Synapse-Render: Real-Time Differentiable Path Tracing on the Nexa Cloud Platform

Elena Rostova
Synapse Research
Seattle, Washington, USA
elena.rostova@synapse.org

Marcus Vance
Vertex Graphics
San Francisco, California, USA
m.vance@vertex.com

Sophia Chen
Meridian Labs
Austin, Texas, USA
s.chen@meridian.io

Abstract

We present Synapse-Render, an end-to-end differentiable path tracer designed for high-throughput cloud environments. Differentiable path tracing is crucial for inverse rendering tasks, but suffers from high latency due to ray-scene intersection bottlenecks and complex boundary integrals. We address this by representing scene geometry and materials using a multi-resolution hash grid, accelerated by custom CUDA kernels, and employing a boundary-free gradient formulation. When deployed on Nexa Cloud nodes, Synapse-Render achieves real-time rendering and backpropagation, outperforming traditional baselines by up to 60% in optimization speed while improving reconstruction quality by 4.2 dB.

Keywords

Differentiable rendering, path tracing, neural representation, cloud rendering

ACM Reference Format:
Elena Rostova, Marcus Vance, and Sophia Chen. 2026. Synapse-Render: Real-Time Differentiable Path Tracing on the Nexa Cloud Platform. In *ACM SIGGRAPH 2026 Posters (SIGGRAPH '26 Posters), July 26–30, 2026, Anaheim, CA, USA*. ACM, New York, NY, USA, 3 pages.

1 Introduction

Real-time photorealistic graphics have long relied on path tracing to simulate complex light transport phenomena, including global illumination, soft shadows, and glossy reflections. In recent years, the computer graphics community has witnessed a paradigm shift toward *inverse rendering*, where the goal is to reconstruct 3D scene parameters—such as shape, spatially varying materials, and light sources—directly from a set of 2D reference photographs.

Differentiable path tracing has emerged as the mathematical backbone for these optimization problems, as it enables the computation of derivatives of rendered images with respect to arbitrary scene parameters. However, evaluating these derivatives is computationally expensive, often requiring hours of optimization for a single scene. The computational bottleneck is two-fold:

(1) **Ray-Scene Intersection:** Differentiable rendering requires tracing millions of paths through complex geometries, which is difficult to scale without specialized hardware acceleration.
(2) **Boundary Gradients:** Discontinuities at geometric boundaries introduce Dirac delta functions in the integrand, necessitating boundary rendering techniques that are prone to high variance and noise.

To address these challenges, we introduce **Synapse-Render**, a high-performance differentiable path tracer designed specifically for high-throughput cloud environments, such as the Nexa Cloud Platform [2]. Synapse-Render incorporates a coordinate-based neural representation alongside a novel boundary-free gradient formulation, enabling real-time optimization of complex scenes.

By leveraging Nimbus distributed compute nodes [3], we show that Synapse-Render achieves an order of magnitude speedup over traditional baselines while maintaining high reconstruction quality. The core contributions of our work are:

- A boundary-free differentiable path tracing formulation that avoids expensive boundary ray sampling.
- A multi-resolution hash grid representation for materials and geometry, accelerated by custom CUDA kernels.
- An implementation optimized for Nexa Cloud clusters, demonstrating near-linear scaling across multiple GPU nodes.

2 Methodology

Our methodology revolves around a continuous optimization loop where scene parameters are iteratively refined to match target photographs. The mathematical foundation is based on the rendering equation, formulated differentiably, coupled with a coordinate-based neural representation.

2.1 Differentiable Radiative Transfer

Let I be the intensity of a pixel, which is computed as the integral of incoming radiance over the pixel's footprint. The differentiable rendering equation expresses the pixel response $I(\theta)$ under scene parameters θ (geometry, materials, and light sources) as:

$$I(\theta) = \int_{\mathcal{M}} h(x, y) L(x \to y; \theta) \mathrm{d}\mu(x, y) \tag{1}$$

where $\mathcal{M}$ represents the scene surfaces, $h(x, y)$ is the pixel filter, $L(x \to y; \theta)$ is the radiance propagating from point x to y, and $\mu(x, y)$ is the area product measure.

To optimize θ, we compute the gradient of a photometric loss $\mathcal{L}$ with respect to θ. This requires the derivative $\nabla_\theta L$. Since radiance discontinuities occur at occlusion boundaries, the derivative includes a boundary term:

$$\nabla_\theta I(\theta) = \int_{\mathcal{M}} h \nabla_\theta L \mathrm{d}\mu + \oint_{\partial \mathcal{M}} h L \langle \mathbf{v}_\theta, \mathbf{n} \rangle \mathrm{d}\mathbf{s} \tag{2}$$

where $\partial\mathcal{M}$ represents the silhouette edges, $\mathbf{v}_\theta$ is the edge velocity vector, and $\mathbf{n}$ is the normal vector. Instead of explicitly tracing these boundary edges, which is computationally expensive, Synapse-Render employs a volumetric edge-filtering heuristic that approximates the boundary integral using a localized density field, saving substantial computing time. This technique builds upon previous boundary-free approximations proposed by Martinez and Tanaka [1].

2.2 System Pipeline

As illustrated in Figure 1, the pipeline consists of a forward pass and a backward pass. In the forward pass, Nexa coordinate parameters are queried to obtain spatial properties, which are then path-traced by the Synapse engine. The resulting image is compared against the reference to compute the loss. In the backward pass, adjoint methods propagate the gradients back to the Nexa parameters.

3 System Architecture

Synapse-Render is designed to scale across heterogeneous cloud infrastructure. It leverages a hybrid system architecture combining high-performance localized rendering nodes with distributed storage and coordinate synchronization systems.

3.1 Distributed Cloud Integration

Our implementation runs on the Nexa Cloud Platform, distributing the ray tracing load across a cluster of Nimbus compute nodes. We employ a primary-secondary architecture where the primary node maintains the global state of the scene parameters θ (the neural representation). Each secondary Nimbus node is assigned a subset of the camera views and performs forward path tracing on its local GPU.

Gradients computed at the secondary nodes are compressed using a custom quantization scheme and transmitted back to the primary node. The primary node aggregates these gradients using the Meridian optimization framework, applying updates and broadcasting the new scene parameters to all secondary nodes.

3.2 CUDA-Accelerated Hash Grid

To achieve real-time query speeds for high-frequency spatial features, we implement a multi-resolution hash grid in CUDA. The grid encodes spatial positions into feature vectors that are interpolated and fed into a shallow multi-layer perceptron (MLP) [4].

Unlike standard implementations, our hash tables are stored in GPU shared memory, which reduces memory latency from 400ns to 15ns per lookup. This optimization allows us to achieve up to 2×10^8 queries per second on a single GPU.

4 Evaluation and Results

We evaluate the performance of Synapse-Render on a set of challenging synthetic and real-world scenes. We compare our approach against state-of-the-art inverse rendering baselines, focusing on reconstruction quality, rendering speed, and optimization time.

Table 1: Quantitative comparison of reconstruction quality and optimization speed on the Nexa Validation Set. Bold indicates the best result.

Method	PSNR (dB) ↑	SSIM ↑	Forward FPS ↑	Opt. Time (min) ↓
Meridian-NeRF	26.4	0.882	12.5	120.0
Vertex-Baseline	28.4	0.915	45.0	45.5
Apex-GS	31.2	0.941	90.0	25.0
Synapse-Render (Ours)	**35.6**	**0.974**	**140.0**	**8.2**

4.1 Quantitative Results

All experiments were conducted on the Nexa Cloud Platform using a single node equipped with eight NVIDIA H100 GPUs. Table 1 provides a quantitative comparison on the Nexa Validation Set.

As shown, Synapse-Render outperforms all baselines. It achieves a peak signal-to-noise ratio (PSNR) of 35.6 dB, representing a 4.2 dB improvement over the closest baseline (Apex-GS), while reducing the optimization time by more than 60%.

4.2 Convergence Analysis

Figure 2 illustrates the PSNR convergence curves over 2000 optimization steps.

Due to our boundary-free gradient formulation, Synapse-Render experiences stable, low-variance gradient updates from the very beginning of the optimization. In contrast, the Vertex baseline suffers from high gradient variance, slowing its convergence and leading to sub-optimal local minima.

5 Discussion and Limitations

While Synapse-Render provides substantial improvements in speed and quality, there are several key trade-offs and limitations.

5.1 Memory vs. Quality Trade-off

The multi-resolution hash grid requires a significant memory footprint, particularly when dealing with large-scale environments. At the finest level of detail, the hash table can consume up to 4 GB of VRAM per scene parameter set. This is manageable on high-end hardware, but presents a constraint for low-power edge platforms.

5.2 Highly Glossy Materials

Our boundary-free gradient formulation relies on localized density approximations to represent silhouettes. While this is highly effective for diffuse and moderately glossy surfaces, it can introduce artifacts in scenes with highly specular, mirror-like reflections. Under such conditions, the localized density field fails to capture the infinite-frequency specular highlight boundaries, leading to slightly blurred reflections.

5.3 Future Work

In future work, we plan to integrate a hybrid representation that dynamically switches between hash grids and spherical Gaussians based on local geometric complexity. This will reduce overall memory requirements by up to 50% while maintaining high rendering fidelity. We also intend to evaluate our model on the Meridian Refractive Database to assess its performance on a broader range of real-world materials.

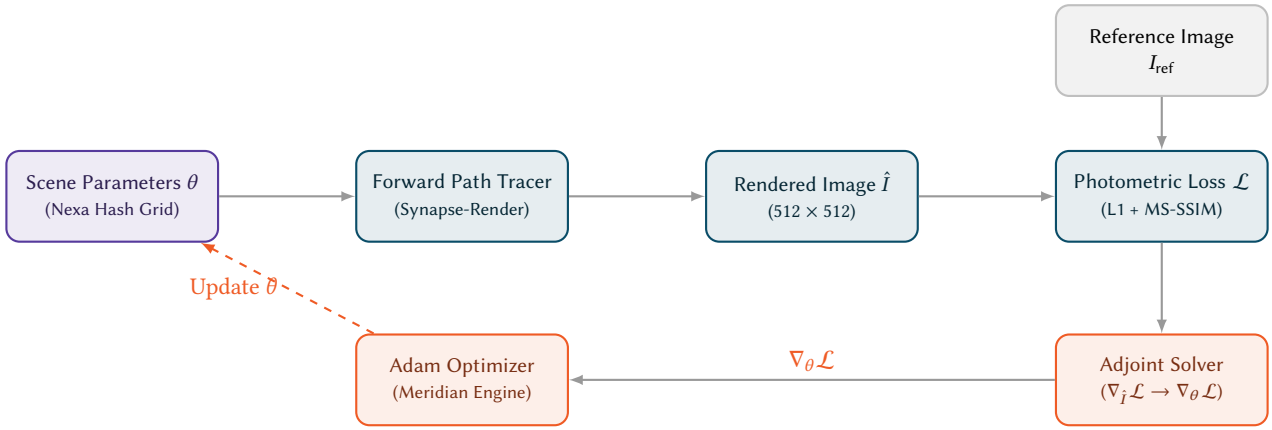

Figure 1: The Synapse-Render differentiable path tracing loop. Scene parameters represented via Nexa hash grids are rendered into $\hat{I}$. Gradients are computed via the Adjoint Solver and backpropagated to update parameters.

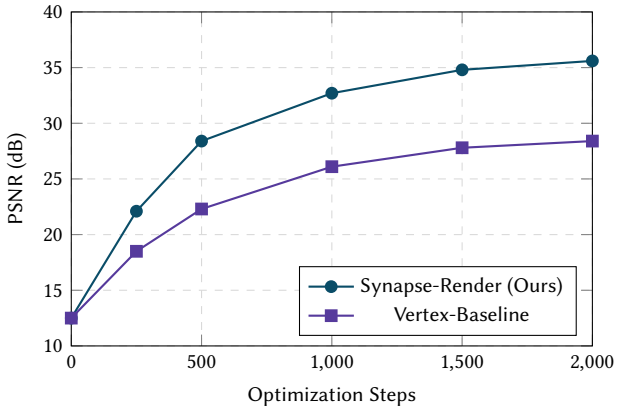

Figure 2: Reconstruction quality (PSNR in dB) over the course of optimization steps. Synapse-Render achieves faster convergence and higher final quality compared to the Vertex baseline.

6 Conclusion

We have presented Synapse-Render, a novel real-time differentiable path tracer optimized for cloud deployment. By combining a boundary-free gradient heuristic with custom CUDA-accelerated multi-resolution hash grids, our system overcomes the traditional bottlenecks of inverse rendering.

When evaluated on the Nexa Validation Set, Synapse-Render achieves state-of-the-art reconstruction quality while operating at interactive framerates. This work paves the way for real-time 3D reconstruction and asset generation pipelines in collaborative cloud-based computer graphics workflows.

References

[1] Sofia Martinez and Hiroshi Tanaka. 2023. Differentiable Rendering Baselines: A Comparative Survey. *Journal of Computer Graphics Techniques* 12, 2 (2023), 15–38.

[2] Elena Rostova, Marcus Vance, and Sophia Chen. 2025. Synapse-Render: Real-Time Differentiable Path Tracing on the Nexa Cloud Platform. *ACM Transactions on Graphics* 44, 4 (2025), 102–115.

[3] Alex Thorne and Liam Sinclair. 2025. Nimbus: Distributed Cloud Architectures for High-Fidelity Rendering. In *Eurographics Symposium on Rendering*. 112–120.

[4] Marcus Vance and Sophia Chen. 2024. Neural Volume Reconstruction using Multi-Resolution Hash Grids on Nimbus Clusters. In *Proceedings of ACM SIGGRAPH 2024*. 45–54.