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

Jiabin Liang*[1], Lanqing Zhang[1], Zhuoran Zhao[2], and Xiangyu Xu[3]

[1] Sea AI Lab, 1 Fusionopolis Place, #17-10, Galaxis, 138522, Singapore
{liangjb,zhanglq}@sea.com
[2] National University of Singapore, Singapore
zhuoran.zhao@u.nus.edu
[3] Xi'an Jiaotong University, 710049, Xi'an, China
xuxiangyu2014@gmail.com

1 Method

1.1 Tree Pruning

Proposed tree pruning is robust as the subtree pruning is only restricted to cases where there are no sparse points in a relatively large cube (approximately 2048 pixel on image) . Even in the rare case that a subtree is pruned by mistake, the affected volume can still be well reconstructed by its parent node, albeit in a coarser manner. This ensures that, while fine details may be omitted, the essential structure of the scene is preserved.

Because most of the sparse points lay on the surface of the object, which is a 2-manifold, so theoretically the octree will converge to a quadtree when $gridsize \to 0$.

1.2 Training

Pyramid Supervision Given the hierarchical nature of image pyramid, which has significantly different amount of pixel at each level, the pixel sampling strategy of conventional NeRF, which samples all images equally, is not suitable for our pyramid supervision. To account for the pixel imbalance across different pyramid levels, we perform uniform sampling in pixel domain instead, where a high-resolution image receives four times the sampling rate compared to its low-resolution counterpart. This strategy facilitates more balanced training across different levels of the image pyramid. An intriguing observation worth noting is that each level of the pyramid contains one-fourth the number of pixels as its upper level, mirroring the number of nodes in the quadtree. This correspondence ensures that the training data is approximately proportional to the parameters across different levels of the hierarchy.

* corresponding author

Furthermore, to encourage sharpness of the rendered results, we divide the GSD of each node by 2 during training, allowing the node to be supervised by images one level lower in the pyramid.

1.3 Complexity Analysis

In this analysis, the variable n denotes the total information encapsulated within the scene at a resolution desired by the user. This resolution, measured in meters per pixel, corresponds to the GSD introduced in paper. Therefore, for a simple 2.5D scene, n can be approximated by:

$$n = \frac{S}{\mathrm{GSD}^2}, \tag{1}$$

where S denotes the area of the scene. In real-world applications, the required resolution tends to vary significantly. This could be exemplified by the contrast between urban and rural areas, or the detailed features of a statue's face compared to its body. Therefore, GSD becomes a function of position $\mathbf{x}$, and the n in our 3D problem is defined as:

$$n = \iiint\limits_{\mathbf{x}\in V} \frac{dV}{\mathrm{GSD}(\mathbf{x})^3}, \tag{2}$$

where V denotes the volumetric representation of the scene.

Considering the total parameters of all the leaf nodes is $\mathcal{O}(n)$, we can assume this is a reasonable lower bound for space complexity to achieve user-defined resolution. The total parameters of the tree are, therefore, still $\mathcal{O}(n + \frac{n}{\lambda} + \frac{n}{\lambda^2} + ...) = \mathcal{O}(\frac{\lambda}{\lambda-1}n) = \mathcal{O}(n)$ where λ represents the average number of children of each node. λ is around 4, depending on the scene's sparseness and the tree pruning performance. The extra training data from the image pyramid is 33% of the original data and is still linear and also acceptable. For rendering, the total number of samples is the total number of pixels times the number of samples per ray. It's a constant. Each sample goes through $\mathcal{O}(\log(n))$ times recursively in the tree and only once in the neural network. The total time complexity is $\mathcal{O}(\log(n))$. Each step of recursion involves a few comparisons and one memory access, which are negligible compared to the computational load of the neural network.

The $\mathcal{O}(\log n)$ rendering space complexity is a crucial aspect in this study. During rendering, as illustrated in Fig. 1, the frustum can be partitioned into many zones exponentially in depth. Each zone comprises samples with a similar radius. Therefore, processed by nodes at the same level in the tree. As the size of each zone grows exponentially in both width and depth, the size of AABB in the corresponding level follows. Therefore, each zone will overlap with a constant number of nodes. On the other hand, let us consider the number of the zones. Without any assumption, the number of zones is theoretically infinite in both near and far directions. However, near-plane clipping, or the user's specified

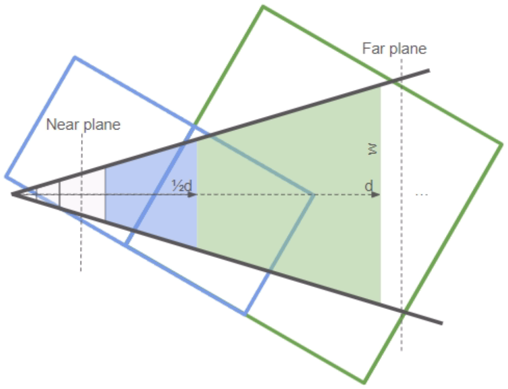

Fig. 1: The frustum is exponentially partitioned in depth into $\mathcal{O}(\log n)$ zones, with each zone overlapping with a constant number of nodes from the corresponding tree level. In the Figure, the green box represents the NeRF responsible for rendering the green zone, and the two blue boxes represent the NeRFs for rendering the blue zone. These nodes are required for rendering this single frame, resulting to an overall rendering space complexity of $\mathcal{O}(\log n)$.

minimum resolution (maximum depth of the tree), imposes constraints, and caps the number of zones in the near direction. For the far direction, when the scene extends infinitely, the number of zones will tend to infinity in an order of $\log(n)$. Combining this with the total number of zones, the overall space requirement in rendering this frustum is $\mathcal{O}(\log n)$. This underscores the algorithm's scalability and efficiency.

2 Experiment

2.1 Implementation Details

A single camera appearance embedding module proposed by [2] is shared by the entire tree. The images from the same pyramid with different resolutions share one appearance embedding feature. For training the appearance embedding, similar to MipNeRf [1] and Mega-NeRf [3], the left half of the test image is used in training. Evaluation is conducted solely on the right half of the image. The tree is limited to 4 levels, resulting a reasonable resolution of 5cm for a scene spanning 1 km. 5-level image pyramid is used in training, starting from the original resolution. Loss weights are set to $w_1 = 0.002, w_2 = 0.01, w_3 = 0.001$. The training employs the ADAM optimizer with a learning rate of 10^{-2}, consistent with Instant-NGP. Each batched is 65,536 rays. For sampling along the ray, a three-stage approach is adopted: initially, 192 sample points are chosen using disparity sampling. Subsequently, PDF sampling is applied in two additional rounds, sampling 96 points and 48 points.

2.2 Tree Pruning

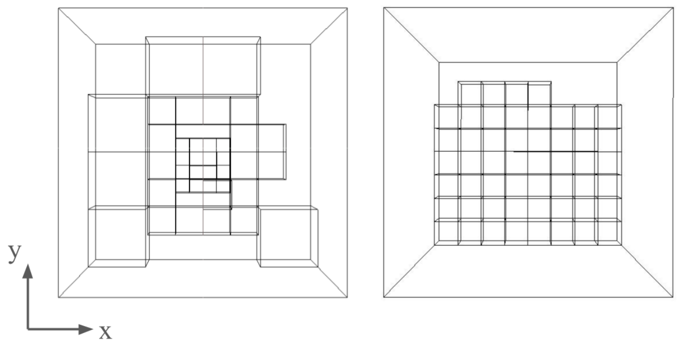

Fig. 2: We show only the scene's AABB and leaf AABBs for the Garden dataset on the left side and the Residential dataset (with depth limited to 3) on the right side. Camera looks down from the top. InfNeRF automatically creates smaller blocks to represent the details in the middle of the Garden scene. Meanwhile, it creates equal-sized blocks to represent the buildings near the ground in the Residential scene.

Without any assumptions and interventions, our tree pruning algorithm automatically allocates resources based on the scene. On the left side of Fig. 2, the leaf AABBs for a small scene "Garden" in Mipmap datasets [1] are depicted. In this scene, photos are taken closely around a table and a vase in the middle of a small garden. Without tree depth limit, InfNeRF automatically generate a tree with a depth of 4 to capture all details. 98.7% of the prefect octree's nodes are pruned. Only 62 nodes are left. The small blocks representing the table and vase are located at the center of the scene, along with larger blocks located at the surrounding area. The right side of Fig. 2 illustrates the leaf AABBs of the residential scene, In this scene, numerous low buildings are distributed across the ground. InfNeRF effectively divides the xy plane into blocks of equal size. 90% nodes are pruned, while only 59 nodes are create. Both cases show that, without any human intervention, the InfNeRF pruning algorithm demonstrates its ability to adapt to different types of scenes effectively.

2.3 Distributed Training

We conduct experiments on the Residence dataset. In order to balance the tree over 4 GPU, the scene is shift to center, for both distributed training and single-machine training. The maximum depth of the octree is 3, with 4 levels. When $L = 1$, only the first level is shared. 8 subtrees are divided into 4 groups and assigned to 4 GPU devices. The batch size is also divided by 4 for each GPU. The training data is split using the method described in paper.

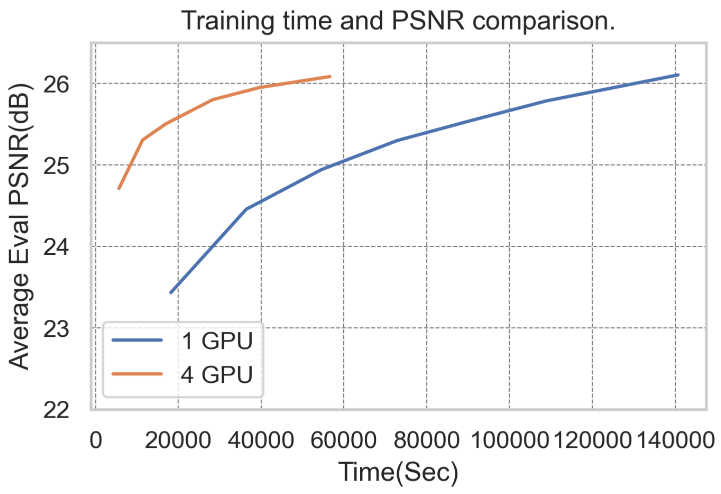

Fig. 3: Training speed and convergence of our distributed training algorithm. Only the top layer of the octree was shared.

The distributed training achieves a PSNR 26.0 and PSNR0 25.7, whereas single-machine training yields a PSNR 26.1 and PSNR0 25.6. Each device in distributed training owns only 25.49% of the total model, training speed is approximately 3 times faster than single GPU. The experiment shows that InfNeRF can effectively parallelize the training workload while essentially maintaining accuracy, which facilitates scalable performance and faster convergence.

References

1. 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) 3, 4
2. Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7210–7219 (2021) 3
3. 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) 3