Adaptive Sampling for Real-time Neural View Synthesis on the Web with Reinforcement Learning

OkHwan Bae and Chung-Pyo Hong*

Division of Computer Engineering, Hoseo University, Republic of Korea
E-mail: foem954@gmail.com; cphong@hoseo.edu
**Corresponding Author*

Received 26 September 2025; Accepted 11 November 2025

Abstract

The proliferation of immersive 3D web applications, from e-commerce product viewers to virtual real estate tours, has created a critical need for high-quality, real-time rendering directly within the browser. Neural radiance fields (NeRF) offer unprecedented photorealism but are hamstrung by immense computational demands, making their deployment on resource-constrained web platforms a significant web engineering challenge. The core bottleneck is NeRF's reliance on dense point sampling for volume rendering. This paper introduces a novel framework that directly tackles this challenge through a pioneering adaptive sampling technique powered by reinforcement learning.

We name this framework PPO-NeRF. It integrates the rapid training capabilities of Instant-NGP's hash encoding with an agent trained via proximal policy optimization (PPO). This agent learns to adaptively predict the minimal set of crucial sample points along each camera ray, dynamically pruning computationally redundant samples to optimize rendering specifically for web-based, real-time scenarios. Experimental results demonstrate that PPO-NeRF significantly lowers the barrier to web deployment. Compared to the original NeRF, it reduces training time by approximately 73.63%, enabling

Journal of Web Engineering, Vol. 25_2, 135–152.
doi: 10.13052/jwe1540-9589.2521

faster content iteration for web developers. More critically, our adaptive sampling slashes rendering time by approximately 44.7% and VRAM usage by approximately 29.9%, while maintaining comparable visual fidelity. These gains directly translate to faster load times, smoother user interaction, and broader device compatibility.

In conclusion, PPO-NeRF provides a practical solution to NeRF's long-standing performance bottlenecks, establishing a viable pathway for deploying high-fidelity, interactive 3D experiences at scale across the modern web.

Keywords: Neural radiance field, multi-resolution hash encoding, reinforcement learning, proximal policy optimization.

1 Introduction

Volume rendering [1] is one of the primary methods for directly visualizing data distributed in three-dimensional space and is widely utilized as a core technology in various fields such as medical imaging (CT, MRI), computer graphics, and virtual reality. In general, volume rendering techniques cast rays from a virtual camera into a data volume and sample points at regular intervals along these rays. The final pixel color and opacity of a 2D image are then calculated by compositing the density and color values obtained from each sample point along the ray's direction. This approach has the distinct advantage of being able to represent the internal structure of the data more accurately and in greater detail as the sampling points become denser. However, this improvement in quality comes at a great computational cost. As the number of sample points increases, the computational load and memory usage grow exponentially, posing a significant constraint for applications in real-time rendering environments or on resource-limited devices such as mobile phones. Consequently, extensive research has been conducted over the past few decades to improve computational efficiency while maintaining rendering quality [2].

Recently, with advancements in deep learning technology, the emergence of neural radiance fields (NeRF) [3] has brought a revolution to the field of 3D scene reconstruction. NeRF is a deep neural network that learns a continuous 3D representation of a scene from 2D images taken from multiple viewpoints, generating highly realistic, high-quality novel view images through volume

rendering. However, NeRF also relies on densely sampling hundreds of points per ray to achieve high-quality results, thereby inheriting the fundamental drawbacks of traditional volume rendering. Specifically, the neural network operations on this vast number of sample points create a severe bottleneck, which can cause model training to take several days. Even after training is complete, the rendering process of generating a new image can take tens of seconds to several minutes.

To address the slow learning speed of NeRF, research has proposed using multi-resolution hash encoding [4, 5]. This technique maps input coordinates to hash tables at multiple resolutions and uses the interpolated feature vectors as input to the neural network, significantly reducing model size and computational load, thereby drastically shortening training time to a matter of minutes. However, as this method focuses on improving training speed, the rendering process still requires the evaluation of a large number of sample points to ensure rendering quality. Consequently, the rendering speed remains proportional to the number of samples, showing limitations similar to the original NeRF. Therefore, this paper proposes a novel hybrid framework that combines multi-resolution hash encoding and reinforcement learning techniques, aiming to simultaneously reduce NeRF's training and rendering times in a single process. First, we adopt the multi-resolution hash encoding architecture to secure fast training speeds. Then, to address the chronic issue of slow rendering, we introduce an adaptive point sampling agent based on the proximal policy optimization (PPO) reinforcement learning algorithm [6]. This agent takes the camera ray corresponding to each pixel as input and learns a policy to predict the minimal set of sampling points essential for the final color composition. In other words, it dynamically identifies and skips unnecessary sampling in empty spaces or occluded regions of the scene, concentrating samples on information-dense areas like object surfaces. This effectively reduces the overall computational load required for rendering while minimizing the loss in rendering quality. Through the synergy of these two techniques, this study seeks to elevate the practicality of NeRF by simultaneously achieving both fast training and fast rendering.

The remainder of this paper is organized as follows: Chapter 2 introduces previous studies aimed at overcoming the limitations of NeRF. Chapter 3 explains our method for improving training and rendering speed using multi-resolution hash encoding and reinforcement learning. Chapter 4 describes the experimental process and results, and Chapter 5 presents the conclusion.

2 Related Work

2.1 Neural Radiance Fields

Neural radiance fields (NeRF) is a technology for synthesizing novel views of a 3D scene by representing it as a continuous 5D function. This 5D function is modeled by a multi-layer perceptron (MLP), a type of neural network, which takes a 3D location and a 2D viewing direction as input and outputs a volume density and an RGB color value for that point. The process of rendering a single image is as follows. First, a ray is cast from the camera origin through a pixel in the image [7]. Multiple points are then sampled along this ray's path. Next, the 5D information (position and direction) of each sampled point is fed into the MLP to obtain its density and color values. Finally, these values are integrated using classical volume rendering techniques to determine the final color of the pixel. Through this method, NeRF has succeeded in realistically capturing the intricate details of complex scenes, including view-dependent effects such as reflections. However, it suffers from a fundamental drawback: the training and rendering processes are extremely time-consuming due to the need to sample numerous points for each ray and perform individual MLP computations for each point.

2.2 Research Trends for Reducing Training Time

The slow training speed of NeRF has been a major limitation for its practical application, leading to the emergence of numerous follow-up studies aimed at overcoming this issue. Early research primarily focused on reducing the computational burden of the MLP or modifying the model's architecture.

Plenoxels [8] introduced an approach that directly optimizes a sparse voxel grid representing the scene, without using a neural network. At each vertex of a voxel, it stores a density value and the coefficients of a spherical harmonic function [9], which represents view-dependent color information. The density and color at a specific location can be efficiently computed through trilinear interpolation of the values stored at the eight corner vertices of the encompassing voxel. As a result, Plenoxels demonstrated training speeds hundreds of times faster than the original NeRF. However, this explicit grid structure has the limitation that memory requirements increase exponentially with higher resolution.

Instant-NGP proposed a technique called multi-resolution hash encoding. This method utilizes multiple virtual grids at different resolutions. For a given input coordinate, it identifies the corresponding voxel in each resolution level.

The vertices of this voxel are then passed through a hash function to retrieve unique feature vectors from a hash table. The feature vectors obtained from all levels are concatenated and fed into a small MLP. While this approach carries the possibility of hash collisions, the multi-resolution design enables the model to effectively learn fine details of the scene while minimizing the side effects of collisions. Instant-NGP thereby made it possible to train high-quality NeRF models in a very short amount of time. However, these methods are not free from the inherent problem that computational speed increases linearly as the number of sampled points increases, and they still require significant memory. Furthermore, the issue of rendering speed degradation proportional to the number of samples during rendering remained a limitation.

2.3 Research Trends for Improving Rendering Speed

Separate from reducing training time, achieving fast rendering speed is essential for web interactive applications such as real-time rendering. To this end, active research has been conducted on making the NeRF model structure more lightweight or making the sampling process more efficient [10].

FastNeRF [11] focused on dramatically increasing rendering speed by caching the rendering results of the neural network. This method separates the model into a position-dependent component and a view-dependent component. It pre-computes the position-dependent outputs and stores them in a 3D grid-based cache. Consequently, during rendering, instead of performing complex MLP computations, it can swiftly look up the cached values, enabling very fast rendering at over 200 frames per second (FPS). However, this technique does not improve the training time, and as the number of sampling points is increased to achieve high-quality results, the memory occupied by the cache also increases rapidly.

TermiNeRF [12], in an effort to minimize unnecessary computations during rendering, introduced a separate lightweight network to predict and sample only important regions. This sampling network first identifies a few key regions along each camera ray where the density is expected to be high. Subsequently, a color network, with a structure similar to the original NeRF, calculates the density and color only at these predicted sparse points to synthesize the pixel color. The sampling network is trained in a supervised manner, using the weights assigned to each sample by a pre-trained standard NeRF model as ground truth. Although this approach improved rendering speed by about $10\times$, it introduced the inconvenience of a two-stage training

process: a standard NeRF must first be trained, and then the sampling network must be additionally trained based on its outputs. As a result, the total training time could be similar to or even longer than that of the original NeRF.

2.4 Research Trends in 3D Reconstruction Based on Reinforcement Learning

In the field of 3D reconstruction, alongside rendering-based approaches like NeRF, significant research has been conducted on methods that directly generate geometry using reinforcement learning. A notable study, "Modeling 3D Shapes by Reinforcement Learning" [13], proposed a novel framework where two cooperative agents incrementally construct a 3D mesh.

The core of this methodology is the division of labor between two distinct agents: a prim-agent and a mesh-agent. Initially, the prim-agent establishes the coarse structure of the shape by placing primitive meshes. Subsequently, the mesh-agent refines the geometry by meticulously adjusting the vertices of these primitives to better match the target shape. The research team evaluated this framework by applying various RL algorithms, such as DDQN [14], DAgger [15], and DQfD [16], and conducted a comparative performance analysis.

However, this RL-based methodology exhibited several distinct limitations. First, the training process suffered from inefficiency because the prim-agent and mesh-agent could not be trained simultaneously within a single, unified pipeline. Instead, they required sequential training, which hindered overall computational efficiency. Second, in contrast to NeRF, which can render high-fidelity, photorealistic results, this model struggled to represent highly intricate or thin structures. Lastly, the final output was restricted to a polygon mesh, devoid of appearance attributes such as color or material information, thus failing to generate properties beyond pure geometry.

Therefore, this paper aims to propose a novel methodology that integrates core ideas from previous research on improving learning and rendering speed. The proposed technique overcomes the limitations of existing methods by simultaneously accelerating learning and rendering as a single process while reducing memory usage.

3 Proposed Scheme

In this paper, we propose a novel end-to-end learning framework that combines reinforcement learning-based adaptive sampling with the fast

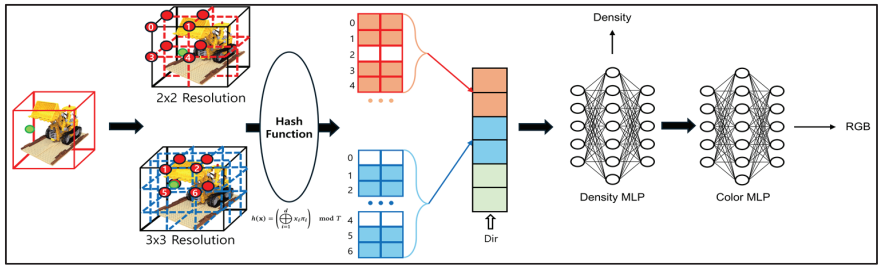

Figure 1 Hash based training optimization.

learning capabilities of multi-resolution hash encoding to overcome the tradeoff between rendering quality and computational efficiency in existing NeRFs. The proposed model consists of three core components. First, a policy network based on the proximal policy optimization (PPO) algorithm dynamically explores and focuses sampling on essential regions required for rendering along each ray. Second, a multi-resolution hash encoding-based NeRF model quickly and accurately infers scene color and density using only a few sampling points selected by a reinforcement learning agent. Finally, rather than training these two components separately, we propose an end-to-end learning process that integrates them to simultaneously optimize sampling efficiency and rendering accuracy.

3.1 Fast Training with Multi-resolution Hash Encoding

The sparse sampling points selected by the reinforcement learning agent are utilized as input for the NeRF model, which represents the 3D structure and appearance of the scene. In this study, instead of a large-scale MLP with numerous parameters, we adopted a multi-resolution hash encoding method as the core architecture of the NeRF model. This approach achieves high expressive power with few parameters and demonstrates remarkable training speed.

This method begins by mapping the 3D sampling points provided by the agent to multiple virtual feature grids, each with a different resolution, as shown in Figure 1. At each resolution level, the indices of the vertices of the voxel containing the point are calculated. Rather than being used directly as memory addresses, these indices are converted into keys for a hash table via a hash function that allows for spatial collisions. The hash table stores unique, learnable feature vectors as its values, and the converted keys are used to quickly look up the corresponding feature vectors for each vertex.

Subsequently, trilinear interpolation is applied to the retrieved feature vectors at each resolution level to compute a final feature vector for the current point, and the feature vectors from all resolution levels are concatenated. This final encoded feature vector, rich in local scene information, is then passed to a very small MLP to predict the volume density and view-dependent color at that point. The final pixel color is calculated by integrating the predicted density and color values along the ray, following the classic volume rendering equation.

3.2 Adaptive Ray Sampling with Reinforcement Learning

Traditional NeRF models use uniform sampling at all points along a ray or employ a hierarchical sampling method with coarse and fine networks. While these approaches are intuitive, they have an inherent inefficiency, consuming the same computational cost in empty spaces or in regions that contribute little to the final image. To improve this, our work introduces a reinforcement learning agent that understands the characteristics of each ray and dynamically determines the sampling distribution based on its importance, as shown in Figure 2. For this purpose, we train a sampling policy network using proximal policy optimization (PPO), a prominent policy gradient algorithm.

The environment for the agent to learn an optimal policy is defined as follows. The state consists of the information for a single ray under consideration: its origin and direction. To overcome the inherent bias of neural networks toward learning low-frequency functions from low-dimensional coordinate inputs, this state vector is passed through positional encoding via Fourier feature mapping.

Notably, to account for the distinct properties of positional information, which encodes fine geometric detail, and directional information, which represents view-dependent color variations, different encoding levels (L) are

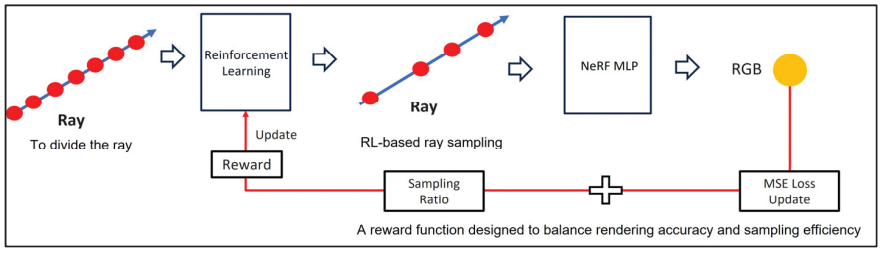

Figure 2 Adaptive sampling based on reinforcement learning.

applied differentially. This process amplifies subtle variations in the coordinate values, transforming them into a high-dimensional feature vector. This allows the network to leverage richer information about the ray's position and direction.

The agent's action space is defined by determining the locations to sample along the ray. A given ray is divided into 192 uniform intervals within its viewable range. For each interval, the agent makes a binary decision: whether to sample it or to skip it. Through these sequential decisions, the agent can execute an adaptive sampling strategy, concentrating computation only on important intervals where surfaces are likely to exist or complex translucent effects occur, while efficiently bypassing unnecessary regions.

The reward function, which guides the training of the policy network, is carefully designed to simultaneously achieve two objectives: rendering accuracy and sampling efficiency. The final calculated reward ***R*** is given by Equation (1).

$$
\begin{aligned}
R_v &= \alpha \cdot round(-MSE(C_{gt}, \hat{C}), 3) \\
S_{ratio} &= \frac{1}{N} \sum_{i=1}^{N} m_i \\
R &= R_v + (\beta \cdot R_v \cdot S_{ratio})
\end{aligned}
\tag{1}
$$

The final reward function ***R*** for the reinforcement learning agent proposed herein is formulated to optimally mediate the trade-off between the visual fidelity of the rendered output and computational expense. This design concurrently addresses two principal objectives: high rendering accuracy and efficient sampling.

The rendering accuracy reward, R_v, is predicated on the mean squared error (MSE) between the ground truth and the agent-rendered pixel values. To transform the loss minimization objective into a reward maximization problem, the MSE is negated, rounded to the third decimal place, and subsequently multiplied by a hyperparameter, α, which modulates the significance of rendering fidelity.

In parallel, the sampling efficiency reward is quantified by the proportion of utilized sample intervals, determined via an indicator variable, m_i, which signifies the selection of the *i*th candidate from a total of 192 samples.

Ultimately, these two reward components are unified into the final reward function, ***R***, through a hyperparameter, β, which serves as a weighting factor for sampling efficiency. This reward architecture compels the agent to learn

Table 1 Hyperparameter configuration for the experiment

Parameter	Symbol	Value
Position encoding dimension/level	L	10
Direction encoding dimension/level	L	4
Rendering fidelity weight	α	1000
Sampling efficiency weight	β	10

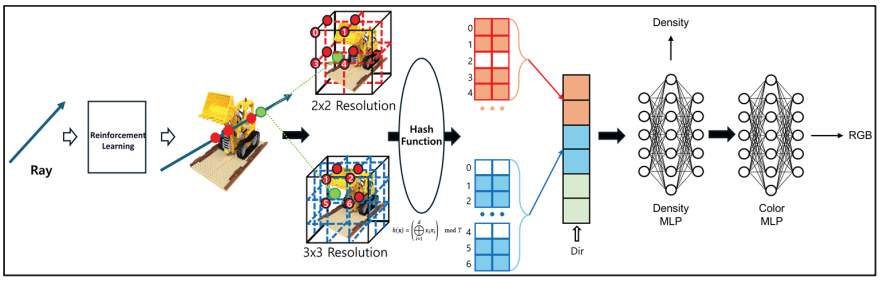

Figure 3 The proposed end-to-end learning pipeline.

a policy that not only minimizes rendering error but also generates results of equivalent or superior quality with a minimal sample count, thereby avoiding perceptual degradation. Consequently, the agent is guided to ascertain an optimal equilibrium between rendering accuracy and computational overhead.

In this reinforcement learning model, the processing of a single ray can be considered a complete episode. This marks a fundamental departure from conventional reinforcement learning (PPO) environments, where a rollout buffer is populated through sequential interactions until an episode terminates. Instead, the structure is analogous to the supervised learning paradigm of NeRF, wherein thousands of independent episodes (rays) are batched together and executed in parallel. As each episode concludes instantaneously with a corresponding reward upon making its sampling decisions, the model can rapidly accumulate the vast amount of experience data required for PPO training within a single training step. This efficiency eliminates the need to gather experience over extended periods, a common requirement in traditional reinforcement learning scenarios.

3.3 End-to-end Learning Framework

As shown in Figure 3, the most significant feature of the proposed method is that the policy network for adaptive sampling and the NeRF model for scene

representation do not operate independently; instead, they interact organically and are optimized together within a single, unified framework. This end-to-end learning approach maximizes the synergy between the two models.

The overall training process proceeds as follows. First, when ray information corresponding to an image pixel is input to the PPO agent, the agent samples points it deems most critical for rendering according to its current policy. These selected points are immediately passed to the multi-resolution hash based model, which predicts the density and color at each point, and the final pixel color is synthesized via volume rendering. As soon as the mean squared error (MSE) between the rendered color and the ground-truth color is calculated, this loss value is propagated through two pathways. The first pathway uses backpropagation to directly update the trainable parameters of the multi-resolution hash based model, that is, the feature vectors stored in the multi-resolution hash tables and the weights of the small MLP. This forces the NeRF model to reconstruct the scene more accurately from the given sampling points. In the second pathway, this MSE value is combined with the sampling efficiency metric to generate a reward signal for the PPO agent. Based on this reward, the agent updates its policy, exploring more sophisticated and efficient sampling strategies in the next iteration that can further improve the performance of the current NeRF model. In conclusion, this end-to-end learning architecture creates a powerful feedback loop between the sampling agent and the NeRF model. The sampling agent progressively refines its sampling strategy to maximize the rendering performance of the NeRF model, while the NeRF model becomes specialized for the efficient sampling points provided by the agent, enabling it to learn faster and more accurately. Through this complementary process, the entire system converges toward generating the highest quality rendering results with minimal computation.

4 Evaluation

In this section, we present the experimental results comparing the proposed NeRF with the original NeRF. The experiments were conducted in two parts. The first part evaluates efficiency in terms of training time, rendering time, and memory usage. The second part assesses rendering quality using various metrics: PSNR, SSIM, and LPIPS. For a detailed comparison, we provide a visual comparison of rendered images from our proposed method, the baseline method, and the ground truth to highlight the visual differences.

The experiments were performed on a single Nvidia L40s GPU. This ensures that the comparison between the proposed and original methods is consistent and fair, conducted under identical hardware conditions.

4.1 Training and Rendering Efficiency

One of the primary goals of the proposed model is to reduce the substantial computational cost of NeRF. To validate this, we compared the total training time, the rendering time required to render a single image, and the peak GPU VRAM usage against the baseline model.

As shown in Table 2, the proposed model demonstrates improved performance over the existing NeRF model across all efficiency metrics. As shown in Figure 4, The total training time was significantly reduced from 18.2 hours to 4.8 hours, achieving a speed improvement of approximately 73.63%. Similarly, as shown in Figure 5, the rendering speed improved by 44.74%, from 11.4 to 6.3 seconds per image, enhancing the potential for real-time rendering.

This is attributed to the reinforcement learning-based adaptive sampling, which effectively eliminates unnecessary computations in empty spaces, and the hash encoding structure of Instant-NGP, which enables faster convergence. Furthermore, as shown in Figure 6, VRAM usage was reduced by approximately 29.94%, making it feasible to operate the model in more constrained hardware environments.

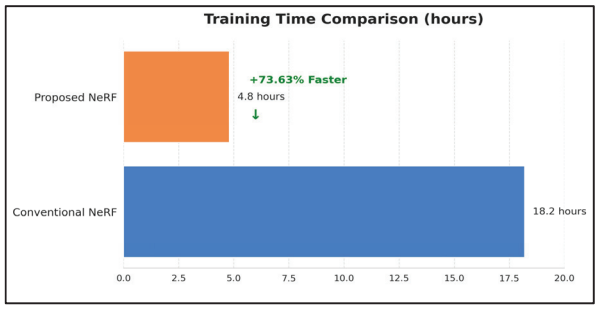

Figure 4 Training time comparison of PPO-NeRF and conventional NeRF.

Table 2 Comparison between the proposed NeRF model and the conventional model

	Training (hours)	Rendering (seconds)	VRAM (MiB)
Proposed NeRF	4.8	6.3	31,714 MiB
Conventional NeRF	18.2	11.4	45,266 MiB

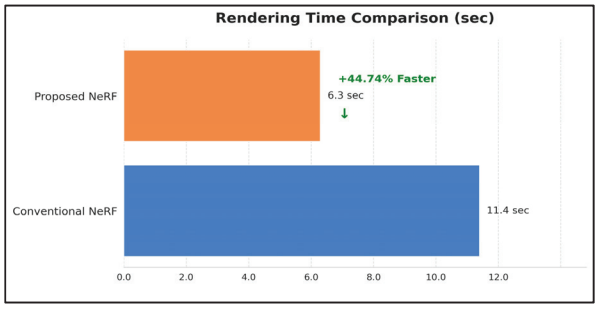

Figure 5 Rendering time comparison of PPO-NeRF and conventional NeRF.

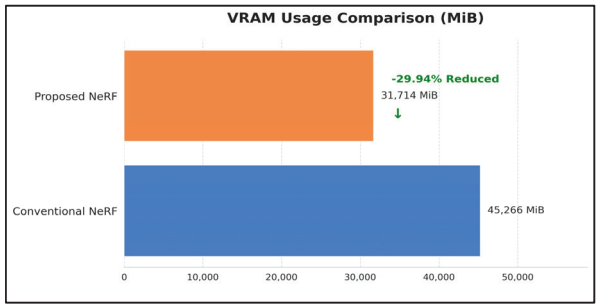

Figure 6 Usage VRAM comparison of PPO-NeRF and conventional NeRF.

Table 3 Comparison of rendering quality between the proposed NeRF model and conventional model

	PSNR	SSIM	LPIPS
Proposed NeRF	32.3664	0.9654	0.0668
Conventional NeRF	30.5785	0.9556	0.0750

4.2 Rendering Quality Comparison

The results in Table 3 and Figure 7 show that the proposed model not only enhances computational efficiency but also improves rendering quality. PSNR significantly increased from 30.5785 to 32.3664, and SSIM rose from 0.9556 to 0.9654. The LPIPS metric, which most closely aligns with human perceptual similarity, also decreased from 0.0750 to 0.0668, confirming an improvement in visual quality. This is because the reinforcement learning agent was trained to concentrate sampling on important areas, such as scene surfaces and regions with complex details, rather than sampling randomly. Consequently, the model was able to reconstruct the scene more faithfully and accurately with fewer points.

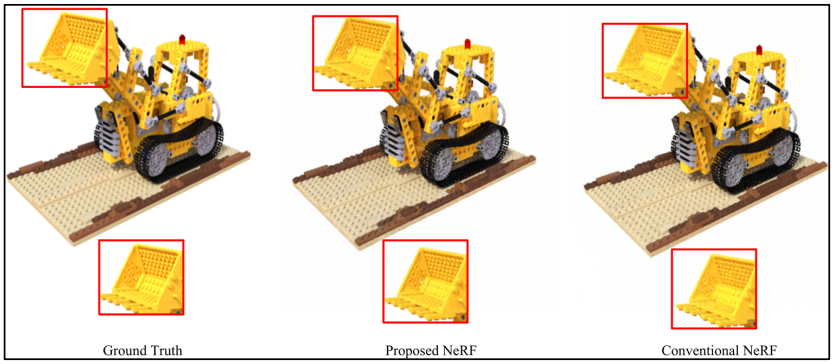

Figure 7 Comparison of rendering quality.

In conclusion, the proposed reinforcement learning-based NeRF model dramatically improves upon the long-standing issues of the original model: slow training and rendering speeds and high memory consumption. More notably, this enhancement in efficiency overcomes the typical trade-off that often involves a loss in quality. Instead, our model achieves superior performance across all quantitative quality metrics. This result demonstrates the success of our proposed end-to-end adaptive sampling framework.

5 Conclusion

Neural radiance fields (NeRF) is a key technology for delivering high-quality 3D content, but it has been difficult to practically apply in web environments due to severe performance bottlenecks from its immense computational load. NeRF suffers from a chronic trade-off between speed and quality, as it must densely sample numerous points along camera rays during the volume rendering process. While subsequent research like Instant-NGP improved training speed, rendering speed has remained an obstacle for real-time web services.

To address this issue, this paper proposes a hybrid approach that simultaneously optimizes the learning and rendering steps in a single process. We adopted Instant-NGP's hash encoding to accelerate training and introduced a novel adaptive point sampling technique based on the proximal policy optimization (PPO) reinforcement learning algorithm to improve rendering speed. This technique trains a reinforcement learning agent to dynamically predict only the minimum number of point locations necessary for rendering

each pixel, thereby eliminating unnecessary computations and maximizing rendering efficiency.

Experimental results show that the proposed method reduces training time by approximately 73.63%, rendering time for 200 images by approximately 44.7%, and VRAM usage by approximately 29.94%, all while maintaining a rendering quality comparable to the original NeRF. In conclusion, the combination of reinforcement learning-based sampling and hash encoding effectively resolves NeRF's long-standing speed issues, providing a crucial technical foundation for the implementation of real-time, high-quality 3D content on the web.

Acknowledgement

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP)-Innovative Human Resource Development for Local Intellectualization program grant funded by the Korea government (MSIT) (IITP-2025-RS-2024-00436765).

References

[1] Brebin, R. A., et al. "Volume Rendering." Seminal Graphics: Pioneering Efforts that Shaped the Field (1998): 363–372.

[2] Tewari, Ashish, et al. "Advances in Neural Rendering." Computer Graphics Forum 41.2 (2022): 703–735.

[3] Mildenhall, Ben, et al. "Nerf: Representing Scenes as Neural Radiance Fields for View Synthesis." Communications of the ACM 65.1 (2021): 99–106.

[4] Müller, Thomas, et al. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding." ACM Transactions on Graphics (TOG) 41.4 (2022): 1–15.

[5] Bae, O., and C. P. Hong. "An Effective Scheme to Accelerate NeRF for Web Applications Using Hash-based Caching and Precomputed Features." Journal of Web Engineering 23.7 (2024): 1041–1056.

[6] Schulman, John, et al. "Proximal Policy Optimization Algorithms." arXiv preprint arXiv:1707.06347 (2017).

[7] Tsai, R. "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses". IEEE Journal on Robotics and Automation, 3(4), (2003): 323–344.

[8] Fridovich-Keil, Sara, et al. “Plenoxels: Radiance Fields Without Neural Networks.” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022): 5501–5510.

[9] Sloan, Peter-Pike, et al. “Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments.” Seminal Graphics Papers: Pushing the Boundaries, Volume 2 (2023): 339–348.

[10] Yu, Alex, et al. “Plenoctrees for Real-Time Rendering of Neural Radiance Fields.” Proceedings of the IEEE/CVF International Conference on Computer Vision (2021): 5752–5761.

[11] Garbin, Stephan J., et al. “FastNeRF: High-Fidelity Neural Rendering at 200fps.” Proceedings of the IEEE/CVF International Conference on Computer Vision (2021): 14346–14355.

[12] Piala, Matthew, and Roman Clark. “TermiNeRF: Ray Termination Prediction for Efficient Neural Rendering.” 2021 International Conference on 3D Vision (3DV) (2021): 1106–1114.

[13] Lin, Cheng, et al. “Modeling 3D Shapes by Reinforcement Learning.” European Conference on Computer Vision (2020): 545–561.

[14] Van Hasselt, Hado, et al. “Deep Reinforcement Learning with Double Q-Learning.” Proceedings of the AAAI Conference on Artificial Intelligence 30.1 (2016).

[15] Ross, Stéphane, et al. “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning.” Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (2011): 627–635.

[16] Hester, Todd, et al. “Deep Q-Learning from Demonstrations.” Proceedings of the AAAI Conference on Artificial Intelligence 32.1 (2018).

Biographies

OkHwan Bae received his master's degree in Computer Engineering from Hoseo University in 2025. His research interests include computer vision, deep learning, and reinforcement learning.

Chung-Pyo Hong received his B.Sc. and M.Sc. degrees in Computer Science from Yonsei University, Seoul, Korea, in 2004 and 2006, respectively. In 2012, he received his Ph.D. degree in Computer Science from Yonsei University, Seoul, Korea. He is currently an associate professor of Computer Engineering at Hoseo University, Asan, Korea. His research interests include machine learning, explainable AI, and data science.