Integration of Neural Radiance Fields for Volumetric Rendering in an Optix Path Tracer

Bryan Fallin

***Abstract*—This paper presents the integration of Neural Radiance Fields (NeRF) with an Optix-based path tracer to facilitate volumetric rendering of complex scenes. The system dynamically alternates between radiance and volume rays within volumetric data, exemplified by a backyard fountain scenario. The path tracer, enhanced with Axis-Aligned Bounding Boxes (AABB) and MISS shaders, manages ray segments for improved volume integration. This approach demonstrates the potential of merging traditional ray tracing techniques with machine learning to enhance visual realism and computational efficiency in 3D rendering.**

I. Introduction

Achieving photorealism in computer graphics is a complex and resource-intensive endeavor. Traditional industry pipelines often require a team of specialized artists, each responsible for different aspects of the production process. A modeling artist creates detailed meshes that accurately represent real-world geometry, leaving finer details to texture artists. These artists are tasked with producing all necessary textures using both procedural techniques and tailored photographic methods. This includes creating albedo textures, normal maps, and displacement textures that deform the mesh. Depending on the tools used, these textures might need to be baked into image files for use in the final rendering engine. Furthermore, dynamic objects within a scene require additional expertise in rigging and animation. This labor-intensive process can be prohibitively expensive, particularly for startup media firms.

In contrast, modern AI-driven approaches offer a cost-effective solution. Specifically, Neural Radiance Fields (NeRF) allow for the representation of fully photorealistic scenes with just a camera and an operator, potentially reducing the time to produce production-ready assets to just a few hours. Photo-realistic rendering is considered the 'holy grail' of the computer graphics domain, with substantial research dedicated to enhancing efficiency and quality. This research is supported by various industries including gaming, film, and scientific visualization. NeRF, one of the more recent innovations, offers exceptional fidelity and simplifies the rendering process by encapsulating scenes in a continuous volumetric representation without the need for complex texture mapping[1], [2].

The objective of this software project is to augment an Optix path tracer to render NeRF volumes along with other traditional assets in a scene, thereby displaying photorealistic objects without necessitating a team of artists to produce high-resolution models and textures.

II. Implementation

A. Technical Background

NeRF offers an alternative to traditional discrete scene representations such as triangle meshes or voxel grids by modeling scenes as continuous volumes embedded within a neural network's parameters. Unlike conventional methods, NeRF processes scene data continuously, rendering pixels by querying spatial coordinates and view directions. This approach returns RGB and density values that are integrated using traditional volume rendering techniques[1].

NVIDIA's Optix is a ray-tracing library that leverages RTX cores on modern NVIDIA graphics hardware. RTX cores are specialized for hardware-accelerated BVH (Bounding Volume Hierarchy) intersection tests, which significantly speed up ray-tracing algorithms. Optix facilitates the automation of acceleration structure creation, geometry intersections, and provides an API for writing shader code in C++/CUDA, thereby enhancing the efficiency of rendering processes[3].

Volume rendering, distinct from ray tracing, involves ray-marching, which takes discrete steps along a ray to sample 3D space. At each step, radiance and attenuation are accumulated from the sampled values. Decisions are then made to either stop the ray due to absorption, scatter it in a new direction, or continue the march, depending on the sampled values and the requirements of the volume rendering implementation[4].

B. System Pipeline Overview

The complete pipeline for rendering a NeRF volume, as depicted in Fig. 1, involves several critical steps. Initially, photographic data is collected, which must then be carefully curated. While some photographic tools can embed gyroscopic metadata about the camera pose directly into the images, other scenarios require the estimation of camera poses using photogrammetry methods. These images, along with the derived or embedded camera poses, are utilized to train the NeRF, creating a continuous volumetric representation. Subsequently, this representation is rendered to the screen, showcasing the integrated visual output of the process.

C. Development Process

1) Data Accumulation: Data capture for the Neural Radiance Field (NeRF) was conducted using an iPhone 11 on a calm, overcast day to ensure uniform lighting conditions, minimize harsh shadows, and reduce the presence of transient movement in the scene. A comprehensive video was recorded by encircling the subject from three different zenith angles, completing a 360-degree rotation at each level.

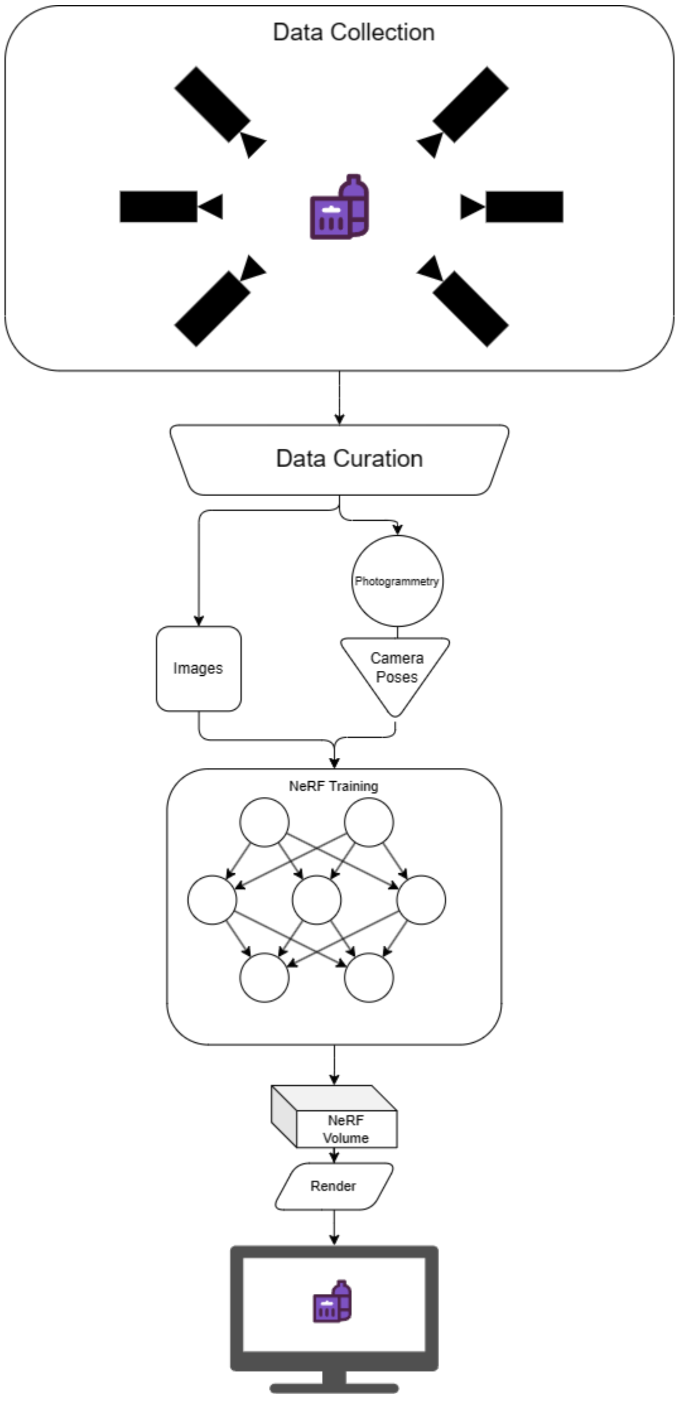

Fig. 1. Flow diagram illustrating the process from data collection through rendering.

The project required the development of a curation tool to extract frames from the video, selectively filtering for clarity and subject centricity. Clarity was assessed by computing the variance of the Laplacian for each frame. Frames with a variance less than one standard deviation below the mean were considered sharp enough for inclusion[5]. Subject centricity was evaluated using HSV color bounds to isolate the subject via a mask, facilitating contour detection. Frames where the centroid of the contour was within eight standard deviations from the mean centroid to image center were selected for the dataset.

The curated images were then processed using the photogrammetry software, COLMAP[6], to estimate camera poses, producing a JSON file with paths and camera transformation matrices for each image.

This JSON file was input into NVIDIA's Instant-NGP software to train a NeRF model[7]. Post-training, users manually adjust an axis-aligned bounding box (AABB) around the scene's target feature, from which a snapshot of the model can be saved. Additionally, a voxel grid representing raw volume data is exported as an RGBA floating-point array.

2) Path Tracer Augmentation: The existing path tracer, built on the Optix framework, was enhanced to support volumetric rendering. This involved integrating a MISS shader for volumetric accumulation. The path tracer identifies the scene's volumetric subject locations using a cube marked with a material designated as ”volume.” It calculates the cube's Axis-Aligned Bounding Box (AABB), which serves as a spatial constraint within the scene.

Optix's support for multiple ray types, each associated with unique payload data structures, enabled the introduction of a specific volumetric ray type. Radiance rays, when intersecting the volume's AABB, trigger a shift to a volume shader via the radiance hit shader. This shader calculates the maximum ray length for volume rendering by dividing the volume's largest dimension into 100 uniform steps. Short rays, expected to miss the geometry, are cast to invoke the MISS shader, which samples RGB and density values from the volume data for accumulation[3].

III. Challenges and Solutions

Software development inherently involves complexity, especially when it entails the integration and understanding of third-party codebases. A primary challenge encountered in this project was the time constraint associated with deciphering and adapting NVIDIA's Instant-NGP codebase. Specifically, the tiny-cuda-nn framework, integral to the project, is typically used for training the NeRF neural network. However, within the scope of this project, its crucial role was to query the already trained network to facilitate real-time rendering. The lack of comprehensive API documentation and the need to deeply understand the framework's data handling on the GPU significantly extended the learning curve and adaptation process.

A. Understanding Complex Codebases

The initial ambition to incorporate a continuous volumetric representation encoded by a trained NeRF neural network directly within the rendering pipeline faced significant hurdles. These included the need to decompose the trained model's snapshot to extract relevant parameters and transfer them to the GPU for use in Optix shaders. The complexity was compounded by the lack of API documentation for tiny-cuda-nn, necessitating an extensive examination of its codebase to understand the GPU data access patterns. Notably, tiny-cuda-nn's main class interface does not expose pointers to its GPU data structures, essential for executing the neural network's matrix multiplications on the device. The time required to unravel and adapt these intricacies would have extended beyond the project's deadline[8], [7].

B. Adopting a Practical Solution

In response to these challenges, a decision was made to adopt a more practical approach: exporting the neural network's continuous volumetric representation to a discrete volumetric grid. This method involved simplifying the data into a format that could be directly transferred to the GPU, byte for byte. Although this approach sacrifices some level of detail inherent in the continuous representation, the RGB values stored in the discrete grid were sufficiently detailed to produce renders with photographic fidelity that accurately reflect the volume learned by the NeRF network. This strategy allowed the project to meet its deadlines while achieving a high level of visual realism in the rendered output.

C. Optix Rendering Domain

Integrating ray marching within the Optix API presented significant challenges. Optix is optimized primarily for ray-tracing, focusing on intersection testing with RTX cores and does not expose intersection data directly[3]. This limitation complicates the implementation of ray marching, as continuously looped marching within a volume might overlook other geometries inside the volume's bounds.

D. Strategic Implementation of Volume Ray Casting

To address these challenges, a non-standard method was adopted: the integration of a volume type ray alongside the standard radiance rays. Instead of conventional ray marching within a loop, volume rays are cast as short segments, each representing the length of a single marching step. This method allows for implicit ray marching, where volumetric effects are accumulated inside the MISS shader when no geometry is intersected by the Optix ray.

IV. Results

The project successfully integrates NeRF-based volume rendering within the Optix framework to produce photorealistic images. While traditional performance gains in terms of speed are not realized due to the inherent costs of path marching, the project demonstrates substantial benefits in resource utilization and visual output quality.

Rendered images exhibit remarkable detail and realism, closely matching actual photographs of the rendered asset. A side-by-side comparison (see Figure 2) illustrates the rendering's ability to capture intricate lighting and textures, validating the NeRF approach in producing photorealistic visuals.

Although the use of a discrete voxel grid as opposed to a direct neural network representation increased the data storage requirements, it eliminated the need for detailed geometric models and texture maps for the asset. This trade-off highlights a potential area for resource optimization, especially in scenarios where storage and memory are less constrained than computational power.

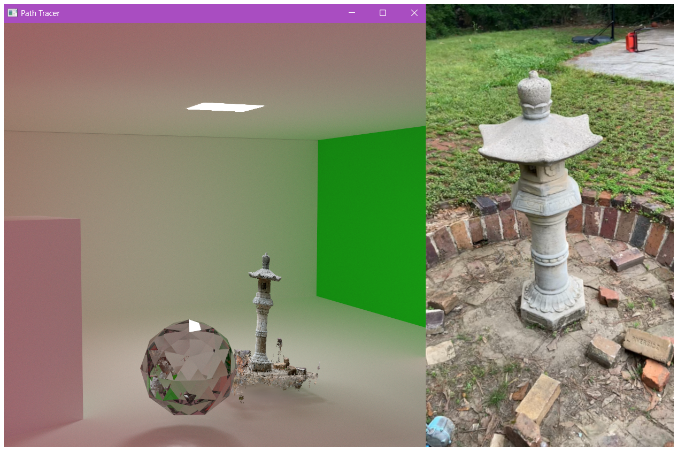

Fig. 2. Comparison of the rendered NeRF asset in a scene (left) against an actual photograph of the same asset (right).

The frame rates output for each render pass indicated stable performance, albeit without definitive conclusions on convergence due to the stochastic nature of Monte Carlo methods. Future work could aim to optimize the convergence metrics to enhance performance predictability.

This project's approach aligns with trends towards more data-driven rendering techniques in the industry, offering a foundation for future explorations in how neural networks can simplify the rendering pipeline while maintaining high visual quality.

V. Conclusion

This study successfully demonstrates the integration of Neural Radiance Fields (NeRF) with an Optix-based path tracer, providing a framework for volumetric rendering in complex 3D environments. The enhancements to the path tracing software, such as the use of Axis-Aligned Bounding Boxes and MISS shaders, facilitate improved management of ray segments, contributing to a more effective rendering process. While this integration reveals significant potential for advancing the realism and performance of 3D rendering, ongoing work will aim to refine the volume sampling methods to optimize rendering times and quality further. The findings encourage continued exploration into the fusion of machine learning with traditional rendering techniques, promising substantial advancements in the fields of gaming, virtual reality, and cinematic visual effects.

VI. Future Work

Given the time constraints and the complexity involved in extracting the neural network from the model snapshot, as well as loading and querying the model within the context of an Optix shader, this implementation opted to use raw volume data to demonstrate the photographic fidelity of NeRF. Future work will focus on modifying the volume sampling function to query the continuous volume representation from the neural network instead of relying on a large voxel grid. This change aims to enable true real-time NeRF rendering.

Additionally, the current implementation naively uses a ray marching step size based on the size of the AABB as a quick heuristic. This approach may exhibit striation artifacts at certain viewing angles due to the non-uniform nature of sampling the volume from oblique angles. Future improvements will involve calculating the actual length of the ray within the bounds of the AABB. Normalizing the step size against the true ray length should minimize these artifacts, leading to a more consistent and visually accurate rendering.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, p. 99–106, dec 2021. [Online]. Available: https://doi.org/10.1145/3503250

[2] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, and J. Li, "Nerf: Neural radiance field in 3d vision, a comprehensive review," 2023.

[3] NVIDIA Corporation, *NVIDIA Optix User's Manual*, 2021, accessed: 2024-05-01. [Online]. Available: https://raytracing-docs.nvidia.com/optix8/guide/index.htmlpreface

[4] M. Pharr and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd ed. Morgan Kaufmann, 2016. [Online]. Available: https://www.pbrt.org

[5] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall, 2002.

[6] J. L. Schoenberger and J.-M. Frahm, "Structure-from-motion revisited," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530127

[8] T. Müller, "tiny-cuda-nn," 4 2021. [Online]. Available: https://github.com/NVlabs/tiny-cuda-nn