

# ProNeRF: Learning Efficient Projection-Aware Ray Sampling for Fine-Grained Implicit Neural Radiance Fields

Juan Luis Gonzalez Bello\*, Minh-Quan Viet Bui\*, Munchurl Kim

Korea Advanced Institute of Science and Technology (KAIST), South Korea

## Abstract

Recent advances in neural rendering have shown that, albeit slow, implicit compact models can learn a scene’s geometries and view-dependent appearances from multiple views. To maintain such a small memory footprint but achieve faster inference times, recent works have adopted ‘sampler’ networks that adaptively sample a small subset of points along each ray in the implicit neural radiance fields. Although these methods achieve up to a  $10\times$  reduction in rendering time, they still suffer from considerable quality degradation compared to the vanilla NeRF. In contrast, we propose ProNeRF, which provides an optimal trade-off between memory footprint (similar to NeRF), speed (faster than HyperReel), and quality (better than  $K$ -Planes). ProNeRF is equipped with a novel projection-aware sampling (PAS) network together with a new training strategy for ray exploration and exploitation, allowing for efficient fine-grained particle sampling. Our ProNeRF yields state-of-the-art metrics, being  $15\text{-}23\times$  faster with  $0.65\text{dB}$  higher PSNR than NeRF and yielding  $0.95\text{dB}$  higher PSNR than the best published sampler-based method, HyperReel. Our exploration and exploitation training strategy allows ProNeRF to learn the full scenes’ color and density distributions while also learning efficient ray sampling focused on the highest-density regions. We provide extensive experimental results that support the effectiveness of our method on the widely adopted forward-facing and 360 datasets, LLFF and Blender, respectively.

## 1 Introduction

Neural radiance fields (NeRFs) (Mildenhall et al. 2020) have gained significant attention in the computer vision community due to their greater ability to compactly represent complex scenes’ 3D geometries and view-dependent specularity, in comparison with other implicit representations (Flynn et al. 2019; Sitzmann et al. 2020). The efficacy of NeRFs can be attributed to several key features such as: (i) the volumetric rendering technique (Drebin, Carpenter, and Hanrahan 1988), which aggregates estimated RGB-density values along rendering rays, (ii) their implicit representation by a multi-layer perception (MLP) network that incorporates positional encoding (Mildenhall et al. 2020), and (iii) their coarse-to-fine rendering strategy that enables *dense* fine-grained ray sampling for high-quality rendering.

\*These authors contributed equally.

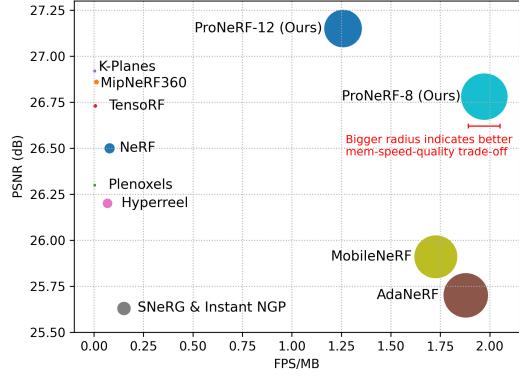


Figure 1: Performance trade-off of neural rendering (memory, speed, quality) on the LLFF dataset.

Although NeRFs offer a compact representation of 3D geometry and view-dependent effects, there is still significant room for improvement in rendering quality and inference times. To speed up the rendering times, recent trends have explored caching diffuse color estimation into an explicit voxel-based structure (Yu et al. 2021a; Hedman et al. 2021; Garbin et al. 2021; Hu et al. 2022) or leveraging texture features stored in an explicit representation such as hash girds (Müller et al. 2022), meshes (Chen et al. 2023), or 3D Gaussians (Kerbl et al. 2023). While these methods achieve SOTA results on object-centric 360 datasets, they underperform for the forward-facing scene cases and require considerably larger memory footprints than NeRF.

In a different line of work, the prior literature of (Neff et al. 2021; Piala and Clark 2021; Lin et al. 2022; Kurz et al. 2022; Attal et al. 2023) has proposed training single-pass lightweight “sampler” networks, aimed to reduce the number of ray samples required for volumetric rendering. Although fast and memory compact, previous sampler-based methods often fall short in rendering quality compared to the computationally expensive vanilla NeRF.

In contrast, our proposed method with a Projection-Aware Sampling (PAS) network and an exploration-exploitation training strategy, denoted as “ProNeRF,” greatly reduces the inference times while simultaneously achieving superior image quality and more details than the current high-quality methods (Chen et al. 2022; Sara Fridovich-Keil and Gi-

acom Meanti et al. 2023). In conjunction with its small memory footprint (as small as NeRF), our ProNeRF yields the best performance profiling (memory, speed, quality) trade-off. Our main contributions are as follows<sup>1</sup>:

- **Faster rendering times.** Our ProNeRF leverages multi-view color-to-ray projections to yield a few precise 3D query points, allowing up to  $23\times$  faster inference times than vanilla NeRF under a similar memory footprint.
- **Higher rendering quality.** Our proposed PAS and exploration-exploitation training strategy allow for *sparse* fine-grained ray sampling in an end-to-end manner, yielding rendered images with improved quality metrics compared to the implicit baseline NeRF.
- **Comprehensive experimental validation.** The robustness of ProNeRF is extensively evaluated on forward-facing and 360 object-centric multi-view datasets. Specifically, in the context of forward-facing scenes, ProNeRF establishes SOTA renders, outperforming implicit and explicit radiance fields, including NeRF, TensoRF, and  $K$ -Planes with a considerably more optimal performance profile in terms of memory, speed, and quality.

## 2 Related Work

The most relevant works concerning our proposed method focus on maintaining the compactness of implicit NeRFs while reducing the rendering times by **learning sampling networks** for efficient ray querying.

Nevertheless, other works leverage data structures for **baking radiance fields**, that is, caching diffuse color and latent view-dependent features from a pre-trained NeRF to accelerate the rendering pipelines (as in SNeRG (Hedman et al. 2021)). Similarly, Yu et al. (2021a) proposed Plenoctrees to store spatial densities and spherical harmonics (SH) coefficients for fast rendering. Subsequently, to reduce the redundant computation in empty space, Plenoxels (Fridovich-Keil et al. 2022) learns a sparse voxel grid of SH coefficients. On the other hand, Efficient-NeRF (Hu et al. 2022) presents an innovative caching representation referred to as “NeRF-tree,” enhancing caching efficiency and rendering performance. However, these approaches require a pre-trained NeRF and a considerably larger memory footprint to store their corresponding scene representations.

Explicit data structures have also been used for storing latent textures in **explicit texture radiance fields** to speed up the training and inference times. Particularly, INGP (Müller et al. 2022) proposes quickly estimating the radiance values by interpolating latent features stored in multi-scaled hash grids. Drawing inspiration from tensorial decomposition, in TensoRF, Chen et al. (2022) factorize the scene’s radiance field into multiple low-rank latent tensor components. Following a similar decomposition principle, Sara Fridovich-Keil and Giacomo Meanti et al. (2023) introduced  $K$ -Planes for multi-plane decomposition of 3D scenes. Recently, MobileNeRF (Chen et al. 2023) and 3DGS (Kerbl et al. 2023) concurrently propose merging the rasterization process with explicit meshes or 3D Gaussians for real-time

<sup>1</sup>Visit our project website at <https://kaist-viclab.github.io/pronerf-site/>

rendering. Similar to the baked radiance fields, MobileNeRF and 3DGS demonstrate the capability to achieve incredibly rapid rendering, up to several hundred frames per second. However, they demand a considerably elevated memory footprint, which might be inappropriate in resource-constrained scenarios where real-time swapping of neural radiance fields is required, such as streaming, as discussed by Kurz et al. (2022).

Inspired by the concept proposed in (Levoy and Hanrahan 1996), recent studies have also explored the learning of **neural light fields** which only require a single network evaluation for each casted ray. Light field networks such as LFNR (Suhail et al. 2022b) and GPNR (Suhail et al. 2022a) presently exhibit optimal rendering performance across diverse novel view synthesis datasets. Nevertheless, they adopt expensive computational attention operations for aggregating multi-view projected features. Additionally, it’s worth noting that similar to generalizable radiance fields (e.g., IBRNet (Wang et al. 2021), or NeuRay (Liu et al. 2022)), LFNR and GPNR necessitate the storage of all training input images for epipolar feature projection, leading to increased memory requirements. Conversely, our method, ProNeRF, leverages color-to-ray projections while guaranteeing consistent memory footprints by robustly managing a small and fixed subset of reference views for rendering any novel view in the target scene. This eliminates the necessity for nearest-neighbor projection among all available training views in each novel scene. To balance computational cost and rendering quality for neural light fields, RSEN (Attal et al. 2022) introduces a novel ray parameterization and space subdivision structure of the 3D scenes. On the other hand, R2L (Wang et al. 2022) distills a compact neural light field with a pre-trained NeRF. Although R2L achieves better inference time and quality than RSEN, it necessitates the generation of numerous pseudo-images from a pre-trained NeRF to perform exhaustive training on dense pseudo-data. This process can extend over days of optimization.

In addition to IBRNet and NeuRay, other generalizable radiance fields have also been explored in (Yu et al. 2021b; Li et al. 2021), but are less relevant to our work.

**Learning sampling networks.** In AutoInt, Lindell, Martel, and Wetzstein (2021) propose to train anti-derivative networks that describe the piece-wise color and density integrals of discrete ray segments whose distances are individually estimated by a sampler network. In DONeRF (Neff et al. 2021) and TermiNeRF (Piala and Clark 2021), the coarse NeRF in vanilla NeRF is replaced with a sampling network that learns to predict the depth of objects’ surfaces using either depth ground truth (GT) or dense depths from a pre-trained NeRF. The requirement of hard-to-obtain dense depths severely limits DONeRF and TermiNeRF for broader applications. ENeRF (Lin et al. 2022) learns to estimate the depth distribution from multi-view images in an end-to-end manner. In particular, ENeRF adopts cost-volume aggregation and 3D CNNs to enhance geometry prediction.

Instead of predicting a continuous depth distribution, AdaNeRF (Kurz et al. 2022) proposes a sampler network that maps rays to fixed and discretized distance probabilities. During test, only the samples with the highest probabilities

are fed into the shader (NeRF) network for volumetric rendering. AdaNeRF is trained in a dense-to-sparse multi-stage manner without needing a pre-trained NeRF. The shader is first trained with computationally expensive dense sampling points, where sparsification is later introduced to prune insignificant samples, and then followed by simultaneous sampling and shading network fine-tuning. In MipNeRF360, Barron et al. (2022) introduce online distillation to train the sampling network. Nevertheless, the sampler utilized in MipNeRF360 remains structured as a radiance field, necessitating a per-point forward pass. Consequently, incorporating this sampler does not yield substantial improvements in rendering latency. On the other hand, in the recent work of HyperReel, Attal et al. (2023) proposed a sampling network for learning the geometry primitives in grid-based rendering models such as TensoRF. HyperReel inherits the fast-training properties of TensoRF but also yields limited rendering quality with a considerably increased memory footprint compared to the vanilla NeRF.

Contrary to the existing literature, we present a sampler-based method, ProNeRF, that allows for fast neural rendering while substantially outperforming the implicit and explicit NeRFs quantitatively and qualitatively in reconstructing forward-facing captured scenes. The main components of ProNeRF are a novel PAS network and a new learning strategy that borrows from the reinforcement learning concepts of exploration and exploitation. Moreover, all the previous sampler-based methods require either pre-trained NeRFs (TerMiNeRF), depth GTs (DoNeRF), complex dense-ray sampling and multi-stage training strategies (AdaNeRF), or large memory footprint (HyperReel). In contrast, our proposed method can more effectively learn the neural rendering in an end-to-end manner from sparse rays, even with shorter training cycles than NeRF.

### 3 Proposed Method

Fig. 2 depicts a high-level overview of our ProNeRF, which is equipped with a projection-aware sampling (PAS) network and a shader network (a.k.a NeRF) for few-point volumetric rendering. ProNeRF performs PAS in a coarse-to-fine manner. First, for a given target ray, ProNeRF maps the ray direction and origin into coarse sampling points with the help of an MLP head ( $F_{\theta_c}$ ). By tracing lines from these sampling points into the camera centers of the reference views in the training set, ProNeRF performs a color-to-ray projection which is aggregated to the coarse sampling points and is processed in a second MLP head ( $F_{\theta_f}$ ).  $F_{\theta_f}$  then outputs the refined 3D points that are fed into the shading network ( $F_{\theta_s}$ ) for the further volumetric rendering of the ray color  $\hat{c}$ . See Section 3.2 for more details.

Training a ProNeRF as depicted in Fig. 2 is not a trivial task, as the implicit shader needs to learn the *full* color and density distributions in the scenes while the PAS network tries to predict ray points that focus on specific regions with the highest densities. Previous works, such as DoNERF, TerMiNeRF, and AdaNeRF go around this problem at the expense of requiring depth GTs, pre-trained NeRF models, or expensive dense sampling. To overcome this issue, we propose an alternating learning strategy that borrows from rein-

forcement learning which (i) allows the shading network to *explore* the scene’s rays and learn the full scene distributions and (ii) leads the PAS network to *exploit* the ray samples with the highest densities. See Section 3.3 for more details.

### 3.1 PAS-Guided Volumetric Rendering

Volumetric rendering synthesizes images by traversing the rays that originate in the target view camera center into a 3D volume of color and densities. As noted by Mildenhall et al. (2020), the continuous volumetric rendering equation (VRE) of a ray color  $c(r)$  can be efficiently approximated by alpha compositing, which is expressed as:

$$\hat{c}(r) = \sum_{i=1}^N \left( \prod_{j=1}^{i-1} 1 - \alpha_j \right) \alpha_i c_i, \quad (1)$$

where  $N$  is the total number of sampling points and  $\alpha_i$  denotes the opacity at the  $i^{th}$  sample in ray  $r$  as given by

$$\alpha_i = 1 - e^{-\sigma_i(t_{i+1}-t_i)}. \quad (2)$$

Here,  $\sigma_i$  and  $c_i$  respectively indicate the density and colors at the 3D location given by  $r(t_i)$  for the  $i^{th}$  sampling point on  $r$ . A point on  $r$  in distance  $t$  is  $r(t) = r_o + r_d t$  where  $r_o$  and  $r_d$  are the ray origin and direction, respectively.

In NeRF (Mildenhall et al. 2020), a large number of  $N$  samples along the ray is considered to precisely approximate the original integral version of the VRE. In contrast, our objective is to perform high-quality volumetric rendering with a smaller number of samples  $N_s \ll N$ . Rendering a ray with a few samples in our ProNeRF can be possible by accurately sampling the 3D particles with the highest densities along the ray. Thanks to the PAS, our ProNeRF can yield a sparse set of accurate sampling distances, denoted as  $T = \{t_1, t_2, \dots, t_{N_s}\}$ , by which the shading network  $F_{\theta_s}$  is queried for each point corresponding to the ray distances in  $T$  (along with  $r_d$ ) to obtain  $c_i$  and  $\sigma_i$  as

$$[c_i, \sigma_i] = F_{\theta_s}(r(t_i), r_d). \quad (3)$$

Furthermore, similar to AdaNeRF, our ProNeRF adjusts the final sample opacities  $\alpha_i$ , which allows for fewer-sample rendering and back-propagation during training. However, unlike the AdaNeRF that re-scales the sample densities, we shift and scale the  $\alpha$  values in our ProNeRF, yielding  $\hat{\alpha}$ :

$$\hat{\alpha}_i = a_i (1 - e^{-(\sigma_i + b_i)(t_{i+1}-t_i)}), \quad (4)$$

where  $a_i$  and  $b_i$  are estimated by the PAS network as  $A_t = \{a_1, a_2, \dots, a_{N_s}\}$  and  $B_t = \{b_1, b_2, \dots, b_{N_s}\}$ . We then render the final ray color in our *PAS-guided VRE* according to

$$\hat{c}(r) = \sum_{i=1}^{N_s} \left( \prod_{j=1}^{i-1} 1 - \hat{\alpha}_j \right) \hat{\alpha}_i c_i. \quad (5)$$

### 3.2 PAS: Projection-Aware Sampling

Similar to previous sampler-based methods, our PAS network in the ProNeRF runs only once per ray, which is a very efficient operation during both training and testing. As depicted in Fig. 2, our ProNeRF employs two MLP heads that map rays into the optimal ray distances  $T$  and the corresponding shift and scale in density values  $A_t$  and  $B_t$  required in the *PAS-guided VRE*.

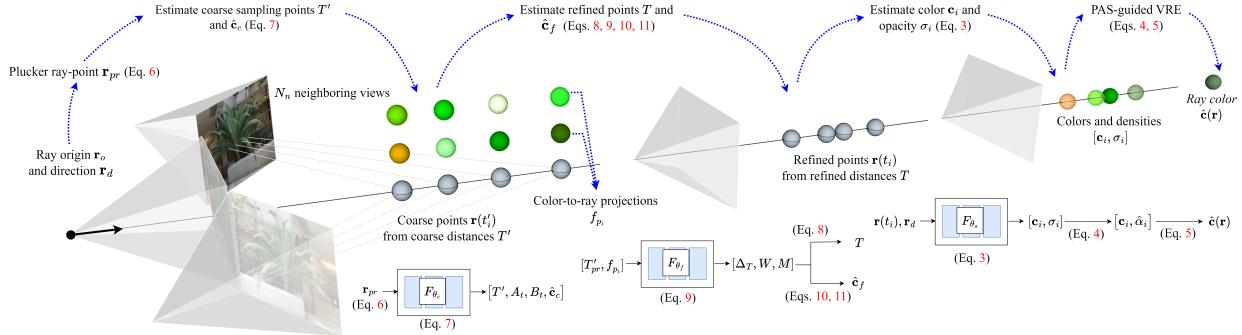


Figure 2: A conceptual illustration of our fast and high-quality projection-aware sampling of neural radiance fields (ProNeRF). The reference views are available during training and testing. The target view is drawn for illustrative purposes only.

The first step in the PAS of our ProNeRF is to map the ray's origin and direction ( $r_o$  and  $r_d$ ) into a representation that facilitates the mapping of training rays and interpolation of unseen rays. Feeding the raw  $r_o$  and  $r_d$  into  $F_{\theta_c}$  can mislead to overfitting, as there are a few ray origins in a given scene (as many as reference views). To tackle this problem, previous works have proposed to encode rays as 3D points (TermiNeRF) or as a Plücker coordinate which is the cross-product  $r_o \times r_d$  (LightFields and HyperReel). Motivated by these works, we combine the Plücker and ray-point embedding into a 'Plücker ray-point representation'. Including the specific points in the ray aids in making the input representation more discriminative, as it incorporates not only the ray origin but also the range of the ray, while the vanilla Plücker ray can only represent an infinitely long ray. The embedded ray  $r_{pr}$  is then given by

$$r_{pr} = [r_d, r_o + r_d \odot t_{nf}, (r_o + r_d \odot t_{nf}) \times r_d] \quad (6)$$

where  $t_{nf}$  is a vector whose  $N_{pr}$  elements are evenly spaced between the scene's near and far bounds ( $t_n$  and  $t_f$ ),  $\odot$  is the Hadamard product, and  $[., .]$  is the concatenation operation. The ProNeRF processes the encoded ray  $r_{pr}$  via  $F_{\theta_c}$  in the first stage of PAS to yield the coarse sampling distances  $T' = \{t'_1, t'_2, \dots, t'_{N_s}\}$  along  $r$ .  $F_{\theta_c}$  also predicts the shifts and scales in opacity values  $A_t$  and  $B_t$ . Furthermore, inspired by light-fields,  $F_{\theta_c}$  yields a light-field color output  $\hat{c}_c$  which is supervised to approximate the GT color  $c(r)$  to further regularize  $F_{\theta_c}$  and improve the overall learning. The multiple outputs of  $F_{\theta_c}$  are then given by

$$[T', A_t, B_t, \hat{c}_c] = F_{\theta_c}(r_{pr}). \quad (7)$$

While the previous sampler-based methods attempt to sample radiance fields with a single network such as  $F_{\theta_c}$ , we propose a coarse-to-fine PAS in ProNeRF. In our ProNeRF, the second MLP head  $F_{\theta_f}$  is fed with the coarse sampling points  $r(t'_i)$  and color-to-ray projections which are obtained by tracing lines between the estimated coarse 3D ray points and the camera centers of  $N_n$  neighboring views from a pool of  $N_t$  available images, as shown in Fig. 2. The pool of  $N_t$  images in the training phase consists of all training images. However, it is worth noticing that only a significantly small number of  $N_t$  views is needed for inference. The color-to-ray projections make ProNeRF projection-aware and enable

$F_{\theta_f}$  to better understand the detailed geometry in the scenes as they contain not only image gradient information but also geometric information that can be implicitly learned for each point in space. That is, high-density points tend to contain similarly-valued multi-view color-to-ray projections.

Although previous image-based rendering methods have proposed to *directly* exploit projected reference-view-features onto the shading network, such as the works of T et al. (2023) and Suhail et al. (2022b), these approaches necessitate computationally expensive attention mechanisms and all training views storage for inference, hence increasing the inference latency and memory footprint. On the other hand, we propose to incorporate color-to-ray projections not for directly rendering the novel views but for *fine-grained ray sampling of radiance fields*. As we learn to sample implicit NeRFs sparsely, our framework provides a superior trade-off between memory, speed, and quality.

The color-to-ray projections are concatenated with the Plücker-ray-point-encoded  $r'_{pr}$  of coarse ray distances  $T'$ , which is then fed into  $F_{\theta_f}$ , as shown in Fig. 2. In turn,  $F_{\theta_f}$  improves  $T'$  by yielding a set of inter-sampling refinement weights, denoted as  $0 \leq \Delta_T \leq 1$ . The refined ray distances  $T$  are obtained by the linear interpolation between consecutive elements of the expanded set of coarse ray distances  $\dot{T} = \{t_n, t'_1, t'_2, \dots, t'_{N_s}, t_f\}$  from  $T'$ , as given by

$$T = \left\{ \frac{1}{2} \left( (\dot{T}_i + \dot{T}_{i+1}) + \Delta_{T_i} (\dot{T}_{i+2} - \dot{T}_i) \right) \right\}_{i=1}^{N_s}. \quad (8)$$

Our inter-sampling residual refinement aids in training stability by reusing and maintaining the order of the coarse samples  $T'$ .  $\Delta_T$  is predicted by  $F_{\theta_f}$  as given by

$$[\Delta_T, W, M] = F_{\theta_f}([r'_{pr}, f_{p_1}, f_{p_2}, \dots, f_{p_{N_s}}]), \quad (9)$$

where  $f_{p_i} = [c_{p_i}^1, c_{p_i}^2, \dots, c_{p_i}^{N_s}]$  and  $c_{p_i}^k$  is the  $k^{th}$  color-to-ray projection from the  $N_n$  views at 3D point  $p_i = r(t'_i)$ . Note that  $W$  and  $M$  in Eq. (9) are the auxiliary outputs of softmax and sigmoid for network regularization, respectively. In contrast with  $F_{\theta_c}$ ,  $F_{\theta_f}$  is projection-aware, thus  $\hat{c}_f$  is obtained by exploiting the color-to-ray projections in an *approximated version of volumetric rendering* (AVR). In AVR,  $c_{p_i}^k$  and  $W \in \mathbb{R}^{N_s}$  are employed to approximate the

VRE (Eq. 1). The terms  $(\prod_{j=1}^{i-1} 1 - \alpha_j) \alpha_i$  in VRE are approximated by  $W$  while  $\mathbf{c}_i$  is approximated by projected color  $\mathbf{c}_{p_i}^k$  for the  $k^{\text{th}}$  view in  $N_n$  neighbors. AVR then yields

$$\mathbf{c}_{avr}^k = \sum_{i=1}^{N_s} W_i \mathbf{c}_{p_i}^k, \quad (10)$$

resulting in  $N_n$  sub-light-field views. The final light-field output  $\hat{\mathbf{c}}_f$  is aggregated by  $M \in \mathbb{R}^{N_n}$  with  $\mathbf{c}_{avr}^k$  as

$$\hat{\mathbf{c}}_f = \sum_{k=1}^{N_n} M_k \mathbf{c}_{avr}^k \quad (11)$$

Algorithm 1: Exploration and exploitation end2end training

```

1: procedure PRONERF TRAINING
2:   Init Data, PAS,  $F_{\theta_s}$ ,  $Opt_s$ ,  $Opt_{cfs}$ 
3:   for  $it = 0$  to  $7 \times 10^5$  do
4:     Sample random ray  $\mathbf{r}$ 
5:      $A_t, B_t, T, \hat{\mathbf{c}}_c, \hat{\mathbf{c}}_f \leftarrow PAS(\mathbf{r})$ 
6:     if  $2|it$  and  $it < 4 \times 10^5$  then ▷ Exploration pass
7:        $N_s^+ \leftarrow RandInt(N_s, N)$ 
8:        $T^+ \leftarrow Sample(T, N_s^+)$ 
9:        $T^+ \leftarrow T^+ + noise$ 
10:       $\{\mathbf{c}_i, \sigma_i\}_{i=1}^{N_s^+} \leftarrow F_{\theta_s}(\mathbf{r}_o + \mathbf{r}_d \odot T^+)$ 
11:       $\hat{\mathbf{c}}(\mathbf{r}) \leftarrow VRE(\{\mathbf{c}_i, \sigma_i\}_{i=1}^{N_s^+}, T^+)$  (Eq. 1)
12:       $loss \leftarrow |\hat{\mathbf{c}}(\mathbf{r}) - \mathbf{c}(\mathbf{r})|_2$ 
13:      Back-propagate and update by  $Opt_s$ 
14:    else ▷ Exploitation pass
15:       $\{\mathbf{c}_i, \sigma_i\}_{i=1}^{N_s} \leftarrow F_{\theta_s}(\mathbf{r}_o + \mathbf{r}_d \odot T)$ 
16:       $\hat{\mathbf{c}}(\mathbf{r}) \leftarrow VRE(\{\mathbf{c}_i, \sigma_i\}_{i=1}^{N_s}, A_t, B_t, T)$  (Eq. 5)
17:       $loss \leftarrow |\hat{\mathbf{c}}(\mathbf{r}) - \mathbf{c}(\mathbf{r})|_2$ 
18:      if  $it < 4 \times 10^5$  then
19:         $loss \leftarrow loss + |\hat{\mathbf{c}}_c - \mathbf{c}(\mathbf{r})|_2 + |\hat{\mathbf{c}}_f - \mathbf{c}(\mathbf{r})|_2$ 
20:      Back-propagate and update by  $Opt_{cfs}$ 
```

### 3.3 Novel Exploration-Exploitation Training

Our training strategy alternates between ray sampling *exploration* and *exploitation* as shown in Algorithm 1. As noted in line(L)-2, we first initialize the dataset (composed of calibrated multi-views) by extracting the target rays and colors, followed by ProNeRF’s networks’ initialization. We implement two optimizers, one for exploration ( $Opt_s$ ) and the other for exploitation ( $Opt_{cfs}$ ).  $Opt_s$  updates the weights in  $F_{\theta_s}$ , while  $Opt_{cfs}$  updates all weights in  $F_{\theta_c}$ ,  $F_{\theta_f}$ ,  $F_{\theta_s}$ . The first step in a training cycle is to obtain the PAS outputs ( $A_t$ ,  $B_t$ ,  $T$ ,  $\hat{\mathbf{c}}_c$ ,  $\hat{\mathbf{c}}_f$ ), as denoted in line 5 of Algorithm 1.

In the *exploration* pass (Algorithm 1 L-7 to 13),  $F_{\theta_s}$  learns the scene’s full color and density distributions by randomly interpolating  $N_s$  estimated  $T$  distances into  $N_s^+$  piece-wise evenly-spaced exploration sample distances  $T^+$ . For example, if the number of estimated ray distances is  $N_s = 8$  and the exploration samples are randomly set to  $N_s^+ = 32$ , the distance between each sample in  $T$  will be evenly divided into four bins such that the sample count is 32. Moreover, we add Gaussian noise to  $T^+$  as shown in of Algorithm 1 L-9, further allowing the  $F_{\theta_s}$  to explore the scene’s full color and density distributions. We then query  $F_{\theta_s}$  for the  $N_s^+$  exploration points to obtain  $\mathbf{c}_i$  and  $\sigma_i$  in the original VRE (Eq. 1). Finally,  $F_{\theta_s}$  is updated in the *exploration* pass.

In the *exploitation* pass, described in Algorithm 1 L-15 to 20, we let the PAS and  $F_{\theta_s}$  be greedy by only querying the samples corresponding to  $T$  and using the PAS-guided VRE (Eq. 5). Additionally, we provide GT color supervision to the auxiliary PAS network light-field outputs  $\hat{\mathbf{c}}_c$  and  $\hat{\mathbf{c}}_f$  for the first 60% of the training iterations. For the remaining 40%, ProNeRF focuses on the *exploitation* and disables the auxiliary loss as described by Algorithm 1 L-18 and 19. Note that for rendering a ray color with a few points during *exploitation* and testing, adjusting  $\alpha_i$  in Eq. 4 is needed to compensate for the subsampled accumulated transmittance which is learned for the full ray distribution in the *exploration* pass.

In summary, during *exploration*, we approximate the VRE with Monte Carlo sampling, where a random number of samples, ranging from  $N_s$  to  $N$ , are drawn around the estimated  $T$ . When training under *exploitation*, we sparsely sample the target ray  $\mathbf{r}$  given by  $T$ . Furthermore, we only update  $F_{\theta_s}$  during the *exploration* pass while using the original VRE (Eq. 1). However, in our *exploitation* pass, we update all MLP heads while using the PAS-guided VRE (Eq. 5). See Section 4 for more implementation details.

### 3.4 Objective functions

Similar to previous works, we guide ProNeRF to generate GT colors from the queried ray points with an  $l_2$  penalty as

$$l = \frac{1}{N_r} \sum_{N_r} \|\hat{\mathbf{c}}(\mathbf{r}) - \mathbf{c}(\mathbf{r})\|_2, \quad (12)$$

which is averaged over the  $N_r$  rays in a batch. In contrast with the previous sampler-based networks (TermiNeRF, AdaNeRF, DoNeRF, HyperReel), our ProNeRF predicts additional light-field outputs, which further regularize learning, and is trained with an auxiliary loss  $l_a$ , as given by

$$l_a = \frac{1}{N_r} \sum_{N_r} \|\hat{\mathbf{c}}_c(\mathbf{r}) - \mathbf{c}(\mathbf{r})\|_2 + \|\hat{\mathbf{c}}_f(\mathbf{r}) - \mathbf{c}(\mathbf{r})\|_2. \quad (13)$$

Our total objective loss is  $l_T = l + \lambda l_a$ , where  $\lambda$  is 1 for 60% of the training and then set to 0 afterward.

## 4 Experiments and Results

We provide extensive experimental results on the LLFF (Mildenhall et al. 2019) and Blender (Mildenhall et al. 2020) datasets to show the effectiveness of our method in comparison with recent SOTA methods. Also, we present a comprehensive ablation study that supports our design choices and main contributions. More results are shown in *Supplemental*.

We evaluate the rendering quality of our method by three widely used metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM) (Wang et al. 2004) and Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al. 2018). When it comes to SSIM, there are two common implementations available, one from *Tensorflow* (Abadi et al. 2015) (used in the reported metrics from NeRF, MobileNeRF, and IBRnet), and another from *scikit-image* (van der Walt et al. 2014) (employed in ENeRF, RSeN, NLF). We denoted the metrics from *Tensorflow* and *scikit-image* as  $SSIM_t$  and  $SSIM_s$ , respectively. Similarly, for LPIPS, we can choose between two backbone options, namely AlexNet (Krizhevsky, Sutskever, and Hinton 2012) and VGG (Simonyan and Zisserman 2014). We present our SSIM and LPIPS results across all available choices to ensure a fair and comprehensive evaluation of our method’s performance.

## 4.1 Implementation Details

We train our ProNeRF with PyTorch on an NVIDIA A100 GPU using the Adam optimizer with a batch of  $N_r = 4,096$  randomly sampled rays. The initial learning rate is set to  $5 \times 10^{-4}$  and is exponentially decayed for 700K iterations. We used TensoRT on a single RTX 3090 GPU with model weights quantized to half-precision FP16 for testing. We set the point number in the Plücker ray-point encoding for our PAS network to 48. We set the maximum number of *exploration* samples to  $N = 64$ .  $F_{\theta_c}$  and  $F_{\theta_f}$  consist of 6 fully-connected layers with 256 neurons followed by ELU nonlinearities. Finally, we adopt the shading network introduced in DOnERF, which has 8 layers with 256 neurons.

## 4.2 Results

**Forward-Facing (LLFF).** This dataset comprises 8 challenging real scenes with 20 to 64 front-facing handheld captured views. We conduct experiments on  $756 \times 1008$  images to compare with previous methods, holding out every 8<sup>th</sup> image for evaluation. We also provide the quantitative results on  $378 \times 504$  images for a fair comparison to the methods evaluated on the lower resolution.

Our quantitative and qualitative results, respectively shown in Table 1 and Fig. 3, demonstrate the superiority of our ProNeRF over the implicit NeRF and the previous explicit methods, e.g., TensoRF and *K*-Planes. Our model with 8 samples, ProNeRF-8, is the first sampler-based method that outperforms the vanilla NeRF by 0.28dB PSNR while being more than **20**× faster. Furthermore, our ProNeRF-12 yields rendered images with **0.65dB** higher PSNR while being about **15**× faster than vanilla NeRF. Our improvements are reflected in the superior visual quality of the rendered images, as shown in Fig. 3. On the lower resolution, ProNeRF-8 outperforms the second-best R2L by 0.28dB and the latest sampler-based HypeReel by 0.58dB with faster rendering. In Table 1, compared to the explicit grid-based methods of INGP, Plenoxtels and MobileNeRF, our ProNeRF shows a good trade-off between memory, speed, and quality.

We also present the quantitative results of the auxiliary PAS light field outputs in Table 1, denoted as PAS-8  $c_f$  for both the regression (Reg) and AVR cases. We observed no difference in the final color output when Reg or AVR were used in ProNeRF-8. However, PAS-8  $c_f$  (AVR) yields considerably better metrics than its Reg counterpart.

Inspired by the higher FPS from PAS-8  $c_f$  (AVR), we also explored *pruning* ProNeRF by running the  $F_{\theta_s}$  only for the “complex rays”. We achieve ProNeRF-8 prune by training a complementary MLP head  $F_{\theta_m}$  which has the same complexity as  $F_{\theta_c}$  and predicts the error between  $\hat{c}_f$  and  $\hat{c}$  outputs. When the error is low, we render the ray by PAS-8  $c_f$  (AVR); otherwise, we subsequently run the shader network  $F_{\theta_s}$ . While *pruning* requires an additional 3.3 MB in memory, the pruned ProNeRF-8 is 23% faster than ProNeRF-8 with a small PSNR drop and negligible SSIM and LPIPS degradations, as shown in Table 1. Note that other previous sampler-based methods cannot be pruned similarly, as they do not incorporate the auxiliary light-field output. Training *pruning* is fast (5min). See more details in *Supplemental*.

Res. Methods	PSNR	SSIM <sub>t/s</sub>	LPIPS <sub>vgg/alex</sub>	FPS	Mem(MB)
NeRF (ECCV20)	26.50	0.811 / -	0.250 / -	0.3	3.8
INGP (SIGGRAPH22)	25.60	0.758 / -	0.267 / -	7.3	64.0
Plenoxtels (CVPR22)	26.30	0.839 / -	0.210 / -	9.1	3629.8
MipNeRF360 (CVPR22)	26.86	<b>0.858</b> / -	- / 0.128	0.1	8.2
TensoRF (ECCV22)	26.73	0.839 / -	0.204 / 0.124	1.1	179.7
<i>K</i> -Planes (CVPR23)	26.92	0.847 / -	0.182 / -	0.7	214
756 SNeRG (ICCV21)	25.63	0.818 / -	0.183 / -	50.7	337.3
× ENeRF (SIGGRAPH22)	24.89	- / 0.865	<b>0.159</b> / -	8.9	10.7
1008 AdaNeRF (ECCV22)	25.70	- / -	- / -	7.7	4.1
Hyperreel (CVPR23)	26.20	- / -	- / -	4.0	58.8
MobileNeRF (CVPR23)	25.91	0.825 / -	0.183 / -	<b>348</b>	201.5
PAS-8 $c_f$ (Reg) (Ours)	24.86	0.787 / 0.855	0.236 / 0.150	29.4	<b>2.7</b>
PAS-8 $c_f$ (AVR) (Ours)	25.15	0.793 / 0.860	0.234 / 0.146	25.6	5.0
ProNeRF-8 Prune (Ours)	26.54	0.825 / 0.883	0.219 / 0.120	8.5	6.8
ProNeRF-8 (Ours)	26.78	0.825 / 0.884	0.228 / 0.119	6.9	3.5
ProNeRF-12 (Ours)	<b>27.15</b>	0.838 / <b>0.894</b>	0.217 / <b>0.109</b>	4.4	3.5
FastNeRF (ICCV21)	26.04	- / 0.856	- / 0.085	<b>700</b>	4100
EfficientNeRF (CVPR22)	27.39	- / 0.912	- / 0.082	219	2800
378 RSEN (CVPR22)	27.45	- / 0.905	- / 0.060	0.34	5.4
× R2L (ECCV22)	27.79	- / -	- / 0.097	5.6	22.6
504 Hyperreel (CVPR23)	27.50	- / -	- / -	4.0	58.8
ProNeRF-8 (Ours)	28.08	0.879 / 0.916	0.129 / 0.060	6.9	<b>3.5</b>
ProNeRF-12 (Ours)	<b>28.33</b>	<b>0.885 / 0.920</b>	0.129 / <b>0.058</b>	4.4	<b>3.5</b>

Table 1: Results on LLFF. Metrics are the lower the better and the higher the better. (-) metrics are not provided in the original literature.

Methods	PSNR	SSIM	LPIPS					
No exploration pass	24.00	0.754	0.299					
No exploitation pass	24.31	0.779	0.278					
		Avg $N_r$	PSNR SSIM LPIPS Mem(MB)					
No $\sigma$ shift (no $B_t$ )	24.2	0.773	0.264	4.00	27.15	<b>0.838</b>	0.217	<b>3.5</b>
No aux. loss (no $I_a$ )	24.26	0.766	0.296	8.00	<b>27.16</b>	<b>0.838</b>	<b>0.216</b>	4.2
No $\hat{\alpha}$ (no $A_t, B_t$ )	24.69	0.785	0.260	12.00	27.15	0.837	0.217	4.9
No Plücker ray-point	24.72	0.782	0.257	32.75	27.15	<b>0.838</b>	<b>0.216</b>	8.4
No color-to-ray proj	24.83	0.789	0.245					
ProNeRF-12 $N_r=4$	<b>25.17</b>	<b>0.809</b>	<b>0.244</b>					

Table 2: ProNeRF ablations on LLFF. (Left) Network designs on Fern. (Right) Ablation of # of available ref. views.

**360 Blender.** This is an object-centric 360-captured synthetic dataset for which our ProNeRF-32 achieves a reasonably good performance of 31.92 dB PSNR, 3.2 FPS (after pruning) and 6.3 MB Mem. It should be also noted that the ProNeRF-32 outperforms NeRF, SNeRG, Plenoxtree, and Plenoxtels while still displaying a favorable performance profiling. See *Supplemental* for detailed results.

## 4.3 Ablation Studies

We ablate our ProNeRF on the LLFF’s Fern scene in Table 2 (left). We first show that infusing exploration and exploitation into our **training strategy** is critical for high-quality neural rendering. As shown in the top section of Table 2 (left), exploration- or exploitation-only leads to sub-optimal results as neither the shading network is allowed to learn the full scene distributions nor the PAS network is made to focus on the regions with the highest densities.

Next, we explore our **network design** by ablating each

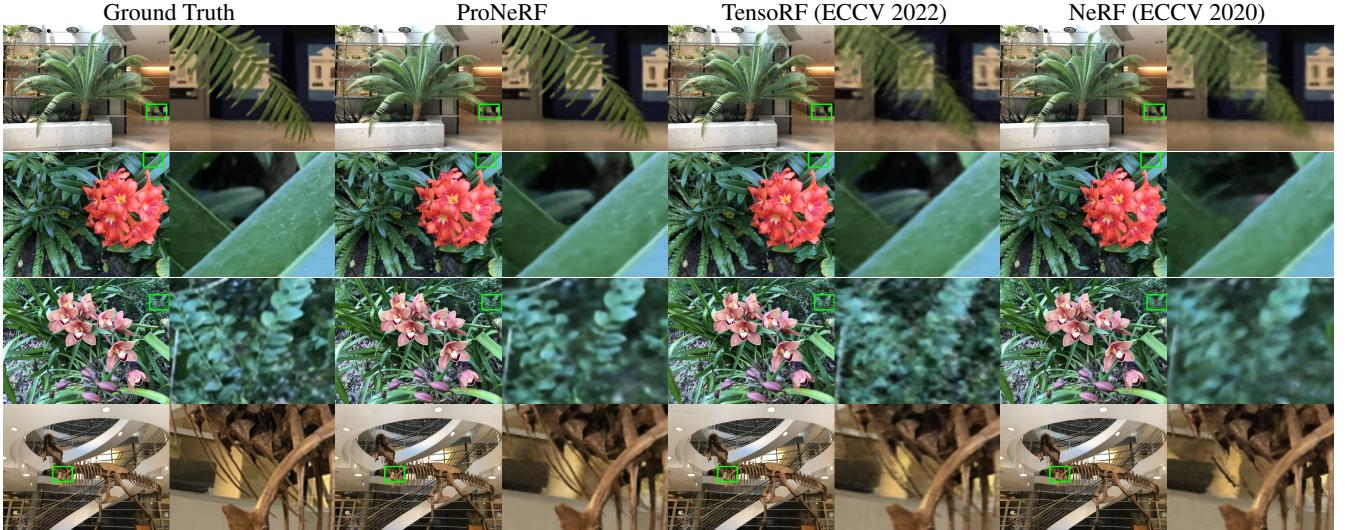


Figure 3: Qualitative comparisons for the LLFF (Mildenhall et al. 2019) dataset. Zoom in for better visualization.

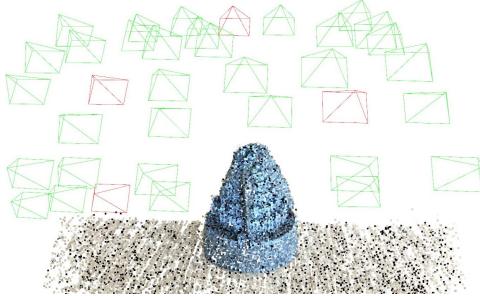


Figure 4: Cameras distribution on the LLFF’s Fortress scene. Green cameras denote available training views. Red cameras denote selected and fixed subset of  $N_t$  frames for projection.

design choice. As noted in Table 2 (left), removing  $\alpha$  scales ( $A_t$ ) and shifts ( $B_t$ ) severely impact the rendering quality. We also observed that the auxiliary loss ( $l_a$ ) is critical to properly train our sampler since its removal causes almost 1dB drop in PSNR. The importance of our Plücker ray-point encoding is shown in Table 2 (left), having an impact of almost 0.5dB PSNR drop when disabled. Finally, we show that the color-to-ray projection in the PAS of our ProNeRF is the key feature for high-quality rendering.

**Memory footprint consistency.** This experiment proves ProNeRF yields a consistent usage of memory footprint. As mentioned in *Section 2*, light-fields and image-based rendering methods, which rely on multi-view color projections, typically require large storage for all available training views for rendering a novel view. This is because they utilize the nearest reference views to the target pose from the entire pool of available images. In contrast, our ProNeRF takes a distinct approach by consistently selecting a **fixed** subset of  $N_t$  reference views when rendering any novel viewpoint in the inference stage. This is possible because (i) we randomly select any  $N_n$  neighboring views (from the entire training

pool) during training; and (ii) our final rendered color is obtained by sparsely querying a radiance field, not by directly processing projected features/colors. As a result, our framework yields a consistent memory footprint for storing reference views, which is advantageous for efficient hardware design. To select the  $N_t$  views, we leverage the sparse point cloud reconstructed from COLMAP and a greedy algorithm to identify the optimal combination of potential frames. As shown in Fig. 4, the  $N_t$  views become a subset across all available training images that comprehensively cover the target scene (see details in *Supplemental*). As shown in Table 2 (right), we set the number of neighbors in PAS to  $N_n = 4$  and adjust  $N_t$  to 4, 8, 12, and all training views (32.75). Please note our ProNeRF’s rendering quality remains stable while modulating  $N_t$ , attesting to the stability and robustness of our approach across varying configurations.

#### 4.4 Limitations

While not technically constrained to forward-facing scenes (such as NeX) and yielding better metrics than vanilla NeRF and several other works, our method is behind grid-based explicit models such as INGP for the Blender dataset. The methods like INGP contain data structures that better accommodate these kinds of scenes. Our method requires more samples for this data type, evidencing that our method is more efficient and shines on forward-facing datasets.

## 5 Conclusions

Our ProNeRF, a sampler-based neural rendering method, significantly outperforms the vanilla NeRF quantitatively and qualitatively for the first time. It also outperforms the existing explicit voxel/grid-based methods by large margins while preserving a small memory footprint and fast inference. Furthermore, we showed that our exploration and exploitation training is crucial for learning high-quality rendering. Future research might extend our ProNeRF for dynamic-scenes and cross-scene generalization.

## Acknowledgements

This work was supported by IITP grant funded by the Korea government (MSIT) (No. RS2022-00144444, Deep Learning Based Visual Representational Learning and Rendering of Static and Dynamic Scenes).

## References

- Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.
- Attal, B.; Huang, J.; Richardt, C.; Zollhöfer, M.; Kopf, J.; O'Toole, M.; and Kim, C. 2023. HyperReel: High-Fidelity 6-DoF Video with Ray-Conditioned Sampling. *CoRR*, abs/2301.02238.
- Attal, B.; Huang, J.-B.; Zollhöfer, M.; Kopf, J.; and Kim, C. 2022. Learning Neural Light Fields with Ray-Space Embedding Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *CVPR*.
- Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; and Su, H. 2022. TensoRF: Tensorial Radiance Fields. In Avidan, S.; Brostow, G. J.; Cissé, M.; Farinella, G. M.; and Hassner, T., eds., *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXII*, volume 13692 of *Lecture Notes in Computer Science*, 333–350. Springer.
- Chen, Z.; Funkhouser, T.; Hedman, P.; and Tagliasacchi, A. 2023. MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures. In *The Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Drebin, R. A.; Carpenter, L.; and Hanrahan, P. 1988. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4): 65–74.
- Flynn, J.; Broxton, M.; Debevec, P.; DuVall, M.; Fyffe, G.; Overbeck, R.; Snavely, N.; and Tucker, R. 2019. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2367–2376.
- Fridovich-Keil, S.; Yu, A.; Tancik, M.; Chen, Q.; Recht, B.; and Kanazawa, A. 2022. Plenoxels: Radiance Fields without Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, 5491–5500. IEEE.
- Garbin, S. J.; Kowalski, M.; Johnson, M.; Shotton, J.; and Valentin, J. P. C. 2021. FastNeRF: High-Fidelity Neural Rendering at 200FPS. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, 14326–14335. IEEE.
- Hedman, P.; Srinivasan, P. P.; Mildenhall, B.; Barron, J. T.; and Debevec, P. E. 2021. Baking Neural Radiance Fields for Real-Time View Synthesis. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, 5855–5864. IEEE.
- Hu, T.; Liu, S.; Chen, Y.; Shen, T.; and Jia, J. 2022. EfficientNeRF - Efficient Neural Radiance Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, 12892–12901. IEEE.
- Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4).
- Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.
- Kurz, A.; Neff, T.; Lv, Z.; Zollhöfer, M.; and Steinberger, M. 2022. AdaNeRF: Adaptive Sampling for Real-Time Rendering of Neural Radiance Fields. In Avidan, S.; Brostow, G. J.; Cissé, M.; Farinella, G. M.; and Hassner, T., eds., *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XVII*, volume 13677 of *Lecture Notes in Computer Science*, 254–270. Springer.
- Levoy, M.; and Hanrahan, P. 1996. Light Field Rendering. In Fujii, J., ed., *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*, 31–42. ACM.
- Li, J.; Feng, Z.; She, Q.; Ding, H.; Wang, C.; and Lee, G. H. 2021. MINE: Towards Continuous Depth MPI with NeRF for Novel View Synthesis. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, 12558–12568. IEEE.
- Lin, H.; Peng, S.; Xu, Z.; Yan, Y.; Shuai, Q.; Bao, H.; and Zhou, X. 2022. Efficient Neural Radiance Fields for Interactive Free-viewpoint Video. In *SIGGRAPH Asia Conference Proceedings*.
- Lindell, D. B.; Martel, J. N. P.; and Wetzstein, G. 2021. AutoInt: Automatic Integration for Fast Neural Volume Rendering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 14556–14565. Computer Vision Foundation / IEEE.
- Liu, Y.; Peng, S.; Liu, L.; Wang, Q.; Wang, P.; Theobalt, C.; Zhou, X.; and Wang, W. 2022. Neural Rays for Occlusion-aware Image-based Rendering. In *CVPR*.
- Mildenhall, B.; Srinivasan, P. P.; Cayon, R. O.; Kalantari, N. K.; Ramamoorthi, R.; Ng, R.; and Kar, A. 2019. Local light field fusion: practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 38(4): 29:1–29:14.
- Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. NeRF: Representing

- Scenes as Neural Radiance Fields for View Synthesis. In Vedaldi, A.; Bischof, H.; Brox, T.; and Frahm, J., eds., *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, 405–421. Springer.
- Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4): 102:1–102:15.
- Neff, T.; Stadlbauer, P.; Parger, M.; Kurz, A.; Mueller, J. H.; Chaitanya, C. R. A.; Kaplanyan, A.; and Steinberger, M. 2021. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Comput. Graph. Forum*, 40(4): 45–59.
- Piala, M.; and Clark, R. 2021. TermiNeRF: Ray Termination Prediction for Efficient Neural Rendering. In *International Conference on 3D Vision, 3DV 2021, London, United Kingdom, December 1-3, 2021*, 1106–1114. IEEE.
- Sara Fridovich-Keil and Giacomo Meanti; Warburg, F. R.; Recht, B.; and Kanazawa, A. 2023. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. In *CVPR*.
- Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Sitzmann, V.; Martel, J.; Bergman, A.; Lindell, D.; and Wetzstein, G. 2020. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33: 7462–7473.
- Suhail, M.; Esteves, C.; Sigal, L.; and Makadia, A. 2022a. Generalizable Patch-Based Neural Rendering. In *European Conference on Computer Vision*. Springer.
- Suhail, M.; Esteves, C.; Sigal, L.; and Makadia, A. 2022b. Light Field Neural Rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, 8259–8269. IEEE.
- T, M. V.; Wang, P.; Chen, X.; Chen, T.; Venugopalan, S.; and Wang, Z. 2023. Is Attention All That NeRF Needs? In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- van der Walt, S.; Schönberger, J. L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J. D.; Yager, N.; Gouillart, E.; Yu, T.; and the scikit-image contributors. 2014. scikit-image: image processing in Python. *PeerJ*, 2: e453.
- Wang, H.; Ren, J.; Huang, Z.; Olszewski, K.; Chai, M.; Fu, Y.; and Tulyakov, S. 2022. R2L: Distilling Neural Radiance Field to Neural Light Field for Efficient Novel View Synthesis. In *ECCV*.
- Wang, Q.; Wang, Z.; Genova, K.; Srinivasan, P. P.; Zhou, H.; Barron, J. T.; Martin-Brualla, R.; Snavely, N.; and Funkhouser, T. A. 2021. IBRNet: Learning Multi-View Image-Based Rendering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 4690–4699. Computer Vision Foundation / IEEE.
- Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4): 600–612.
- Yu, A.; Li, R.; Tancik, M.; Li, H.; Ng, R.; and Kanazawa, A. 2021a. PlenOctrees for Real-time Rendering of Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, 5732–5741. IEEE.
- Yu, A.; Ye, V.; Tancik, M.; and Kanazawa, A. 2021b. pixelNeRF: Neural Radiance Fields From One or Few Images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 4578–4587. Computer Vision Foundation / IEEE.
- Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 586–595. Computer Vision Foundation / IEEE Computer Society.