–5761. 1

[YSHWSH16] YÜCER K., SORKINE-HORNUNG A., WANG O., SORKINE-HORNUNG O.: Efficient 3d object segmentation from densely sampled light fields with applications to 3d reconstruction. *ACM Transactions on Graphics (TOG) 35*, 3 (2016), 1–15. 1, 6

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 586–595. 7

[ZRSK20] ZHANG K., RIEGLER G., SNAVELY N., KOLTUN V.: Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492* (2020). 2