Tri-MipRF: Tri-Mip Representation for Efficient Anti-Aliasing Neural Radiance Fields

Wenbo Hu[1] Yuling Wang[1,2] Lin Ma[1] Bangbang Yang[1]
Lin Gao[3] Xiao Liu[1] Yuewen Ma[1†]
[1]PICO, ByteDance, Beijing [2]Tsinghua University [3]Institute of Computing Technology, CAS

Abstract

Despite the tremendous progress in neural radiance fields (NeRF), we still face a dilemma of the trade-off between quality and efficiency, e.g., MipNeRF [3] presents fine-detailed and anti-aliased renderings but takes days for training, while Instant-ngp [37] can accomplish the reconstruction in a few minutes but suffers from blurring or aliasing when rendering at various distances or resolutions due to ignoring the sampling area. To this end, we propose a novel Tri-Mip *encoding (à la "mipmap") that enables both instant reconstruction and anti-aliased high-fidelity rendering for neural radiance fields. The key is to factorize the pre-filtered 3D feature spaces in three orthogonal mipmaps. In this way, we can efficiently perform 3D area sampling by taking advantage of 2D pre-filtered feature maps, which significantly elevates the rendering quality without sacrificing efficiency. To cope with the novel* Tri-Mip *representation, we propose a cone-casting rendering technique to efficiently sample anti-aliased 3D features with the* Tri-Mip *encoding considering both pixel imaging and observing distance. Extensive experiments on both synthetic and real-world datasets demonstrate our method achieves state-of-the-art rendering quality and reconstruction speed while maintaining a compact representation that reduces* 25% *model size compared against Instant-ngp. Code is available at the project webpage:* https://wbhu.github.io/projects/Tri-MipRF

1. Introduction

Neural radiance field (NeRF) [35], emerged as a groundbreaking implicit 3D representation, models the geometry and view-dependent appearance by a multi-layer perceptron (MLP) for rendering photo-realistic novel views. MipNeRF [3] further pushes the boundaries of rendering quality by integrated position encoding to model the pre-filtered radiance fields. Such impressive visual quality, however, requires expensive computation in both reconstruction and rendering stages, *e.g.*, MipNeRF [3] takes more than three days for the reconstruction and minutes for rendering a frame. On the other hand, recent works proposed explicit or hybrid

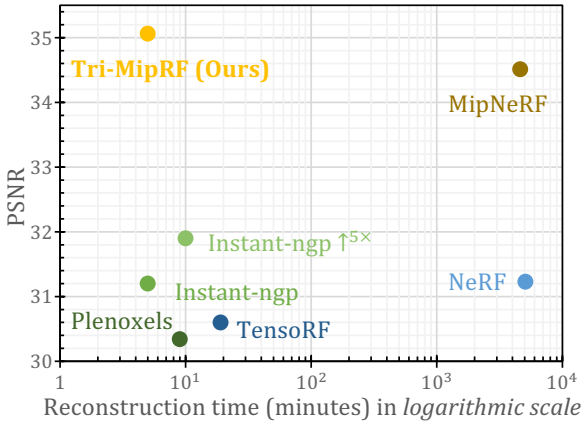


Figure 1. Rendering quality *vs*. reconstruction time on the multi-scale Blender dataset [3]. Our Tri-MipRF achieves state-of-the-art rendering quality while can be reconstructed efficiently, compared with cutting-edge radiance fields methods, *e.g.*, NeRF [35], MipNeRF [3], Plenoxels [14], TensoRF [9], and Instant-ngp [37]. Equipping Instant-ngp with super-sampling (named Instant-ngp $\uparrow^{5\times}$) improves the rendering quality to a certain extent but significantly slows down the reconstruction.

representation for efficient rendering [43, 17, 56, 20, 10, 6], or reconstruction [14, 47, 9, 37], *e.g.*, the hash encoding [37] greatly reduces the reconstruction time from days to minutes and achieves real-time rendering. But all their rendering model is flawed in point-based sampling, which would cause the renderings excessively blurred in close-up views and aliased in distant views. We face a dilemma of the trade-off between quality and efficiency due to the lacking of a representation to support efficient area sampling.

In this paper, we aim to design a radiance field representation that supports both high-fidelity anti-aliased renderings and efficient reconstruction. To address the aliasing and blurring issue, super-sampling and pre-filtering (*a.k.a.* area-sampling) are two popular streams of strategies in the offline and real-time rendering literature, respectively. But super-sampling each pixel by casting multiple rays through its footprint significantly increases the computation cost, and directly pre-filtering the 3D volume is also memory- and computation-intensive, which conflicts with the goal of efficiency. Also, it is not trivial to pre-filter the radiance field represented with hash encoding, due to the hash col-

†Corresponding author.

lisions. We achieve this challenging goal with our novel Tri-Mip radiance fields (Tri-MipRF). As shown in Fig. 1, our Tri-MipRF achieves state-of-the-art rendering quality that presents high-fidelity details in close-up views and is free of aliasing in distant views. Meanwhile, it can be reconstructed super-fast, *i.e.*, within five minutes on a single GPU, while the super-sampling variant of hash encoding, Instant-ngp $\uparrow^{5\times}$, takes around ten minutes for the reconstruction and has much lower rendering quality.

The key to achieving our goal is the proposed Tri-Mip encoding, *i.e.*, featurizing the 3D space by three 2D mip (*multum in parvoto*) maps. The Tri-Mip encoding first decomposes the 3D space into three planes (plane_{XY}, plane_{XZ}, and plane_{YZ}) inspired by the factorization for 3D content generation in [8], and then represent each plane by a mipmap. It ingeniously models the pre-filtered 3D feature space by taking advantage of different levels of the 2D mipmaps. Our Tri-MipRF belongs to the hybrid representation since it models the radiance fields by Tri-Mip encoding and a tiny MLP, which makes it converge fast during the reconstruction. And the model size of our method is relatively compact since the MLP is very shallow and the Tri-Mip encoding only requires three 2D maps to store the base levels of the mipmaps. To cope with the Tri-Mip encoding, we propose an efficient cone-casting rendering technique that formulates the pixel as a disc and emits a cone for each pixel. Different from MipNeRF [3] that samples the cone with multivariate Gaussian, we adopt spheres that are inscribed with the cone. The spheres are further featurized by the Tri-Mip encoding according to their occupied area. The reason for doing so is that the features in mipmaps are pre-filtered isotropically. The Tri-Mip encoding models the pre-filtered 3D feature space while the cone-casting is adaptive to the rendering distance and resolution, and they are effectively connected by the occupied area of the sampling sphere, which makes the renderings of our Tri-MipRF free of blurring in close-up views and aliasing in distant views. Besides, we also develop a hybrid volume-surface rendering strategy to enable real-time rendering on consumer-level GPUs, *e.g.*, 60 FPS on an Nvidia RTX 3060 graphics card.

We extensively evaluated our Tri-MipRF on both public benchmarks and images captured in the wild. Both quantitative and qualitative results demonstrate the effectiveness of our method for high-fidelity rendering and fast reconstruction. Our contributions are summarized below.

- We propose a novel Tri-Mip encoding to model the pre-filtered 3D feature space by leveraging multi-level 2D mipmaps, which enables anti-aliased volume rendering with efficient area sampling.
- We propose a new cone-casting rendering technique that efficiently emits a cone for each pixel while gracefully sampling the cone with spheres on the Tri-Mip encoded 3D space.
- Our method achieves both state-of-the-art rendering quality and reconstruction speed (within five minutes on a single GPU), while still maintaining a compact representation (with a 25% smaller model size than Instant-ngp). Thanks to the hybrid volume-surface rendering strategy, our method also achieves real-time rendering when deploying on consumer-level devices.

2. Related Work

Anti-aliasing in rendering. Anti-aliasing is a fundamental problem in computer graphics and image processing and has been extensively explored in the rendering community. Mathematically, aliasing is the effect of overlapping frequency components resulting from an insufficient sampling rate. Super-sampling and pre-filtering (area-sampling) are two typical streams of approaches to reduce the aliasing artifacts in offline and real-time rendering algorithms, respectively. Super-sampling anti-aliasing (SSAA) methods [15, 11, 32, 19, 51] directly increases the sampling rate to approach the Nyquist frequency, and multi-sampling anti-aliasing (MSAA) [1] is the de facto method supported by modern graphics processors and APIs. Pre-filtering-based methods [25, 39, 22, 2, 52, 23] relieve this burden by pre-compute the filtered version of content ahead of rendering, thus, this streams of methods are more suitable for real-time rendering.

In the context of NeRF, super-sampling can be achieved by casting multiple rays per pixel and aggregating rendered results to produce the final color. This straightforward strategy is useful but expensive since the computation cost grows significantly with the sampling rate increasing. On the other hand, recent works [3, 4, 28] introduce the pre-filtering idea into neural radiance fields by the proposed integrated position encoding or band-limited coordinate networks to learn a pre-filtered representation of the scene, such that the renderings of them are free of blurring in close-up views and aliasing in distant views. However, the rendering and reconstruction of them are extremely slow, *e.g.*, MipNeRF [3] takes around three days to reconstruct a scene and minutes to render a frame, which hinders the applicability. In contrast, our Tri-MipRF can be reconstructed within *five minutes* and achieves real-time rendering on the same hardware, meanwhile, our method even has better-rendering quality in both close-up and distant views compared with MipNeRF.

Accelerating NeRF. NeRF [35] implicitly represents the scene in the MLP, which leads to a very compact storage, but the reconstruction and rendering of it are extremely slow. A thread of works is devoted to speeding up the rendering, by splitting a scene into many cells [42, 43] to reduce the inference complexity, learning to reduce samples per ray [27, 38], or caching trained fields values [20, 17, 56, 6] to reduce the computation in rendering. Another line of works focuses on reducing the reconstruction time by directly optimizing the explicit representation [14, 47], or utilizing hybrid representations, *e.g.*, low-rank tensor [9] and hash table [37], to speed up the converging. Especially, hash encoding achieves instant reconstruction in around five minutes and rendering in real-time.

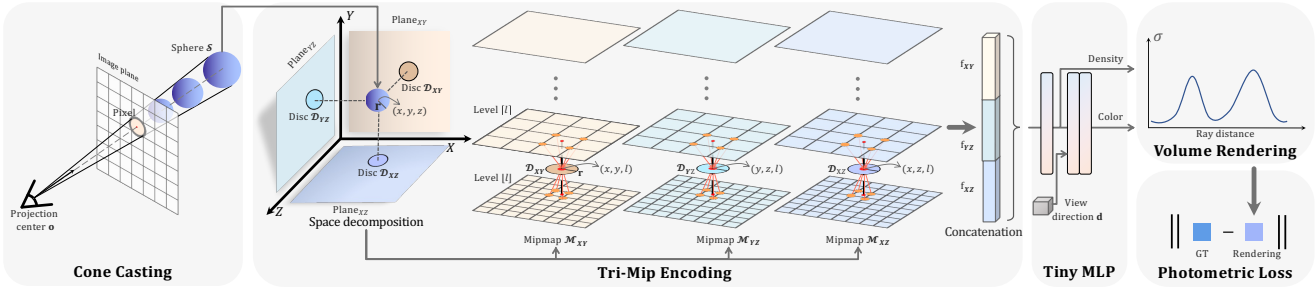


Figure 2. Overview of our Tri-MipRF. To render a pixel, we emit a cone from the camera's projection center to the pixel on the image plane, and then we cast a set of spheres inside the cone, next, the spheres are orthogonally projected on the three planes and featurized by our Tri-Mip encoding, after that the feature vector is fed into the tiny MLP to non-linearly map to density and color, finally, the density and color of the spheres are integrated using volume rendering to produce final color for the pixel.

However, the rendering model of all the above methods is flawed in formulating the pixel as a single point and sampling with ignorance of the corresponding area, which would cause the renderings excessively blurred in close-up views and aliased in distant views. The super-sampling technique mentioned above can relieve this issue but requires casting multiple rays per pixel, which significantly increases the reconstruction and rendering cost. And incorporating pre-filtering with the hash encoding [37] is non-trivial due to the hash collisions. Our method addresses this issue by the proposed Tri-Mip encoding to effectively model the pre-filtered 3D feature space, which is as efficient as the hash encoding but able to produce anti-aliased high-fidelity renderings.

Compact 2D representation for 3D content. Directly representing 3D contents in volumes is memory- and computation-intensive, as well as redundant since 3D contents are always sparse. Peng *et al*. [41] propose to project features of point cloud to multiple planes for 3D geometry reconstruction. And recent works [21, 54, 55] have demonstrated that 3D content can be compactly represented in 2D images with faithful restoration. In the context of generative models, EG3D [8] proposed a tri-plane representation to decompose 3D volume into three 2D planes for 3D content generation, and this representation is adopted in many follow-up generative methods [16, 44, 45, 48, 5, 53, 12]. Besides, this representation is further generalized into 4D space to model dynamic scenes [7, 13]. Our Tri-Mip encoding is inspired by this line of works, but none of the above representations can realize our goal, *i.e*., modeling the pre-filtered 3D feature space for efficient area sampling.

3. Method

3.1. Overview

Given a set of calibrated multi-view images of static scenes, our goal is to efficiently reconstruct the radiance fields that can be further rendered into anti-aliased high-fidelity images. The rendering of radiance fields is performed one pixel at a time, so we describe the rendering procedure of a pixel of interest here, as shown in Fig. 2. We formulate the pixel as a disc on the image plane and perform cone casting for each pixel, rather than ray casting that ignores the area of a pixel. The cone casting emits a cone $\mathcal{C}$ from the projection center of the camera to the pixel disc on the image plane, and samples the cone with a set of spheres $\mathcal{S}$ that are inscribed with the cone. Further, we featurize the spheres to feature vectors $\mathbf{f}$ by our proposed Tri-Mip encoding that is parameterized by three mipmaps $\mathcal{M}$. This is the key to making our renderings contain fine-grained details in close-up views and free of aliasing in distant views, since the Tri-Mip encoding effectively models the pre-filtered 3D feature space by taking advantage of different levels in the mipmap. Then, we employ a tiny MLP parameterized by weights Θ to non-linearly map the feature vector $\mathbf{f}$ of spheres $\mathcal{S}$ and view direction $\mathbf{d}$ to density τ and color c of the spheres,

$$[\tau, c] = \mathrm{MLP}(\mathbf{f}, \mathbf{d};\ \Theta). \tag{1}$$

Finally, the estimated densities and colors of spheres inside a cone are used to approximate the volume rendering integral by numerical quadrature as in [33] to render the final color of the pixel corresponding to the cone:

$$\mathbf{C}(\mathbf{t}, \mathbf{d}, \Theta, \mathcal{M}) = \sum_i T_i(1 - \exp(-\tau_i(t_{i+1} - t_i)))c_i,$$
$$\text{with} \quad T_i = \exp\left(-\sum_{k<i} \tau_k(t_{k+1} - t_k)\right), \tag{2}$$

where $\mathbf{t}$ is the distance between the sampled spheres and the projection center of the camera. During training, the photometric loss will be computed between the rendered colors and captured colors to back-propagate gradients to the weights Θ of MLP and parameters $\mathcal{M}$ of Tri-Mip encoding to jointly optimize them.

In the following sections, we will present the cone casting, Tri-Mip encoding, as well as the hybrid volume-surface rendering in detail, while omitting the procedures of tiny MLP and volume rendering as they are similar to the original NeRF [35]. Please refer to the supplemental material for more details.

3.2. Cone Casting

NeRF renders a pixel by emitting a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, and sampling points $\mathbf{x}$ along the ray, *a.k.a.* ray casting, as

shown in Fig. 3 (a). And the points $\mathbf{x}$ are further featurized by position encoding (PE) γ to produce the feature vectors for the points $\gamma(\mathbf{x})$. This formulation models the pixel as a single point while ignoring the area of the pixel, which is quite different from the real-world imaging sensors. Most NeRF works [47, 56, 14, 9, 10], including instant-ngp [37], followed this formulation. It can approximate the real-world case when the captured/rendered views are at a roughly constant distance but will lead to obvious artifacts when viewing at very different distances, *e.g.*, blurring in close-up views and aliasing in distant views since the sampling is distance-agnostic. To this end, MipNeRF emits a cone for each pixel and samples the cone by the multivariate Gaussian, which is further featurized by integrated position encoding (IPE). The IPE is derived by the integral $\mathrm{E}[\gamma(\mathbf{x})]$ over the PE of the points within the Gaussian, as shown in Fig. 3 (b). This strategy, however, is not trivial to be extended to explicit or hybrid representations for efficient reconstruction and rendering, *e.g.*, hash encoding [37], since IPE is the integral of coordinate-based positional encoding, which is not compatible with explicit or hybrid volumetric feature encoding.

In contrast, our efficient cone-casting strategy can efficiently work with our Tri-Mip encoding for area sampling during the volume rendering. As shown in Fig. 3 (c), we formulate the pixel as a disc on the image plane rather than a single point that ignores the area of a pixel. The radius of the disc can be calculated by $\dot{r} = \sqrt{\Delta x \cdot \Delta y / \pi}$, where Δx and Δy are the width and height of the pixel in world coordinates that can be derived from the calibrated camera parameters. For each pixel, we emit a cone $\mathcal{C}$ from the camera's projection center $\mathbf{o}$ along the direction $\mathbf{d} = \mathbf{p_o} - \mathbf{o}$, where $\mathbf{p_o}$ is the pixel's center. The apex of the cone is located at the optical center of the camera and the intersection between the cone and the image plane is the disc corresponding to the pixel. We can derive the central axis of the cone as $\mathbf{a}(t) = \mathbf{o} + t\mathbf{d}$. To sample the cone, we cannot follow MipNeRF [3] to use the multivariate Gaussian, since the multivariate Gaussian is anisotropic but the pre-filtering in our Tri-Mip encoding is isotropic. Thus, we sample the cone with a set of spheres $\mathcal{S}(\mathbf{x},\ \mathbf{r})$ parameterized by their centers $\mathbf{x}$ and radiuses $\mathbf{r}$. The centers $\mathbf{x}$ are located at the central axis of the cone and the radiuses $\mathbf{r}$ are set to make the spheres inscribed with the cone, which can be written as:

$$\begin{aligned} \mathbf{x} &= \mathbf{o} + t\mathbf{d}\,, \\ \mathbf{r} &= \frac{\|\mathbf{x}-\mathbf{o}\|_2 \cdot f\dot{r}}{\|\mathbf{d}\|_2 \cdot \sqrt{\left(\sqrt{\|\mathbf{d}\|_2^2 - f^2} - \dot{r}\right)^2 + f^2}}, \end{aligned} \tag{3}$$

where f is the focal length. Based on Eq. 3, the sampling spheres $\mathcal{S}(\mathbf{x},\ \mathbf{r})$ can be determined by a sorted distance vector $t_i \in \mathbf{t}$, since the center location $\mathbf{x}_i$ and radius $\mathbf{r}_i$ of a sphere $\mathcal{S}_i$ is the function of the distance t_i. We uniformly sample $t_i \in \mathbf{t}$ between the camera's predefined near t_n and far t_f planes or the two intersections between the central axis of the cone and the axis-aligned bounding-box (AABB)

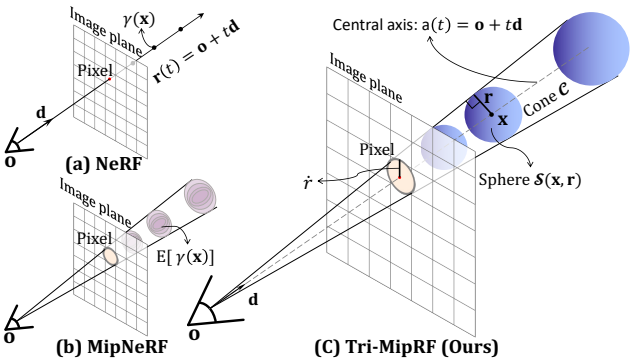


Figure 3. NeRF [35] renders a pixel by ray-casting and the sampling points $\mathbf{x}$ on the ray are featurized by position encoding $\gamma(\mathbf{x})$. MipNeRF [3] emits a cone for each pixel and featurize the sampling multivariate Gaussian by integrated position encoding $\mathrm{E}[\gamma(\mathbf{x})]$. Our Tri-MipRF renders a pixel by cone casting and the cone is sampled by a set of spheres that are inscribed with the cone.

of the interested 3D space. To further speed up the cone casting by utilizing the sparsity of the 3D space, we employ a binary occupancy grid that coarsely marks empty *vs.* non-empty space similar to [37, 26], such that we can cheaply skip samples in the empty area and concentrate samples near surfaces to avoid wasted computation.

3.3. Tri-Mip Encoding

To realize our goal, *i.e.* rendering fine-grained details in close-up views and avoiding aliasing in distant views while maintaining the reconstruction and rendering efficiency, we should constructively featurize the sampled spheres $\mathcal{S}(\mathbf{x},\ \mathbf{r})$ according to their occupied area, which shares similar motivation of area-sampling (*a.k.a.* pre-filtering) in computer graphics. Hash encoding proposed in instant-ngp [37] can efficiently featurize the sampled *points* by looking up the hash table and trilinear interpolation, however, it cannot be easily extended to featurize the spheres $\mathcal{S}(\mathbf{x},\ \mathbf{r})$. One plausible workaround is to incorporate the super-sampling strategy with hash encoding to approximate the featurization of spheres. However, super-sampling significantly increases the computation cost, which unexpectedly sacrifices the ability of efficient reconstruction and rendering.

To this end, we propose a novel Tri-Mip encoding parameterized by three trainable mipmaps $\mathcal{M}$ to featurize the sampling spheres $\mathcal{S}(\mathbf{x},\ \mathbf{r})$:

$$\begin{aligned} \mathbf{f} &= \text{Tri-Mip}(\mathbf{x}, \mathbf{r};\ \mathcal{M}), \\ \mathcal{M} &= \{\mathcal{M}_{XY}, \mathcal{M}_{XZ}, \mathcal{M}_{YZ}\}. \end{aligned} \tag{4}$$

As shown in Fig. 2, the Tri-Mip encoding decomposes the 3D space into three planes (plane_{XY}, plane_{XZ}, and plane_{YZ}) using orthographic projection, and then represent each plane by a mipmap ($\mathcal{M}_{XY}$, $\mathcal{M}_{XZ}$, and $\mathcal{M}_{YZ}$) to model the pre-filtered feature space. For each mipmap, the base level $\mathcal{M}^{L_0}$ is a feature map with the shape of $H \times W \times C$, where H, W, C are the height, width, and number of channels, respectively. The base level $\mathcal{M}^{L_0}$ is

randomly initialized and can be trained during the reconstruction, and other levels ($\mathcal{M}^{L_i}, i = 1, 2, ..., N$) are derived from the previous level $\mathcal{M}^{L_{i-1}}$ by downscaling $2\times$ along the height and width. This pre-filtering strategy maintains consistency among the levels of mipmap, which is the key to making the reconstructed objects coherent at different distances.

To query the feature vectors $\mathbf{f}$ corresponding to the spheres $\mathcal{S}(\mathbf{x},\ \mathbf{r})$, we first orthogonally project $\mathcal{S}$ on the three planes to obtain three discs $\mathcal{D} = \{\mathcal{D}_{XY}, \mathcal{D}_{XZ}, \mathcal{D}_{YZ}\}$, as shown in Fig. 2. For each disc, we query a feature vector from the corresponding mipmap. Take disc $\mathcal{D}_{XY}$ as an example, we query its feature $\mathbf{f}_{XY}$ from the mipmap $\mathcal{M}_{XY}$. Based on the property of orthogonal projection, the disc $\mathcal{D}_{XY}$ shares the same radius $\mathbf{r}$ as the sampled sphere, and the 2D coordinate of the $\mathcal{D}_{XY}$'s center $\mathbf{x}_{\mathcal{D}_{XY}}$ is the partial coordinate (x, y) of the sampled sphere's center $\mathbf{x}(x, y, z)$. For the disc $\mathcal{D}_{XY}$'s query level l of the mipmap $\mathcal{M}_{XY}$, we assign it to:

$$
\begin{aligned}
l &= log_2\left(\frac{\mathbf{r}}{\ddot{r}}\right), \\
\ddot{r} &= \sqrt{\frac{(\mathcal{B}_{max} - \mathcal{B}_{min})_X \cdot (\mathcal{B}_{max} - \mathcal{B}_{min})_Y}{HW \cdot \pi}},
\end{aligned}
\tag{5}
$$

where $\ddot{r}$ is the radius of the feature elements in the base level of the mipmap $\mathcal{M}^{L_0}$, $\mathcal{B}_{max}$ and $\mathcal{B}_{min}$ are the maximum and minimum corners of the Axis Aligned Bounding Box (AABB) of the interested 3D space, respectively. The motivation of Eq. 5 is to match the sphere's radius $\mathbf{r}$ with the feature elements' radius in a certain level of the mipmap $\mathcal{M}^l_{XY}$. After obtaining the query coordinate (x, y, l), we can get the feature vector $\mathbf{f}_{XY}$ from the mipmap $\mathcal{M}_{XY}$ by the trilinear interpolation. As shown in Fig. 2, we first find the two nearest levels of the mipmap $\mathcal{M}^{\lfloor l \rfloor}_{XY}$ and $\mathcal{M}^{\lceil l \rceil}_{XY}$; and then we project the center coordinate (x, y) of the disc $\mathcal{D}_{XY}$ to the two levels of the mipmap (shown as red dots); next, we find four neighbors of them (shown as orange dots), respectively; finally, we interpolate the eight neighbors based on their distance to the center of the disc $\mathcal{D}_{XY}$ to produce the feature vector $\mathbf{f}_{XY}$. The trilinear interpolation increases the effective precision of both levels and spatial resolutions, also, it yields a continuous encoding space that is beneficial for efficient training. Similarly, we can get the feature vectors $\mathbf{f}_{XZ}$ and $\mathbf{f}_{YZ}$ for the disc $\mathcal{D}_{XZ}$ and $\mathcal{D}_{YZ}$, respectively. The final queried feature vector $\mathbf{f}$ for the sampled sphere $\mathcal{S}$ is a concatenation of the three discs' feature vectors $\{\mathbf{f}_{XY}, \mathbf{f}_{XZ}, \mathbf{f}_{YZ}\}$.

Our Tri-Mip encoding effectively featurize the 3D space in a pre-filtered way, such that we can perform area-sampling for the volume rendering to produce high-quality renderings that are free of aliasing. And the feature query process is also efficient, *i.e.*, querying mipmap has been highly optimized in modern GPUs, which promotes fast reconstruction. Besides, the storage of our Tri-Mip encoding is three 2D feature maps, *i.e.* the base levels of the three mipmaps $\mathcal{M}^{l_0}$ as other levels are derived by the base level by downscaling, which makes our model compact enough for easy distribution. Note that, Tri-Mip encoding also promotes the training converges faster than implicitly representing the scene in MLP, *e.g.*, our method only takes $25K$ iterations to converge while MipNeRF [3] requires $1M$ iterations, since features in the mipmap $\mathcal{M}$ can be optimized directly rather than mapped from the IPE by optimizable weights of MLP.

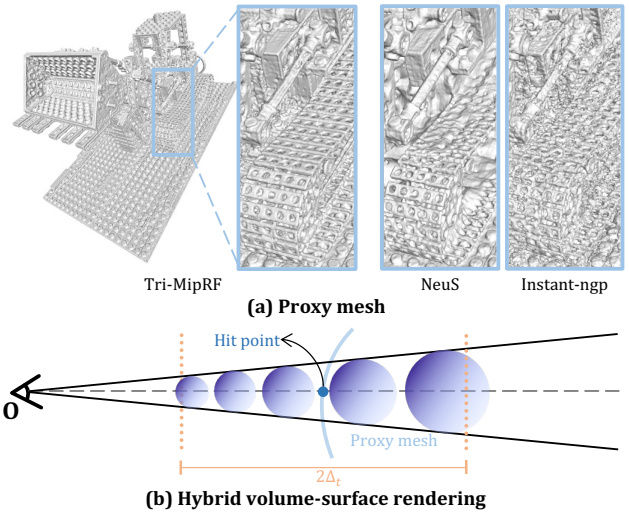


Figure 4. Visual comparison of the proxy mesh produced by our Tri-MipRF, Instant-ngp [37], and NeuS [49] (a); and our proposed hybrid volume-surface real-time rendering strategy (b).

3.4. Hybrid Volume-Surface Rendering

Though our method can efficiently reconstruct the radiance fields, directly rendering it on consumer-level GPUs, *e.g.*, an Nvidia RTX 3060 graphics card, only achieves around 30 FPS. This is because the volume rendering inherently samples multiple spheres inside the cone for each pixel, though we can skip some samples by the binary occupancy grid. Observing the real-time surface rendering benefited from the efficient rasterization of the polygon mesh, we develop a hybrid volume-surface rendering strategy to further boost the rendering speed. Besides the reconstructed radiance field, our hybrid volume-surface rendering strategy requires a proxy mesh to efficiently determine a rough distance from the camera's optical center to the object. Fortunately, we can obtain the proxy mesh by marching cubes [30] on the reconstructed density field followed by mesh decimation. The proxy mesh produced by our Tri-MipRF presents high-fidelity quality even in complicated structure details, as shown in the left-hand side of Fig. 4 (a), while the results produced by Instant-ngp [37] and NeuS [49] are shown in the right-hand side as references.

Once the proxy mesh is available, we first efficiently rasterize it to obtain the hit point (shown as a blue dot) on the surface for the central axis of the cone, as shown in Fig. 4 (b), then we uniformly sample spheres within the distance of Δ_t from the hit point in the central axis of the cone, which yields a $2\Delta_t$ sampling interval. This hybrid volume-surface rendering strategy significantly reduces the

	Train. ↓	PSNR ↑					SSIM ↑					LPIPS ↓				
		Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Avg.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res	Avg.
NeRF w/o $\mathcal{L}_{\text{area}}$	3 days	31.20	30.65	26.25	22.53	27.66	0.950	0.956	0.930	0.871	0.927	0.055	0.034	0.043	0.075	0.052
NeRF [35]	3 days	29.90	32.13	33.40	29.47	31.23	0.938	0.959	0.973	0.962	0.958	0.074	0.040	0.024	0.039	0.044
MipNeRF [3]	3 days	32.63	34.34	35.47	35.60	34.51	0.958	0.970	0.979	0.983	0.973	0.047	0.026	0.017	0.012	0.026
Plenoxels [14]	9 min	31.60	32.85	30.26	26.63	30.34	0.956	0.967	0.961	0.936	0.955	0.052	0.032	0.045	0.077	0.051
TensoRF [9]	19 min	32.11	33.03	30.45	26.80	30.60	0.956	0.966	0.962	0.939	0.956	0.056	0.038	0.047	0.076	0.054
Instant-ngp [37]	5 min	30.00	32.15	33.31	29.35	31.20	0.939	0.961	0.974	0.963	0.959	0.079	0.043	0.026	0.040	0.047
Instant-ngp $\uparrow^{5\times}$	10 min	30.96	32.87	33.10	30.82	31.94	0.945	0.965	0.973	0.970	0.963	0.070	0.038	0.025	0.029	0.041
Tri-MipRF w/o $\mathcal{M}$	4.5 min	30.25	32.52	33.73	29.44	31.48	0.938	0.961	0.975	0.964	0.959	0.081	0.045	0.026	0.039	0.048
Tri-MipRF (Ours)	5 min	33.32	35.02	35.78	36.13	35.06	0.961	0.974	0.981	0.986	0.976	0.043	0.024	0.017	0.011	0.024

Table 1. Quantitive comparison of our Tri-MipRF against several cutting-edge methods and their variants on the multi-scale Blender dataset.

number of samples, thus, enabling real-time rendering (>60 FPS) on consumer-level GPUs. Please refer to the video in the supplemental material for the real-time interactive rendering demo.

4. Experimental Evaluation

4.1. Implementation

Our Tri-Mip radiance fields (Tri-MipRF) learns the 3D structure solely from the calibrated multi-view 2D images and is trained with the photometric metric loss between the rendered pixels and the captured ones. Following MipNeRF [3], we scale the loss of each pixel by the area of its footprint on the image plane, noted as "area loss $\mathcal{L}_{\text{area}}$". We implement our Tri-MipR using PyTorch [40] framework with tiny-cuda-nn [36] extension. The mipmap query process is well optimized in the rendering community for texture sampling, thus, we employ the nvdiffrast [24] library to implement our Tri-Mip encoding efficiently. The shape of the base level of the mipmap $\mathcal{M}^{l_0}$ in Tri-Mip encoding is empirically set to $H = 512, W = 512, C = 16$, which is a compact representation for the scene. We train our Tri-MipRF using the AdamW optimizer [31] for $25K$ iterations with the weight decay set to 1×10^{-5} and the learning rate set to 2×10^{-3} and scheduled by MultiStepLR in PyTorch. Note, the learning rate for the Tri-Mip encoding is further scaled up $10\times$ since the parameter $\mathcal{M}$ of Tri-Mip encoding directly represents the scene while that for tiny MLP keeps unchanged.

4.2. Evaluation on the Multi-scale Blender Dataset

The Blender dataset presented in the original NeRF [35] is a synthetic dataset where all training and testing images observe the scene content from a roughly constant distance, which is very different from real-world captures. MipNeRF [3] presents a multi-scale Blender dataset to better probe the reconstruction accuracy and anti-aliasing on multi-resolution scenes. It is compiled by downscaling the original dataset with a factor of 2, 4, and 8, and combining them together. Due to the nature of projecting geometry, this is almost equivalent with re-rendering the original dataset where the distance to the camera has been increased by scale factors of 2, 4, and 8.

Quantitative results. We compared our Tri-MipRF with several cutting-edge methods, *i.e.*, NeRF [35], MipNeRF [3], Plenoxels [14], TensoRF [9], and Instant-ngp [37]. Following previous works, we report three metrics: PSNR, SSIM [50], and VGG LPIPS [57], as shown in Tab. 1. We also report the rough reconstruction time on the same hardware, *i.e.* a single Nvidia A100 GPU. Except for MipNeRF, other comparison methods are not designed for multi-scale captures or imaging at various distances, thus, we equipped all of them with the aforementioned area loss $\mathcal{L}_{\text{area}}$ by default, which yields a better performance as evidenced by comparing the results of "NeRF w/o $\mathcal{L}_{\text{area}}$" and "NeRF". From Tab. 1, we can see that MipNeRF presents high-quality renderings, however, the reconstruction of it is extremely slow (up to around three days) which greatly prevents the applicability. Besides, the reconstruction times of Plenoxels, TensoRF, and Instant-ngp are greatly faster than that of MipNeRF, but the rendering qualities are unsatisfactory no matter in terms of PSNR, SSIM, or LPIPS. For Instant-ngp, we further design a super-sampling variant of it, Instant-ngp $\uparrow^{5\times}$, which means casting five rays in the quincunx sample pattern for each pixel and aggregating the samples in these rays. We can find that super-sampling makes it render higher-quality images, however, super-sampling also significantly increases the reconstruction time from five to ten minutes. In contrast, our Tri-MipRF not only produces the highest-quality renderings for all four types of resolutions but also can be reconstructed super-fast, *i.e.* 5 minutes. To verify the effectiveness of the Tri-Mip encoding, we also evaluate an ablation of our method, Tri-MipRF w/o $\mathcal{M}$, that replaces the three mipmaps with three 2D feature maps with the same shape as the base level of the mipmap. As shown in Tab. 1, the Tri-MipRF w/o $\mathcal{M}$ performs comparable with Instant-ngp but significantly worse than our full method, Tri-MipRF, even though it can be reconstructed slightly faster than Tri-MipRF since it gets rid of the mipmap query procedure. These quantitative comparisons demonstrate the effectiveness of our Tri-Mip encoding and cone casting, such that our Tri-MipRF can effectively model the pre-filtered 3D feature space and efficiently perform area sampling on it for anti-aliased high-fidelity rendering.

Qualitative results. We further qualitatively compared our Tri-MipRF with the Instant-ngp [37] and its super-sapling variant, Instant-ngp $\uparrow^{5\times}$, since their reconstruction speed is similar to ours, *i.e.*, five minutes for our Tri-MipRF, five and ten minutes for Instant-ngp and Instant-ngp $\uparrow^{5\times}$, respectively. In Fig. 5, we show examples of full-resolution renderings

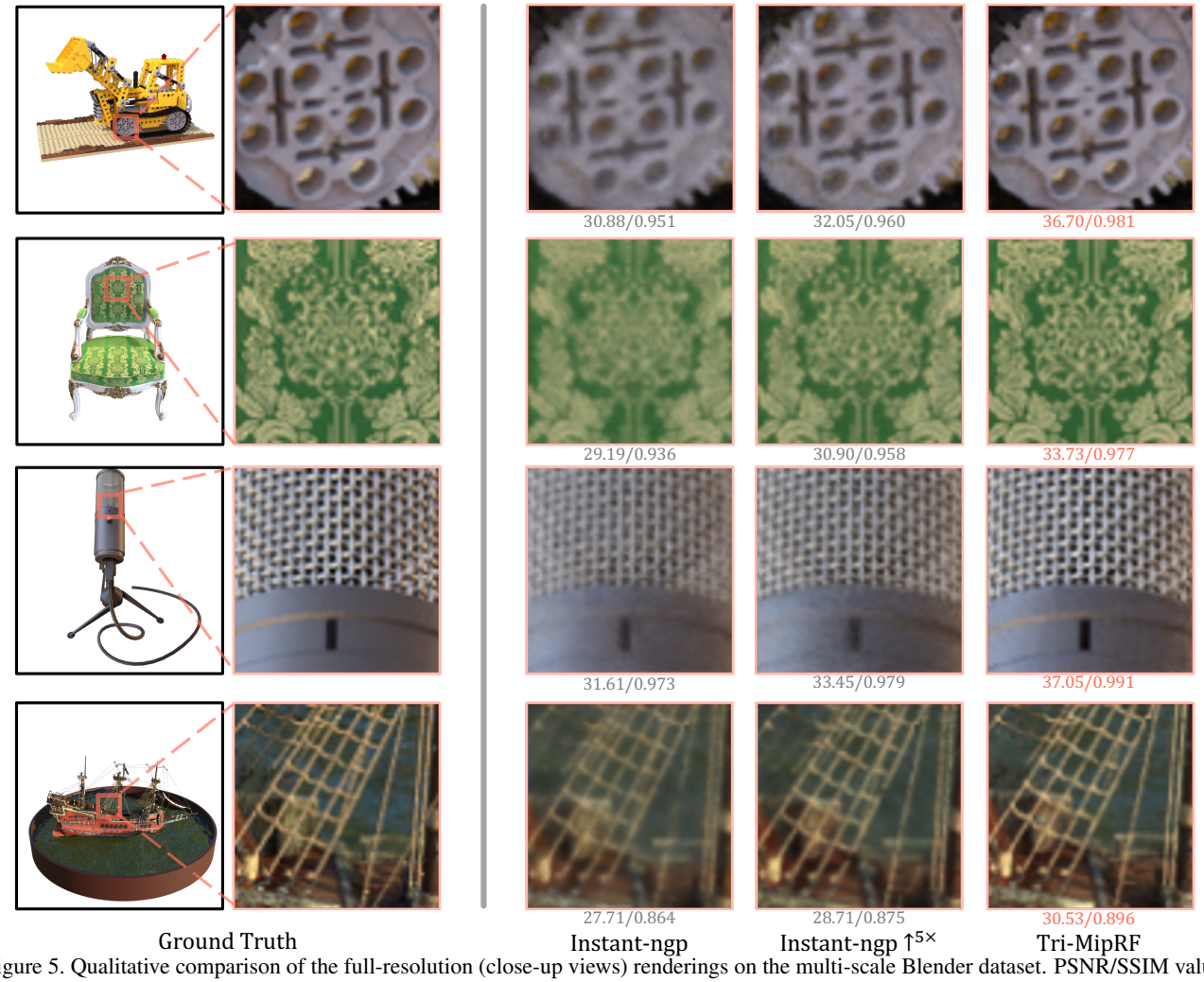


Figure 5. Qualitative comparison of the full-resolution (close-up views) renderings on the multi-scale Blender dataset. PSNR/SSIM values are shown at the bottom of each result.

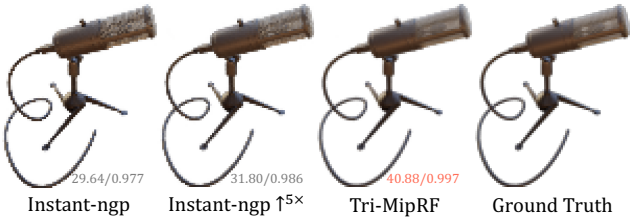


Figure 6. Qualitative comparison of the low-resolution renderings (distant view) on the multi-scale Blender dataset. PSNR/SSIM values are shown in the bottom right corners of each result.

that can be treated as close-up views. we can see that the results of Instant-ngp suffer from blurriness for structure and texture details, Instant-ngp $\uparrow^{5\times}$ improves the quality but significantly increases the reconstruction time. In contrast, our method faithfully renders the fine-grained details while keeping the reconstruction super-fast. On the other hand, we compare the renderings of $1/8$ resolution that can simulate the distant views in Fig. 6. We can see renderings of Instant-ngp exhibit severe aliasing and "jaggies" artifacts and Instant-ngp $\uparrow^{5\times}$ slightly relieves this issue, while our Tri-MipRF faithfully renders smooth appearance and fine-grained structure details, thanks to the Tri-Mip encoding that efficiently models the pre-filtered 3D feature space. We highly recommend readers to watch the supplemental video to better evaluate the anti-aliasing feature.

4.3. Evaluation on the Single-scale Blender Dataset

The easier single-scale Blender dataset captures images at a roughly constant distance, which is friendly to the point-sampling-based methods, *e.g.*, NeRF [35], Plenoxels [14], TensoRF [9], Instant-ngp [37], and *etc*. We also compared our Tri-MipRF with multiple cutting-edge methods on this dataset, as shown in Tab. 2. We can see that our Tri-MipRF still outperforms all of them no matter in terms of PSNR, SSIM, or LPIPS, in the meanwhile, achieving the fastest reconstruction together with Instant-ngp. Besides, we also report the model size, *i.e.* the storage consumption, in Tab. 2. We can find that the implicit methods have extremely small model sizes, *e.g.*, the model size of NeRF and MipNeRF is 5.00 MB and 2.50 MB, respectively, but are reconstructed

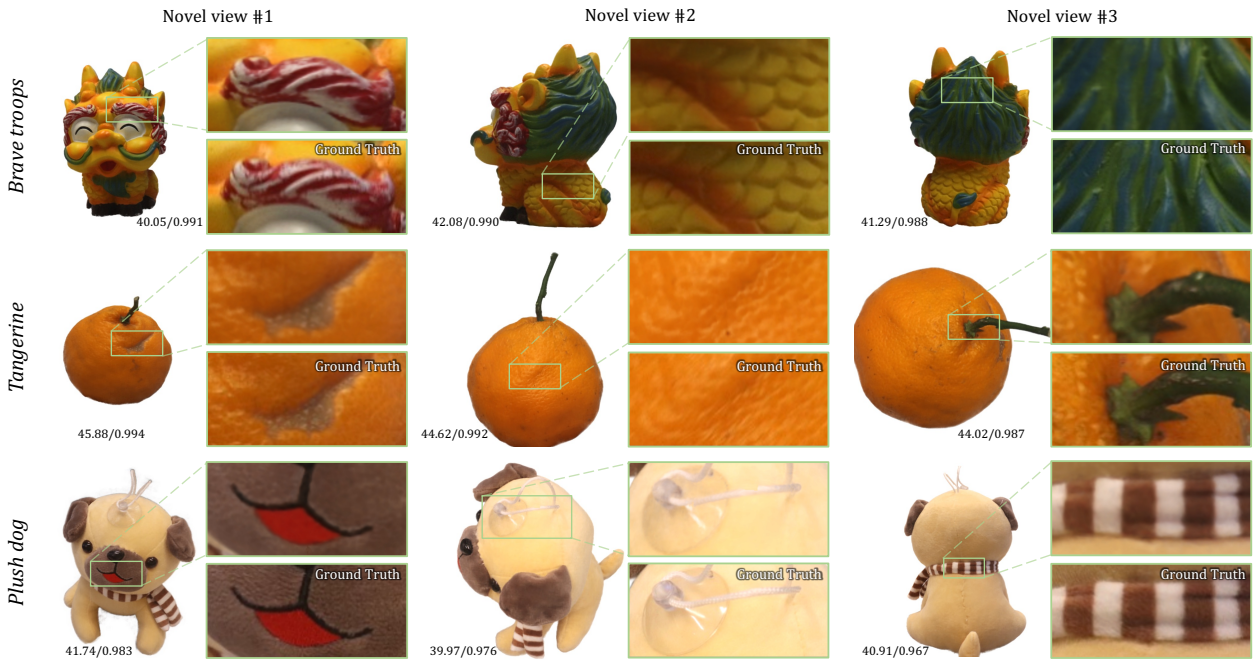


Figure 7. Example rendering results of our method from in-the-wild captures. The PSNR/SSIM values are shown below the renderings of our method.

	PSNR ↑	SSIM ↑	LPIPS ↓	# Size ↓
SRN [46]	22.26	0.846	0.170	-
LLFF [34]	24.88	0.911	0.114	-
Neural Volumes [29]	26.05	0.893	0.160	-
Plenoxels [14]	31.71	0.958	0.049	778 MB
NeRF [35]	31.74	0.953	0.050	5.00 MB
DVGO [47]	31.95	0.957	0.053	612 MB
MipNeRF [3]	33.09	0.961	0.043	2.50 MB
TensoRF [9]	33.14	0.963	0.047	71.8 MB
Instant-ngp [37]	33.18	0.963	0.045	64.1 MB
Tri-MipRF	33.65	0.963	0.042	48.2 MB

Table 2. Results on the single-scale Blender dataset of our Tri-MipRF and several cutting-edge methods.

very slowly ($\sim$ 3 days); the model sizes of explicit methods, *e.g.*, Plenoxels and DVGO, are very large ($>$ 500 MB); and the hybrid methods, *e.g.*, Instant-ngp, TensoRF, and our Tri-MipRF, have relative small model size ($<$ 100 MB) while our Tri-MipRF has the smallest model size (48.2 MB) in the hybrid methods, which reduces 25% storage consumption compared against Instant-ngp. Please refer to the supplemental materials for more detailed statistics.

4.4. Applicability on the In-the-wild Captures

To further demonstrate the applicability of our method, we captured several objects in the wild. We performed SFM on the sequence to estimate the camera's intrinsic and extrinsic parameters and employed multi-view segmentation methods to separate the object from the background scenes. Each captures contain 200 $\sim$ 300 images with the resolution of 1200×800, and we uniformly sample 70% of them for the reconstruction and the remains are used for evaluation. We show three example results in Fig. 7, where we can see the rendered novel views faithfully reproduce the detailed structures and appearances, and the PSNR/SSIM values marked below the images also evidence the applicability of our method. Interestingly, we find our renderings even have "better" details than the ground truth in some cases, *e.g.*, the brave troops' eyebrow shown in the blow-up figure of the novel view #1 in the first line of Fig. 7. This is because the ground truth, *i.e.* the captured image by the camera in the wild, may suffer from motion blur artifacts due to the fast movements, while this issue is relieved during the reconstruction by fusing multiple observations.

5. Conclusion

In this work, we propose a Tri-Mip radiance fields, Tri-MipRF, to make the renderings contain fine-grained details in close-up views and free of aliasing in distant views while maintaining efficient reconstruction, *i.e.* within five minutes, and compact representation, *i.e.* 25% smaller model size than Instant-ngp. This is realized by our novel Tri-Mip encoding and cone casting. The Tri-Mip encoding featurizes the 3D space by three mipmaps to model the pre-filtered 3D feature space, such that the sample spheres from the cone casting can be encoded in an area-sampling manner. We also develop a hybrid volume-surface rendering strategy to enable real-time rendering ($>$ 60 FPS) on consumer-level devices. Extensive quantitative and qualitative experiments demonstrate our Tri-MipRF achieves state-of-the-art rendering quality while having a super-fast reconstruction speed. Also, the reconstruction results on the in-the-wild captures demonstrate the applicability of our Tri-MipRF.

References

[1] Kurt Akeley. Reality engine graphics. In *Conference on Computer graphics and interactive techniques*, 1993. 2

[2] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-time rendering*. AK Peters/crc Press, 2019. 2

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021. 1, 2, 4, 5, 6, 8, 11, 12, 13

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 2

[5] Miguel Angel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, Afshin Dehghan, and Josh Susskind. Gaudi: A neural architect for immersive 3d scene generation. *arXiv*, 2022. 3

[6] Aljaž Božič, Denis Gladkov, Luke Doukakis, and Christoph Lassner. Neural assets: Volumetric object capture and rendering for interactive environments. *arXiv preprint arXiv:2212.06125*, 2022. 1, 2

[7] Ang Cao and Justin Johnson. Hexplane: a fast representation for dynamic scenes. *arXiv preprint arXiv:2301.09632*, 2023. 3

[8] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2022. 2, 3

[9] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 1, 2, 4, 6, 7, 8, 11, 12, 13

[10] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277*, 2022. 1, 4

[11] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. *ACM SIGGRAPH Computer Graphics*, 22(4):21–30, 1988. 2

[12] Kangle Deng, Gengshan Yang, Deva Ramanan, and Jun-Yan Zhu. 3d-aware conditional image synthesis. *arXiv*, 2023. 3

[13] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. *arXiv preprint arXiv:2301.10241*, 2023. 3

[14] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 1, 2, 4, 6, 7, 8, 11, 12, 13

[15] Henry Fuchs, Jack Goldfeather, Jeff P Hultquist, Susan Spach, John D Austin, Frederick P Brooks Jr, John G Eyles, and John Poulton. Fast spheres, shadows, textures, transparencies, and imgage enhancements in pixel-planes. *ACM SIGGRAPH Computer Graphics*, 19(3):111–120, 1985. 2

[16] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. In *NeurIPS*, 2022. 3

[17] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *ICCV*, 2021. 1, 2

[18] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010. 11

[19] Paul Haeberli and Kurt Akeley. The accumulation buffer: Hardware support for high-quality rendering. *ACM SIGGRAPH Computer Graphics*, 24(4):309–318, 1990. 2

[20] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *ICCV*, 2021. 1, 2

[21] Wenbo Hu, Menghan Xia, Chi-Wing Fu, and Tien-Tsin Wong. Mononizing binocular videos. *TOG*, 39(6):228:1–228:16, December 2020. 3

[22] Anton S Kaplanyan, Stephan Hill, Anjul Patney, and Aaron E Lefohn. Filtering distributions of normals for shading antialiasing. 2016. 2

[23] Alexandr Kuznetsov. Neumip: Multi-resolution neural materials. *TOG*, 40(4), 2021. 2

[24] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *TOG*, 39(6), 2020. 6

[25] William J Leler. Human vision, anti-aliasing, and the cheap 4000 line display. *SIGGRAPH*, 14(3):308–313, 1980. 2

[26] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: A general nerf accleration toolbox. *arXiv preprint arXiv:2210.04847*, 2022. 4

[27] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *CVPR*, 2021. 2

[28] David B Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. Bacon: Band-limited coordinate networks for multiscale scene representation. In *CVPR*, 2022. 2

[29] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *TOG*, 38(4):65:1–65:14, 2019. 8, 13

[30] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *TOG*, 21(4):163–169, 1987. 5

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6, 11

[32] Abraham Mammen. Transparency and antialiasing algorithms implemented with the virtual pixel maps technique. *IEEE Computer Graphics and Applications*, 9(4):43–55, 1989. 2

[33] Nelson Max. Optical models for direct volume rendering. *TVCG*, 1(2):99–108, 1995. 3

[34] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *TOG*, 38(4):1–14, 2019. 8, 13

[35] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 3, 4, 6, 7, 8, 11, 12, 13

[36] Thomas Müller. tiny-cuda-nn, 4 2021. 6

[37] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *TOG*, 41(4):1–15, 2022. 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 16, 17

[38] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. 40(4):45–59, 2021. 2

[39] Marc Olano and Dan Baker. Lean mapping. In *ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, 2010. 2

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 6

[41] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 3

[42] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *CVPR*, 2021. 2

[43] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021. 1, 2

[44] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. *arXiv preprint arXiv:2211.16677*, 2022. 3

[45] Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, and Yaniv Taigman. Text-to-4d dynamic scene generation. *arXiv:2301.11280*, 2023. 3

[46] Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. 2019. 8, 13

[47] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 1, 2, 4, 8, 13

[48] Jingxiang Sun, Xuan Wang, Yichun Shi, Lizhen Wang, Jue Wang, and Yebin Liu. Ide-3d: Interactive disentangled editing for high-resolution 3d-aware portrait synthesis. *TOG*, 41(6):1–10, 2022. 3

[49] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. 5

[50] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 13(4):600–612, 2004. 6

[51] Turner Whitted. An improved illumination model for shaded display. In *ACM Siggraph 2005 Courses*, pages 4–es. 2005. 2

[52] Lifan Wu, Shuang Zhao, Ling-Qi Yan, and Ravi Ramamoorthi. Accurate appearance preserving prefiltering for rendering displacement-mapped surfaces. *TOG*, 38(4):1–14, 2019. 2

[53] Rundi Wu and Changxi Zheng. Learning to generate 3d shapes from a single example. *TOG*, 41(6), 2022. 3

[54] Yue Wu, Guotao Meng, and Qifeng Chen. Embedding novel views in a single jpeg image. In *ICCV*, 2021. 3

[55] Menghan Xia, Jose Echevarria, Minshan Xie, and Tien-Tsin Wong. Lf2mv: Learning an editable meta-view towards light field representation. *TVCG*, 2022. 3

[56] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 1, 2, 4, 11

[57] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

Supplementary Material

A. Characteristics Matrix

We compared various crucial characteristics of cutting-edge NeRF methods and our Tri-MipRF in Tab. 3, including quality aspect, *e.g.*, anti-aliasing, and efficiency aspect, *e.g.*, fast reconstruction, real-time rendering, and compact model. We can see that, except for our Tri-MipRF, none of them can support anti-aliasing, fast reconstruction, real-time rendering, and compact model, at the same time. These positive characteristics are enabled by our Tri-Mip encoding and cone casting.

Method	Anti-aliasing	Fast Reconstruction	Real-time Rendering	Compact Model
NeRF [35]	✗	✗	✗	✓
MipNeRF [3]	✓	✗	✗	✓
Instant-ngp [37]	✗	✓	✓̸	✓
TensoRF [9]	✗	✓̸	✓̸	✓
PlenOctrees [56]	✗	✗	✓	✗
Plenoxels [14]	✗	✓	✓̸	✗
Tri-MipRF (ours)	✓	✓	✓	✓

Table 3. Characteristics matrix of cutting-edge NeRF methods and ours. "✓" means "yes", "✓̸" means "moderate", and "✗" means "no".

B. Model Details

B.1. Tiny MLP

The goal of the tiny MLP is to nonlinearly map the feature vector $\mathbf{f}$ produced by the Tri-Mip encoding and the view direction $\mathbf{d}$ to density τ and color c of the sampled sphere $\mathcal{S}$. The feature vector $\mathbf{f}$ has a dimension of 48 since the mipmaps $\mathcal{M}$ in Tri-Mip encoding has a shape of $512 \times 512 \times 16$ as described in Sec.4.1 of the main paper. The first two layers of the MLP take $\mathbf{f}$ as input and produce the density τ and a geometric feature $\mathbf{f}_{\text{geo}}$ with a dimension of 15. And the view direction $\mathbf{d}$ is encoded by the spherical harmonics basis and then fed into the last three layers the MLP together with the $\mathbf{f}_{\text{geo}}$ to estimate the final view-dependent color c, which is similar to [37]. The width of the tiny MLP is empirically set to 128. The activation function of all the layers is the ReLU, except for the output layer of density τ, where we adopt the truncated exponential function followed [37]. This shallow MLP is implemented with tiny-cuda-nn that is well-optimized for fused and half-precision MLP.

B.2. Optimization

The optimizable parameters of our Tri-MipRF include the model weights Θ of the tiny MLP and the mipmaps $\mathcal{M}$ in the Tri-Mip encoding. The model weights Θ is initialized by the method proposed in [18], while the mipmaps $\mathcal{M}$ is initialized by a uniform distribution of the interval $[-0.01, 0.01]$ to encourage the sparsity of $\mathcal{M}$. We employ the AdamW optimizer [31] to train Θ and $\mathcal{M}$, where we set base learning rate for Θ and scale up the base learning rate $10\times$ for $\mathcal{M}$ since $\mathcal{M}$ is a direct representation the reconstructed scene. We set the base learning rate to 2×10^{-3} and scale it up $0.6\times$ at steps $12K$, $18K$, $20K$, and $22K$, while the total number of iteration is $25K$. And followed [37], we adopt the dynamic batch-size strategy that keep the total number of spheres in a batch to be roughly $256K$. We will release our source code for better reproducibility upon publication.

C. Detailed Results

Multi-scale Blender Dataset To demonstrate more detailed per-scene results of our Tri-MipRF, compared with other cutting-edge methods, we provide quantitative results of them under three metrics in Tab. 4. We can see our Tri-MipRF outperforms all the other methods on almost all the scenes. Visual comparisons of the renderings from Instant-ngp, Instant-ngp $\uparrow^{5\times}$, and our Tri-MipRF can be found in Fig. 8 and Fig. 9. Our method consistently renders more fine-grained and anti-aliased images compared with Instant-ngp and Instant-ngp $\uparrow^{5\times}$. The aliasing artifacts are not easy to be observed in still pictures, so readers are highly recommended to watch the supplemental video for better evaluation.

Single-scale Blender Dataset To show more detailed per-scene results of our Tri-MipRF, compared with other cutting-edge methods, on the single-scale Blender dataset, we provide quantitative results of them under three metrics in Tab. 5. Even the

	PSNR								
	chair	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	*Average*
NeRF w/o $\mathcal{L}_{\text{area}}$	29.92	23.27	27.15	32.00	27.75	26.30	28.40	26.46	27.66
NeRF [35]	33.39	25.87	30.37	35.64	31.65	30.18	32.60	30.09	31.23
MipNeRF [3]	37.14	27.02	33.19	39.31	35.74	32.56	38.04	33.08	34.51
Plenoxels [14]	32.79	25.25	30.28	34.65	31.26	28.33	31.53	28.59	30.34
TensoRF [9]	32.47	25.37	31.16	34.96	31.73	28.53	31.48	29.08	30.60
Instant-ngp [37]	32.95	26.43	30.41	35.87	31.83	29.31	32.58	30.23	31.20
Instant-ngp $\uparrow^{5\times}$	34.15	26.79	31.50	36.47	32.51	29.49	33.81	30.78	31.94
Tri-MipRF w/o $\mathcal{M}$	33.09	26.85	31.07	36.08	32.09	29.85	32.66	30.17	31.48
Tri-MipRF (Ours)	37.72	28.55	33.77	39.96	36.51	32.35	38.06	33.59	35.06

	SSIM								
	chair	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	*Average*
NeRF w/o $\mathcal{L}_{\text{area}}$	0.944	0.891	0.942	0.959	0.926	0.934	0.958	0.861	0.927
NeRF [35]	0.971	0.932	0.971	0.979	0.965	0.967	0.980	0.900	0.958
MipNeRF [3]	0.988	0.945	0.984	0.988	0.984	0.977	0.993	0.922	0.973
Plenoxels [14]	0.968	0.929	0.972	0.976	0.964	0.959	0.979	0.892	0.955
TensoRF [9]	0.967	0.930	0.974	0.977	0.967	0.957	0.978	0.895	0.956
Instant-ngp [37]	0.971	0.940	0.973	0.979	0.966	0.959	0.981	0.904	0.959
Instant-ngp $\uparrow^{5\times}$	0.979	0.943	0.978	0.982	0.972	0.959	0.985	0.909	0.963
Tri-MipRF w/o $\mathcal{M}$	0.971	0.941	0.974	0.980	0.967	0.960	0.980	0.901	0.959
Tri-MipRF (Ours)	0.990	0.957	0.986	0.989	0.986	0.972	0.992	0.935	0.976

	LPIPS								
	chair	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	*Average*
NeRF w/o $\mathcal{L}_{\text{area}}$	0.035	0.069	0.032	0.028	0.041	0.045	0.031	0.095	0.052
NeRF [35]	0.028	0.059	0.026	0.024	0.035	0.033	0.025	0.085	0.044
MipNeRF [3]	0.011	0.044	0.014	0.012	0.013	0.019	0.007	0.062	0.026
Plenoxels [14]	0.040	0.070	0.032	0.037	0.038	0.055	0.036	0.104	0.051
TensoRF [9]	0.042	0.075	0.032	0.035	0.036	0.063	0.040	0.112	0.054
Instant-ngp [37]	0.035	0.066	0.029	0.028	0.040	0.051	0.032	0.095	0.047
Instant-ngp $\uparrow^{5\times}$	0.025	0.059	0.023	0.025	0.031	0.049	0.023	0.089	0.041
Tri-MipRF w/o $\mathcal{M}$	0.036	0.066	0.030	0.028	0.039	0.051	0.032	0.099	0.048
Tri-MipRF (Ours)	0.010	0.042	0.014	0.012	0.012	0.029	0.008	0.062	0.024

Table 4. Quantitative per-scene results on the test set of the multi-scale Blender dataset. For each scene, we report the arithmetic mean of each metric averaged over the four scales used in the dataset. The best, second-best, and third-best results are marked in red, orange, and yellow, respectively.

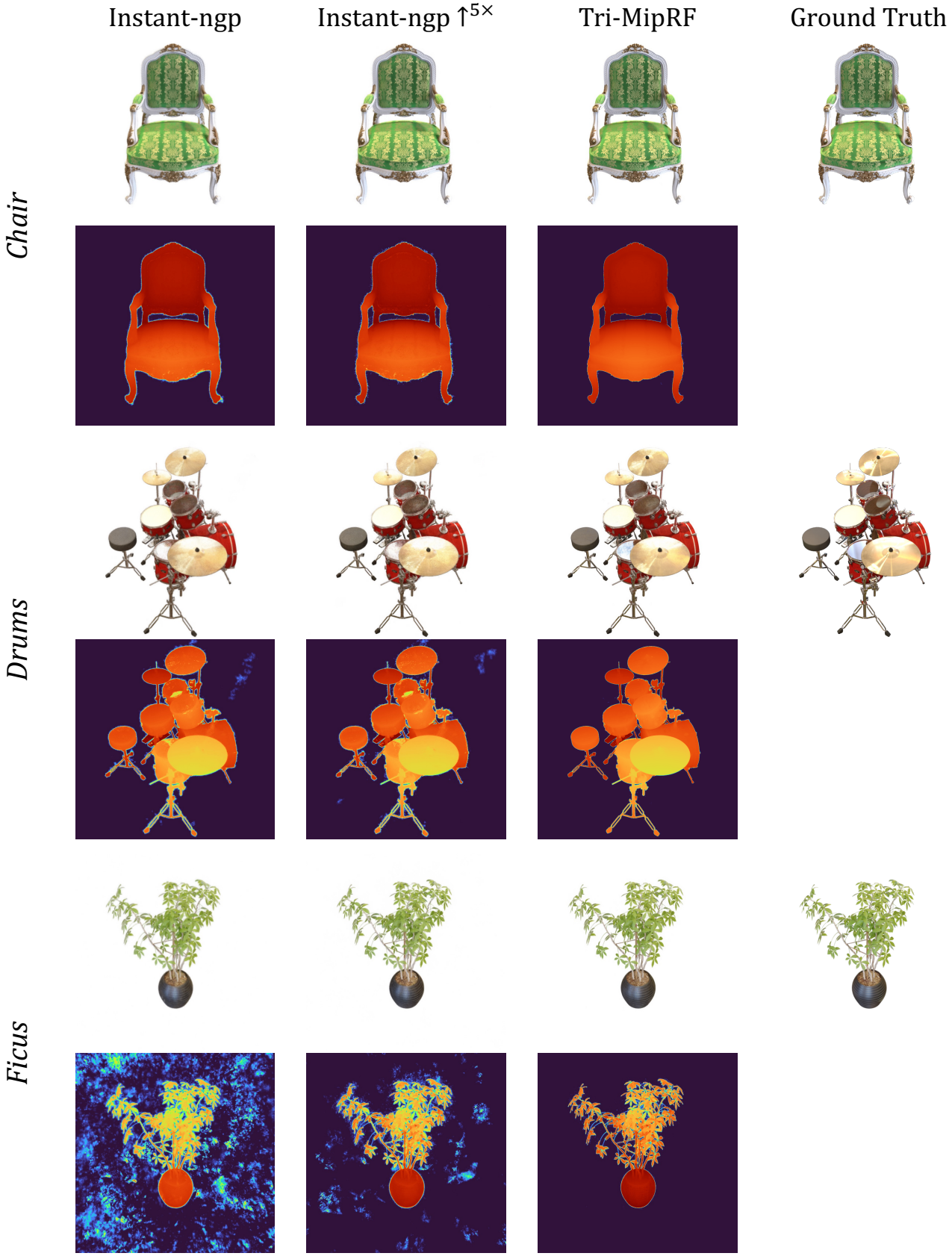
single-scale Blender dataset observes the scene at a roughly constant distant, where the point-sampling-based methods would not suffers from the scale issue, our Tri-MipRF still outperforms them in terms all the three metrics. It demonstrates the high applicability of our method for reconstructing objects at a constant or varying observing distances.

	PSNR								
	chair	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	*Average*
SRN [46]	29.96	17.18	20.73	26.81	20.85	18.09	26.85	20.60	22.26
LLFF [34]	28.72	21.13	21.79	31.41	24.54	20.72	27.48	23.22	24.88
Neural Volumes [29]	28.33	22.58	24.79	30.71	26.08	24.22	27.78	23.93	26.05
Plenoxels [14]	33.98	25.35	31.83	36.43	34.10	29.14	33.26	29.62	31.71
NeRF [35]	34.17	25.08	30.39	36.82	33.31	30.03	34.78	29.30	31.74
DVGO [47]	34.09	25.44	32.78	36.74	34.64	29.57	33.20	29.13	31.95
MipNeRF [3]	35.14	25.48	33.29	37.48	35.70	30.71	36.51	30.41	33.09
TensoRF [9]	35.76	26.01	33.99	37.41	36.46	30.12	34.61	30.77	33.14
Instant-ngp [37]	35.00	26.02	33.51	37.40	36.39	29.78	36.22	31.10	33.18
Tri-MipRF (Ours)	36.10	26.59	34.51	38.54	36.15	30.73	37.75	28.78	33.65

	SSIM								
	chair	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	*Average*
SRN [46]	0.910	0.766	0.849	0.923	0.809	0.808	0.947	0.757	0.846
LLFF [34]	0.948	0.890	0.896	0.965	0.911	0.890	0.964	0.823	0.911
Neural Volumes [29]	0.916	0.873	0.910	0.944	0.880	0.888	0.946	0.784	0.893
Plenoxels [14]	0.977	0.933	0.976	0.980	0.976	0.949	0.985	0.890	0.958
NeRF [35]	0.975	0.925	0.967	0.979	0.968	0.953	0.987	0.869	0.953
DVGO [47]	0.977	0.930	0.978	0.980	0.976	0.951	0.983	0.879	0.957
MipNeRF [3]	0.981	0.932	0.980	0.982	0.978	0.959	0.991	0.882	0.961
TensoRF [9]	0.985	0.937	0.982	0.982	0.983	0.952	0.988	0.895	0.963
Instant-ngp [37]	0.979	0.937	0.981	0.982	0.982	0.951	0.990	0.896	0.963
Tri-MipRF (Ours)	0.985	0.939	0.983	0.984	0.982	0.953	0.992	0.879	0.963

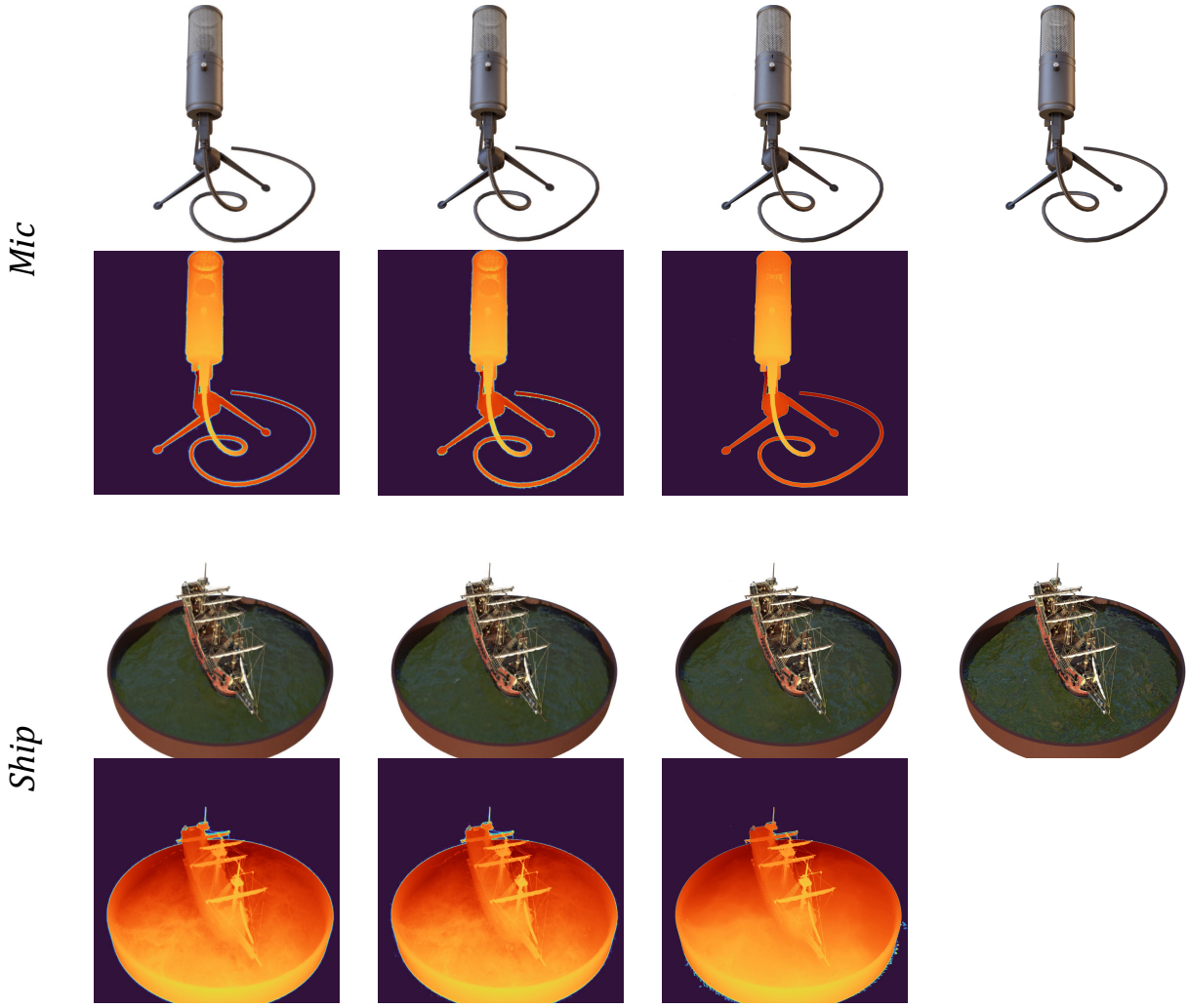
	LPIPS								
	chair	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	*Average*
SRN [46]	0.106	0.267	0.149	0.100	0.200	0.174	0.063	0.299	0.170
LLFF [34]	0.064	0.126	0.130	0.061	0.110	0.117	0.084	0.218	0.114
Neural Volumes [29]	0.109	0.214	0.162	0.109	0.175	0.130	0.107	0.276	0.160
Plenoxels [14]	0.031	0.067	0.026	0.037	0.028	0.057	0.015	0.134	0.049
NeRF [35]	0.026	0.071	0.032	0.030	0.031	0.047	0.012	0.150	0.050
DVGO [47]	0.027	0.077	0.024	0.034	0.028	0.058	0.017	0.161	0.053
MipNeRF [3]	0.021	0.065	0.020	0.027	0.021	0.040	0.009	0.138	0.043
TensoRF [9]	0.022	0.073	0.022	0.032	0.018	0.058	0.015	0.138	0.047
Instant-ngp [37]	0.022	0.071	0.023	0.027	0.017	0.060	0.010	0.132	0.045
Tri-MipRF (Ours)	0.016	0.066	0.020	0.021	0.016	0.052	0.008	0.136	0.042

Table 5. Quantitative per-scene results on the test set of the single-scale Blender dataset. The best, second-best, and third-best results are marked in red, orange, and yellow, respectively.


Instant-ngp
Instant-ngp ↑5×
Tri-MipRF
Ground Truth
Chair
Drums
Ficus

(a)

(b)



(c)

Figure 8. Qualitative full-resolution rendering results of Instant-ngp [37], Instant-ngp $\uparrow^{5\times}$, and our Tri-MipRF on the multi-scale Blender dataset. We show the rendered depth map under the RGB renderings.

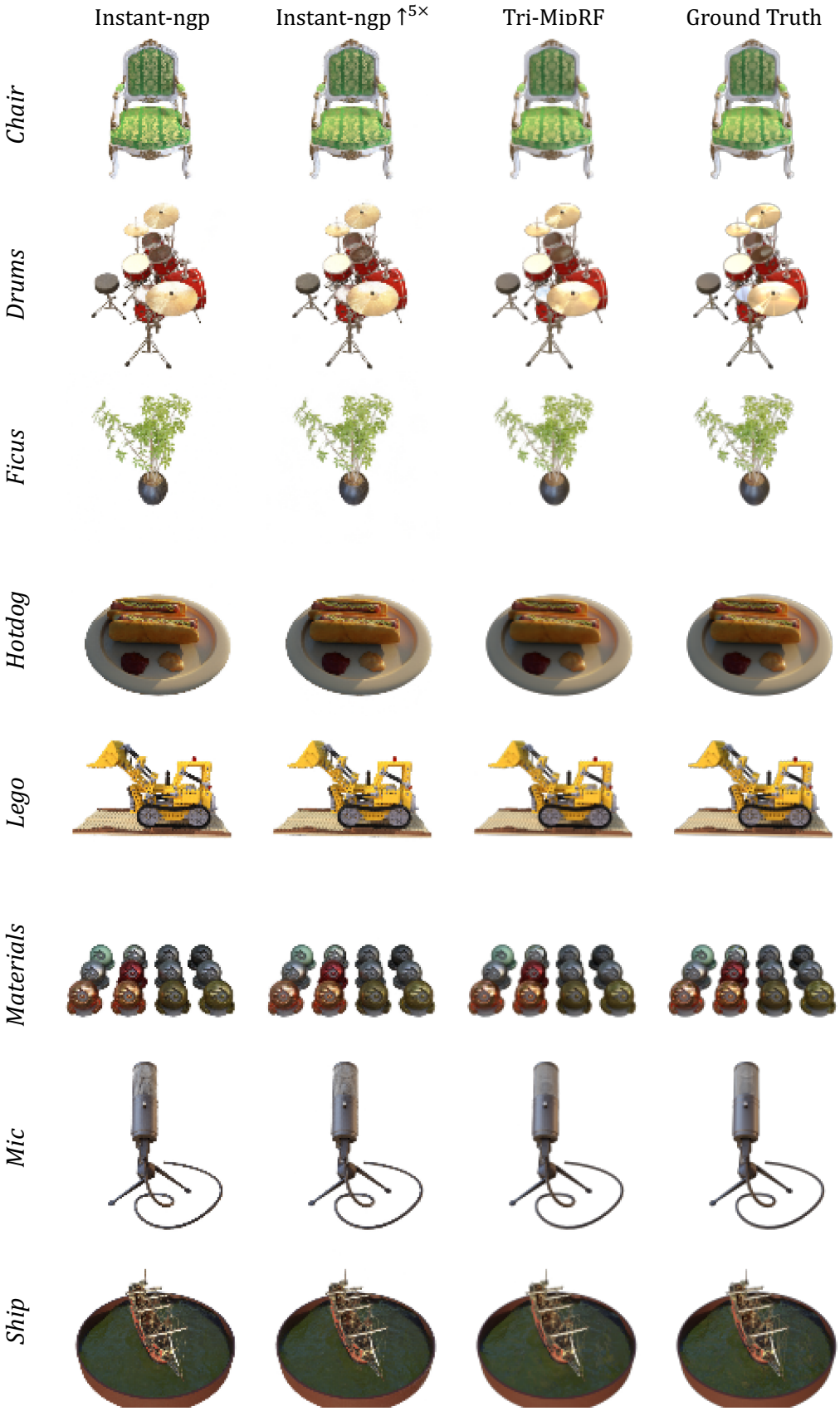


Figure 9. Qualitative $1/8$ resolution rendering results of Instant-ngp [37], Instant-ngp $\uparrow^{5\times}$, and our Tri-MipRF on the multi-scale Blender dataset.