Fast Learning Radiance Fields by Shooting Much Fewer Rays

Wenyuan Zhang, Ruofan Xing, Yunfan Zeng, Yu-Shen Liu, *Member, IEEE*, Kanle Shi, and Zhizhong Han

***Abstract*— Learning radiance fields has shown remarkable results for novel view synthesis. The learning procedure usually costs lots of time, which motivates the latest methods to speed up the learning procedure by learning without neural networks or using more efficient data structures. However, these specially designed approaches do not work for most of radiance fields based methods. To resolve this issue, we introduce a general strategy to speed up the learning procedure for almost all radiance fields based methods. Our key idea is to reduce the redundancy by shooting much fewer rays in the multi-view volume rendering procedure which is the base for almost all radiance fields based methods. We find that shooting rays at pixels with dramatic color change not only significantly reduces the training burden but also barely affects the accuracy of the learned radiance fields. In addition, we also adaptively subdivide each view into a quadtree according to the average rendering error in each node in the tree, which makes us dynamically shoot more rays in more complex regions with larger rendering error. We evaluate our method with different radiance fields based methods under the widely used benchmarks. Experimental results show that our method achieves comparable accuracy to the state-of-the-art with much faster training.**

***Index Terms*— Novel view synthesis, neural rendering, radiance fields, neural networks, quadtree.**

I. Introduction

LEARNING a scene representation from single or multiple rendered views is an effective way to understand 3D shapes and scenes. Recent work has made great progress in reconstructing 3D meshes or point clouds from given 2D images for scene understanding [3], [4], [5], [6], [7], [8], [9]. However, it is still a challenge to render realistic views from meshes and point clouds. As a solution, neural radiance field builds up a bridge between 2D images and 3D representations because it has the ability to both learn the implicit representation of 3D scenes and render novel views from any given perspectives [1], [10], [11], [12], [13].

A radiance field describes a scene by providing the color and density at each queried location in the scene. Hence, with a known camera pose, we can render a radiance field into an image from a specific view angle using volume rendering, which emits a ray on each pixel and accumulates the color and density of sampled points along each ray. Given multiple images taken from a scene as input, we can learn a radiance field of the scene by pushing the radiance field to be rendered as similar to the input images as possible.

Current methods leveraging deep learning models to learn radiance fields have made significant progress. Radiance fields have been used in various applications such as novel view synthesis [1], [11], [14], scene editing [15], [16], stylization [17], [18] and generation [19], [20], [21]. Despite great success, the unaffordable learning time is still a big limitation suffering from the state-of-the-art methods.

Current methods speed up the learning of radiance fields by different strategies [2], [22], [23], [24], [25]. For example, Plenoxels [2] replaced neural networks by a sparse voxel model to learn radiance fields, which achieves a speedup of two orders of magnitude compared to NeRF [1]. TensoRF [24] models the radiance field of a scene as a 4D tensor, which represents a 3D voxel grid with per-voxel multi-channel features. Differently, Instant-NGP [23] introduced multiresolution hash encoding that permits the use of a smaller network without sacrificing quality, which also significantly reduces the training cost. Although these methods can train radiance fields fast, they require specific architectures, such as sparse voxel grids, discrete tensor coordinates or hash coding, which are not easy to adapt to improve the training efficiency of different neural radiance fields variations.

To resolve this issue, we introduce a general strategy to speed up the learning of radiance fields, as illustrated in Figure 1. Our key idea is to shoot much fewer rays to the radiance fields in volume rendering, which is a universal procedure during training in different neural radiance fields methods. Our contribution lies in our founding that we can perceive a radiance field without dramatically sacrificing accuracy by just shooting rays at pixels with dramatic color change, which significantly reduces redundancy of training rays from other pixels. Moreover, we also adaptively subdivide each view into

Manuscript received 11 July 2022; revised 28 December 2022; accepted 24 March 2023. Date of publication 8 May 2023; date of current version 16 May 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFC3800600, in part by the National Natural Science Foundation of China under Grant 62272263 and Grant 62072268, and in part by the Tsinghua-Kuaishou Institute of Future Media Data. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Aswin Sankaranarayanan. *(Corresponding author: Yu-Shen Liu.)*

Wenyuan Zhang and Ruofan Xing are with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: zhangwen21@mails.tsinghua.edu.cn; xingrf20@mails.tsinghua.edu.cn).

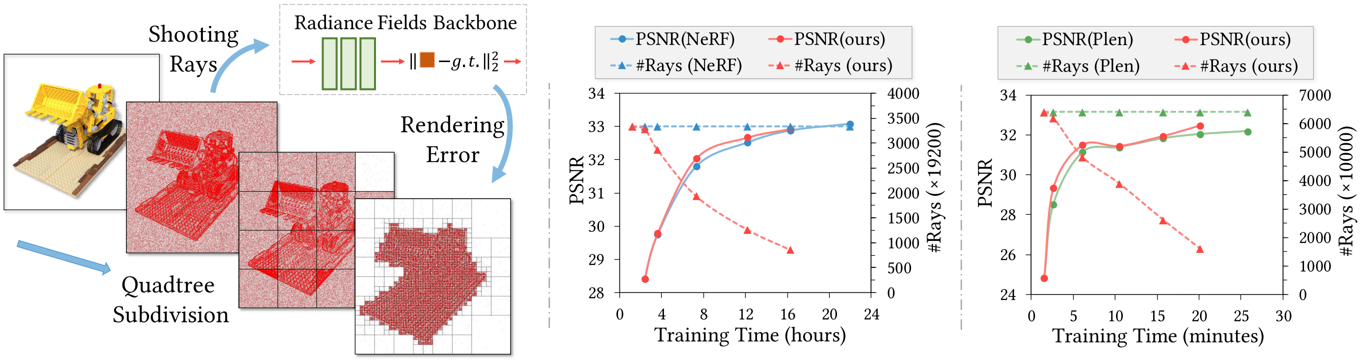
Yunfan Zeng is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: zengyf20@mails.tsinghua.edu.cn).

Yu-Shen Liu is with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: liuyushen@tsinghua.edu.cn).

Kanle Shi is with Kuaishou Technology, Beijing 100085, China (e-mail: shikanle@kuaishou.com).

Zhizhong Han is with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA (e-mail: h312h@wayne.edu).

Digital Object Identifier 10.1109/TIP.2023.3267049



(a) Method Overview **(b) Comparing with NeRF** **(c) Comparing with Plenoxels**

Fig. 1. Illustration of our general strategy to learn radiance fields faster by shooting much fewer rays. **(a)** We use image context based probability distribution and adaptive quadtree subdivision to determine where to shoot more rays or to reduce rays on training images. Take "lego" in Synthetic dataset [1] for example. **(b)** Based on NeRF [1], our method decreases the number of training rays from 3334 to 852, which reduces 30% training time. **(c)** Based on Plenoxels [2], our method decreases the number of rays from 12800 to 3218 and reduces 22% training time. Both of our methods achieve a higher accuracy on test views.

a quadtree according to the average rendering error in each node in the tree, which makes us shoot more rays in more complex regions with larger rendering error. We evaluate our method with different radiance fields based methods under different benchmarks. Experimental results show that our method achieves comparable accuracy to the state-of-the-art with much faster training. Our contributions are listed below.

- We present a method to speed up the learning of radiance fields. Our method can generally work for different radiance field based methods without requiring specific modifications.
- We justify the founding that the rays starting from different pixels have different perceiving ability of radiance fields.
- Our method can generalize to different radiance field based methods, and significantly reduce the training time under different scenes.

II. Related Work

A. Neural Radiance Fields

Recently, NeRF [1] has achieved impressive results in novel view synthesis task and has attracted lots of follow-up work. Unlike traditional explicit and discretized volumetric representations, NeRF uses a continuous 5D function to represent a static scene and optimizes a deep fully-connected neural network to learn this function. It can render high-resolution photorealistic novel views of objects and scenes from RGB images captured in natural settings. The follow-up work resolves the deficiencies, and expands possible applications of NeRF. NeRF++ [10] extends NeRF to work for unbounded scenes, while NeRF-W [26] tackles the learning of radiance fields from unstructured photo collections. PixelNeRF, etc. [11], [27], [28], [29], [30] are devoted to solve the distortion problems when input views are sparse while others [31], [32], [33] remove the requirement for pose estimation in volume rendering. In addition, there is still much work to study the solutions of defects and various applications of NeRF, e.g., mesh reconstruction [12], [34], [35], relighting [36], [37], dynamic scene modeling [38], [39], [40], [41] and deformation [42], [43], [44].

B. Fast NeRF Rendering

Although NeRF has achieved amazing novel view synthesis effects, its training and rendering phase costs lots of time. Rendering a novel view usually takes about one minute, which is not conductive to real-time rendering and interaction. To solve this problem, NSVF [45] uses both sparse voxel fields and classical techniques like empty space skipping and early ray termination to speed up the rendering procedure in NeRF. The idea of decomposition has also been adopted by many other methods [46], [47], [48]. DeRF [46] decomposes the scene into sixteen irregular Voronoi cells to speed up the rendering. FastNeRF [47] splits the same task into two neural networks that are amenable to caching. Similarly, Kilo-NeRF [48] decomposes the scene into thousands of tiny MLPs arranged on a regular 3D grid, leading to significantly higher speedups. Moreover, DONeRF [49] uses another network to predict the intersection of rays and mesh surfaces. Only 4 sampled points near surface are needed for neural rendering. AutoInt [50] replaces the need for numerical integration in NeRF by learning a closed form solution of the antiderivative.

C. Fast NeRF Convergence

Training NeRF [1] and its variants [10], [26], [32] usually takes 1 to 2 days, which limits its further applications. Many recent papers have proposed methods to improve the training efficiency of radiance fields. Some methods focus on rendering quality or sparse input views and bring faster convergence as a side benefit, such as generalizable pre-training [14], [51], [52], anti-aliasing [53], [54], external depth [45], [52] and so on. Other methods apply practical tricks to reduce training redundancy, such as skipping empty space [13], [22], [23], pruning sample points on rays [23], [25], [55], simplifying the two-stage coarse to fine training [24], [25], [54], block decomposition [56] and so on.

Recent studies [2], [22], [23], [57] adopt voxel grids to represent neural radiance field without a prior consultation. In these methods, parameters to be optimized are stored in voxel grid vertices. Parameters of sampled points are trilinear-interpolated from its neighboring vertices instead of directly using its 5D coordinates. Through this way, the amount of parameters to be optimized are reduced from infinite space to the same scale as the number of voxel grid vertices. Specifically, PlenOctrees [57] extracts the NeRF structure into a sparse voxel grid in which each voxel represents view-dependent color using spherical harmonic (SH) coefficients. Plenoxels [2] extends PlenOctrees by additionally assign density parameters to voxel grids. The color and density of sample points are directly interpolated from grid vertices instead of predicted by MLP (Multi-Layer Perceptron). DVGO [22] achieves a similar speed as Plenoxels by splitting color and density into two voxel grids, where one is feature grid and the other is density grid. The color is generated by a shallow MLP, using interpolated feature from feature grid as input, and the density is directly interpolated from density grid. Latest work Instant-NGP [23] uses hash encoding to store features in multiresolution voxel grids. The feature of a sample point is queried, interpolated and then concatenated from multiresolution features. It achieves both very fast speed and quite good rendering results in different tasks.

Although the above methods have made some progress in terms of NeRF accelerating, they are task-specific and difficult to migrate to different radiance fields variations because of their specific and complex architectures. To resolve this issue, we introduce a general strategy to speed up the learning procedures for mainstream radiance fields based methods and achieves much faster training speed and comparable accuracy than the state-of-the-art works.

III. Method

Our method is a general framework of accelerating training procedure, which can be easily integrated with mainstream radiance fields based methods. This generalization ability comes from the way of how we shoot much fewer rays in volume rendering to perceive radiance fields better, which is a common procedure in radiance fields based methods. Our method is formed by two main components: (1) a probability-based sampling function which samples rays according to the input image context and (2) an adaptive quadtree subdivision strategy that learns where to reduce rays in simple regions and where to increase rays in complex regions. Figure 2 illustrates our method, which will be detailed in the following.

A. Volume Rendering

NeRF [1] proposed to use a continuous 5D function to represent a static scene and achieved great results in the view synthetic task. We use the same differentiable model for volume rendering as in NeRF, where the color $\hat{\mathbf{c}}$ of a camera ray is rendered using 3D points sampled along the ray by the discretized volume rendering, defined by

$$\hat{\mathbf{c}} = \sum_{j=1}^{N} T_j(1-\exp(-\sigma_j\delta_j))\mathbf{c}_j$$
$$T_j = \exp(-\sum_{t=1}^{j-1}\sigma_t\delta_t), \tag{1}$$

where $\mathbf{c}_j$, σ_j and T_j represent the color, the opacity and the transmittance of the ray at j-th sampled point, respectively, δ_j indicates the distance between j-th and $(j+1)$-th adjacent sampled points. This function for calculating $\hat{\mathbf{c}}$ from the set of $(\mathbf{c}_j, \sigma_j)$ values is trivially differentiable and reduces to traditional alpha compositing with alpha values $\alpha_j = 1 - \exp(-\sigma_j\delta_j)$.

B. Ray Sampling Probability

Specifically, given a 3D point $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ and a related viewing direction $\mathbf{d} = (\theta, \phi) \in \mathbb{R}^2$, a neural network $F_\Theta : (\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma)$ maps a 5D point $(\mathbf{x}, \mathbf{d})$ to an emitted color $\mathbf{c} = (r, g, b)$ and a volume density σ. The parameters of the network are optimized by minimizing the pixel-level color error obtained by volume rendering.

During training, an amount of 5D points are sampled on each ray $\mathbf{r}_i^k \in \mathbf{R}_i$, where $\mathbf{r}_i^k$ represents the k-th ray from the i-th training image I_i, and the set $\mathbf{R}_i$ of the rays is generated from $I_i \in \mathcal{I}, i = 1, 2 \ldots N$, where N is the number of images and $\mathcal{I}$ is the set of all the training images. Volume rendering $V_\Theta : \mathbf{r}_i^k \mapsto \hat{\mathbf{c}}_i^k$ is then used for rendering per-pixel color $\hat{\mathbf{c}}_i^k$ using color and density predicted from the sampled 5D points along $\mathbf{r}_i^k$. The object function is to minimize the following volume rendering loss from all the emitted rays

$$\mathcal{L} = \frac{1}{NM}\sum_{I_i\in\mathcal{I}}\sum_{\mathbf{r}_i^k\in\mathbf{R}_i} ||V_\Theta(\mathbf{r}_i^k) - \mathbf{c}_i^k||_2^2, \tag{2}$$

where $\mathbf{c}_i^k$ is the pixel's ground truth color and M is the size of $\mathbf{R}_i$, that is, the amount of emitted rays in each image I_i.

With regard to how to obtain R_i of all the training images, almost all radiance fields based methods randomly select M pixels in the image I_i, and then emit M rays from the selected pixels along the viewing direction. Given a pixel (u, v), we use $\mathbf{r}_i(u, v)$ to represent a ray starting from the pixel and the ray orientation is consistent with the viewing direction of image I_i. In this way, the sampled ray positions obey uniform distribution on the training images:

$$\mathbf{r}_i(u, v) \sim U(I_i), u \in [0, H_i], v \in [0, W_i], \tag{3}$$

where H_i and W_i are the height and width of image I_i, respectively.

Although such distribution is straight and proved to be effective in practice [1], [10], there are still some potential problems. (1) The uniformly distributed rays cannot well capture the non-uniformly distributed information. In practice, there are many areas of continuous pixels with the similar color on images, which we call *trivial areas*, such as the background of synthetic images, the sky in outdoor scene images, the wall in indoor scene images. Pixels in such kinds of trivial areas usually contain less information due to semantic consistency with their neighborhood pixels.

As a result, the rendering error of the rays shot from trivial areas can converge fast, because the density of most sampled points along the rays is close to 0. Therefore, we only need to shoot fewer rays in the trivial areas where the color changes slightly to perceive the radiance fields. (2) In contrast, pixels in

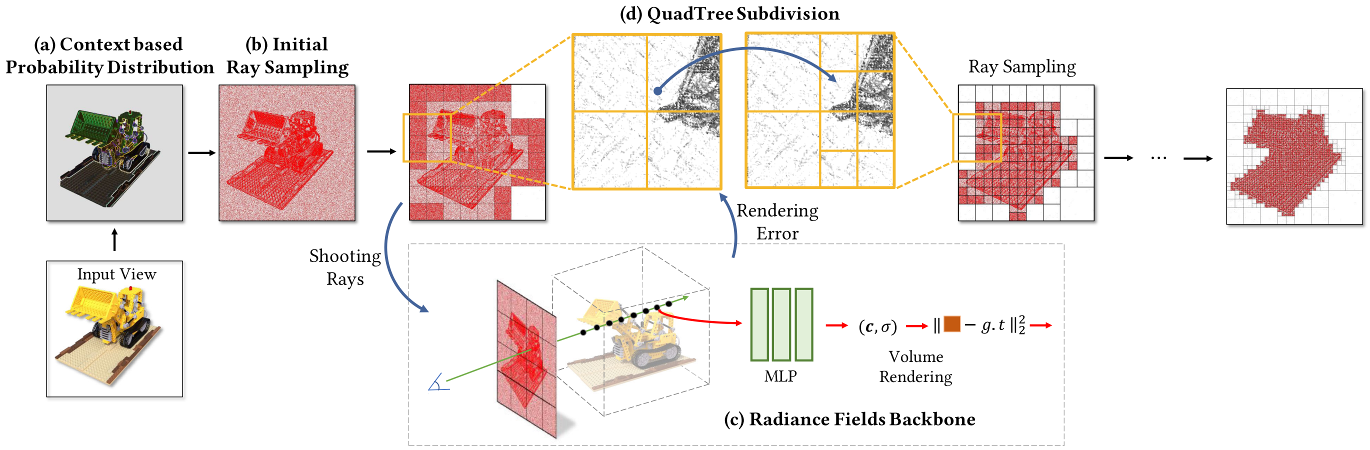


Fig. 2. Overview of our method. Given an input view, we first (a) calculate a probability distribution according to the context around each pixel. Then we (b) use it as a prior probability distribution to sample pixels from which we shoot rays. These rays are fed into (c) a radiance fields backbone network and an average rendering loss is gathered for quadtree leaf nodes. Then (d) an adaptive quadtree subdivision algorithm is applied on each leaf node according to its average rendering loss to adjust the distribution of sampling rays.

the *nontrivial areas* where color changes greatly contain more information, so more rays are required to capture the detailed information and learn how to distinguish these pixels' colors from its neighboring ones.

Figure 3 illustrates an example. We observe that the learning of radiance fields in trivial areas (highlighted by green circles) converges in a very short time, while it takes a long time to finely render the nontrivial areas (highlighted by red circles) with a great change in colors. Therefore, there is no need to keep shooting a large number of rays at the trivial areas for training. In contrast, it makes more sense to shoot more rays in nontrivial areas to perceive the radiance field. Based on the above observation, we propose two strategies to optimize ray distribution on input images. The first one is to calculate a prior probability distribution based on the image context, and the second one is to apply an adaptive quadtree subdivision algorithm to dynamically adjust ray distribution.

C. Context Based Probability Distribution

In order to identify the trivial and nontrivial areas in an image, we introduce the *image context* to capture the area importance of this image, which will be used to non-uniformly sample rays on the image. Specially, we use a center surround mechanism for computing image context, because it has the intuitive appeal of being able to identify areas that are different from their surrounding context. Intuitively, a pixel, that has the same color with its neighbors, has only a few context. In contrast, a pixel, whose color is quite different from its surrounding pixels, has more context. Therefore, we use the color variation of pixels relative to their surroundings to quantitatively identify the image context.

Given an input view before training, we design a probability density function $g(u, v)$, which maps a pixel (u, v) position into a prior probability, indicating how likely a ray is sampled at this position. We define g as the standard variation of the color $\mathbf{c}$ of pixel (u, v) and its 8 one-order neighboring pixels,

$$g(u, v) = \mathrm{std}(\mathbf{c}(u, v)) = \sqrt{\frac{1}{9}\sum_{x,y}[\mathbf{c}(x, y) - \bar{\mathbf{c}}]^2},$$

$$x \in \{u-1, u, u+1\}, \quad y \in \{v-1, v, v+1\}. \tag{4}$$

where $\bar{\mathbf{c}}$ is the mean color among the center pixel (u, v) and its 8 adjacent pixels.

In 2D image space, pixels with high values of g usually correspond to the areas where color changes greatly. Accordingly, in 3D space, these pixels usually correspond to the positions where density changes greatly, and the positions are often on the boundary surfaces of 3D objects. How to accurately detect the surface of objects has proved to be significant to 3D reconstruction and novel views synthesis [12], [34], [35], and our image context based probability distribution function naturally helps to estimate where surfaces are located.

To balance the difference between maximum and minimum values of g, we utilize a normalization operation on g below

$$g'(u, v) = \frac{\mathrm{clamp}(s, \max(g(u, v)))}{\max(g(u, v))}, \tag{5}$$

where we typically define threshold $s = 0.01 \times \mathrm{mean}(g(u, v))$. Values less than s will be clamped to s to avoid sampling too few rays at the corresponding positions. After normalization, $g'(u, v)$ is distributed in interval $[0, 1]$. We use the distribution g' instead of uniform distribution to generate rays for neural radiance field training, i.e.

$$\mathbf{r}_i(u, v) \sim g'(u, v), u \in [0, H_i], v \in [0, W_i]. \tag{6}$$

where $\mathbf{r}_i(u, v)$ is the ray emitted from pixel (u, v) on i-th image.

Compared with the uniform distribution, our probability distribution takes into account the context information of the image. We sample more rays in areas with large probability and sample less rays in the areas with small probability. Figures 5, 6, 7 provide several visualization examples of our sampling strategy. In the lines of "Sampled Rays Distribution", we sample 50% rays according to the context based probability distribution and randomly sample the other 50% rays, where each red point represents a sampled ray. It can be observed that more rays are distributed in the areas with complex shapes and colors, i.e. nontrivial areas, where color changes a lot. On the other hand, fewer rays are distributed in trivial areas, such as the white background, the walls and the floors.

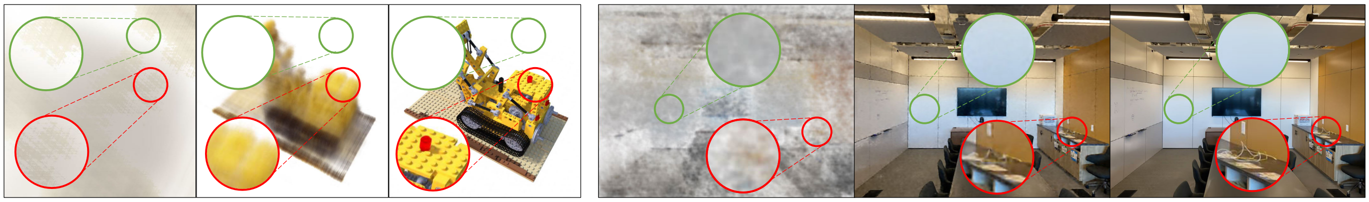

(a) NeRF (15s) (b) NeRF (1min) (c) NeRF (∼8h) (d) Plenoxels (5s) (e) Plenoxels (4min) (f) Plenoxels (∼1.2h)

Fig. 3. Comparison of convergence speed in different areas. (a) - (c) is the training process of NeRF [1] on the synthetic dataset. The blank background area converges rapidly in 1 minute, while the lego building block area doesn't become clear until about 8 hours of training. (d) - (f) is the training process of Plenoxels [2] on the forward-facing dataset. Similarly, the wall of the room becomes the same color in 4 minutes, while the wires on the table cannot be distinguished until 1.2 hours.

D. Adaptive QuadTree Subdivision

Based on the prior probability distribution on images, we further utilize an adaptive quadtree subdivision to adjust the position of shooting rays and to reduce the number of training rays. We firstly construct a quadtree on each input view, in which each leaf node represents one block of the image. As the training procedure goes on, we constantly traverse every leaf node. For each node F, we first sample and shoot rays on F according to the prior probability distribution, and then calculate the average rendering error e_F over all the emitted rays $\mathbf{r}_i \in F$ by

$$e_F = \mathrm{mean}_{\mathbf{r}_i \in F}(V_\Theta(\mathbf{r}_i) - \mathbf{c}_i), \tag{7}$$

where $\mathbf{c}_i$ is the ground truth color of the pixel emitting $\mathbf{r}_i$. If e_F is smaller than the pre-defined threshold a, we conclude that the training in this block has converged well, so the leaf node F is marked to be left aside. Only a few rays will be sampled on F and it won't be subdivided again anymore. Otherwise, if the average rendering error e_F is larger than threshold a, we conclude the rendered color is still far from the ground truth pixel color, which indicates that further training is still needed, so this node will be subdivided into four smaller leaf nodes. The selection of threshold a will be discussed in detail in ablation study in Section IV-E.

Let M_i denote the number of rays emitted from image I_i. Before subdivision, rays are randomly selected on image I_i at the times of total number of pixels, so $M_i = H_i \times W_i$ at the beginning of training, where H_i and W_i are the height and width of I_i. Let Q_1^l and Q_2^l denote the number of unmarked leaf nodes and marked leaf nodes separately, and l denote the times of subdivision. In an unmarked node, the number of sampled rays is equal to the number of pixels of the node, while in a marked node, only a few rays are sampled. Then we get the total number of emitted rays after l times of quadtree subdivision:

$$M_i^l = Q_1^l \times \frac{H_i}{2^l} \times \frac{W_i}{2^l} + Q_2^l \times n_0, \tag{8}$$

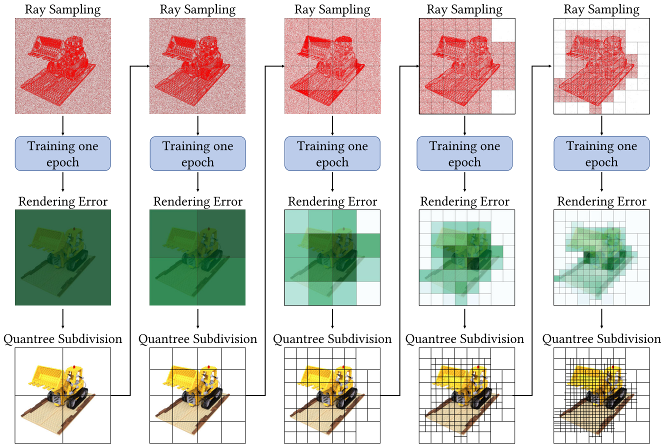


Fig. 4. A demonstration of the overall procedure of ray sampling, training, obtaining rendering error and quadtree subdivision.

where n_0 is a constant number of rays sampled on marked leaf nodes (we typically set $n_0 = 10$ in practice). With the increase of subdivision times, the growth rate of Q_2^l is far lower than that of Q_1^l. As a result, much fewer rays are sampled on marked leaf nodes, so the total amount of rays to be trained will gradually decrease. We draw a flow chart in Figure 4 to illustrate the overall process of ray sampling, training, rendering error and quadtree subdivision as below. This figure will help clarify the sampling and quadtree subdivision procedure.

We noticed that iMap [58] similarly uses rendering error as sampling guidance. Here we clarify the main differences between iMap and our sampling strategies. (1) *The applications of rendering loss are different.* iMap uses the rendering loss distribution on image blocks to decide how many points should be sampled on each block, while we use the rendering loss on each leaf node to decide whether this node should be subdivided into 4 child nodes. (2) *The number of sampled points and image blocks are different.* In our method, the number of sampled points in each leaf node is identical, but the number and area of the image blocks (i.e. leaf nodes) changes during training. In contrast, iMap samples different numbers of rays according to a render loss based distribution in each one of the same size blocks. (3) *The sampling strategies in image blocks are different.* In each image block, iMap uniformly samples points for rendering, while we sample points according to the image context. More points are sampled in the nontrivial areas where color changes a lot, while fewer points are sampled in the trivial areas where color changes slightly. Our sampling strategy helps to capture the detailed information in the nontrivial areas and reduce the training burden in the trivial areas. The ablation study (Table VIII: "Ours w/o prob")

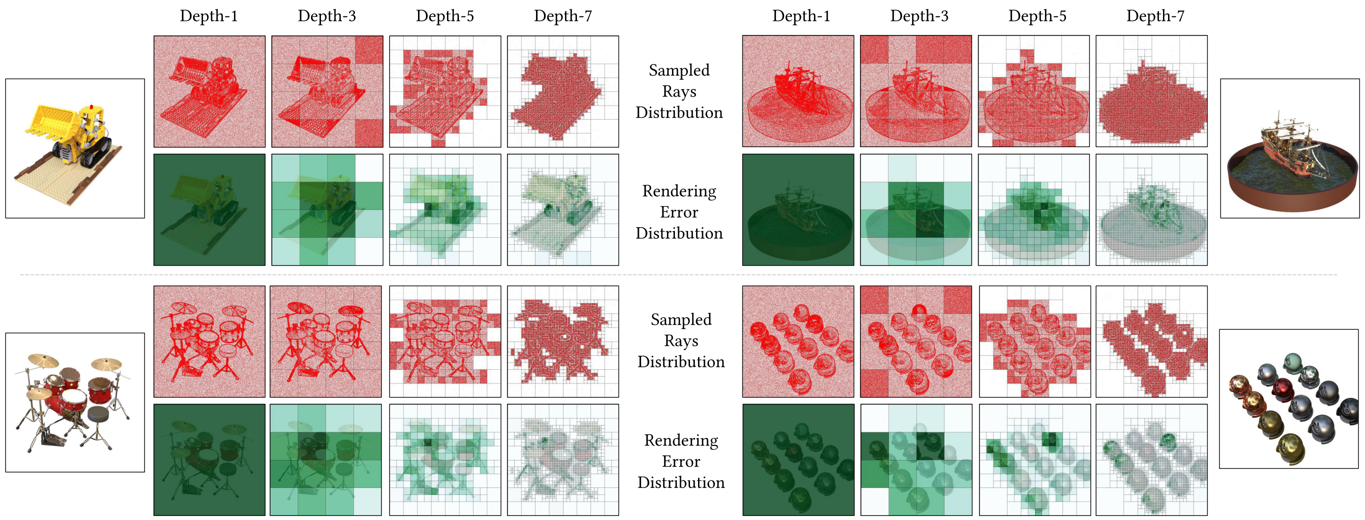

Depth-1
Depth-3
Depth-5
Depth-7
Depth-1
Depth-3
Depth-5
Depth-7
Sampled Rays Distribution
Rendering Error Distribution
Sampled Rays Distribution
Rendering Error Distribution

Fig. 5. Visualization of sampled rays distribution and rendering error distribution at different quadtree depths. We select “Lego”, “Ship”, “Drums”, “Materials” in Synthetic Dataset [1] as examples. In the lines of “Sampled Rays Distribution”, each red point represents a sampled ray and in the lines of “Rendering Error Distribution”, the intensity of green represents the value of average rendering error of quadtree nodes. Dark green represents large rendering error of the node, while light green represents small rendering error.

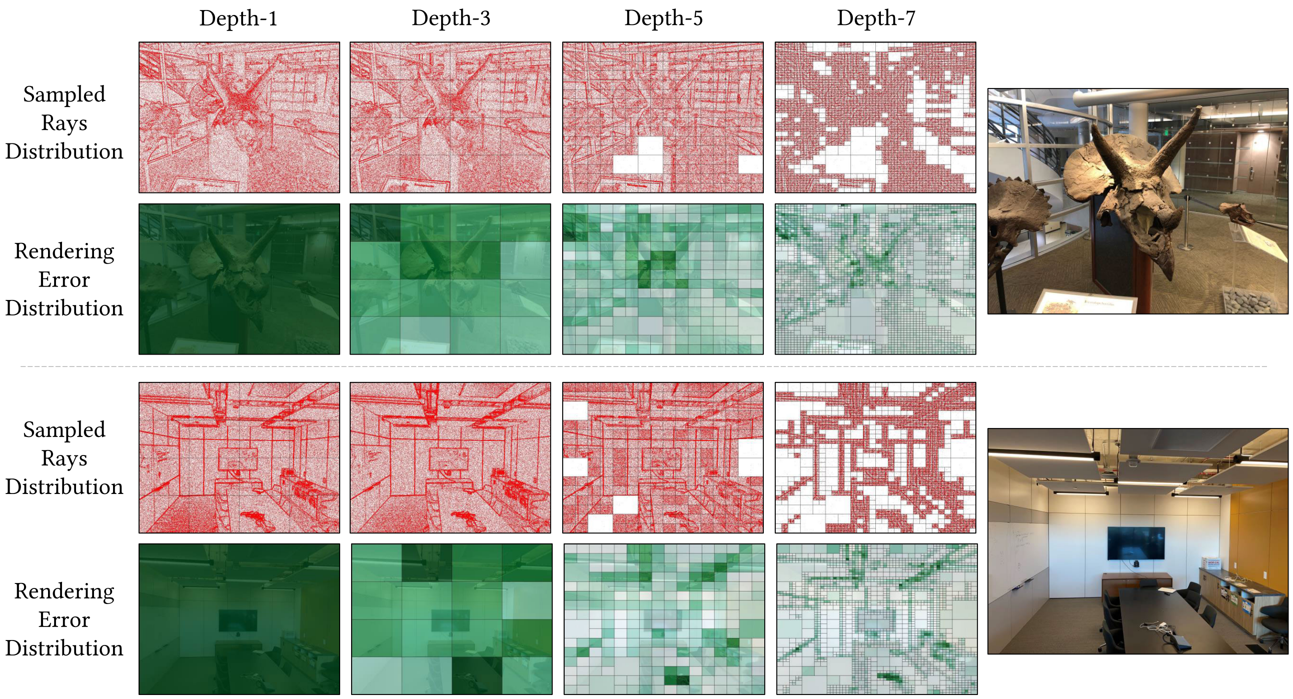

Depth-1
Depth-3
Depth-5
Depth-7
Sampled Rays Distribution
Rendering Error Distribution
Sampled Rays Distribution
Rendering Error Distribution

Fig. 6. Visualization of sampled rays distribution and rendering error distribution at different quadtree depths. We select “horns” and “room” in LLFF Dataset [59] as examples. The meanings of different colors of the images are the same as Figure 5.

further demonstrates that our context-based sampling method is efficient and effective. In conclusion, although rendering loss is both used in our method and iMap, there are essential differences between these two methods on the applications of rendering loss, the number and sampling strategies of points, and the problems to be solved.

Figures 5, 6, 7 demonstrate the process of quadtree subdivision. We select several scenes from different datasets and select quadtree depths of 1, 3, 5, 7 for display. In each scene, red images demonstrate the sampled rays distribution and each red point represents a sampled ray. Green images demonstrate the rendering error distribution and the intensity of green represents the value of average rendering error of quadtree nodes. Dark green represents large rendering error on the node while light green represents small rendering error. The distribution of quadtree leaf nodes and rendering error proves the effectiveness of our subdivision strategy. The chosen subdivided leaf nodes are always around the nontrivial areas, so the distribution of leaf nodes will approach the details of the scene or the silhouette of the object. With the increase

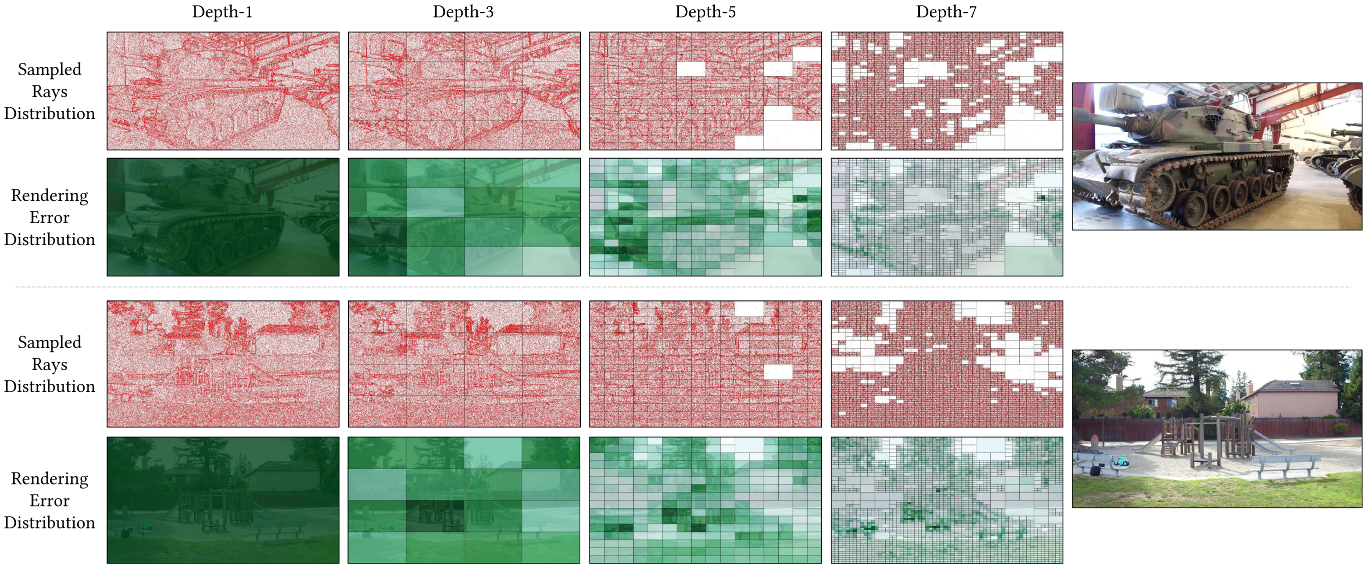


Fig. 7. Visualization of sampled rays distribution and rendering error distribution at different quadtree depths. We select "M60" and "Playground" in Tanks And Temples Dataset [60] as examples. The meanings of different colors of the images are the same as Figure 5.

of subdivision, more leaf nodes are marked and sampled rays are more concentrated in the nontrivial areas, which lead the deep network to learn more details of the scene.

Section III-C and III-D respectively provide the prior and posterior probability distribution for training rays on images. Prior distribution guides how rays are sampled on each leaf nodes, and posterior distribution determines which redundant blocks are to be discarded. Our method of sampling rays not only improves the rendering effect but also reduces the training time. It should be noted that our method is only available for the training stage. During test stage, the number of the rays to be rendered always equals the number of pixels of the image and can not be reduced, so the rendering procedure can not be accelerated through quadtree subdivision or adaptive sampling.

E. Implementation Details

1) All-Pixel Sampling: To avoid under-fitting, we randomly sample rays from all the pixels in the image instead of using quadtrees for sampling at the last epoch. The color and density in 3D space construct a continuous radiance field. The field will be continuously updated when optimizing the network parameters. Even if the training on some area of an image has converged, the corresponding radiance field of the area may change slowly and slightly in account of the changing of its neighborhood fields. Therefore, the rendered color on marked leaf nodes may gradually deviate from the ground truth color. To avoid this problem, when it comes to the last epoch of training, we just randomly sample rays from the whole image instead of using quadtrees for sampling, where the number of sampled rays is equal to the total number of pixels. Through this way, rays on the marked leaf nodes are trained again at the last epoch to finetune the radiance field. Because the rays pruned by quadtree subdivision have reached the local optimum before, they can converge very quickly at the last epoch of finetuning. We find that the operation of *all-pixel sampling* helps to further improve the rendering accuracy, which will be demonstrated in ablation study below.

2) Epoch-Based Subdivision: The subdivision of quadtrees starts only when loss calculation of all the leaf nodes is complete (i.e. at the end of each epoch). If the quadtrees are subdivided during an epoch of training, it will be difficult to count the reduction proportion of sampled rays and to control the subdividing process. Following Plenoxels [2], we also generate training rays for all training views before each epoch starts, unlike NeRF [1] selecting only one view and sampling certain rays from the view at every iteration. At the end of each epoch, we collect the rendering error of rays sampled from each leaf node, which will guide the quadtree subdivision. In this way, we are able to synchronously subdivide all the quadtrees after each epoch ends.

3) QuadTree Initialization: In practice, we initially subdivide the quadtrees into 2 or 3 depths at the begin of training. This helps our method to distinguish the trivial and nontrivial areas faster among the quadtree leaf nodes.

IV. Experiments

A. Experimental Settings

1) Dataset: We evaluate our method quantitatively and qualitatively under four widely used benchmarks for novel view synthesis, including Realistic Synthetic 360° [1], Light Field (LF) [61], Local Light Field Fusion (LLFF) [62], Tanks and Temples (T&T) [60], Real-World 360° [54] and one widely used benchmark for multi-view reconstruction, DTU [63]. Synthetic dataset contains pathtraced images of 8 objects that exhibit complex geometry and realistic non-Lambertian materials. LF dataset is formed by hand-held capturing or a circular boom that rotates around a fixed object. Following NeRF++ [10] settings, we use 4 scenes from the LF dataset: Africa, Basket, Ship and Torch. LLFF dataset consists of 8 complex real-world scenes captured with roughly forward-facing images. T&T dataset consists of hand-held 360° captures of large-scale scenes. In our experiments we use two benchmarks of T&T dataset following the default settings

TABLE I

QUANTITATIVE RESULTS ON REALISTIC SYNTHETIC 360° DATASET. RESULTS ARE AVERAGED OVER THE 8 SYNTHETIC SCENES

Methods	Training Time	Reduction	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [1]	22.0h	23%	31.14	0.950	0.067
Ours(NeRF)	**16.9h**		**31.21**	**0.951**	**0.066**
Plenoxels [2]	25.6min	23%	**30.87**	0.959	**0.051**
Ours(Plenoxels)	**19.6min**		30.84	**0.960**	0.054
Instant [23]	264s	18%	32.04	0.961	0.045
Ours(Instant)	**217s**		**32.44**	**0.962**	**0.043**

of different baselines. The first one contains four scenes with real-world background, including M60, Playground, Train and Truck. The other one contains five scenes with transparent background, including Barn, Caterpillar, Family, Ignatius and Truck. And Real-World 360° dataset consists of 9 indoor and outdoor scenes, each containing a complex central object or area and a detailed background. Additionally, DTU dataset contains 15 scenes with a wide variety of materials, appearance and geometry, including challenging cases for reconstruction algorithms, such as non-Lambertian surfaces and thin structures.

2) Baselines: To demonstrate our method as a general framework that can work for almost all radiance field based methods, we apply our method to six representative methods, including NeRF [1], NeRF++ [10], Plenoxels [2], Instant-NGP [23], Mip-NeRF 360 [54] and NeuS [12]. Specially, NeRF is the pioneer of neural radiance fields, NeRF++ improves NeRF to represent large-scale unbounded 3D scenes, Plenoxels and Instant-NGP is the latest work on speeding up the learning of radiance fields, Mip-NeRF 360 is the latest work on rendering unbounded scenes and NeuS is a widely adopted method which combines radiance field and neural implicit surfaces to reconstruct multi-view surfaces. Moreover, for fair comparison with Plenoxels, we report the results without the octree pruning in 3D space, since the octree pruning affects the number of rays. We report our improvement in terms of both speed and accuracy. The rendering accuracy is evaluated by PSNR, SSIM and LPIPS, which have been widely adopted by most radiance fields based methods (e.g. [2]). Additionally, for DTU dataset, We use Chamfer Distance (CD) [12] to evaluate the quality of reconstructed meshes.

3) Training Details: During training, the quadtree is initialized to 2 depths and subdivided every three epochs with a threshold of 1e-3. On each leaf node, 50% rays are sampled according to the prior distribution and other 50% rays are randomly sampled. The parameter of random sampling ratio will be discussed in detail in ablation study. Additionally, we randomly sample rays across the whole image at the last epoch to finetune the radiance field. All of the training time of experiments is counted on a single NVIDIA 3090Ti GPU.

B. Results on Synthetic Dataset

Table I shows the quantitative comparisons with NeRF [1], Plenoxels [2] and Instant-NGP [23] under Realistic Synthetic 360° dataset. We report our results upon NeRF, Plenoxels and Instant-NGP, as denoted by "Ours(NeRF)", "Ours(Plenoxels)" and "Ours(Instant)", respectively. In Table I, the bold items indicate better results in the numerical comparison, which is the same setting as used below. Our method can save 23% training time (from 22 hours to 16.9 hours) for NeRF and 23% training time (from 25.6 minutes to 19.6 minutes) for Plenoxels on synthetic dataset, where we achieve comparable results to Plenoxels. Althogh Instant-NGP is the state-of-the-art work on accelerating NeRF training and is able to train a radiance field in less than 5 minues, our method upon Instant-NGP further saves 18% training time (from 264s to 217s) and achieves better performance than Instant-NGP. Figure 8 provides some qualitative results between NeRF and our method, where more details are demonstrated in our results. We compare with NeRF in two training settings in the visualization results. In the first setting, we train NeRF in the same time as ours, and in the second setting, we train NeRF in the same epochs as ours, as denoted by "NeRF(Time)" and "NeRF(Epoch)", respectively. The two settings of other methods below are the same as those of NeRF.

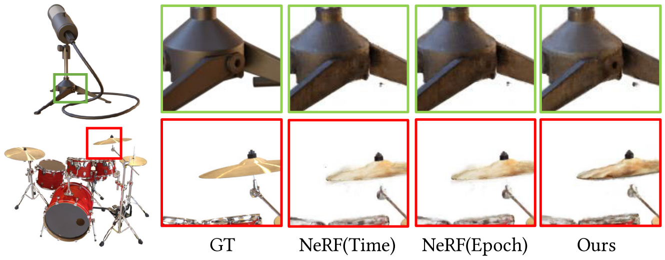


Fig. 8. Qualitative comparison of our method vs. NeRF on the Realisitic Synthetic 360° dataset [1].

We noticed that our numerical results are merely a little bit better than baselines in Table I, III, V. However, the marginal improvements are caused by the optimization of NeRF. A remarkable phenomenon of NeRF-based methods is that the accuracy increases rapidly at the beginning of training, while increasing extremely slowly after some epochs. For example, PSNR of NeRF in synthetic dataset increases less than 1.0 at the last 5 hours, which is demonstrated in (b) and (c) of Figure 1. Additionally, we provide some qualitative results between our method and baseline methods, in which our method shows obvious advantages on the scene details, as shown in Figure 8 - 11 in the paper.

We also conduct an experiment to illustrate the above phenomenon. We train "Plenoxels" and "Ours(Plenoxels)" on the "Lego" scene from synthetic dataset for 10 epochs. We report the PSNR and time after each one of the first 10 epochs. The initial depth of quadtrees is 3 and the quadtrees are subdivided every 2 epochs (same as the settings in paper). The PSNR (the larger the better) and the training time of some epochs are listed in Table II. As shown in the table, at the beginning of training, the PSNR of both our method and Plenoxels increases rapidly after the first epoch. Benefiting from our context-based probability sampling strategy, we achieved larger PSNR improvements over Plenoxels after each epoch. Additionally, as the training progresses, our PSNR improvements over Plenoxels become smaller. It is because that the bottleneck capacity of Plenoxels restricts the upper

TABLE II

THE COMPARISON OF PSNR AND TRAINING TIME BETWEEN PLENOXELS [2] AND OUR METHOD AFTER EACH EPOCH

	Epoch	0	1	2	3	4	5	6	7	8	9
PSNR ↑	Plenoxels [2]	10.427	30.198	31.218	31.696	31.965	32.140	32.245	32.327	32.373	32.405
	Ours	10.427	30.336	31.317	31.755	32.010	32.201	32.315	32.375	32.449	32.464
	Improvements	0.000	**+0.138**	**+0.099**	**+0.059**	**+0.045**	**+0.061**	**+0.070**	**+0.048**	**+0.076**	**+0.059**
Time(s) ↓	Plenoxels [2]	0	176.08	325.85	473.6	619.34	764.34	909.64	1053.4	1197.67	1342.52
	Ours	0	185.65	346.21	465.69	590.4	682.89	789.62	875.16	976.27	1049.91
	Improvements	0.00	+9.57	+20.36	**-7.91**	**-28.94**	**-81.45**	**-120.02**	**-178.24**	**-221.40**	**-292.61**

TABLE III

QUANTITATIVE RESULTS ON FORWARD-FACING DATASET. RESULTS ARE AVERAGED OVER THE 8 SCENES

Methods	Training Time	Reduction	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [1]	4.4h	16%	**25.85**	0.782	**0.288**
Ours(NeRF)	**3.8h**		25.79	**0.784**	0.294
Plenoxels [2]	66.6min	39%	24.60	**0.768**	**0.316**
Ours(Plenoxels)	**40.4min**		**25.12**	0.767	0.322
Instant [23]	350s	15%	26.03	0.810	0.265
Ours(Instant)	**296s**		**26.39**	**0.812**	**0.264**

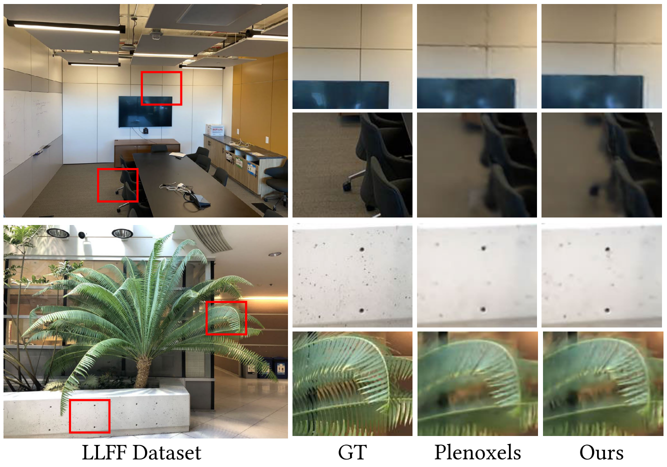


Fig. 9. Qualitative comparison of our method vs. Plenoxels on the LLFF dataset [62]. Take cracks in walls, spots in flower beds for examples.

limit of PSNR. At the same time, our training speed shows significant improvements because of the quadtree subdivision strategy (the tiny increase of training time at the beginning is because of sampling and subdividing). Therefore, these results demonstrate that our method can achieve a balance of efficiency and effectiveness, i.e. better performance improvements at the early training stages or larger training speed improvements at the later training stages.

C. Results on Real-World Dataset

We then evaluate our method under Forward-Facing (LLFF) dataset by comparing with NeRF, Plenoxels and Instant-NGP. As shown in Table III, with the help of our framework, we can speed up NeRF by 16% (from 4.4 hours to 3.8 hours), Plenoxels by 39% (from 66.6 minutes to 40.4 minutes) and Instant-NGP by 15% (from 350s to 296s), respectively. Figure 9 provides some qualitative results between Plenoxels and our method. The results show that our method can predict more details such as the office chair leg, the cracks of the walls, the spots on the flower beds and the gap between leaves in the novel synthetic views.

TABLE IV

QUANTITATIVE RESULTS ON LIGHT FIELD DATASET. RESULTS ARE AVERAGED OVER THE 5 SCENES

Methods	Training Time	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF++(Time)	6.7h	24.59	0.798	0.331
NeRF++(Epoch)	7.6h	25.14	0.811	0.318
Ours(NeRF++)	6.7h	25.06	0.807	0.324

TABLE V

QUANTITATIVE RESULTS ON TANKS AND TEMPLES DATASET [60]. RESULTS ARE AVERAGED OVER ALL THE SCENES

Methods	Training Time	Reduction	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF++ [10]	7.7h	18%	**20.42**	0.659	**0.417**
Ours(NeRF++)	**6.3h**		20.39	**0.663**	0.421
Plenoxels [2]	41.0min	20%	19.88	0.669	0.473
Ours(Plenoxels)	**32.7min**		**19.98**	**0.671**	**0.469**
Instant [23]	354s	19%	27.91	0.832	0.141
Ours(Instant)	**285s**		**28.25**	**0.837**	**0.132**

TABLE VI

QUANTITATIVE RESULTS ON REAL-WORLD 360° DATASET [54]. RESULTS ARE AVERAGED OVER THE 7 SCENES

Methods	Training Time	Reduction	PSNR ↑	SSIM ↑	LPIPS ↓
Mip-NeRF 360 [54]	41.5h	23%	26.45	0.785	0.242
Ours(Mip-NeRF 360)	**31.9h**		**26.89**	**0.787**	**0.240**

TABLE VII

QUANTITATIVE RESULTS ON DTU DATASET [63]. RESULTS ARE AVERAGED OVER THE 15 SCENES

Methods	Training Time	Reduction	PSNR ↑	CD ↓
NeuS [12]	9.2h	21%	31.97	0.87
Ours(NeuS)	**7.3h**		**33.41**	**0.73**

In Table IV, we further report numerical comparison with NeRF++ under Light Field (LF) dataset. Note that almost all follow-up approaches [59], [64], [65] reporting results under LF dataset use different dataset splits, thus their results are not comparable with ours. Hence, we only list the results of NeRF++ and ours on this dataset. We report the result of NeRF++ by training it for the same time or the same number of epochs as ours, as denoted by "NeRF++(Time)"

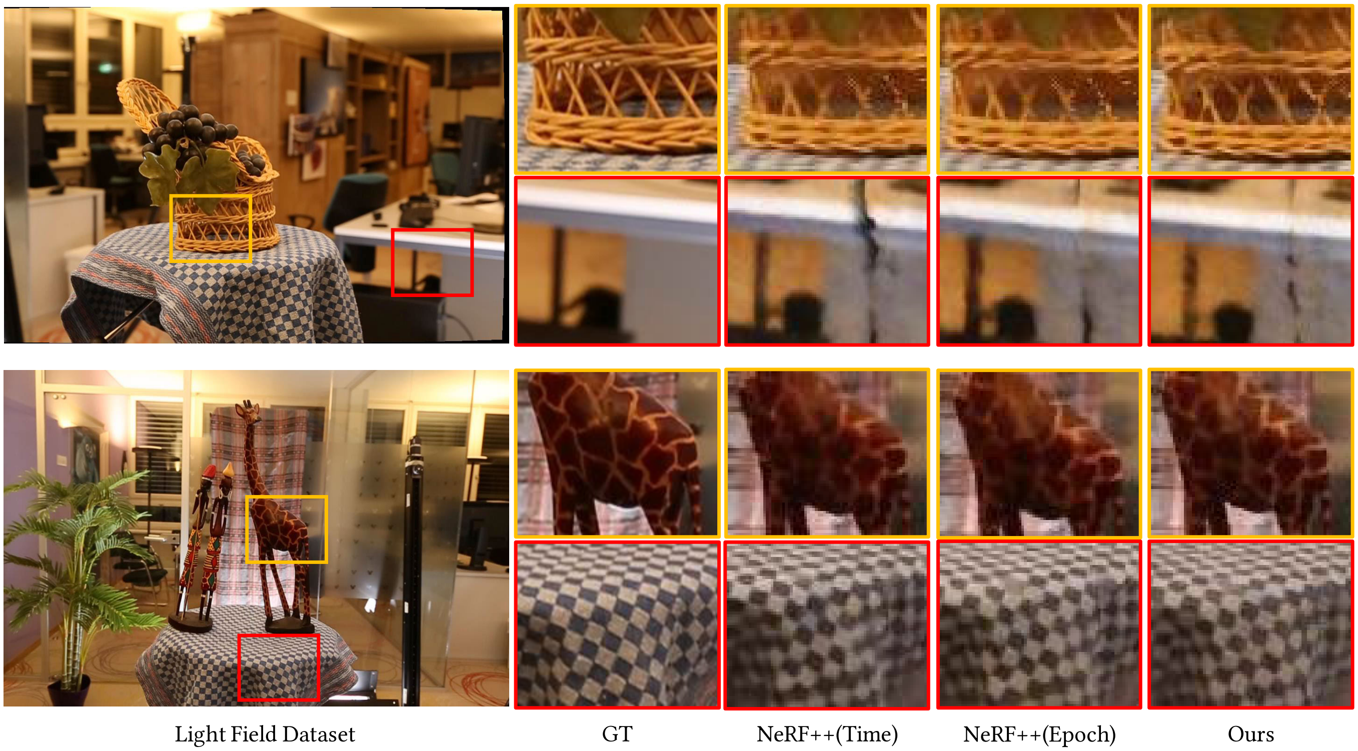


Fig. 10. Qualitative comparison of our method vs. NeRF++ [10] on the Light Field dataset [61]. Comparing with NeRF++, our method is faster and captures the scene details well, taking reticular formation of basket, patterns on giraffes for examples.

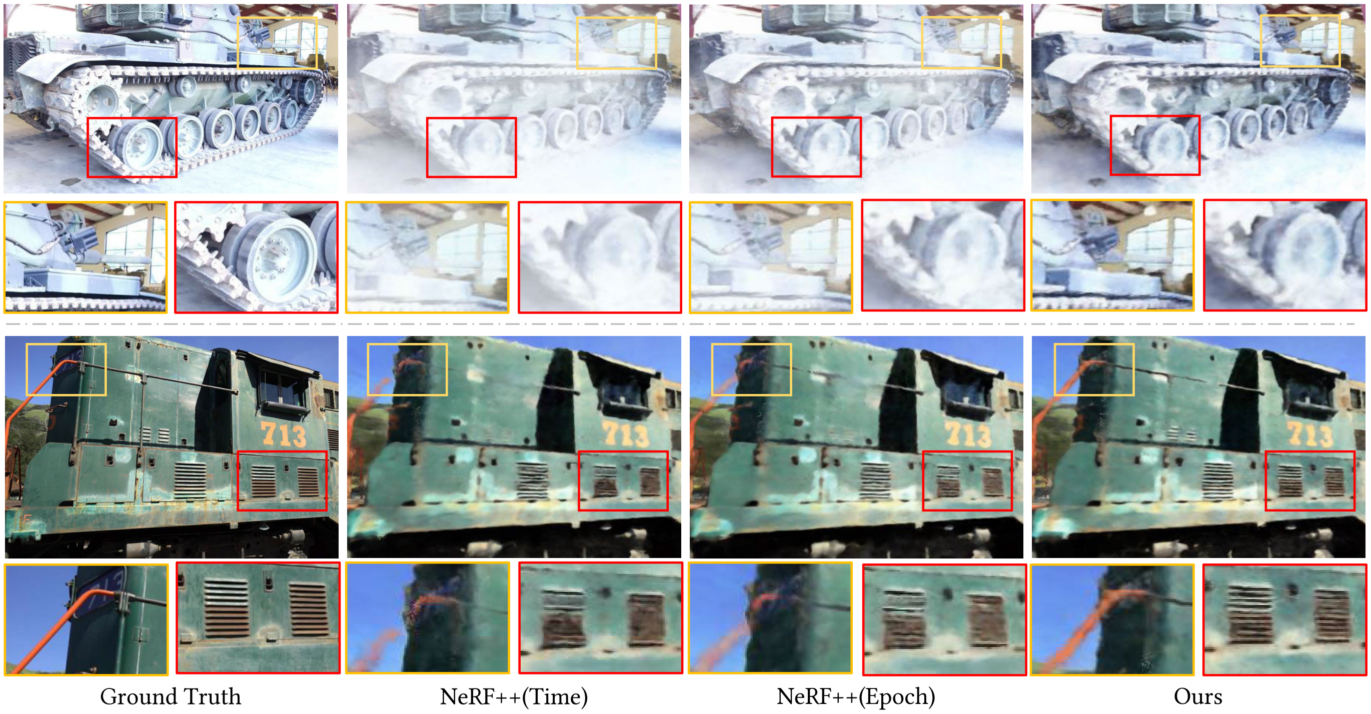


Fig. 11. Qualitative comparison of our method vs. NeRF++ [10] on the Tanks and Temples dataset [60]. Comparing with NeRF++, our method achieves both faster speed and better visual performance, taking gear of tank, exhaust fan of train for examples.

and "NeRF++(Epoch)", respectively. The settings of "Time" and "Epoch" are the same as the ones in Figure 8. With the same training time, our method achieves better accuracy than NeRF++, while NeRF++ achieves comparable results to ours with more time cost. We further highlight our advantage in visual comparison in Figure 10. Our method can reveal more fine texture and structures in novel views.

We report numerical comparisons under Tanks and Temples dataset in Table V, which further demonstrates that our method is a general framework for most radiance fields based methods.

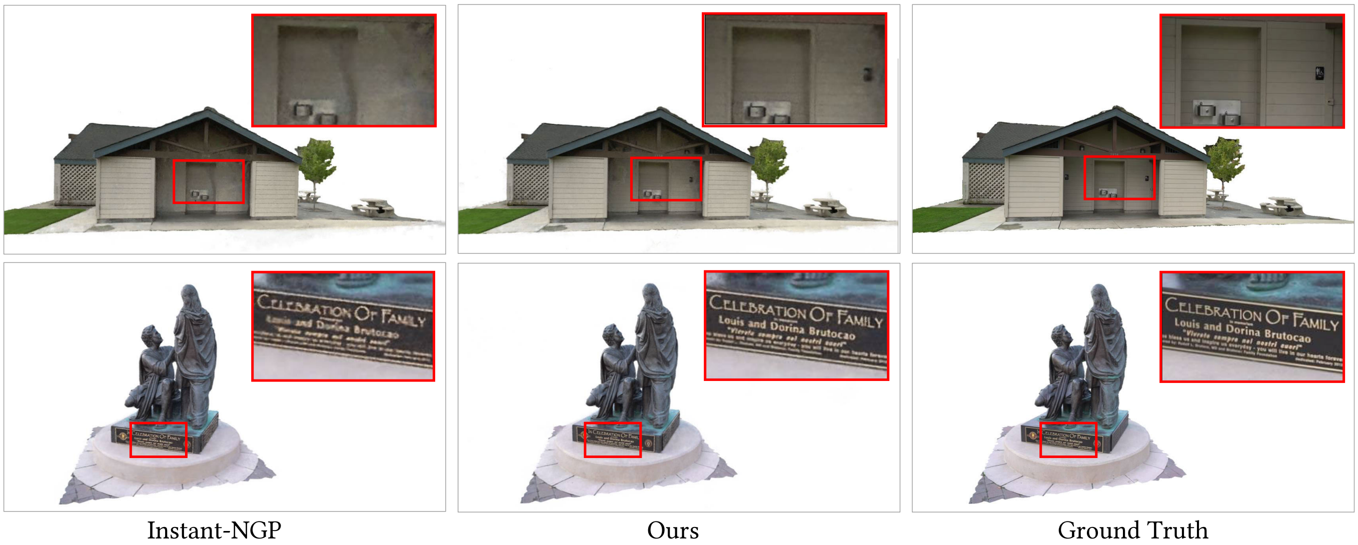


Fig. 12. Qualitative comparison of our method vs. Instant-NGP [23] on the Tanks and Temples dataset [60]. Our method further improves the training speed and the performance of Instat-NGP, which is the state-of-the-art among NeRF accelerating methods.

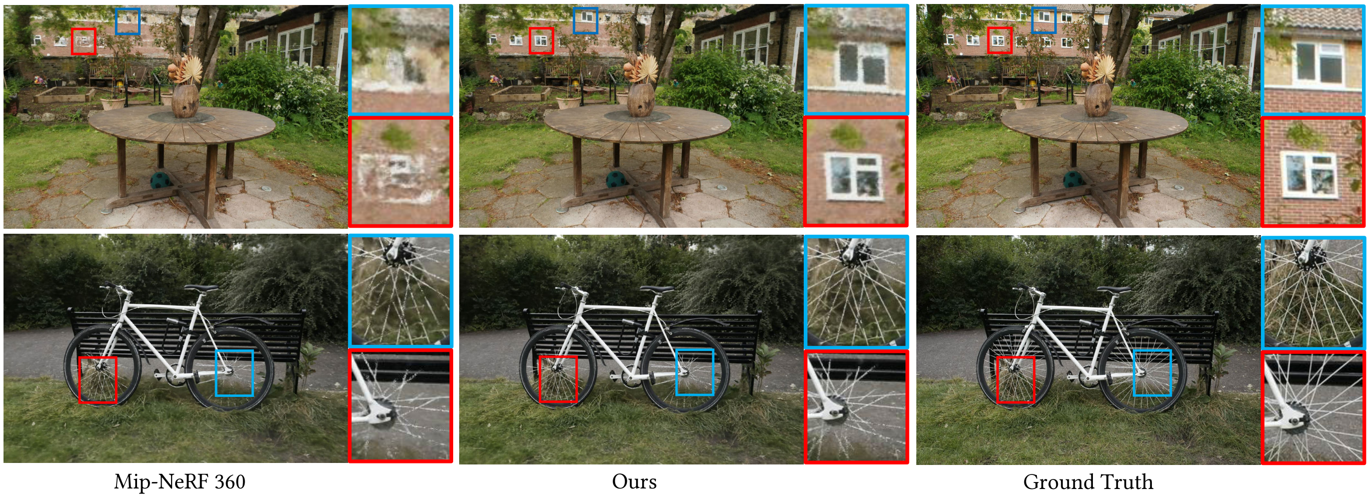


Fig. 13. Qualitative comparison of our method vs. Mip-NeRF 360 [54] on the Real-World 360° [54]. Our method improves both the training speed and the performance of Mip-NeRF 360, which is the state-of-the-art among Realistic 360° rendering methods.

Our methods upon NeRF++, Plenoxels and Instant-NGP can significantly speed up their training without scarifying accuracy. Figure 12 and Figure 11 are visual comparison with NeRF++ and Instant-NGP and our method under T&T dataset. Note that we use the benchmark of 4 scenes with real background for NeRF++ and Plenoxels while use the benchmark of 5 scenes with transparent background for Instant-NGP, following the same settings of the baselines. The results show that our method can show more details with fewer artifacts.

We then evaluate our method by comparing with Mip-NeRF 360 [54] under Real-World 360° dataset [54], as reported in Table VI. Mip-NeRF 360 is the state-of-the-art work on rendering unbounded scenes and our method further speeds up Mip-NeRF 360 by 23% (from 41.5h to 31.9h) and achieves better numerical performance. Figure 13 provides some visual examples between our method and Mip-NeRF 360, where our method shows significant advantages in scene details, such as the windows in the distance.

D. Results on Multi-View-Stereo Dataset

In Sections IV-B and IV-C, we evaluate our method by comparing with several representative methods which focus on novel view synthesis. Our method is able to reduce the training time from 15% to 40% and achieve comparable numerical results under different baselines. However, our method is not limited for novel view synthesis task. To further demonstrate that our method is a general strategy for almost all radiance fields based methods, we integrate our method with NeuS [12], which combines radiance fields and neural implicit surfaces and is designed for multi-view reconstruction task. We evaluate our method under all the 15 scenes on DTU dataset [63] and report the quality of rendering views (evaluated by PSNR) and reconstructed meshes (evaluated by CD) in Table VII. The results show that our methods reduce 21% training time (from 9.2h to 7.3h) for NeuS and achieves better performance on both rendering views and reconstructed meshes. Figure 14 provides visual comparison between our

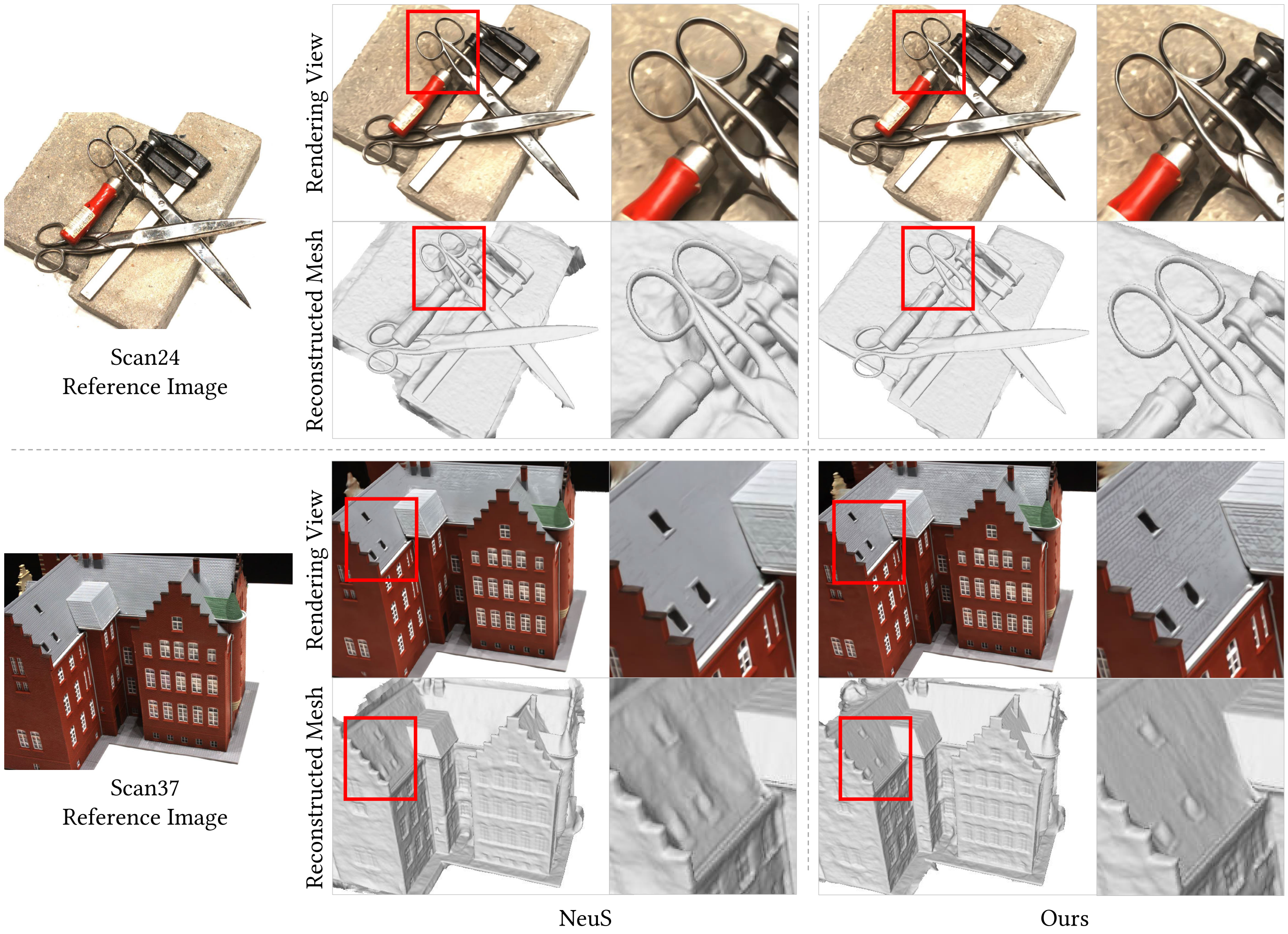
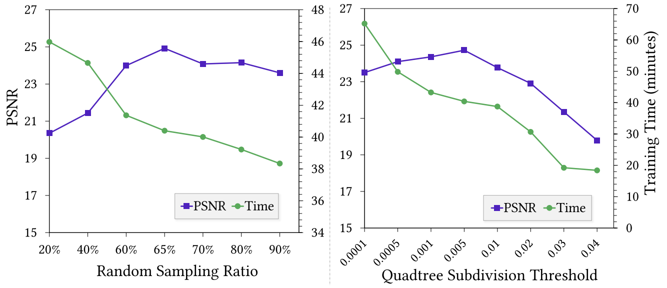


Fig. 14. Qualitative comparison of our method vs. NeuS [12] on DTU dataset [63]. For each scene, we display the reference view, rendering view and the reconstructed mesh between our method and NeuS. The results show that our method can improve both the rendering quality and the reconstruction details of NeuS.



(a) Ablation on Random Sampling Ratio

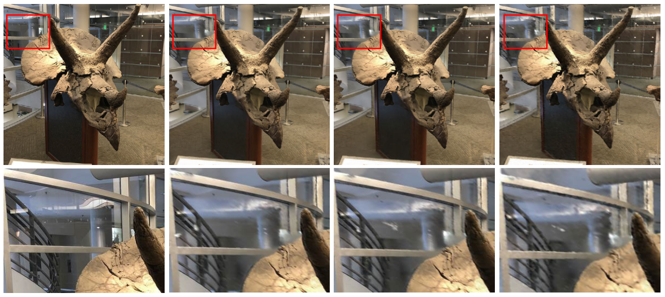
(b) Ablation on Quadtree Subdivision Threshold

Fig. 15. Ablation study on random sampling ratio and quadtree subdivision thresholds.

method and NeuS. With the help of probability sampling and quadtree subdivision, our method is able to render high-fidelity images with smooth clean surfaces, taking scissors handles in scan24 and roofs in scan37 as examples.

E. Ablation Study

1) Ratio of Random Sampling: We find that adding a certain ratio of random sampling in probability distribution sampling is helpful to learn the radiance field better. It is because that the context based probability distribution may ignore some trivial but important areas on the image, such as the areas of smooth surfaces. Moreover, random sampling helps to pay more attention to the trivial areas, which usually have small color change, so this strategy intuitively plays the same role as threshold s in Eq. (5). In this part of ablation study, we freeze quadtree settings and report the correlation between



(a) Ground Truth

(b) Ours (All)
PSNR: 25.40

(c) Ours (w/o prob)
PSNR: 24.33

(d) Ours (w/o allPixel)
PSNR: 23.30

Fig. 16. Qualitative comparison of ablation study on each module of our method.

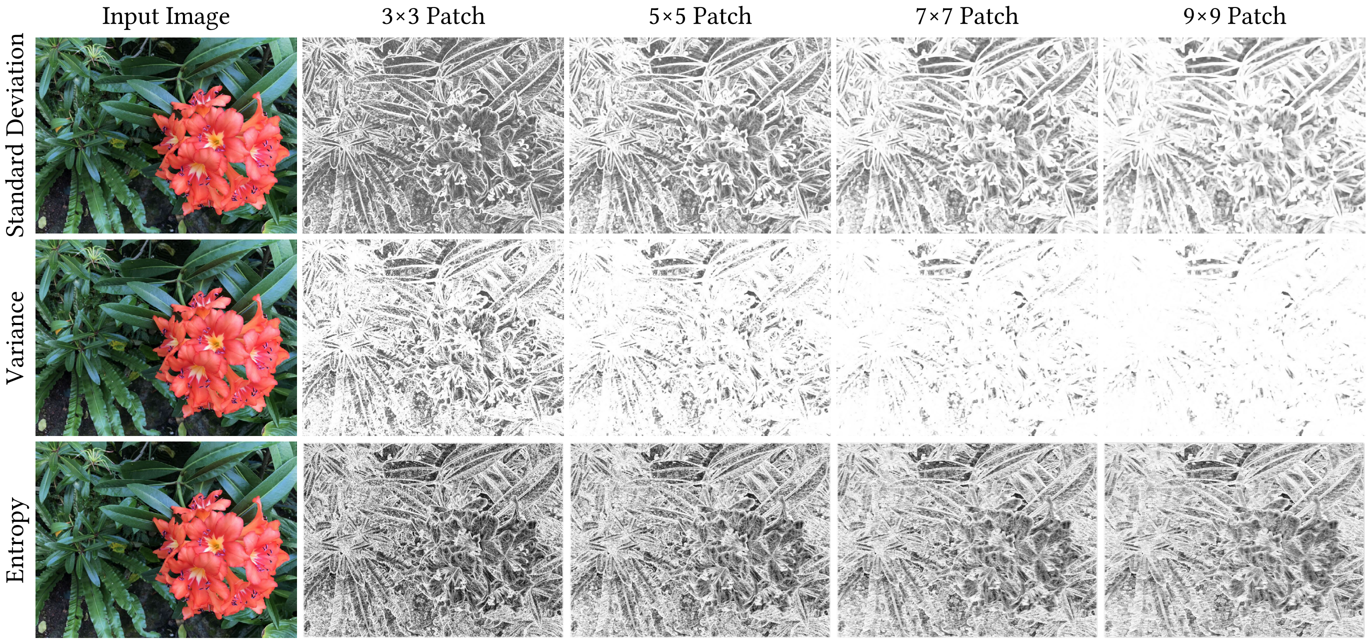


Fig. 17. Visualization of different choices of measuring context based probability distribution. Each line represents patches of different scales, and each column represents different approaches of calculating context information.

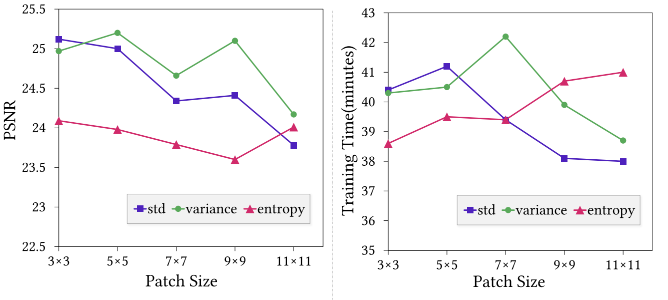
PSNR, training time and random sampling ratio on Forward-Facing dataset [62] based on Plenoxels [2]. The metrics are averaged across the eight scenes of the dataset, as shown in Figure 15 (a). With the increase of ratio from 20% to 90%, a peak value occurs to PSNR, while the training time keeps decreasing. This is because that with the increase of random sampling ratio, our method samples more rays in trivial areas, which (1) changes the ability of learning details, leading to a wave of PSNR, and (2) results in leaf nodes being marked more easily, leading to a decline of training time. The results are not sensitive with large random sampling ratio (>60%) while suffer from low random sampling ratio. It is because that low random sampling ratio leads to a lack of learning in trivial areas, while the metrics do not reflect subtle differences on the ratios with high random sampling ratio.

2) Threshold of Quadtree Subdivision: The threshold a introduced in Section III-D is an important hyper-parameter in our experiments. The leaf node with average rendering error larger than a will be subdivided into four smaller leaf nodes, while the leaf node with error smaller than a will be marked and will not be subdivided anymore. Therefore, the threshold a greatly affects the training speed and effect. In this part of ablation study, we freeze the random sampling ratio as 65% and report the correlation between subdivision threshold a, PSNR and training time on the same dataset as the previous ablation experiment based on Plenoxels [2], as shown in Figure 15 (b). With the increasing of the threshold from 0.0001 to 0.04, the training time keeps decreasing, because larger threshold results in fewer leaf nodes to be subdivided and fewer rays to be trained. However, the metrics are not monotonous with the threshold. If the threshold is too small, most leaf nodes are subdivided, so the rays are sampled on very small leaf nodes, which is similar to random sampling, losing the advantage of our context-based sampling strategy. On the other hand, if the threshold is too large, a leaf node may be marked to be not subdivided anymore although the training on this node has not converged yet.

3) Metrics of Context Based Probability Distribution: In Eq. 4, we propose using the standard deviation of the color in 3×3 patch to represent the probability density of a given pixel. In this part, we compare the performance of different sizes of patches and different kinds of metrics. We first change the receptive field from 3×3 patch to 5×5, 7×7, 9×9 and then change the metric method from standard deviation to variation and entropy. The formulation of variation and entropy are listed below, given 3×3 patch as an example.

$$\begin{aligned}&\text{variance}(\mathbf{c}(u, v)) = \frac{1}{9}\sum_{x,y}[\mathbf{c}(x, y) - \bar{\mathbf{c}}]^2,\\&\text{entropy}(\mathbf{c}(u, v)) = \sum_{x,y} -\mathbf{c}(x, y)\log \mathbf{c}(x, y),\\&x \in \{u-1, u, u+1\},\quad y \in \{v-1, v, v+1\}.\end{aligned} \tag{9}$$

For each combination of patch size and metric method, we visualize the calculated probability distribution of an input image, as shown in Figure 17. The lighter color represents the larger probability in the grayscale image. The variation of patch sizes and metric methods typically change the difference range of probability distribution between trivial areas and nontrivial areas. Therefore, the hyperparameter of patch size and the different choices of metric methods play the same role as random sampling ratio, which also controls the difference range between small probability and large probability. We also report the PSNR and the training time of each combination on LLFF dataset [62] based on Plenoxels [2], as shown in Figure 18. The results show that the performance of entropy metric is a little worse than standard deviation and variance. However, the performances between different patch sizes do



(a) PSNR of Different Patch Sizes and Metrics

(b)Training Time of Different Patch Sizes and Metrics

Fig. 18. Ablation study on different sizes of patches and approaches of context information metrics.

TABLE VIII
ABLATION STUDY ON VARIOUS COMPONENTS OF OUR METHOD

Methods	Training Time	PSNR ↑
Plenoxels	66.6min	24.60
Ours	40.4min	25.12
Ours w/o quadtree	71.5min	**25.31**
Ours w/o prob	38.4min	23.97
Ours w/o allPixel	**37.8min**	23.01
Ours w/o random	42.8min	23.39

not differ a lot, which means that patch size is an insensitive hyperparameter.

4) Ablation Studies on Other Modules: As reported in Table VIII and visualized in Figure 16, we perform extensive ablation studies of various components of our method to demonstrate their effectiveness. Our experiments are based on Plenoxels [2] under the Forward-Facing dataset [62]. We first remove quadtree subdivision ("w/o quadtree"), which takes more time than Plenoxels to converge. This is because probability sampling takes some extra time in traversing leaf nodes and calculating probability. We then remove probability distribution sampling ("w/o prob"), which takes less time, but the performance greatly degenerates. Our probability sampling method takes into account the context information of the image and dynamically shoot more rays in more complex regions. Therefore it significantly improves the performance. In the third experiment, we remove all-pixel sampling in the last epoch ("w/o allPixel"), which takes the shortest time and shows the lowest accuracy. It is because that fewer rays are sampled at the last epoch without the all-pixel sampling strategy, which leads to a slight decrease of training time. At last, we remove random sampling ratio ("w/o random"), and we find that both of the time and the performance degenerates slightly. The reason is detailed in the previous ablation section. Our method combines the above strategies and takes a balance between time and accuracy.

V. CONCLUSION

We introduce a general framework to speed up the learning of radiance fields. Our method successfully leverages a context based probability distribution and adaptive quadtree subdivision to shoot much fewer rays in volume rendering without sacrificing accuracy. Different from the existing approaches which require specific data structures or networks, our method can effectively speed up the training of almost all radiance fields based methods. Our analysis justifies that our method can dynamically shoot more rays to perceive the regions with complex geometry and shoot much fewer rays in simple regions, which significantly reduces redundancy in shooting rays. The evaluation under the widely used benchmarks shows that our method can significantly speed up the learning of radiance fields and achieve comparable accuracy for different methods.

ACKNOWLEDGMENT

Project Page is available at https://zparquet.github.io/Fast-Learning and code is available at https://github.com/zParquet/Fast-Learning.

REFERENCES

[1] 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 *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 405–421.

[2] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5501–5510.

[3] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4104–4113.

[4] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Fine-grained 3D shape classification with hierarchical part-view attention," *IEEE Trans. Image Process.*, vol. 30, pp. 1744–1758, 2021.

[5] Z. Han, X. Wang, Y.-S. Liu, and M. Zwicker, "Hierarchical view predictor: Unsupervised 3D global feature learning through hierarchical prediction among unordered views," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 3862–3871.

[6] X. Wen, J. Zhou, Y.-S. Liu, H. Su, Z. Dong, and Z. Han, "3D shape reconstruction from 2D images with disentangled attribute flow," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 3803–3813.

[7] Z. Han, C. Chen, Y.-S. Liu, and M. Zwicker, "DRWR: A differentiable renderer without rendering for unsupervised 3D structure learning from silhouette images," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3994–4005.

[8] M. Baorui, H. Zhizhong, L. Yu-Shen, and Z. Matthias, "Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 7246–7257.

[9] J. Zhou, B. Ma, L. Yu-Shen, F. Yi, and H. Zhizhong, "Learning consistency-aware unsigned distance functions progressively from raw point clouds," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2022.

[10] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "NeRF++: Analyzing and improving neural radiance fields," 2020, *arXiv:2010.07492*.

[11] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "PixelNeRF: Neural radiance fields from one or few images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 4578–4587.

[12] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 27171–27183.

[13] Q. Xu et al., "Point-NeRF: Point-based neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5438–5448.

[14] Q. Wang et al., "IBRNet: Learning multi-view image-based rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 4690–4699.

[15] H.-X. Yu, L. Guibas, and J. Wu, "Unsupervised discovery of object radiance fields," in *Proc. Int. Conf. Learn. Represent.*, 2022.

[16] Y.-J. Yuan, Y.-T. Sun, Y.-K. Lai, Y. Ma, R. Jia, and L. Gao, "NeRF-editing: Geometry editing of neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 18353–18364.

[17] K. Zhang et al., "ARF: Artistic radiance fields," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2022, pp. 717–733.

[18] 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 18342–18352.

[19] C. Wang, M. Chai, M. He, D. Chen, and J. Liao, "CLIP-NeRF: Text-and-image driven manipulation of neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 3835–3844.

[20] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole, "Zero-shot text-guided object generation with dream fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 867–876.

[21] S. Yao, R. Zhong, Y. Yan, G. Zhai, and X. Yang, "DFA-NeRF: Personalized talking head generation via disentangled face attributes neural rendering," 2022, *arXiv:2201.00791*.

[22] C. Sun, M. Sun, and H.-T. Chen, "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5449–5459.

[23] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022.

[24] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "TensoRF: Tensorial radiance fields," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2022, pp. 333–350.

[25] T. Hu, S. Liu, Y. Chen, T. Shen, and J. Jia, "EfficientNeRF–efficient neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12902–12911.

[26] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the wild: Neural radiance fields for unconstrained photo collections," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7210–7219.

[27] A. Jain, M. Tancik, and P. Abbeel, "Putting NeRF on a diet: Semantically consistent few-shot view synthesis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5885–5894.

[28] M. Niemeyer, J. T. Barron, B. Mildenhall, M. S. M. Sajjadi, A. Geiger, and N. Radwan, "RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5470–5480.

[29] M. Kim, S. Seo, and B. Han, "InfoNeRF: Ray entropy minimization for few-shot neural volume rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12902–12911.

[30] K. Rematas, R. Martin-Brualla, and V. Ferrari, "ShaRF: Shape-conditioned radiance fields from a single view," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8948–8958.

[31] S.-Y. Su, F. Yu, M. Zollhöfer, and H. Rhodin, "A-NeRF: Articulated neural radiance fields for learning human shape, appearance, and pose," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 12278–12291.

[32] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "NeRF–: Neural radiance fields without known camera parameters," 2021, *arXiv:2102.07064*.

[33] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 1323–1330.

[34] M. Oechsle, S. Peng, and A. Geiger, "UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5589–5599.

[35] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 4805–4815.

[36] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron, "NeRFactor: Neural factorization of shape and reflectance under an unknown illumination," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 1–18, Dec. 2021.

[37] 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7495–7504.

[38] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt, "Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12959–12970.

[39] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-NeRF: Neural radiance fields for dynamic scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10318–10327.

[40] Y. Du, Y. Zhang, H.-X. Yu, J. B. Tenenbaum, and J. Wu, "Neural radiance flow for 4D view synthesis and video processing," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 14304–14314.

[41] Y. Guo, K. Chen, S. Liang, Y.-J. Liu, H. Bao, and J. Zhang, "AD-NeRF: Audio driven neural radiance fields for talking head synthesis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5784–5794.

[42] A. Noguchi, X. Sun, S. Lin, and T. Harada, "Neural articulated radiance field," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5762–5772.

[43] K. Park et al., "Nerfies: Deformable neural radiance fields," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5865–5874.

[44] K. Park et al., "HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 1–12, Dec. 2021.

[45] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 15651–15663.

[46] D. Rebain, W. Jiang, S. Yazdani, K. Li, K. M. Yi, and A. Tagliasacchi, "DeRF: Decomposed radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14153–14161.

[47] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin, "Fast-NeRF: High-fidelity neural rendering at 200FPS," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 14346–14355.

[48] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 14335–14345.

[49] T. Neff et al., "DONeRF: Towards real-time rendering of compact neural radiance fields using depth oracle networks," *Comput. Graph. Forum*, vol. 40, no. 4, pp. 45–59, 2021.

[50] D. B. Lindell, J. N. P. Martel, and G. Wetzstein, "AutoInt: Automatic integration for fast neural volume rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14556–14565.

[51] A. Chen et al., "MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 14124–14133.

[52] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised NeRF: Fewer views and faster training for free," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022.

[53] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5855–5864.

[54] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded anti-aliased neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5470–5479.

[55] R. Li, M. Tancik, and A. Kanazawa, "NeRFAcc: A general NeRF acceleration toolbox," 2022, *arXiv:2210.04847*.

[56] V. Saragadam, J. Tan, G. Balakrishnan, R. G. Baraniuk, and A. Veeraraghavan, "MINER: Multiscale implicit neural representation," in *Proc. 17th Eur. Conf. Comput. Vis. (ECCV)*, Tel Aviv, Israel. Cham, Switzerland: Springer, Oct. 2022, pp. 318–333.

[57] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "PlenOctrees for real-time rendering of neural radiance fields," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5752–5761.

[58] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "iMAP: Implicit mapping and positioning in real-time," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6229–6238.

[59] G. Wu, Y. Liu, L. Fang, Q. Dai, and T. Chai, "Light field reconstruction using convolutional network on EPI and extended applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 7, pp. 1681–1694, Jul. 2019.

[60] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, Jul. 2017.

[61] K. Yücer, A. Sorkine-Hornung, O. Wang, and O. Sorkine-Hornung, "Efficient 3D object segmentation from densely sampled light fields with applications to 3D reconstruction," *ACM Trans. Graph.*, vol. 35, no. 3, pp. 1–15, Jun. 2016.

[62] B. Mildenhall et al., “Local light field fusion: Practical view synthesis with prescriptive sampling guidelines,” *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–14, 2019.

[63] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs, “Large scale multi-view stereopsis evaluation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 406–413.

[64] H. Turki, D. Ramanan, and M. Satyanarayanan, “Mega-NeRF: Scalable construction of large-scale NeRFs for virtual fly-throughs,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12922–12931.

[65] J. Shen, A. Agudo, F. Moreno-Noguer, and A. Ruiz, “Conditional-flow NeRF: Accurate 3D modelling with reliable uncertainty quantification,” 2022, *arXiv:2203.10192*.

Wenyuan Zhang received the B.S. degree from the School of Software, Tsinghua University, China, in 2021, where he is currently pursuing the Ph.D. degree. His research interests include deep learning, novel view synthesis, and 3D scene reconstruction.

Ruofan Xing is currently pursuing the bachelor’s degree with the School of Software, Tsinghua University. His research interests include deep learning, novel view synthesis, and point cloud upsampling.

Yunfan Zeng is currently pursuing the bachelor’s degree with the Department of Computer Science and Technology, Tsinghua University. His research interests include deep learning, volume rendering, and novel view synthesis.

Yu-Shen Liu (Member, IEEE) received the B.S. degree in mathematics from Jilin University, China, in 2000, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2006. From 2006 to 2009, he was a Postdoctoral Researcher with Purdue University. He is currently an Associate Professor with the School of Software, Tsinghua University. His research interests include 3D computer vision, digital geometry processing, and 3D reconstruction.

Kanle Shi received the B.S. and Ph.D. degrees from Tsinghua University in 2007 and 2012, respectively. From 2012 to 2013, he was a Postdoctoral Researcher in computational geometry with INRIA. From 2014 to 2019, he was an Assistant/Associate Researcher with the School of Software, Tsinghua University. He is currently a Researcher with Kuaishou Technology. His research interests include computer graphics, geometry, and machine learning.

Zhizhong Han received the Ph.D. degree from Northwestern Polytechnical University, China, in 2017. He was a Postdoctoral Researcher with the Department of Computer Science, University of Maryland, College Park, MD, USA. He is currently an Assistant Professor of computer science with Wayne State University, USA. His research interests include 3D computer vision, digital geometry processing, and artificial intelligence.