InfNeRF: Towards Infinite Scale NeRF Rendering with O(log n) Space Complexity

Anonymous ECCV 2024 Submission

Paper ID #11289

Abstract. The conventional mesh-based Level of Detail (LoD) technique, exemplified by applications such as Google Earth and many game engines, exhibits the capability to holistically represent a large scene even the Earth, and achieves rendering with a space complexity of $\mathcal{O}(\log n)$. This constrained data requirement not only enhances rendering efficiency but also facilitates dynamic data fetching, thereby enabling a seamless 3D navigation experience for users.

In this work, we extend this proven LoD technique to Neural Radiance Fields (NeRF) by introducing an octree structure to represent the scenes in different scales. This innovative approach provides a mathematically simple and elegant representation with a rendering space complexity of $\mathcal{O}(\log n)$, aligned with the efficiency of mesh-based LoD techniques. We also present a novel training strategy that maintains a complexity of $\mathcal{O}(n)$. This strategy allows for parallel training with minimal overhead, ensuring the scalability and efficiency of our proposed method. Our contribution is not only in extending the capabilities of existing techniques but also in establishing a foundation for scalable and efficient large-scale scene representation using NeRF and octree structures.

Keywords: novel view synthesis · radiance fields · level of detail

1 Introduction

Recent progress in Neural Radiance Fields (NeRF) has significantly advanced the field of photo-realistic novel-view synthesis [2,4,25,34]. Surpassing traditional 3D representations like meshes, NeRF demonstrates a superior capability in generating accurate novel views, particularly for transparent or reflective geometries. However, despite these advancements, a predominant focus of NeRF research has been confined to single objects or limited-scale scenes. The challenge of effectively representing extensive, large-scale scenes remains largely underexplored and unresolved.

Large-scale scene reconstruction has a wide range of applications, such as land surveys, search and rescue operations, construction planning, and navigation. A prime example of its potential can be seen in platforms like Google Earth, enabling users to virtually traverse the globe, which not only showcases the magnificent beauty of natural landscapes but also facilitates the exploration

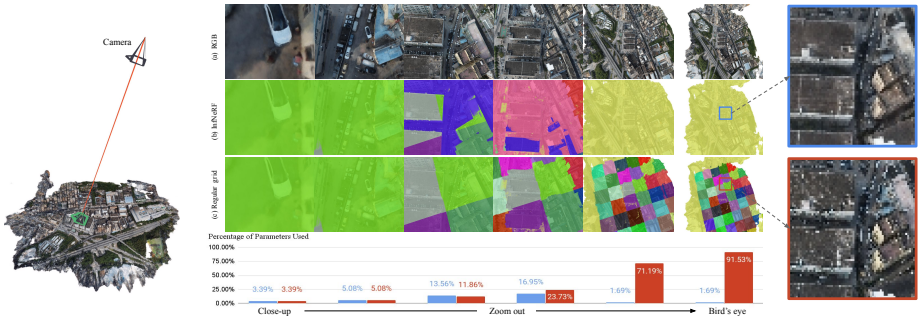

Fig. 1: Efficient large-scale scene rendering with the proposed InfNeRF. (a) shows the rendered RGB frames by InfNeRF under a zoom-out camera trajectory. Unlike existing methods that employ regular grid partition at a single scale (c), we propose a hierarchical NeRF framework to decompose the scene in both space and scale dimensions (b), in a spirit similar to the classical LoD techniques. This innovation is color-coded in (b) and (c), with each color representing a unique NeRF node within the whole model. During rendering, InfNeRF only requires a fraction of the nodes, considerably diminishing the memory footprint and expediting the model retrieval process. As demonstrated in the middle bottom figure, a mere maximum of 16.95% parameters of the entire model are used for the rendering process of InfNeRF, in stark contrast to the 91.53% demanded by the baseline approach. In addition to the improved efficiency, the hierarchical structure of InfNeRF naturally improves its anti-aliasing capability. As shown in the rightmost column, InfNeRF shows notably reduced noise and enhanced smoothness while the existing methods suffer from severe aliasing artifacts.

of intricate historic buildings. Extending NeRF to handle large-scale scenes is intriguing and important because it opens up possibilities for more immersive and realistic representations of our world.

1.1 Block-NeRF and Mega-NeRF

Simply scaling up the architecture of NeRF proves insufficient to address this challenge, primarily due to the escalating space complexity. As the scenes expand, it becomes impractical to store the entire model in the memory of standard devices such as mobile phones or consumer PCs. Consequently, dividing the model and storing it in disk or cloud systems becomes necessary.

Towards this direction, recent large-scale NeRF approaches, such as Block-NeRF [32] and Mega-NeRF [36] divide the whole neural field equally in space into smaller, more manageable blocks, referred to as grid partition in this paper. This allows the rendering device to load only the data relevant to the camera's current location. However, these methods encounter challenges in scenarios requiring a bird's eye view of the entire scene. Such a panoramic view plays a crucial role in human interactive navigation. For example, to adjust the position of the camera, it is much more efficient to first zoom out, identify the target location, and then zoom in again, compared to the laborious process of navigating at a constant zoomed-in level. However, in such scenarios, the existing methods need to load

all blocks simultaneously, which can overwhelm the memory capacity of the device. In addition to the memory issue, these approaches are prone to aliasing artifacts that arise from an insufficient sampling rate for high-frequency signals as illustrated in Fig.1.

1.2 Proposed InfNeRF

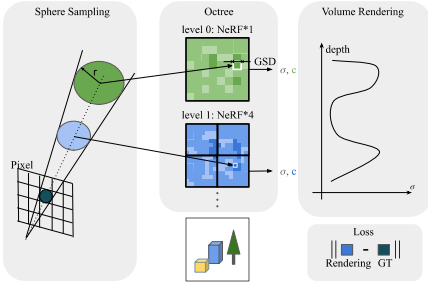

Fig. 2: Each sampling sphere along the ray is assigned a radius corresponding to its pixel size. Depending on the radius, the sampling spheres will be sent to different nodes of the octree to query their densities and colors. The densities and colors are integrated to produce the final color for the pixel.

In this paper, we propose InfNeRF, a novel method that addresses the above challenges. The core concept of InfNeRF is inspired by the classical mesh-based Level of Detail (LoD) techniques [12,16,29,34]. An overview is illustrated in Fig.2. Specifically, we construct an LoD tree for the target scene, where each node encapsulates a small NeRF with a roughly equivalent number of parameters. The root node provides a coarse representation of the entire scene, while its successive children progressively describe more detailed sub-regions. During rendering, each sampling point is forwarded by a different NeRF, depending on its location and radius. The density and color then are accumulated to determine the pixel color. As illustrated in Fig. 3, InfNeRF only requires a minimum subset of the nodes in rendering, chosen based on their proximity to the camera and their intersection within the frustum. This significantly reduces the memory burden and the I/O time for retrieving parameters from disk or cloud storage.

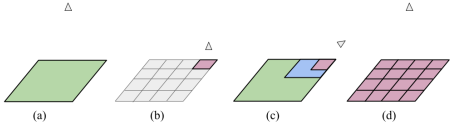

Fig. 3: As shown in (a) and (b), no matter zooming out or in, only one node is required. In (c), when looking at the horizon, approximately $\mathcal{O}(\log n)$ nodes are required which is the upper bound of InfNeRF. In (d), in contrast, other methods require all the blocks when zoom out resulting in an upper bound of $\mathcal{O}(n)$.

The memory complexity is notably efficient at $\mathcal{O}(\log n)$, where n denotes the total information captured in the reconstruction. In the real world, with $\log n \approx 20$, we can represent the Earth down to a resolution of centimeters.

One remarkable feature of InfNeRF is that the parent nodes in the LoD tree act as a smoother, low-pass-filtered version of the scene. Therefore, the proposed method naturally removes high-frequency components in distant views, which effectively reduces the aliasing artifacts and en-

hances the visual quality of the generated images (Fig. 1). In our experiments, InfNeRF achieves superior rendering quality for large-scene reconstruction, with an improvement of over 2.4dB in PSNR while accessing only 17% of the total parameters.

We also propose a pruning algorithm to prune the octree into an effective and compact representation using sparse points from SfM. This algorithm adapts the tree to each scene intelligently without any human intervention.

In addition, to facilitate efficient training of InfNeRF, we introduce a novel training strategy that maintains a time and space complexity of $\mathcal{O}(n)$. Moreover, this strategy allows for the training process to be effectively parallelized across machines.

InfNeRF distinguishes itself from existing methods such as Block-NeRF [32] and Mega-NeRF [36] in that it divides the scene in both spatial and scale dimensions while existing approaches typically only consider the spatial aspect. InfNeRF imposes no constraints on the underlying NeRF model and makes no assumptions about the scene, making it an adaptable framework that can seamlessly accommodate and leverage the rapid advancements in NeRF models.

In summary, our contributions are as follows:

- We propose a novel InfNeRF for large-scale scene reconstruction. By integrating neural fields with the classic LoD, InfNeRF achieves efficient rendering with a significantly reduced memory footprint.
- Our InfNeRF demonstrates a substantial improvement in rendering quality, with over 2.4dB increase in PSNR, and exhibits superior anti-aliasing capabilities.
- We develop an efficient and scalable training strategy tailor-made for InfNeRF, addressing the complexities inherent in large-scale neural field training.

2 Related Work

Photogrammetry. Among the established reconstruction techniques, mesh-based reconstruction methods called photogrammetry have been used widely for it efficiency and robustness. This workflow includes successive steps: Structure from Motion (SfM) [1,26,30,42] for camera pose estimation, Multi-View Stereo (MVS) techniques [5,18,31] for depth computation, mesh reconstruction [23,39], and texture mapping [40]. These steps form the basic workflow to create a textured mesh that can be efficiently rendered by a traditional graphics pipeline. Open-source tools, such as COLMAP [30] and OpenMVS [7], have seamlessly integrated this workflow and gained widespread adoption in the academic community. Furthermore, commercial software solutions like ContextCapture and DJI Terra [14] have significantly elevated the robustness and efficiency to meet industrial standards.

Level of Detail. When the scene is large, it becomes impractical to store it entirely in memory. In this case, the Level of Detail (LoD) or Hierarchical Level

of Detail (HLoD) [12,16,19,20,24,38] provides a framework for efficiently storing, fetching, and rendering the scene.

Take tiled 2D map [35] used widely in 2D navigation system for an example. The maps in different resolutions are cut into 256×256 PNG images referred to as tiles. These tiles are organized into a quadtree structure, typically comprising 22 levels. This hierarchical arrangement allows web browsers to rapidly fetch data and render any location at any zoom level with constant space complexity.

A similar technique can also be applied to 3d mesh. After the reconstruction of the scene into one detailed mesh, this textured mesh undergoes multiple downsampling [13,41] resulting in a mesh pyramid with varying levels of detail, ranging from fine to coarse. Subsequently, these meshes are partitioned into numerous tiles and organized hierarchically, resembling a tree structure similar to a quadtree in 2D mapping. During rendering, open source 3D engines such as CesiumJS [8] or OpenSceneGraph [6] walk the tree, then progressively fetch tiles based on their proximity and intersection with the frustum. This LoD technique enables users to have a seamless 3D navigation experience in a very large scene while maintaining a low memory footprint.

NeRF and Large-Scale NeRF. NeRF [25] employs Multi-Layer Perceptrons (MLPs) for NeRF to capture a continuous density and viewpoint-dependent radiance of a scene. Plenoxel introduces a voxel-dense grid [17], which significantly improves the training and rendering speeds of NeRF, albeit with a significant VRAM requirement. Methods like TensoRF [10] and MERF [28] decompose the dense grid into low dimension feature tensor, while InstantNGP [27] opts to store these vectors in a sparse hash table. These strategies greatly enhance model compactness while maintaining speed.

Large-Scale Scene and Partition at Space Dimension. While previous NeRF approaches primarily focus on scenes of limited scale, scaling up NeRF to handle large-scale scenes, such as cities or even the entire Earth, would empower a broader range of applications. Mip-NeRF 360 [3] proposed a scene contraction technique to compress the unbounded scene into a bounded region. However, this approach compromises quality in the compressed areas. Grid-Guided NeRF [45] extends its capacity by combining a 2D feature grid and MLP, but its reliance on a 2D plane assumption limits its capability to handling complicated 2D scene. Block-NeRF [32], Mega-NeRF [36], SMERF [15] and ScaNERF [43] divide the scene into several sub-regions and reconstruct them separately. In its specialized indoor scenario, SMERF can even achieve impressive real-time rendering. However, all these methods suffer from a common issue: when rendering a bird's eye view, the entire model needs to be loaded into VRAM, limiting their scalability. Additionally, they may suffer from severe aliasing artifacts.

Anti-aliasing and Partition at Scale Dimension. Anti-aliasing has been a fundamental problem in computer graphics and image processing for a long time

and has been extensively explored in the rendering community. The common approaches to tackle this problem can be categorized into two ways: super-sampling and pre-filtering. In the context of NeRF, super-sampling is casting multiple rays per pixel as in MobileNeRF [11], or sampling multiple points with perturbation, as demonstrated in Zip-NeRF [4]. This straightforward strategy is useful but also computationally expensive. On the other hand, Mip-NeRF [2] introduces the concept of pre-filtering through the integrated position encoding. Other research has explored scene partitioning at different scales to mitigate aliasing. For instance, BungeeNeRF [44] proposes a multi-scale MLP that progressively adds detail when the camera zooms in. However, akin to Mip-NeRF, it may only accurately represent the center of the scene, lacking detail elsewhere. Tri-MipRF [22] and D.Hu et al. [21] adopt a down-sampling strategy to create a smoother feature grid at each scale after training. PyNeRF [37] trained a multi-resolution hashed feature grid. These methods demonstrate impressive anti-aliasing capabilities and consistently high-quality rendering across the entire scene. However, their lack of space partitioning leads to challenges when scaling up, requiring the whole finest layer to render a single frame.

In contrast to the methods mentioned in this subsection, the anti-aliasing problem is addressed with inherently no additional cost in InfNeRF. Our parent node in the octree is a downsampled copy of the child nodes. Rendering these nodes at an appropriate zoom level yields a result free from aliasing artifacts.

3 Method

We present InfNeRF, a framework for efficient large-scale scene reconstruction using a LoD octree structure (Section 3.1). This tree structure naturally enables anti-aliasing rendering (Section 3.2), which significantly enhances the visual quality of rendered images. We also introduce tree pruning to improve model sparsity and compactness, along with a rendering approach tailored for the pruned structure (Section 3.3). Moreover, we propose an efficient training strategy for InfNeRF in Section 3.4. Finally, we also provide a comprehensive analysis of the computational complexity of InfNeRF in supplementary materials, offering insights into the efficiency and scalability of our approach in handling large-scale scene reconstructions.

3.1 Tree Structure

The core of our InfNeRF is a LoD structure, essentially an octree where each node represents a specific cubic space of the scene, known as the node's Axis-Aligned Bounding Box (AABB). The octree is illustrated in the middle of Fig. 2. InfNeRF begins with the entire scene encapsulated within a root node, which is then divided into eight smaller cubes, corresponding to the child nodes. This division process continues recursively, partitioning the scene into increasingly finer parts, both in terms of space (size of the area) and scale (level of detail).

Each node in this tree is paired with its own NeRF, which represents the node's AABB at a certain scale. Our approach is flexible regarding the choice of NeRF model. It can be a simple MLP [25], Instant-NGP [27], tri-plane [9,22], or even a combination of them, as long as it can query the density σ and the color $\mathbf{c}$ for a given 3D point $\mathbf{x}$ and viewing direction $\mathbf{d}$:

$$\sigma, \mathbf{c} = \text{NeRF}(\mathbf{x}, \mathbf{d}). \tag{1}$$

In this work, we use Instant-NGP for its efficiency in training. Instant-NGP discretizes each cube into a grid of voxels, and the number of voxels along each axis of the cube is defined as grid size. Correspondingly, the Ground Sampling Distance (GSD) of a node can be approximated as:

$$\text{GSD} = \frac{\text{length of AABB}}{\text{grid size}}, \tag{2}$$

whose unit is in meters. The GSD, in this context, serves as a measure of resolution or granularity, indicating the level of detail that the node can capture about the physical world - the smaller the GSD, the finer the details. For example, if a node $\mathcal{A}$ represents a 1 km^3 cube with a grid size of 2048, the GSD will be $\frac{1000}{2048} \approx 0.48$ meters. As each node in our octree shares the same grid size, the child nodes of $\mathcal{A}$ have a GSD of 0.24 meters, thereby capturing more detailed information. This is also illustrated in Fig. 2, where the GSD of a cube at level 1 is half that of a cube at level 0.

3.2 Anti-Aliasing Rendering

Similar to the original NeRF, the rendering process of InfNeRF accumulates the density and color of sampling points along a ray to determine the RGB of a pixel. However, the key difference is that the sampling points are queried from different nodes of our octree rather than a single NeRF as shown in Fig. 2. In the following, we show how we find the corresponding node for a given sampling point.

According to the sampling theory [41], a sample on the ray is not merely an infinitely small point, but rather a Gaussian sphere with a certain volume. For simplicity, we model this as a sphere with radius r that corresponds to the pixel size,

$$r \approx \text{depth} \times \frac{\text{pixel width}}{2 \times \text{focal length}}, \tag{3}$$

as illustrated on the left of Fig. 2. Given a sampling sphere with position $\mathbf{x} \in \mathbb{R}^3$ and radius $r \in \mathbb{R}$, we aim to find the node whose granularity (GSD) matches r and whose AABB contains $\mathbf{x}$. In this way, we ensure InfNeRF samples a proper region in the 3D space rather than just a single point, leading to a desirable anti-aliasing effect.

Since the GSD is discrete in our tree structure, we can either interpolate between two tree levels, as done in PyNeRF [37], or simply sample from the nearest

level. InfNeRF opts for the latter, rounding towards the root for computational efficiency. Specifically, the level to sample a given sphere is determined by:

$$\text{level} = \lfloor \log_2 \frac{root.gsd}{r} \rfloor, \tag{4}$$

where $root.gsd$ denotes the GSD of the root node, *i.e.*, level 0.

As illustrated in Fig. 3(a), when the camera is far away from the scene, all spheres are relatively large, and therefore, their densities and colors are estimated by the root node, resulting in a smooth anti-aliasing effect. Conversely, as shown in Fig. 3(b), zooming in with the camera reduces the size of the spheres, which shifts the estimation to the leaf nodes, yielding a sharp and detailed image. As InfNeRF necessitates only a minimum subset of the nodes for rendering a frame, it requires loading merely $\mathcal{O}(\log n)$ number of parameters from the whole model into memory. The upper bound $\mathcal{O}(\log n)$ is reached when the camera looks at the distant horizon, as illustrated in Fig. 3(c). In this scenario, multiple-level nodes are engaged simultaneously, from the root node representing the blurring horizon to the leaf node representing foreground details.

3.3 Tree Pruning

While we can build a perfect octree as in Section 3.1, this may not be efficient, particularly for non-uniformly sampled scenes. To optimize the structure, it is important to identify the essential nodes necessary for rendering all input images, effectively pruning the superfluous parts of the tree.

In this work, we utilize sparse points $\mathbf{x} \in \mathbb{R}^3$ obtained from the Structure from Motion (SfM) technique to approximate scene geometry. Each sparse point $\mathbf{x}_i$ observed in M_i images is treated as M_i spheres, each with a radius $r_{ij}, j = 1, \cdots, M_i$ as defined in Section 3.2. We find the corresponding nodes for all the $\sum M_i$ spheres in an ideal, infinitely deep octree as described in Section 3.2. We retain only the nodes that correspond to sparse spheres and their respective parent nodes, while the others are pruned away.

Compared to MegaNeRF or BlockNeRF, which split the scene uniformly based on space occupancy, our approach offers a more intelligent resource allocation strategy. It adaptively creates deeper branches in areas with rich details and shallower branches in coarse regions. In addition, our solution provides users with the flexibility to control the depth of the tree, enabling a tailored balance between reconstruction quality and computational efficiency.

Pruned Tree Rendering. The pruned tree structure introduces new challenges in InfNeRF rendering, as the optimal node determined by Eq. 4 may no longer exist after pruning. In such scenarios, we instead process the sampling sphere with the nearest available ancestor of the intended node.

Specifically, given a sampling sphere $(\mathbf{x}, r)$, the rendering process recursively travels down the tree. If the sphere successfully arrives at the node of the expected level, the density and color of the sphere will be estimated by that node.

Algorithm 1 Forward run of the pruned tree in Section 3.3. It processes a sampling sphere centered at $\mathbf{x}$ with radius r (Eq. 3) and viewing direction $\mathbf{d}$, starting from $node = root$.

```
function FORWARD(node, x, r, d)
    if r ≥ node.gsd then
        return node.NeRF(x, d)                    ▷ Process at current node
    end if
    i ← find_subcube(node.AABB, x)
    child ← node.children[i]
    if child = nil then
        return node.NeRF(x, d)                    ▷ Revert to parent node
    else
        return FORWARD(child, x, r, d)            ▷ Descend to child node
    end if
end function
```

However, if this process is interrupted by a pruned node, the rendering of the sphere reverts to its parent node. We outline this rendering strategy with the pseudo-code in Algorithm 1. Finally, each sampling sphere is processed by the NeRF of the located node, ensuring accurate and efficient rendering despite the pruned tree structure.

3.4 Training

To train the proposed InfNeRF, we accumulate the density and color of all spheres sampled on a ray using volume rendering [25] and compare the resulting RGB values with the pixels of the observed images using L2 loss $\mathcal{L}_{\text{RGB}}$.

To remove the cloudy effect from the untrained areas, particularly in the sky, and enhance visual quality in the bird's-eye view, we introduce a transparency loss $\mathcal{L}_{\text{trans}} = -\|\exp(-\sigma(\mathbf{x}))\|_1$, which uniformly samples points within the AABB and uses L1 loss to encourage their density to approach 0.

To further improve the result, we also incorporate the distortion loss $\mathcal{L}_{\text{distort}}$ from [3] and a regularization loss $\mathcal{L}_{\text{reg}}$ to promote consistency across different scales by encouraging the same locations from different scales have similar density. The total loss of our training is summarized as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{RGB}} + w_1 * \mathcal{L}_{\text{distort}} + w_2 * \mathcal{L}_{\text{reg}} + w_3 * \mathcal{L}_{\text{trans}} \tag{5}$$

Pyramid Supervision. The photographs in most datasets are captured by drones from relatively close proximity, resulting a lack of multi-resolution information. While this offers sharp and detailed pixels that can well supervise the lower-level nodes, it leads to insufficient training for the higher-level nodes close to the root, which require low-frequency supervision signals.

To mitigate this issue, we augment the training set using an image pyramid constructed by repeatedly downsampling the high-resolution images. As illustatrated at Fig. 4, this pyramid supervision ensures that the upper nodes receive

adequate training, effectively addressing the undertraining problem and facilitating a more balanced learning across all levels of the model.

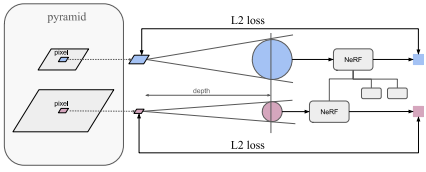

Fig. 4: As illustrated in the bottom row, the high-resolution image creates smaller spheres that are estimated by the leaf node, supervising its training. To ensure effective supervision of the upper nodes, it is necessary to augment the training set with an image pyramid, as illustrated in the top row.

Note that the pyramid supervision is not applicable to conventional NeRF models. The capability of InfNeRF to decompose the scene along the scale dimension is what enables it to effectively utilize multi-resolution training data. The benefit it extends beyond simple pyramid supervision. For instance, when a particular location is captured in photographs taken from varying distances, conventional NeRF models tend to compromise the low-frequency and high-frequency supervision from these images, leading to sub-optimal results. In contrast, InfNeRF naturally separates the blurry and sharp information to supervise different nodes, resulting in more accurate rendering. This concept applies vice versa as well. For instance, within a single image, detailed foreground pixels may supervise to lower-level nodes, while blurry background pixels may automatically supervise to higher-level nodes.

Distributed Training. As the scene grows, the demands on VRAM and training time also increases. Straightforwardly using existing distributed training approaches, such as Distributed Data Parallel (DDP), leads to increased VRAM consumption and substantial communication overhead. To address these limitations, we propose a new distributed training strategy.

Given a specific tree level $L \in \mathbb{Z}^+$, we split the octree into two parts: the upper tree which includes all nodes from level 0 to level $L-1$, and the lower part which is a forest including a maximum of 8^L subtrees. During training, the upper tree is shared across all devices, and the parameters are updated concurrently. Meanwhile, individual subtrees or clusters of adja-

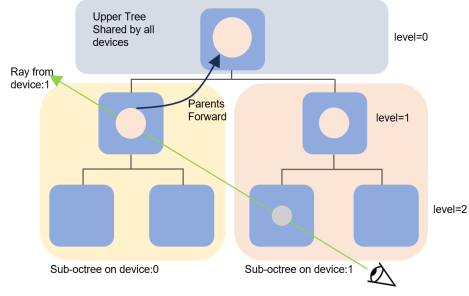

Fig. 5: Illustration of the distributed octree as a binary tree for simplicity. The tree can be divided into the root and 2 branches. The root is shared by all devices like Conventional DDP. And each branch is owned by one device without sharing the weights. When a sampling sphere descends to a pruned node, its density and color will be estimated by the shared parent node.

cent subtrees in the lower part, are assigned among different devices and updated independently. In other words, each device only maintains the shared upper tree and a subset of the lower subtrees, effectively reducing the VRAM footprint and communication overhead. During forward pass, if a sampling sphere descends to a subtree that is not owned by the current device, it will be forwarded by its parent node in the shared tree as demonstrated in Fig. 5. Consequently, during training, the only data transfer is the all-reduce operation on the gradient of the shared upper tree, which leads to a more efficient and scalable training process.

It is worth mentioning that the training data can also be divided among devices for further acceleration. Intuitively, if a ray on a particular device does not intersect with the AABB of any subtree on that device, it would be not sensible to sample the corresponding pixel of the ray. Therefore, we find a 2D AABB for each device by projecting the sparse points inside the subtrees' 3D AABBs back onto images. Then, during the training phase, only the pixels that fall within the 2D AABB are sampled and utilized.

4 Experiments

We show the main results in this section and present more analysis and evaluations in the supplementary material. The source code and trained models will be released to the public.

Implementation Details. Inf-NeRF is implemented using the NeRFStudio framework [33]. Each node's NeRF is an Instant-NGP implemented by tiny-cuda-nn, with default parameters such as a grid size of 2048 and 16 levels of hash tables, each with a size of 2^{19}. The tree is limited to 4 levels, resulting a reasonable resolution of 5cm for a scene spanning 1 km. More implementation details are provided in supplementary materials.

4.1 Experiment on Space Complexity

The crucial metric of our study is the required model size during rendering. It determines how much information must be retrieved through the network and stored in the GPU. We compare three methods with similar total model sizes, including (1) Nerfacto [33], an improved Instant-NGP [27] and whose capacity is scaled up 64 times to match our model. (2) Nerfacto-s, a 4 levels Nerfacto, representing the scale partition methods like PyNeRF [37]. (3) InfNeRF. (4) InfNeRF leaf, using the same space partition as InfNeRF, but only

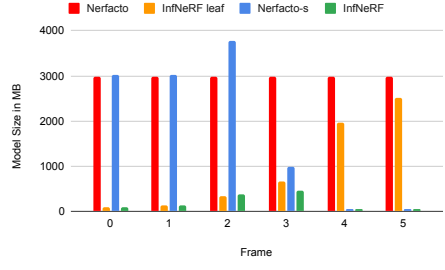

Fig. 6: We rendered a video starting from a close-up view and zooming out to the full scene and recorded the model size required during rendering. Thanks to the LoD tree structure, InfNeRF demonstrated a much smaller memory footprint.

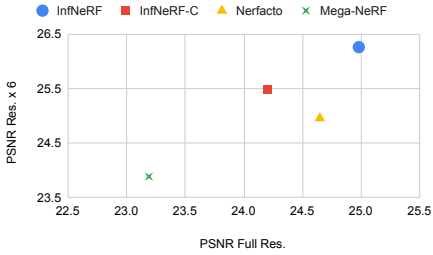

Fig. 7: The PSNR for full resolution and across 6 resolutions of 4 methods are presented in the above plots. InfNeRF is superior to other methods in both dimensions.

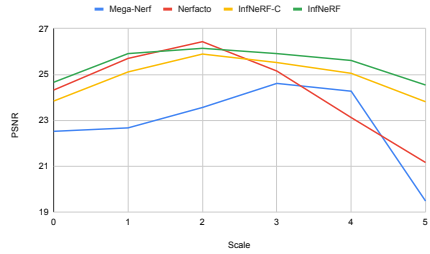

Fig. 8: Due to the aliasing effects, both Nerfacto and Mega-NeRF perform poorly at high scale levels, In contrast, InfNeRF exhibited more stable performance across all levels.

having leaf nodes. It can be considered as an improved version of xy grid partition.

In this experiment, a zoom-out trajectory from a close-up view to a bird's-eye view in VGA resolution is rendered by these 4 methods, and the model size in Megabytes required for 6 keyframes is presented in Fig 6. InfNeRF exhibits a memory footprint that is only 17% of InfNeRF leaf and 12.3% of Nerfacto-s. The trajectory and the rendered images are shown in Fig. 1 and in the video of supplement material.

4.2 Experiment on Rendering Quality

Baseline. We compare the rendering quality of our approach with (1) Nerfaco [33]. (2) Mega-NeRF [36] with 25 partitions. In addition to these two methods, we also train and evaluate a smaller version of InfNeRF, called InfNeRF compact (InfNeRF-C), whose model size is similar to Mega-NeRF.

Metrics. Quantitative results are reported based on metrics such as PSNR, SSIM, and LPIPS using the Alex implementation. As zoom-out is a crucial use case in our paper, we not only report metrics at the finest resolution,denoted as PSNR0, but more importantly, the mean metrics over 6 scaling levels, denoted as PSNR, SSIM, LPIPS.

Comparison. The PSNR0 (full resolution) and all 3 metrics for multi-resolution are listed in Table 1. The mean of 2 PSNR across the four datasets are also plotted in Fig.7. Comparing between two methods with similar rendering model sizes, InfNeRF outperforms Mega-NeRF by 2.4dB in multi-resolution PSNR. Even the InfNeRF compact with a much smaller rendering model size, still outperforms Mega-NeRF by 1.6 dB. This illustrates the overall performance of our proposed method. When comparing two models with large model sizes, InfNeRF and Nerfacto, theoretically, they should exhibit similar PSNR0. Our experiments confirm this hypothesis. InfNeRF slightly outperforms Nerfacto by 0.3

Table 1: Quantitative comparison on four public large-scale scene datasets. We report PSNR0 from full resolution and PSNR, SSIM, and LPIPS metrics from multi-resolution. The **best results** are highlighted.

Scene	Residential				Sci-Art			
Metric	PSNR0↑	PSNR↑	SSIM↑	LPIPS↓	PSNR0↑	PSNR↑	SSIM↑	LPIPS↓
Nerfacto	24.32	24.31	0.772	0.216	26.75	26.48	0.827	**0.182**
Mega-NeRF	22.52	22.85	0.748	0.265	24.37	24.22	0.808	0.202
InfNeRF compact	23.84	24.87	0.785	0.198	25.83	25.81	0.805	0.212
InfNeRF	**24.66**	**25.46**	**0.820**	**0.158**	**27.03**	**26.96**	**0.839**	0.1918
Scene	Rubble				Building			
Metric	PSNR0↑	PSNR↑	SSIM↑	LPIPS↓	PSNR0↑	PSNR↑	SSIM↑	LPIPS↓
Nerfacto	24.99	26.09	0.692	0.336	22.53	22.94	0.663	0.325
Mega-NeRF	24.17	26.02	0.73	0.294	21.70	22.43	0.696	0.282
InfNeRF compact	24.62	27.27	0.731	0.274	22.52	24.02	0.703	0.280
InfNeRF	**25.12**	**27.92**	**0.781**	**0.203**	**23.11**	**24.70**	**0.740**	**0.242**

dB, demonstrating that the octree partitioning of InfNeRF does not compromise rendering quality. However, when considering multi-resolution, thanks to the LoD anti-aliasing effect, the difference enlarges to 1.3 dB. The same conclusion can be drawn from the quality comparison in Fig.9. In the last two rows of each dataset, which display the low-resolution images, we can observe that without the anti-aliasing effect, Nerfacto and Mega-NeRF generate noisy results, especially the heavy moire pattern in the left bottom dataset green region. And InfNeRF generates satisfying results at both fine and coarse levels.

It would be insightful to delve deeper into the performance of different scales. The PNSR of 6 scale of residential dataset is plotted in Fig.8, where the horizontal axis represents the scale of the image. A scale of 0 corresponds to full resolution, while a scale of 5 corresponds to 1/32 resolution. In our experiment, Nerfacto and Mega NeRF reconstruct the scene with only one level, resulting in a lack of anti-aliasing effects and consequently lower PSNR at higher scale levels (lower resolution). As the resolution increased, these 2 methods' performance improved. However, as the resolution continued to rise, the visual information surpassed the network capacity limit, leading to a drop in PSNR. Combining the aliasing and under-fit effects, the curves of the two methods form a peak at the intermediate scale. On the other hand, the two versions of InfNeRF, compact and full, distribute the model capacity to each scale intelligently, resulting in more consistent PSNR values across all scales.

5 Conclusion and Future work

In this work, we introduce InfNeRF for large-scale scene reconstruction. InfNeRF employs an LoD tree to divide the scene in space and scale into cubes, which are reconstructed using many NeRFs. During rendering, InfNeRF selectively re-

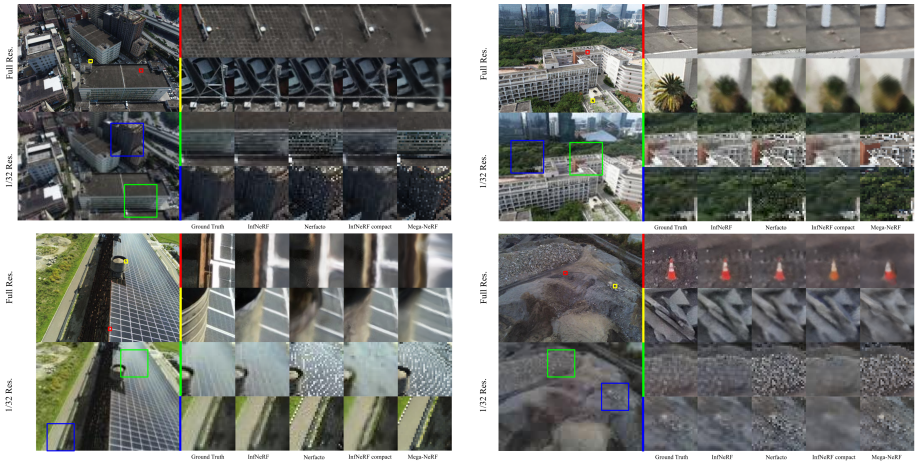

Fig. 9: Qualitative comparison between the baseline and InfNeRF on four public large-scale datasets reveals outstanding rendering quality of InfNeRF. Mega-NeRF appears blurry and fails to capture details effectively. In low resolution, both Mega-NeRF and Nerfacto suffer severe aliasing, particularly evident in moire patterns on the solar panel. Leveraging the LoD octree, InfNeRF not only captures most details in full resolution but also renders a smooth anti-aliasing image in low resolution, with much less memory footprint.

trieves a minimal subset of nodes, significantly reducing memory requirements and I/O time for parameter retrieval from disk or cloud storage. The memory complexity for rendering with InfNeRF is notably efficient at $\mathcal{O}(\log n)$. Another outstanding feature of InfNeRF is that parent nodes in the LoD tree act as a smoother version of the scene, effectively reducing aliasing artifacts. In our experiments, InfNeRF achieves superior rendering quality, with an improvement of over 2.4dB in PSNR while accessing only 17% of the total parameters. Furthermore, InfNeRF imposes no constraints on the underlying NeRF, allowing seamless integration with faster and smaller NeRF models in the future to enhance their scalability and anti-aliasing quality.

InfNeRF demonstrates its potential for large-scale scene reconstruction. However, there are several aspects for future exploration and improvement. Firstly, the reconstruction time and computational burden still exceed traditional, well-optimized photogrammetry methods [14, 30], necessitating further performance enhancements. Secondly, exploring the combination of octree from diverse image sources, such as satellites, planes, and drones, at different times and scales, could enable the creation of larger scenes. With these works, digitizing the Earth using NeRF will become a compelling possibility for the future.

References

1. Agarwal, S., Furukawa, Y., Snavely, N., Curless, B., Seitz, S.M., Szeliski, R.: Reconstructing rome. Computer **43**(6), 40–47 (2010) 4

2. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021) 1, 6
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022) 5, 9
4. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. arXiv preprint arXiv:2304.06706 (2023) 1, 6
5. Bleyer, M., Rhemann, C., Rother, C.: Patchmatch stereo-stereo matching with slanted support windows. In: Bmvc. vol. 11, pp. 1–11 (2011) 4
6. Burns, D., Osfield, R.: Open scene graph a: Introduction, b: Examples and applications. In: Virtual Reality Conference, IEEE. pp. 265–265. IEEE Computer Society (2004) 5
7. Cernea, D.: Openmvs: Multi-view stereo reconstruction library. City **5**(7) (2020) 4
8. CesiumJS: CesiumJS Homepage (2024), https://cesium.com/ 5
9. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., Mello, S.D., Gallo, O., Guibas, L., Tremblay, J., Khamis, S., Karras, T., Wetzstein, G.: Efficient geometry-aware 3D generative adversarial networks. In: CVPR (2022) 7
10. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision. pp. 333–350. Springer (2022) 5
11. Chen, Z., Funkhouser, T., Hedman, P., Tagliasacchi, A.: Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16569–16578 (2023) 6
12. Clark, J.H.: Hierarchical geometric models for visible surface algorithms. Communications of the ACM **19**(10), 547–554 (1976) 3, 5
13. Daniels, J., Silva, C.T., Shepherd, J., Cohen, E.: Quadrilateral mesh simplification. ACM transactions on graphics (TOG) **27**(5), 1–9 (2008) 5
14. DJI: Terra (2024), https://enterprise.dji.com/dji-terra 4, 14
15. Duckworth, D., Hedman, P., Reiser, C., Zhizhin, P., Thibert, J.F., Lučić, M., Szeliski, R., Barron, J.T.: Smerf: Streamable memory efficient radiance fields for real-time large-scene exploration (2023) 5
16. Erikson, C., Manocha, D., Baxter III, W.V.: Hlods for faster display of large static and dynamic environments. In: Proceedings of the 2001 symposium on Interactive 3D graphics. pp. 111–120 (2001) 3, 5
17. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5501–5510 (2022) 5
18. Hirschmuller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Transactions on pattern analysis and machine intelligence **30**(2), 328–341 (2007) 4
19. Hoppe, H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. In: Proceedings Visualization'98 (Cat. No. 98CB36276). pp. 35–42. IEEE (1998) 5
20. Hoppe, H.: Progressive meshes. In: Seminal Graphics Papers: Pushing the Boundaries, Volume 2, pp. 111–120 (2023) 5
21. Hu, D., Zhang, Z., Hou, T., Liu, T., Fu, H., Gong, M.: Multiscale representation for real-time anti-aliasing neural rendering. arXiv preprint arXiv:2304.10075 (2023) 6

22. Hu, W., Wang, Y., Ma, L., Yang, B., Gao, L., Liu, X., Ma, Y.: Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19774–19783 (2023) 6, 7
23. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. vol. 7, p. 0 (2006) 4
24. Luebke, D.: Level of detail for 3D graphics. Morgan Kaufmann (2003) 5
25. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021) 1, 5, 7, 9
26. Moulon, P., Monasse, P., Perrot, R., Marlet, R.: Openmvg: Open multiple view geometry. In: Reproducible Research in Pattern Recognition: First International Workshop, RRPR 2016, Cancún, Mexico, December 4, 2016, Revised Selected Papers 1. pp. 60–74. Springer (2017) 4
27. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022) 5, 7, 11
28. Reiser, C., Szeliski, R., Verbin, D., Srinivasan, P., Mildenhall, B., Geiger, A., Barron, J., Hedman, P.: Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. ACM Transactions on Graphics (TOG) **42**(4), 1–12 (2023) 5
29. Ribelles, J., López, A., Belmonte, O.: An improved discrete level of detail model through an incremental representation. In: TPCG. pp. 59–66 (2010) 3
30. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4104–4113 (2016) 4, 14
31. Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (ECCV) (2016) 4
32. Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H.: Block-nerf: Scalable large scene neural view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8248–8258 (2022) 2, 4, 5
33. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., Kanazawa, A.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH 2023 Conference Proceedings. SIGGRAPH '23 (2023) 11, 12
34. Tanner, C.C., Migdal, C.J., Jones, M.T.: The clipmap: a virtual mipmap. In: Proceedings of the 25th annual conference on Computer graphics and interactive techniques. pp. 151–158 (1998) 1, 3
35. Tiled web map: Tiled web map— Wikipedia, The Free Encyclopedia (2024), https://en.wikipedia.org/wiki/Tiled_web_map 5
36. Turki, H., Ramanan, D., Satyanarayanan, M.: Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12922–12931 (2022) 2, 4, 5, 12
37. Turki, H., Zollhöfer, M., Richardt, C., Ramanan, D.: Pynerf: Pyramidal neural radiance fields. arXiv preprint arXiv:2312.00252 (2023) 6, 7, 11
38. Ulrich, T.: Rendering massive terrains using chunked level of detail control. In: Proc. ACM SIGGRAPH 2002 (2002) 5

39. Vu, H.H., Labatut, P., Pons, J.P., Keriven, R.: High accuracy and visibility-consistent dense multiview stereo. IEEE transactions on pattern analysis and machine intelligence **34**(5), 889–901 (2011) 4
40. Waechter, M., Moehrle, N., Goesele, M.: Let there be color! large-scale texturing of 3d reconstructions. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. pp. 836–850. Springer (2014) 4
41. Williams, L.: Pyramidal parametrics. In: Proceedings of the 10th annual conference on Computer graphics and interactive techniques. pp. 1–11 (1983) 5, 7
42. Wu, C.: Towards linear-time incremental structure from motion. In: 2013 International Conference on 3D Vision-3DV 2013. pp. 127–134. IEEE (2013) 4
43. Wu, X., Xu, J., Zhang, X., Bao, H., Huang, Q., Shen, Y., Tompkin, J., Xu, W.: Scanerf: Scalable bundle-adjusting neural radiance fields for large-scale scene rendering. ACM Transactions on Graphics (TOG) **42**(6), 1–18 (2023) 5
44. Xiangli, Y., Xu, L., Pan, X., Zhao, N., Rao, A., Theobalt, C., Dai, B., Lin, D.: Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In: European conference on computer vision. pp. 106–122. Springer (2022) 6
45. Xu, L., Xiangli, Y., Peng, S., Pan, X., Zhao, N., Theobalt, C., Dai, B., Lin, D.: Grid-guided neural radiance fields for large urban scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8296–8306 (2023) 5