IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2024–07–26.
Submitted on 2024–05–30, accepted on 2024–06–22.
ISSN 2105–1232 © 2024 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2024.553

Accelerating NeRF with the Visual Hull

Roger Marí

Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, Gif-sur-Yvette, France
roger.mari@ens-paris-saclay.fr

Communicated by Pablo Musé *Demo edited by* Roger Marí

Abstract

Neural rendering methods for learning the appearance and geometry of 3D scenes have gained tremendous popularity since 2020. In this field, NeRF or Neural Radiance Fields is the best-known methodology. Given a collection of multi-view images and their camera models, NeRF optimizes a neural network to learn the color and scene geometry that render the input images according to classical volumetric rendering techniques. NeRF operates in a self-supervised manner and provides a remarkable level of detail, but the time-consuming optimization process remains a major limitation. This paper reviews the Voxel-Accelerated NeRF (VaxNeRF), a simple acceleration strategy for NeRF proposed in 2021. VaxNeRF reduces the number of point queries required in training and inference time by considering only the region of space corresponding to the visual hull, i.e., the maximum volume compatible with the object silhouettes given by the multi-view collection. VaxNeRF requires only coarse foreground-background segmentation masks and minimal changes to the original NeRF code to improve speed by a factor of 2-8, without any performance degradation.

Source Code

The source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the README file of the archive. The authors' original method implementation is available here[2].
This is an MLBriefs article, the source code has not been reviewed!

Keywords: neural rendering; NeRF; visual hull; multi-view 3D reconstruction

[1] https://doi.org/10.5201/ipol.2024.553
[2] https://github.com/naruya/VaxNeRF

1 Introduction

The modeling of 3D scenes from image collections is a long standing problem in computer vision. Photogrammetric 3D models obtained from classic structure from motion or dense matching tools (e.g., COLMAP [39], Bundler [40], OpenMVG [31], PMVS [10]) are usually represented as colored point clouds or textured meshes. These well-known formats are easy to manipulate, but provide a discretized 3D representation limited to a given resolution.

Neural rendering is a new and rapidly evolving field in 3D modeling from multiple views that combines machine learning methods with physical knowledge from computer graphics [46]. In 2020, Mildenhall et al. [30] introduced the neural rendering approach known as NeRF (or Neural Radiance Field) to simultaneously address novel view synthesis and 3D reconstruction in controlled multi-view acquisitions. This led to a growing number of variants for all kinds of applications. For instance, NeRF has been extended for in-the-wild photo collections [26, 4], high dynamic range rendering [29], controllable relighting [41, 25], data compression [1, 43], super-resolution [48], camera calibration [22, 50], street view navigation [44, 38], digital elevation modeling [7, 24, 25], animatable avatars [11], style transfer [5, 16] or text-to-3D diffusion models [34].

NeRF achieves a finer representation of 3D objects with respect to classical methods by learning them as a continuous function or field $\mathcal{F}$. The radiance field $\mathcal{F}$ is parameterized using a neural network of fully-connected layers, also known as a multi-layer perceptron (MLP). $\mathcal{F}$ is learned in a self-supervised and multi-view manner, by training the MLP to produce realistic renderings from different viewpoints. In practice, this is done by casting rays of 3D points from the available images and using them as input to the MLP. The MLP outputs are then processed using differentiable volume rendering techniques to render the color of each ray. At each training iteration, the network parameters are optimized so that the rendered colors according to $\mathcal{F}$ are consistent with the actual colors observed in the images.

One of the main limitations of NeRF is the slow training and inference speed of its original and simplest form [30]. In 2021, Kondo et al. [18] proposed the Voxel-Accelerated NeRF (VaxNeRF) to address this problem in multi-view image collections of 3D objects. This variant is striking for its simplicity and efficiency: it only requires coarse foreground-background segmentation masks and minimal changes to the original NeRF code to improve speed by a factor of 2-8, without any performance degradation. Foreground-background masks delimit the silhouette of the object and are used to construct a visual hull, a classic concept of multi-view 3D reconstruction, which restricts the search space and thus the number of entry points. Figure 1 illustrates this idea. Nowadays image segmentation algorithms enable fast and cheap automatic foreground-background annotation, making VaxNeRF potentially interesting for applications that work with 3D objects of complex shapes, such as toys, industrial components or video game items.

This paper reviews the VaxNeRF methodology. First, Section 2 reviews the fundamentals of neural radiance fields and the state of the art of NeRF accelerations. Section 3 delves into the theoretical and implementation details of VaxNeRF. Section 4 presents an experimental evaluation of NeRF and VaxNeRF. Conclusions and ideas for future work are drawn in Section 5.

2 Related Work

The advantages of NeRF for multi-view 3D modeling are manifold. NeRF is capable of capturing fine small-scale details using a simple and fully multi-view logic, which is founded on the physical model of light transport described in Section 2.1. The NeRF solution is unique and optimal for each specific scene and does not require any supervision-oriented labeling.

Despite its appealing advantages, NeRF also has a number of weaknesses. It usually requires

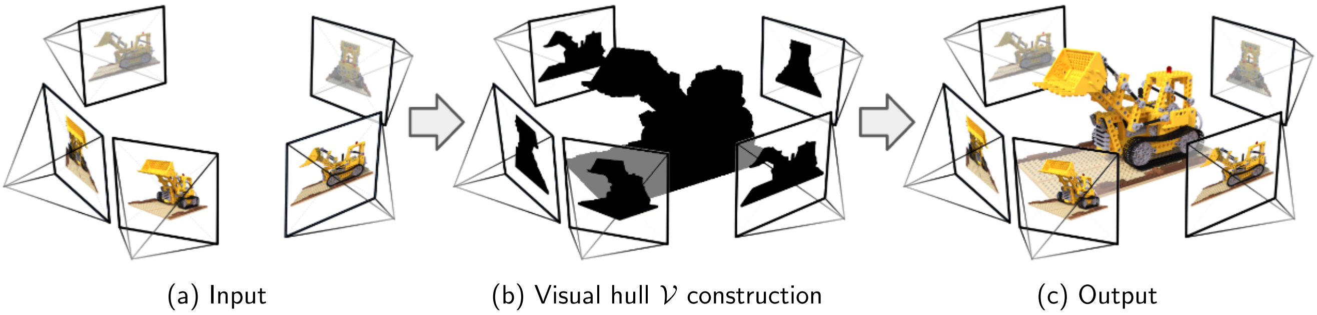


Figure 1: Illustration of the VaxNeRF methodology. Given a collection of input views of an object, as in (a), the visual hull $\mathcal{V}$ is constructed using the object silhouette masks. $\mathcal{V}$ corresponds to the black area of the 3D space in (b) and is used to reduce the volume that is visited to learn a NeRF representation of the scene. NeRF models the 3D geometry and appearance of the scene, as in (c). In contrast to VaxNeRF, the original NeRF approach aims to directly derive (c) from (a).

at least a few tens of input views and cannot generalize across different scenes [52, 3]. The input images must be geometrically calibrated [22, 28] and radiometrically consistent [26]. The illumination conditions must also be invariant between different views [41, 7], as well as the scene geometry [35, 33]. On top of that, the original NeRF [30] optimization time usually takes several hours to reach convergence, potentially days. VaxNeRF primarily addresses the latter limitation. Other concurrent work that also aims to accelerate NeRF is reviewed in Section 2.2.

2.1 NeRF in a Nutshell

A NeRF [30] is an MLP that learns a continuous function $\mathcal{F}$ that models the geometry and appearance of a 3D scene. Given a 3D point $\mathbf{x} \in \mathbb{R}^3$ of the scene and a viewing direction $\mathbf{d} \in \mathbb{R}^2$, $\mathcal{F}$ predicts the emitted RGB color $\mathbf{c} \in [0, 1]^3$ and a scalar volume density $\sigma \in [0, \infty)$, i.e.,

$$\mathcal{F} : (\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma). \tag{1}$$

The volume density σ defines the geometry of the scene and depends only on the spatial coordinates $\mathbf{x}$, while the color $\mathbf{c}$ also depends on the viewing direction $\mathbf{d}$ to recreate non-Lambertian reflectance.

The MLP is trained to render the pixel colors observed in a set of input images. For this purpose, each 2D pixel location is back-projected into 3D space by casting the corresponding camera ray $\mathbf{r}$. The network is optimized by minimizing the mean squared error (MSE) between the rendered color predicted for each ray $\mathbf{r}$, denoted by $\mathbf{c}(\mathbf{r})$, and the real color of the pixel, denoted by $\mathbf{c}_{\mathrm{GT}}(\mathbf{r})$:

$$\sum_{\mathbf{r}\in\mathcal{R}} \|\mathbf{c}(\mathbf{r}) - \mathbf{c}_{\mathrm{GT}}(\mathbf{r})\|_2^2, \tag{2}$$

where $\mathcal{R}$ is the set of rays selected at each optimization step.

Each ray $\mathbf{r}$ originates at the camera center $\mathbf{o}$ and intersects the associated pixel following a direction vector $\mathbf{d}$. In practice, $\mathbf{r}$ is discretized into N 3D points $\{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i = \mathbf{o} + t_i\mathbf{d}$ and t_i is a scalar within the depth limits of the scene. The rendered color $\mathbf{c}(\mathbf{r})$ is obtained using a simple differentiable volume rendering operation [27], i.e.,

$$\mathbf{c}(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i. \tag{3}$$

In (3), the physical contribution of the color $\mathbf{c}_i$ predicted by $\mathcal{F}$ at the i-th point of $\mathbf{r}$ is consistent with the geometry of the scene thanks to the transmittance and opacity coefficients, T_i and α_i respectively, which are defined by the volume density σ_j, $j = 1, \ldots, i$.

$$\alpha_i = 1 - \exp(-\sigma_i(t_{i+1} - t_i)), \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)\,. \tag{4}$$

According to (4), higher σ_i implies higher opacity α_i, meaning that $\mathbf{x}_i$ is a solid and non-transparent point. Occlusions are handled by the transmittance T_i, that only lets $\mathbf{x}_i$ contribute to the rendered color (3) if it is not preceded by previous non-transparent points along the ray $\mathbf{r}$.

Similarly to (3), the observed depth $d(\mathbf{r})$ in the direction of a ray $\mathbf{r}$ can be rendered as

$$d(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i t_i. \tag{5}$$

2.2 NeRF Accelerations

The acceleration of NeRF is a hot research topic. The first acceleration methods proposed in this field (e.g., FastNeRF [12], SNeRG [14], NSVF [23] or PlenOctrees [51]) pre-computed NeRF-like MLP models without major changes or gains in the time-consuming training strategy, but showed that it was possible to deploy the result in different data structures to support real-time inference. E.g., using a sparse voxel octree structure, the rendering of novel views can be accelerated by skipping voxels containing no relevant scene content [23, 51].

To reduce training time, other methods have explored subdividing the scene into multiple smaller MLPs. KiloNeRF [37] and Recursive-NeRF [49] introduce regular uniform and hierarchical subdivisions, respectively, and reduce training time by a few hours. Similarly, DeRF [36] subdivides the scene using a non-regular Voronoi learnable decomposition and assigns an independent MLP to each region of the space. EfficientNeRF [15] does not use multiple MLPs, but incorporates a regular grid of voxels that is updated during training to progressively reduce the number of input points in areas with low volume density σ.

The latest works for accelerating NeRF propose to reduce the size of the MLP or even discard it in favor of voxel grids that cache complex scene information (not only σ) and can be interpolated for a continuous representation. This is the case of DVGO [42] or Plenoxels [8], which optimize vectors of neural features or spherical harmonics, respectively, associated with each voxel. These methods can achieve a visual quality similar to that of a conventional NeRF while reducing the training time to the order of minutes, at the cost of higher memory requirements. Instant-NGP [32] follows a similar philosophy but a multi-resolution hash table of trainable feature vectors is used instead of a voxel grid, for efficient encoding and high compactness. TenorRF [2] is one of the latest works in the literature and explores tensor decomposition techniques to retain the best of both worlds, allowing fast processing and strong compression.

In addition to these new methodologies, the emergence of improved implementations and libraries that provide greater efficiency for neural rendering is also making a critical impact on the acceleration of NeRF (e.g., JaxNeRF [6], NerfAcc [21], Nerfstudio [45]).

3 VaxNeRF Methodology

One of the main reasons that slows down the NeRF optimization process is the need to sample points over the entire space containing the 3D scene. The original NeRF methodology addresses this problem using two different models, one coarse-scale MLP and one fine-scale MLP. For each input ray, N_{coarse} uniformly distributed points are initially sampled and processed by the coarse MLP.

The output of the coarse MLP is then used to sample N_{fine} points per ray using inverse transform sampling based on the distribution of weights $\{w_i\}_{i=1,\ldots,N}$ of the N_{coarse} samples. The weight of the i-th point of a ray is defined as $w_i = T_i \alpha_i$ according to (3).

VaxNeRF [18] proposes to improve this sampling strategy by evaluating only points inside the visual hull, denoted by $\mathcal{V}$. The color rendering operation (3) is modified as

$$\mathbf{c}(\mathbf{r}) = \sum_{i\,:\,\mathbf{x}_i \in \mathcal{V}}^{N} T_i \alpha_i \mathbf{c}_i. \tag{6}$$

where $\mathbf{x}_i$ is the i-th point of a ray $\mathbf{r}$.

The computation of the visual hull is a classic technique of multi-view 3D reconstruction that requires foreground-background segmentation masks and the camera models associated with the input views [20]. The visual hull is obtained from the intersection of the set of foreground silhouettes back-projected into the 3D space [9], as illustrated in Figure 2. The corresponding pseudocode is given in Algorithm 1. The interest of the visual hull is that it can be seen as the least compromised 3D reconstruction of an object (in the foreground) observed by multiple cameras: it extracts a volume containing 100% of the object and can be used as a starting point for further refinement based on shape or color assumptions [19].

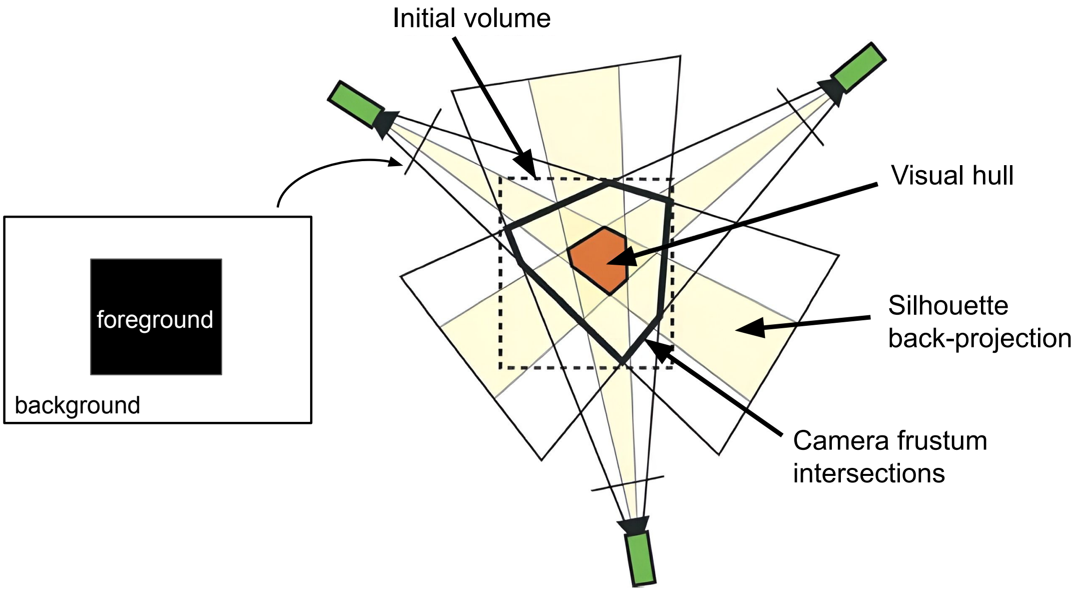


Figure 2: The visual hull is obtained from the intersection of the set of foreground silhouettes back-projected into the 3D space. Toy example using 3 input cameras, reproduced from [47].

Advantages of VaxNeRF. The authors of VaxNeRF list the following advantages with respect to the original NeRF approach:

- Integration into NeRF implementations is simple and requires minimal code modifications.
- Training is 2-8x faster with respect to the original NeRF.
- There is no loss of quality in the resulting scene representation.
- Only one MLP is needed to match or surpass the performance achieved by the original NeRF with two different MLPs dedicated to the coarse and fine stages.

Implementation details. VaxNeRF uses a regular grid of voxels to represent the visual hull $\mathcal{V}$. Specifically, a grid of $D \times D \times D$ voxels is built between the scene boundaries, using $D = 400$ by default. Algorithm 1 details the visual hull construction step, which is extremely fast and uses two auxiliary grids: $\mathcal{G}_1$ to handle the back-projection of foreground silhouettes and $\mathcal{G}_2$ to delimit the camera frustum intersections.

In the open-source implementation [17], the authors of VaxNeRF apply a moderate dilation to the foreground-background segmentation mask $\mathcal{M}_i$ of each view I_i to ensure that no part of the object is outside the silhouette boundary. It is also possible to directly apply the dilation to the voxel grid of the visual hull, as it can be seen as a 3D binary mask.

Lastly, VaxNeRF uses a single MLP with uniform point sampling along the input rays. It is suggested to use $N = 600$ or $N = 800$ point samples to discretize each ray. Although this number may seem large, it should be noted that most of these points fall outside the visual hull in practice. The rest of the method is the same as the original NeRF: the same MLP architecture, loss function and training parameters are adopted with a batch size of 1024 rays.

Algorithm 1: Visual hull construction

```
input  : $N$ input views $\{I_i\}_{i=1,\dots,N}$ with their camera models and foreground-background masks.
output : voxel grid representation of the visual hull $\mathcal{V}$.
Build a grid of voxels $\mathcal{G}_1$ with size $D \times D \times D$. The initial value of each voxel is 0.
Build a second grid of voxels $\mathcal{G}_2$ with size $D \times D \times D$. The initial value of each voxel is 0.
for each input view $I_i$ do
    Load the camera model $\mathcal{P}_i$ and the foreground-background mask $\mathcal{M}_i$ associated with $I_i$.
    Update $\mathcal{G}_1$ as follows:
        Use $\mathcal{P}_i$ to cast a set of rays $\mathcal{R}(\mathcal{M}_i)$ from the foreground pixels in $\mathcal{M}_i$.
        Add 1 to all voxels of $\mathcal{G}_1$ intersected by $\mathcal{R}(\mathcal{M}_i)$.        // See Comment 1
    Update $\mathcal{G}_2$ as follows:
        Use $\mathcal{P}_i$ to cast a set of rays $\mathcal{R}(I_i)$ from all pixels in $I_i$.
        Add 1 to all voxels of $\mathcal{G}_2$ intersected by $\mathcal{R}(I_i)$.
The visual hull $\mathcal{V}$ corresponds to all voxel positions where $\mathcal{G}_1 \geq \mathcal{G}_2$ and $\mathcal{G}_1 > 0$.

Comment 1: The set of voxels intersected by a set of rays $\mathcal{R}$ is obtained by discretizing $\mathcal{R}$ into
$D$ points per ray within the depth limits and marking each voxel containing at least one of the points.
```

4 Experiments

The authors of VaxNeRF evaluate their method using the *Synthetic-NeRF* [30] and *Synthetic-NSVF* [23] datasets. Their results are compared with the original NeRF [30] and the concurrent acceleration techniques NSVF [23] and PlenOctrees [51] mentioned in Section 2.2. Quantitative results indicate that VaxNeRF consistently achieves the highest mean PSNR for the rendered views. It also offers the fastest training time among the considered variants.

This section reproduces from scratch some of the experiments proposed in VaxNeRF, discusses the results, and details step by step how to perform novel view synthesis of a learned scene from any input viewpoint. Each input viewpoint is parameterized using a 3-valued vector $\mathbf{v}$ with the azimuth θ and elevation ϕ angles and a radial distance r to the scene. The resulting vector $\mathbf{v} = (\theta, \phi, r)$ indicates the camera position, in spherical coordinates, with respect to the target object located at the origin $\mathbf{o} = (0, 0, 0)$.

4.1 Data Description and Results

Each scene in the *Synthetic-NeRF* and *Synthetic-NSVF* contains 100 RGB views of 800×800 pixels, offering a 360 degrees coverage of a target object. *Synthetic-NSVF* objects have more complex geometry and lighting effects [23]. Foreground-background segmentation masks are obtained instantly, either because the alpha channel is available or because the background is homogeneous (white).

We used the JAX implementation of VaxNeRF [17] to train some scenes from scratch for 512000 steps. We chose the *lego* and *ship* scenes from *Synthetic-NeRF*, and *palace* and *steamtrain* from *Synthetic-NSVF* [23]. For comparison, the original JAX implementation of NeRF was also trained from scratch [6]. As shown in Figure 3, VaxNeRF trains 2-8x faster and achieves small improvements in terms of PSNR, confirming the authors' claims. The speed gain factor depends on the size of the visual hull with respect to the total volume of the scene (e.g., 4 or less for larger objects such as *palace* or *ship*, in contrast to almost 8 for finer objects such as *steamtrain*). Only in the *steamtrain* scene VaxNeRF appears to show slightly lower PSNR with respect to NeRF after 512000 training steps, but convergence had not yet been reached. For a qualitative inspection of the results, Figure 4 shows the same rendered view after 2000 and 512000 training steps for both VaxNeRF and NeRF. Note that the construction of the visual hull in VaxNeRF (Algorithm 1) takes around 10 seconds or less for this kind of input photo collections. However, storing the visual hull in a $400 \times 400 \times 400$ voxel grid requires ~ 70 MB, while a basic NeRF MLP requires ~ 7 MB.

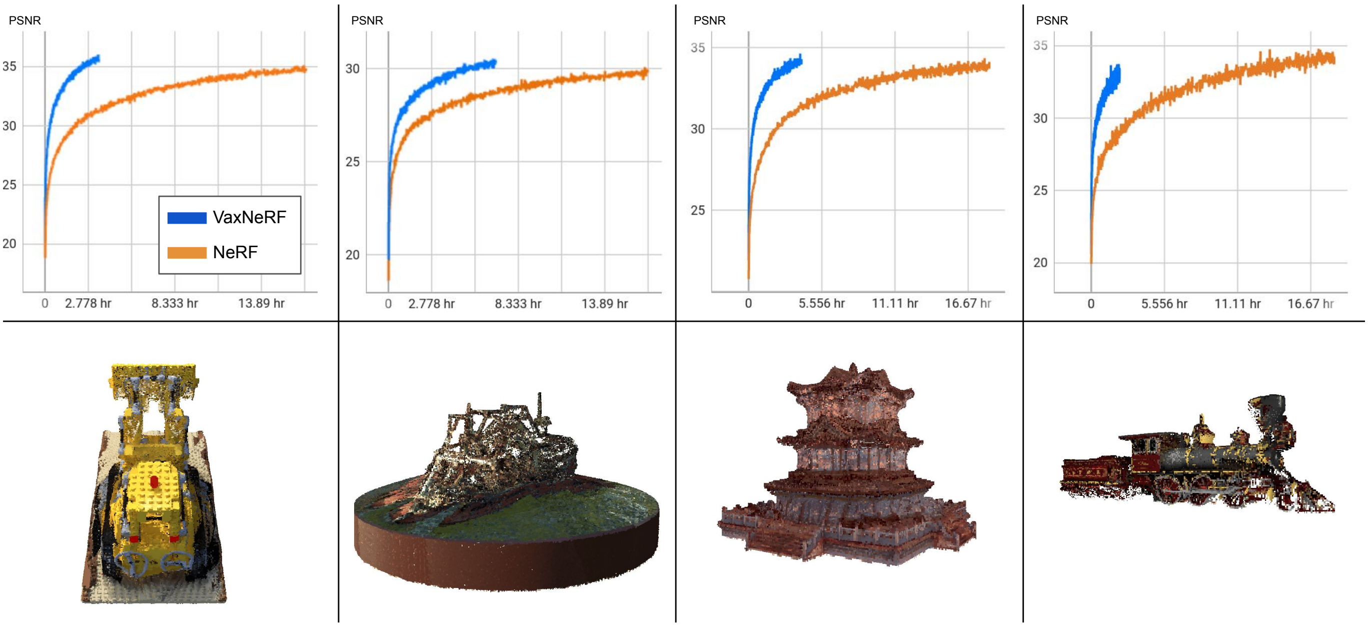
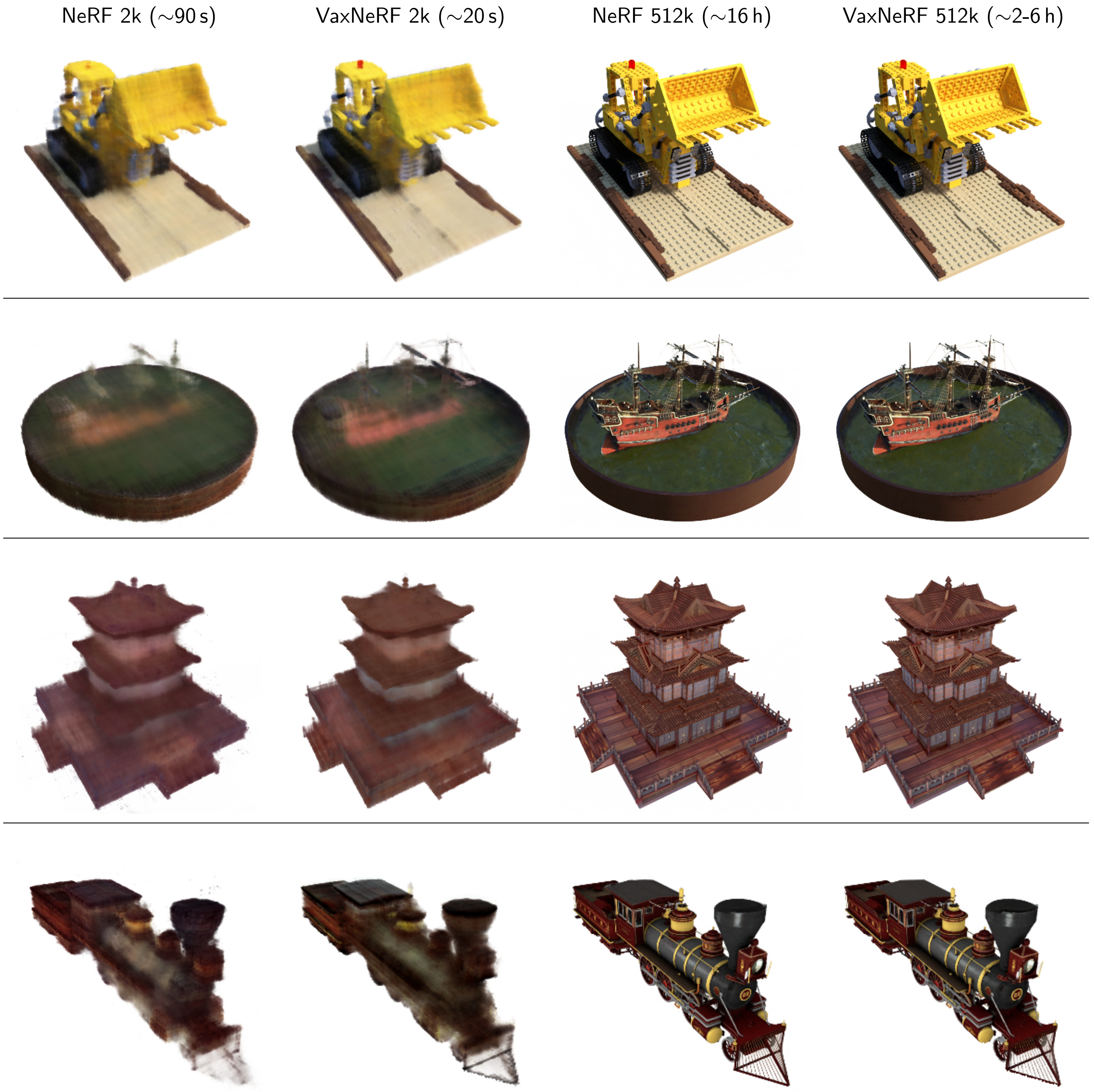


Figure 3: Upper row: PSNR evolution of VaxNeRF and NeRF for the first 512000 training steps, where the x-axis represents the number of training hours (on a 12 GB GPU). The sampling configuration was $N = 600$ points per ray for VaxNeRF and $N_{\text{coarse}} = 64$, $N_{\text{fine}} = 128$ for NeRF, as in [30]. The rest of parameters were set to the default values. Bottom row: colored visualization of the visual hull employed in VaxNeRF. Left to right: *lego*, *ship*, *palace*, *steamtrain*.

4.2 Breaking Down Ray Sampling for Novel View Synthesis

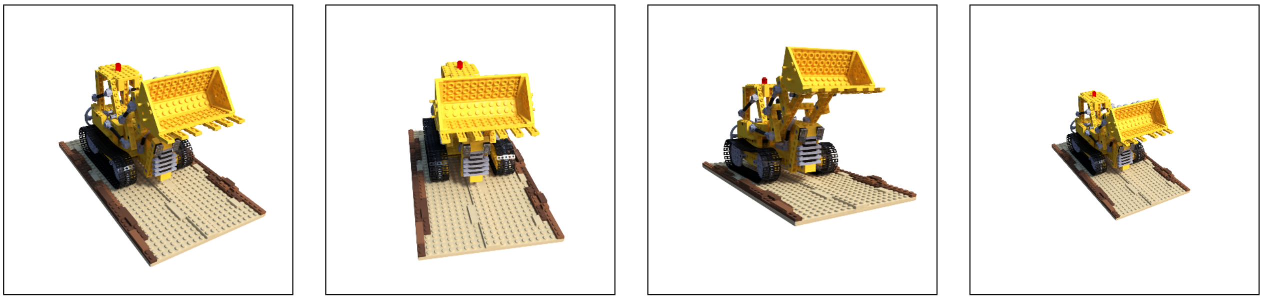
The online demo[3] associated with this article can be used to run VaxNeRF or NeRF to render new views from any viewpoint, as shown in Figure 5. This section details how to render novel views, which were not seen at training time, using a pre-trained NeRF variant. To do this, it is necessary to build the pinhole camera model associated with the novel view.

[3] https://doi.org/10.5201/ipol.2024.553



NeRF 2k (∼90 s) VaxNeRF 2k (∼20 s) NeRF 512k (∼16 h) VaxNeRF 512k (∼2-6 h)

Figure 4: Examples of the same view rendered at different training steps. VaxNeRF requires significantly less time to obtain similar or better results than NeRF. In these examples the input viewpoint is defined by an azimuth of 30 degrees, an elevation of 30 degrees and a radial distance of 1.



$(\theta, \phi, r) = (30, 30, 1)$ $(\theta, \phi, r) = (10, 30, 1)$ $(\theta, \phi, r) = (30, 10, 1)$ $(\theta, \phi, r) = (30, 30, 1.5)$

Figure 5: Novel view synthesis with VaxNeRF using different azimuth θ and elevation ϕ angles, in degrees, and radial distance r to the scene. See Section 4.2 for a detailed explanation of the method.

The first step is defining the 4×4 matrix of external parameters, denoted by M_{pose}, that characterizes the position and orientation of the camera in the 3D space. M_{pose} is obtained from 3-input values: θ, ϕ, r. The azimuth angle θ and elevation angle ϕ define the camera orientation, while the radius r indicates the distance to the observed scene

$$M_{pose} = R_\theta R_\phi T_r = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{7}$$

where R_θ defines a rotation around the y-axis according to the azimuth angle θ, R_ϕ defines a rotation around the x-axis according to the elevation angle ϕ, and T_r is a translation given by the radius r.

The second step is defining the 3×3 matrix of internal parameters, denoted by K, that characterizes the internal configuration of the camera

$$K = \begin{pmatrix} f & 0 & w/2 \\ 0 & f & h/2 \\ 0 & 0 & 1 \end{pmatrix}, \tag{8}$$

where w and h are the width and height of the novel view and f is the focal length. For simplicity, we assume that f is the same for both axis and the principal point is at the center of the image $(w/2, h/2)$. Based on K, the pinhole camera model states that a 2D pixel $\mathbf{x} = (x, y)^\top$ of the image plane corresponds to a 3D point $\mathbf{X}_{cam} = (X, Y, Z)^\top$ such that

$$x = fX/Z + w/2, \quad y = fY/Z + h/2, \tag{9}$$

as illustrated in Figure 6.

To render the view corresponding to the pinhole camera resulting from M_{pose} (7) and K (8), NeRF needs to cast the ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ that originates at the center of projection of the camera and intersects each pixel. Solving for X/Z and Y/Z in (9) yields the direction vector of the ray $\mathbf{r}$,

$$\mathbf{d}_{cam} = (\ (x - w/2)/f,\ (y - h/2)/f,\ -1)^\top. \tag{10}$$

The subscript *cam* of $\mathbf{d}_{cam}$ denotes that vector (10) works with 3D point coordinates expressed in the local camera coordinate frame, where the center of projection of the pinhole camera is at the origin [13]. The direction vector $\mathbf{d}$ can be expressed in the global world coordinate frame using the matrix of external parameters M_{pose},

$$\mathbf{d} = M_{pose_{3\times3}} \mathbf{d}_{cam}, \tag{11}$$

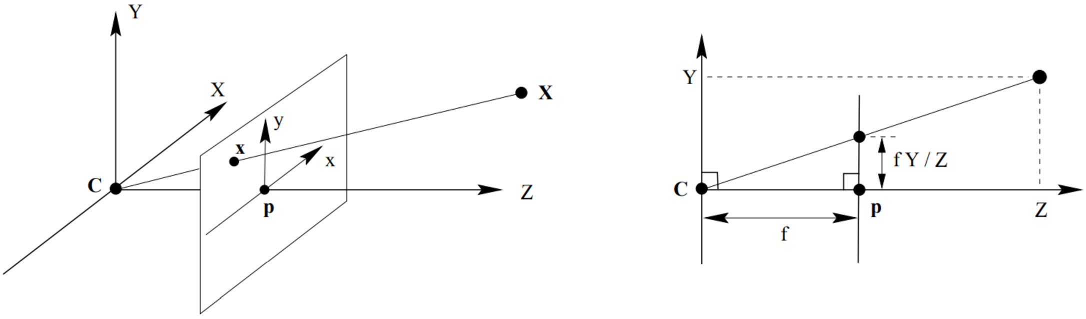


Figure 6: Pinhole camera geometry, reproduced from [13]. $\mathbf{C}$ is the camera center of projection, f is the focal length and $\mathbf{p}$ is the principal point. $\mathbf{X}$ is a 3D point that projects to a pixel $\mathbf{x}$ in the image plane.

where $M_{pose_{3\times3}}$ denotes the upper-left 3×3 subset of M_{pose}.

The origin $\mathbf{o}$ of the ray $\mathbf{r}$ is the center of projection of the camera expressed in the world coordinate frame, which corresponds to the first 3 values of the last column of M_{pose}.

In practice, for simplicity, the online demo does not use the absolute radial distance r, but treats r as a relative factor that multiplies the average radial distance of the input images of each scene.

5 Conclusion

This paper reviewed the VaxNeRF [18] method for accelerating the optimization of neural radiance fields (NeRFs), which are one of the most popular state-of-the-art methods for modeling the appearance and geometry of 3D scenes from multi-view image collections. VaxNeRF reduces the number of input points required by a conventional NeRF by considering only the region of the space corresponding to the visual hull, i.e., the maximum volume compatible with the object silhouettes observed in the input views.

This review reproduced from scratch some of the experiments proposed in VaxNeRF and confirmed the advantages listed by the original authors. Most notably, VaxNeRF reduces the optimization time by a factor of 2-8 (depending on the size of the visual hull) without loss of quality. However, the method is also subject to limitations. VaxNeRF can be extremely useful in large collections of synthetic 360-degree photos, but the benefits for real forward-facing scenes can be expected to be more modest. This will depend on the number of available views and the quality of the segmentation masks, which are key elements to obtain a well-fitted visual hull. Another non-negligible weakness is that VaxNeRF requires additional memory cost to store the voxel grid that encodes the visual hull.

We can conclude with a remark on the terminology: why call this method *Voxel-Accelerated NeRF*? The authors of VaxNeRF used a mask of voxels to encode the visual hull, but the visual hull could be represented in a different format and the idea would be of equal interest. For this reason, another title such as *Visual Hull-Accelerated NeRF* might have been more descriptive of the true nature of the method.

References

[1] T. Bird, J. Ballé, S. Singh, and P. A. Chou, *3D Scene Compression Through Entropy Penalized Neural Representation Functions*, in Picture Coding Symposium (PCS), 2021, pp. 1–5. https://doi.org/10.1109/PCS50896.2021.9477505.

[2] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, *TensoRF: Tensorial Radiance Fields*, in European Conference on Computer Vision (ECCV), 2022, pp. 333–350. https://doi.org/10.1007/978-3-031-19824-3_20.

[3] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, *MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 14104–14113. https://doi.org/10.1109/ICCV48922.2021.01386.

[4] X. Chen, Q. Zhang, X. Li, Y. Chen, Y. Feng, X. Wang, and J. Wang, *Hallucinated Neural Radiance Fields in the Wild*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 12933–12942. https://doi.org/10.1109/CVPR52688.2022.01260.

[5] P.-Z. Chiang, M.-S. Tsai, H.-Y. Tseng, W.-S. Lai, and W.-C. Chiu, *Stylizing 3D Scene Via Implicit Representation and HyperNetwork*, in IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022, pp. 215–224. https://doi.org/10.1109/WACV51458.2022.00029.

[6] B. Deng, J. T. Barron, and P. P. Srinivasan, *JaxNeRF: An Efficient JAX Implementation of NeRF*, 2020. https://github.com/google-research/google-research/tree/master/jaxnerf.

[7] D. Derksen and D. Izzo, *Shadow Neural Radiance Fields for Multi-View Satellite Photogrammetry*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2021, pp. 1152–1161. https://doi.org/10.1109/CVPRW53098.2021.00126.

[8] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, *Plenoxels: Radiance Fields Without Neural Networks*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 5491–5500. https://doi.org/10.1109/CVPR52688.2022.00542.

[9] Y. Furukawa and J. Ponce, *Carved Visual Hulls for Image-Based Modeling*, in European Conference on Computer Vision (ECCV), 2006, pp. 564–577. https://doi.org/10.1007/11744023_44.

[10] ——, *Accurate, Dense, and Robust Multiview Stereopsis*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32 (2010), pp. 1362–1376. https://doi.org/10.1109/TPAMI.2009.161.

[11] G. Gafni, J. Thies, M. Zollhofer, and M. Niessner, *Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 8645–8654. https://doi.org/10.1109/CVPR46437.2021.00854.

[12] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin, *FastNeRF: High-Fidelity Neural Rendering at 200FPS*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 14326–14335. https://doi.org/10.1109/ICCV48922.2021.01408.

[13] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Second Ed., 2004. https://doi.org/10.1017/CBO9780511811685.

[14] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, *Baking Neural Radiance Fields for Real-Time View Synthesis*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 5855–5864. https://doi.org/10.1109/ICCV48922.2021.00582.

[15] T. Hu, S. Liu, Y. Chen, T. Shen, and J. Jia, *EfficientNeRF -- Efficient Neural Radiance Fields*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 12892–12901. https://doi.org/10.1109/CVPR52688.2022.01256.

[16] Y.-H. Huang, Y. He, Y.-J. Yuan, Y.-K. Lai, and L. Gao, *StylizedNeRF: Consistent 3D Scene Stylization as Stylized NeRF Via 2D-3D Mutual Learning*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 18321–18331. https://doi.org/10.1109/CVPR52688.2022.01780.

[17] N. Kondo, *VaxNeRF*, 2021. https://github.com/naruya/VaxNeRF.

[18] N. Kondo, Y. Ikeda, A. Tagliasacchi, Y. Matsuo, Y. Ochiai, and S. S. Gu, *VaxNeRF: Revisiting the Classic for Voxel-Accelerated Neural Radiance Field*, ArXiv Preprint ArXiv:2111.13112, (2021). https://arxiv.org/abs/2111.13112.

[19] K. N. Kutulakos and S. M. Seitz, *A Theory of Shape by Space Carving*, IEEE International Conference on Computer Vision (ICCV), 1 (1999), pp. 307–314. https://doi.org/10.1109/ICCV.1999.791235.

[20] A. Laurentini, *The Visual Hull Concept for Silhouette-Based Image Understanding*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 16 (1994), pp. 150–162. https://doi.org/10.1109/34.273735.

[21] R. Li, M. Tancik, and A. Kanazawa, *NerfAcc: A General NeRF Acceleration Toolbox*, ArXiv Preprint ArXiv:2210.04847, (2022). https://arxiv.org/abs/2210.04847.

[22] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, *BARF: Bundle-Adjusting Neural Radiance Fields*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 5721–5731. https://doi.org/10.1109/ICCV48922.2021.00569.

[23] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, *Neural Sparse Voxel Fields*, Advances in Neural Information Processing Systems, 33 (2020), pp. 15651–15663. https://dl.acm.org/doi/10.5555/3495724.3497037.

[24] R. Marí, G. Facciolo, and T. Ehret, *Sat-NeRF: Learning Multi-View Satellite Photogrammetry With Transient Objects and Shadow Modeling Using RPC Cameras*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2022, pp. 1310–1320. https://doi.org/10.1109/CVPRW56347.2022.00137.

[25] ——, *Multi-Date Earth Observation NeRF: The Detail Is in the Shadows*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2023, pp. 2035–2045. https://doi.org/10.1109/CVPRW59228.2023.00197.

[26] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, *NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 7206–7215. https://doi.org/10.1109/CVPR46437.2021.00713.

[27] N. Max, *Optical Models for Direct Volume Rendering*, IEEE Transactions on Visualization and Computer Graphics, 1 (1995), pp. 99–108. https://doi.org/10.1109/2945.468400.

[28] Q. Meng, A. Chen, H. Luo, M. Wu, H. Su, L. Xu, X. He, and J. Yu, *GNeRF: GAN-Based Neural Radiance Field Without Posed Camera*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 6331–6341. https://doi.org/10.1109/ICCV48922.2021.00629.

[29] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron, *NeRF in the Dark: High Dynamic Range View Synthesis from Noisy Raw Images*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 16169–16178. https://doi.org/10.1109/CVPR52688.2022.01571.

[30] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*, in European Conference on Computer Vision (ECCV), 2020, pp. 405–421. https://doi.org/10.1007/978-3-030-58452-8_24.

[31] P. Moulon, P. Monasse, R. Perrot, and R. Marlet, *OpenMVG: Open Multiple View Geometry*, in Reproducible Research in Pattern Recognition, Springer, 2017, pp. 60–74. https://doi.org/10.1007/978-3-319-56414-2_5.

[32] T. Müller, A. Evans, C. Schied, and A. Keller, *Instant Neural Graphics Primitives with a Multiresolution Hash Encoding*, ACM Transactions on Graphics, 41 (2022). https://doi.org/10.1145/3528223.3530127.

[33] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, *Nerfies: Deformable Neural Radiance Fields*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 5845–5854. https://doi.org/10.1109/ICCV48922.2021.00581.

[34] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, *DreamFusion: Text-To-3D Using 2D Diffusion*, ArXiv Preprint ArXiv:2209.14988, (2022). https://arxiv.org/abs/2209.14988.

[35] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, *D-NeRF: Neural Radiance Fields for Dynamic Scenes*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 10313–10322. https://doi.org/10.1109/CVPR46437.2021.01018.

[36] D. Rebain, W. Jiang, S. Yazdani, K. Li, K. M. Yi, and A. Tagliasacchi, *DeRF: Decomposed Radiance Fields*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 14148–14156. https://doi.org/10.1109/CVPR46437.2021.01393.

[37] C. Reiser, S. Peng, Y. Liao, and A. Geiger, *KiloNeRF: Speeding Up Neural Radiance Fields with Thousands of Tiny MLPs*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 14315–14325. https://doi.org/10.1109/ICCV48922.2021.01407.

[38] K. Rematas, A. Liu, P. P. Srinivasan, J. T. Barron, A. Tagliasacchi, T. Funkhouser, and V. Ferrari, *Urban Radiance Fields*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 12922–12932. https://doi.org/10.1109/CVPR52688.2022.01259.

[39] J. L. Schönberger and J.-M. Frahm, *Structure-From-Motion Revisited*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4104–4113. https://doi.org/10.1109/CVPR.2016.445.

[40] N. Snavely, S. M. Seitz, and R. Szeliski, *Photo Tourism: Exploring Photo Collections in 3D*, ACM Transactions on Graphics, 25 (2006), pp. 835–846. https://doi.org/10.1145/1141911.1141964.

[41] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron, *NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 7491–7500. https://doi.org/10.1109/CVPR46437.2021.00741.

[42] C. Sun, M. Sun, and H.-T. Chen, *Direct Voxel Grid Optimization: Super-Fast Convergence for Radiance Fields Reconstruction*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 5449–5459. https://doi.org/10.1109/CVPR52688.2022.00538.

[43] T. Takikawa, A. Evans, J. Tremblay, T. Müller, M. McGuire, A. Jacobson, and S. Fidler, *Variable Bitrate Neural Fields*, in ACM SIGGRAPH, 2022, pp. 1–9. https://doi.org/10.1145/3528233.3530727.

[44] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, *Block-NeRF: Scalable Large Scene Neural View Synthesis*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 8238–8248. https://doi.org/10.1109/CVPR52688.2022.00807.

[45] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa, *Nerfstudio: A Modular Framework for Neural Radiance Field Development*, in ACM SIGGRAPH, 2023. https://github.com/nerfstudio-project/nerfstudio.

[46] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Niessner, R. Pandey, S. Fanello, G. Wetzstein, J. Zhu, C. Theobalt, M. Agrawala, D. Goldman, and M. Zolhöfer, *State of the Art on Neural Rendering*, in Computer Graphics Forum, vol. 39, 2020, pp. 701–727. https://doi.org/10.1111/cgf.14022.

[47] M. C. V. Uriol, *Video-Based Avatar Reconstruction and Motion Capture*, PhD thesis, University of California, Irvine, 2005.

[48] C. Wang, X. Wu, Y.-C. Guo, S.-H. Zhang, Y.-W. Tai, and S.-M. Hu, *NeRF-SR: High Quality Neural Radiance Fields Using Supersampling*, in ACM International Conference on Multimedia, 2022, pp. 6445–6454. https://doi.org/10.1145/3503161.3547808.

[49] G.-W. Yang, W.-Y. Zhou, H.-Y. Peng, D. Liang, T.-J. Mu, and S.-M. Hu, *Recursive-NeRF: An Efficient and Dynamically Growing NeRF*, IEEE Transactions on Visualization and Computer Graphics, (2022), pp. 1–14. https://doi.org/10.1109/TVCG.2022.3204608.

[50] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, *INeRF: Inverting Neural Radiance Fields for Pose Estimation*, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 1323–1330. https://doi.org/10.1109/IROS51168.2021.9636708.

[51] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, *PlenOctrees for Real-Time Rendering of Neural Radiance Fields*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 5732–5741. https://doi.org/10.1109/ICCV48922.2021.00570.

[52] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, *PixelNeRF: Neural Radiance Fields from One or Few Images*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 4576–4585. https://doi.org/10.1109/CVPR46437.2021.00455.