Efficient 3D Perception from Images

Ruilong Li

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-134
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-134.html

June 2, 2025

Efficient 3D Perception from Images

By

Ruilong Li

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Angjoo Kanazawa, Chair
Professor Alexei Efros
Professor Trevor Darrell
Professor Sanja Fidler

Spring 2025

Efficient 3D Perception from Images

Abstract

Efficient 3D Perception from Images

by

Ruilong Li

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Angjoo Kanazawa, Chair

The ability to capture, reconstruct, and interact with the world in three dimensions is transforming how we experience, understand, and shape our environment. From virtual reality and digital heritage to robotics and scientific discovery, 3D perception is opening new frontiers across art, science, and technology. Yet, despite remarkable progress, creating high-fidelity 3D models from images remains a computationally demanding challenge—one that limits the accessibility and scalability of these powerful tools.

This thesis addresses the core bottleneck of efficiency in differentiable volume rendering, a foundational technique behind recent breakthroughs such as Neural Radiance Fields (NeRF) and Gaussian Splatting. While these methods have demonstrated that continuous volumetric representations and differentiable rendering pipelines can achieve photorealistic results from sparse or unconstrained inputs, their high computational and memory costs pose significant barriers to real-time and large-scale applications.

To overcome these challenges, I present a series of algorithmic and systems-level innovations aimed at making 3D perception faster, more scalable, and more practical. First, I introduce compact and expressive scene representations that reduce memory overhead without sacrificing quality. Second, I develop smarter sampling and visibility strategies that exploit the inherent sparsity of 3D space, focusing computation where it matters most. Third, I design parallelization techniques tailored for modern multi-GPU systems, enabling distributed training and rendering at unprecedented scales. Finally, I explore learning-based approaches that leverage multi-view geometry and attention mechanisms to further accelerate and generalize 3D perception.

Through a combination of theoretical insights, open-source tools, and empirical validation, this work charts a path toward real-time, high-resolution 3D reconstruction that is accessible beyond specialized labs and supercomputers. By making 3D perception more efficient, this thesis aims to unlock new possibilities and bring us closer to a future where anyone can capture, share, and explore the spaces they love in all their depth and richness.

Contents

List of Figures

List of Tables

Acknowledgments

Time flies—it has already been four years since I began my PhD at UC Berkeley in 2021. Throughout this journey, I have been incredibly fortunate to be surrounded by so many wonderful people. This dissertation would not have been possible without their support and encouragement. While it is impossible to mention everyone, I would like to take this opportunity to express my heartfelt gratitude to those who have profoundly influenced me along the way.

First and foremost, I would like to express my deepest gratitude to my advisor, Angjoo Kanazawa. I first met Angjoo a year before joining Berkeley, during my internship at Google in 2020, where I had the privilege of being her intern. My initial impression was of someone exceptionally kind, approachable, and deeply passionate about both life and research. As I got to know her better during my time at Berkeley, these feelings only grew stronger. Her passion and kindness have had a profound impact on me, transforming me from a shy and introverted student into a more confident and open-minded person. I am also incredibly grateful for her unwavering support, not only in my research but also in my personal life. She granted me the flexibility to travel between China and the US multiple times and to take time off to be with my family—something that was invaluable in making my PhD journey both enjoyable and fulfilling.

I would also like to express my sincere gratitude to my other dissertation committee members: Alyosha Efros, Trevor Darrell, and Sanja Fidler. I have always enjoyed Alyosha and Trevor's talks and our conversations—their unique perspectives on both life and research have continually inspired me to think more deeply. I am also deeply appreciative of Sanja's support in my career development. Her guidance, along with the opportunities and industry connections she provided, has broadened my horizons and helped me see the bigger picture of research.

I was also very fortunate to have had many valuable experiences interning in industry during my PhD journey, where I met many great people. I would like to thank my internship mentors at Google, especially Shan Yang—my coding skills truly began with our pair programming sessions. I am grateful to my mentor at Meta, Christoph Lassner, who was extremely kind and helped me overcome many challenges during my time there. I also thank my mentor at NVIDIA, Francis Williams, who trained me to be a competent CUDA programmer, opening many doors for my research.

Of course, none of my research would have been possible without the help of my amazing labmates and collaborators. I would like to thank Hang Gao, Alex Yu, Junchen Liu, Brent Yi, Vickie Ye, Ethan Weber, Chenfeng Xu, Jeffery Hu, Mingh Vo, David Ross, Michael Zollhoefer, Zeng Huang, Yuliang Xiu, Shunsuke Saito, Julian Tanke, Kyle Olszewski, and many others. Working with Hang has been especially memorable—we have collaborated on many projects and shared a lot of fun along the way. Hang was also the first person I met at Berkeley and has always looked out for me, both inside and outside of research. I feel very lucky to have had Hang, along with all my labmates and collaborators, by my side.

Prior to joining Berkeley, I spent two years at the University of Southern California, where I was fortunate to be advised by Hao Li. Hao is not only a great advisor but also a great friend. He has a strong heart and helped me through many difficult times in my life. Although our paths have since diverged, I will always cherish the memories and lessons I learned from him. During my time

at USC, I was also fortunate to meet many wonderful labmates and friends, including Jiaman Li, Zhengfei Kuang, Yi Zhou, Xinglei Ren, Shichen Liu, Sida Peng, Carl Yang, Zeng Huang, Yuliang Xiu, Shunsuke Saito, Tianye Li, and many others. I am grateful to all of them for making my time at USC so memorable.

Lastly, I would like to thank my parents and my wife. Being away from home for so long has not been easy, but I am very lucky to have had my parents' constant support through our weekly online calls. My wife has also supported me through all the ups and downs, traveling multiple times to visit me. The love and support from my family and my wife have been my greatest motivation throughout this journey.

Chapter 1

Introduction

Imagine trying to capture the feeling of standing in your favorite place—a sunlit park, a bustling city street, or a cozy room—and sharing that experience with someone far away. For centuries, people have used paintings, photographs, and even sculptures to try to bring scenes to life. But while a photo can freeze a moment, it flattens the world, losing the sense of depth and space that makes being there so special.

Today, technology offers us new ways to see and understand places in three dimensions. From video games and movies to virtual tours and navigation apps, 3D experiences are becoming part of everyday life. Yet, creating these digital worlds remains surprisingly challenging. Turning a handful of photos into a lifelike 3D scene can require significant time, effort, and computational power.

Researchers have developed two main approaches to tackle this challenge. The first is akin to an artist who carefully studies each new scene, learning its details from scratch. These methods, known as *optimization-based approaches*, work by adjusting a digital model until it matches the photos as closely as possible. They can produce beautiful, detailed results, but must start over for every new place—like painting a new portrait each time.

The second approach is more like teaching a student who, after seeing thousands of different places, learns to recognize patterns and can quickly sketch a new scene. These *learning-based approaches* use vast collections of images to "train" a computer to understand what the world looks like. Once trained, they can create 3D scenes much faster, but the initial training process demands enormous resources—like building a library with millions of books just to answer a few questions.

No matter which path we take, there are significant hurdles. Making these methods faster and less demanding—so they can run on everyday computers or even phones—would open up new possibilities for art, science, education, and beyond. Imagine students exploring ancient ruins in 3D from their classrooms, families sharing memories in virtual spaces, or doctors planning surgeries with detailed models—all without needing a supercomputer.

Being able to see and understand places in three dimensions is at the heart of what we call 3D perception. It's not just about making things look real—it's about truly feeling present, being able to move around, and interacting with the world as if you were really there. 3D perception algorithms are a driving force behind many of today's most transformative technologies. In digital heritage preservation, 3D reconstruction techniques allow us to capture and archive culturally

significant sites and artifacts with unprecedented fidelity, enabling future generations to study and experience them virtually. In architecture and engineering, advanced 3D modeling and visualization tools empower professionals to design, simulate, and evaluate buildings and infrastructure before construction begins, reducing errors and improving outcomes. Virtual prototyping and simulation are now essential in manufacturing, allowing for rapid iteration and testing of complex products in a digital environment. Telepresence and remote collaboration platforms leverage real-time 3D perception to enable immersive communication and shared experiences across distances. Moreover, the development of intelligent agents—such as autonomous vehicles, service robots, and augmented reality systems—relies fundamentally on the ability to perceive, reconstruct, and interact with the world in three dimensions. In all these domains, efficient and accurate 3D perception is not just a convenience, but a necessity for progress and innovation.

However, despite these exciting advances and applications, there remain significant challenges that stand in the way of making 3D perception truly practical and widespread. One of the most pressing issues is efficiency. Many promising applications are currently limited—not by a lack of ideas, but by the sheer computational cost and resource demands of existing methods. For learning-based approaches, the process of training large models often requires vast amounts of data, time, and specialized hardware, making it inaccessible for most users and organizations. On the other hand, optimization-based methods, while flexible and powerful, can be slow to run and consume large amounts of memory, especially as scenes become more complex or detailed. As a result, tasks like real-time 3D reconstruction, interactive design, or large-scale simulation are often out of reach, or only possible with significant compromises in quality or speed. Improving efficiency—making 3D perception faster, lighter, and more scalable—is therefore the key to unlocking its full potential and enabling new generations of applications and intelligent systems.

This thesis addresses the efficiency problem in 3D perception by developing new algorithms and systems that advance both optimization-based and learning-based approaches. The first part of this work focuses on improving optimization-based methods—such as Neural Radiance Fields (NeRF) and Gaussian Splatting—by designing faster, more memory-efficient sampling and rendering pipelines. This includes novel strategies for importance sampling, visibility culling, and parallelization techniques that enable scalable processing of large and complex scenes, making real-time or high-resolution 3D reconstruction more practical.

In the latter part of the thesis, I turn to learning-based solutions, focusing on methods to enhance the efficiency and effectiveness of 3D perception models. Specifically, I investigate advanced multi-view attention mechanisms that enable models to more effectively aggregate and reason over geometric information from multiple images. By improving how these models consume and integrate 3D knowledge from training data, the proposed techniques reduce the amount of data and computation required for training, while also enhancing the model's ability to generalize to novel scenes and tasks. This work aims to make learning-based 3D perception not only faster and more resource-efficient, but also more robust and widely applicable in real-world scenarios.

In the chapters ahead, I share ideas, tools, and discoveries that bring us closer to a world where anyone can capture, share, and explore the spaces they love—in all their depth and beauty. My hope is that, by making these advances, we can help everyone—from curious children to pioneering scientists to future AI agents—perceive and interact with the world in new and meaningful ways.

1.1 Dissertation Overview

This thesis is organized into five chapters, each addressing a key aspect of making 3D perception more efficient, scalable, and accessible. Together, these chapters trace a path from foundational innovations in optimization-based methods to advances in learning-based approaches, all with the goal of enabling richer and more practical 3D experiences.

Chapter 2: NerfAcc – Smarter Sampling Accelerates NeRFs

This chapter introduces NerfAcc, a flexible toolbox designed to accelerate Neural Radiance Fields (NeRFs) by rethinking how we sample and process 3D data. By unifying and generalizing sampling strategies under the concept of transmittance estimation, NerfAcc enables significant speedups in NeRF training and rendering across a range od NeRF representations, making high-quality 3D reconstruction more practical and accessible.

Chapter 3: NeRF-XL – An Efficient Multi-GPU Framework for NeRFs

Here, the focus shifts to scaling up 3D reconstruction for large and complex scenes, and explore how to efficiently harness the power of multi-GPU systems. NeRF-XL presents a principled method for distributing NeRFs across multiple GPUs, allowing for the training and rendering of models with far greater capacity. This chapter details how careful algorithm and system design can unlock new levels of efficiency and quality through distributed computing, even as scenes grow in size and detail.

Chapter 4: gsplat – A Library for Efficient Gaussian Splatting

This chapter explores Gaussian Splatting, a recent paradigm for real-time 3D scene reconstruction and rendering. Together with collaborators, we introduce the gsplat library as an open-source, modular platform that brings new levels of speed and memory efficiency to this approach. The chapter highlights how thoughtful software and algorithm design can make advanced 3D techniques more usable for both researchers and practitioners.

Chapter 5: PRoPE – Multi-view Attention Leads to Faster Convergence

The final technical chapter turns to learning-based 3D perception. It presents Projective Positional Encoding (PRoPE), a novel method for conditioning multiview transformers on camera parameters. By enabling models to more effectively aggregate geometric information from multiple images, PRoPE improves both the efficiency and generalization of learning-based 3D systems, paving the way for broader adoption of these powerful tools.

Chapter 6: Conclusion

The thesis concludes by synthesizing the key findings and reflecting on the broader impact of efficient 3D perception. It discusses remaining challenges and outlines promising directions for future research, with the hope of inspiring continued progress toward a world where 3D understanding is as natural and widespread as photography is today.

Chapter 2

NerfAcc: Efficient Sampling Accelerates NeRFs

Optimizing and rendering Neural Radiance Fields is computationally expensive due to the vast number of samples required by volume rendering. Recent works have included alternative sampling approaches to help accelerate their methods, however, they are often not the focus of the work. In this paper, we investigate and compare multiple sampling approaches and demonstrate that improved sampling is generally applicable across NeRF variants under an unified concept of transmittance estimator. To facilitate future experiments, we develop NerfAcc, a Python toolbox that provides flexible APIs for incorporating advanced sampling methods into NeRF related methods. We demonstrate its flexibility by showing that it can reduce the training time of several recent NeRF methods by 1.5× to 20× with minimal modifications to the existing codebase. Additionally, highly customized NeRFs, such as Instant-NGP, can be implemented in native PyTorch using NerfAcc. Our code are open-sourced at `https://www.nerfacc.com`.

2.1 Introduction

Neural volume rendering has revolutionized the inverse rendering problem, with the Neural Radiance Field (NeRF) [61] being a key innovation. The continuous radiance field representation allows for rendering novel views of a scene from any camera position. However, the optimization of a NeRF can be computationally expensive due to the neural representation of radiance field and the large number of samples required by volume rendering. These challenges have limited practical applications of NeRF-based optimization and rendering.

Several recent works have successfully reduced the computational cost of neural volume rendering by proposing more efficient radiance field representations [63, 124, 123, 10, 91, 25]. While there are differences in the specific radiance field representations and their applications, most of these methods share a similar volume rendering pipeline which involves creating samples along the ray and accumulating them through alpha-composition.

However, compared to the considerable efforts focused on developing efficient radiance field

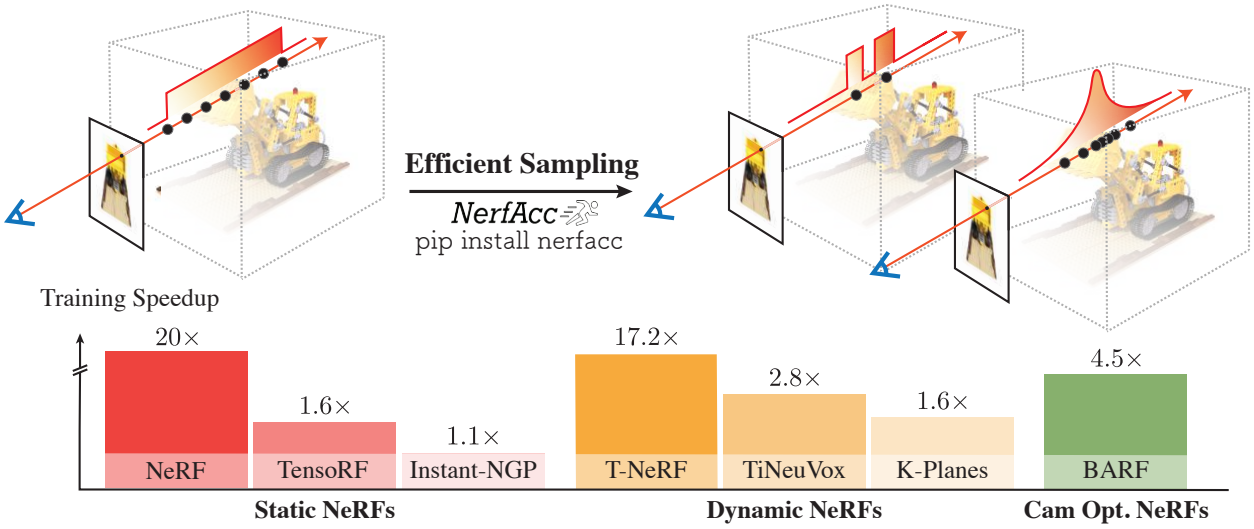


Figure 2.1: **NerfAcc Toolbox**. Our proposed toolbox, *NerfAcc*, integrates advanced efficient sampling techniques that lead to significant speedups in training various recent NeRF papers with minimal modifications to existing codebases.

representations, there has been limited attention devoted to reducing the computational cost of neural volume rendering through *efficient sampling*. While a few recent works have included alternative sampling approaches to accelerate their methods [63, 124, 91, 2], these methods are often not the main focus of the paper. Moreover, implementing advanced sampling approaches typically requires non-trivial efforts. For example, Instant-NGP [63] and Plenoxels [124] both employ highly customized CUDA implementations to achieve spatial skipping during ray marching, which are tightly coupled with their respective radiance field implementations. Consequently, it can be challenging for researchers to benefit from these advanced sampling approaches in their own research.

In this paper, we investigate and compare several advanced sampling approaches from the literature and provide mathematical proofs demonstrating that they can all be viewed in a unified way of creating an *estimation of transmittance* for importance sampling. Our analysis shows that by understanding the spectrum of sampling through transmittance estimator, novel sampling strategies can be explored. Based on this, we decouple the sampling procedure from the neural volumetric rendering pipeline and demonstrate that improved sampling is generally applicable across different variants of NeRF. Furthermore, to facilitate future experiments, we propose NerfAcc, a plug-and-play toolbox that provides a flexible Python API for integrating advanced sampling approaches into NeRF-related methods, ready for researchers to incorporate into their own codebases. We demonstrate that with less than 100 lines of code change using NerfAcc, various NeRF methods [61, 10, 22, 25, 53, 72, 63] can enjoy 1.5× to 20× training speedup with better performance. Notably, using the NerfAcc library, one can train an Instant-NGP [63] model with pure Python code in the same amount of time, and achieve slightly better performance (+0.2dB) than the official pure

CUDA implementation.

Our paper presents a unique contribution to the community. Unlike other papers that introduce novel algorithms, our work sheds light on the intricacies of various sampling approaches, which are often overlooked despite their significance. As far as we know, this work is the first paper that dives deep into this crucial aspect in the context of neural radiance field, and offer a novel, unified concept that allows researchers to view existing sampling algorithms in a fresh perspective. In addition, we provide a toolbox that facilitates research and development in this area. We believe translation of mathematical ideas into efficient, easy-to-modify implementation is fundamental to the research development. Overall, we hope that our proposed concept, along with the toolbox, can inspire new research ideas and accelerate progress in this field.

2.2 Related Works

NeRF Codebases. The recent explosion of NeRF-related research has led to numerous papers, many of which have released their own codebases [3, 2, 10, 25, 50, 61, 67, 122, 27]. These codebases address various tasks related to NeRF, including surface reconstruction [65, 108, 119], radiance field representation [3, 2, 63, 91, 124], dynamic modeling [22, 25, 50, 72], and camera optimization [53, 109]. However, each codebase is tailored to a specific task and supports only a single approach. While most of these methods share the same volume rendering pipeline of accumulating samples along the ray, transferring the implementation from one codebase to another requires non-trivial efforts. In this work, we address this common sampling problem by introducing NerfAcc, a plug-and-play toolbox that supports all the aforementioned tasks and methods, making it easy to integrate into any existing codebase.

NeRF Frameworks. Recently, several projects have been introduced to integrate different NeRF variants into a single framework, such as NeRF-Factory [36], Nerfstudio [98], and Kaolin-Wisp [93]. These frameworks have made significant progress in facilitating future NeRF-related research. NeRF-Factory offers a collection of NeRF variants [61, 129, 91, 124, 3, 2, 105] with original implementations and focuses on comprehensive benchmarking. Nerfstudio consolidates critical techniques introduced in existing literature [61, 3, 2, 63, 56] and provides modular components for the community to easily build on. Kaolin-Wisp builds upon the Kaolin [26] framework and implements a set of voxel-based NeRF papers [94, 63, 95]. However, these frameworks are designed to encourage researchers to develop within the framework, and do not benefit users working on their own codebases. Moreover, due to the high activity in the NeRF-related research, it is almost impossible to keep track of the latest developments and integrate advanced techniques into a single framework. Therefore, NerfAcc is designed as a standalone library that can be plugged into any codebase. It supports a wide range of NeRF related methods and can be easily maintained as new methods emerge.

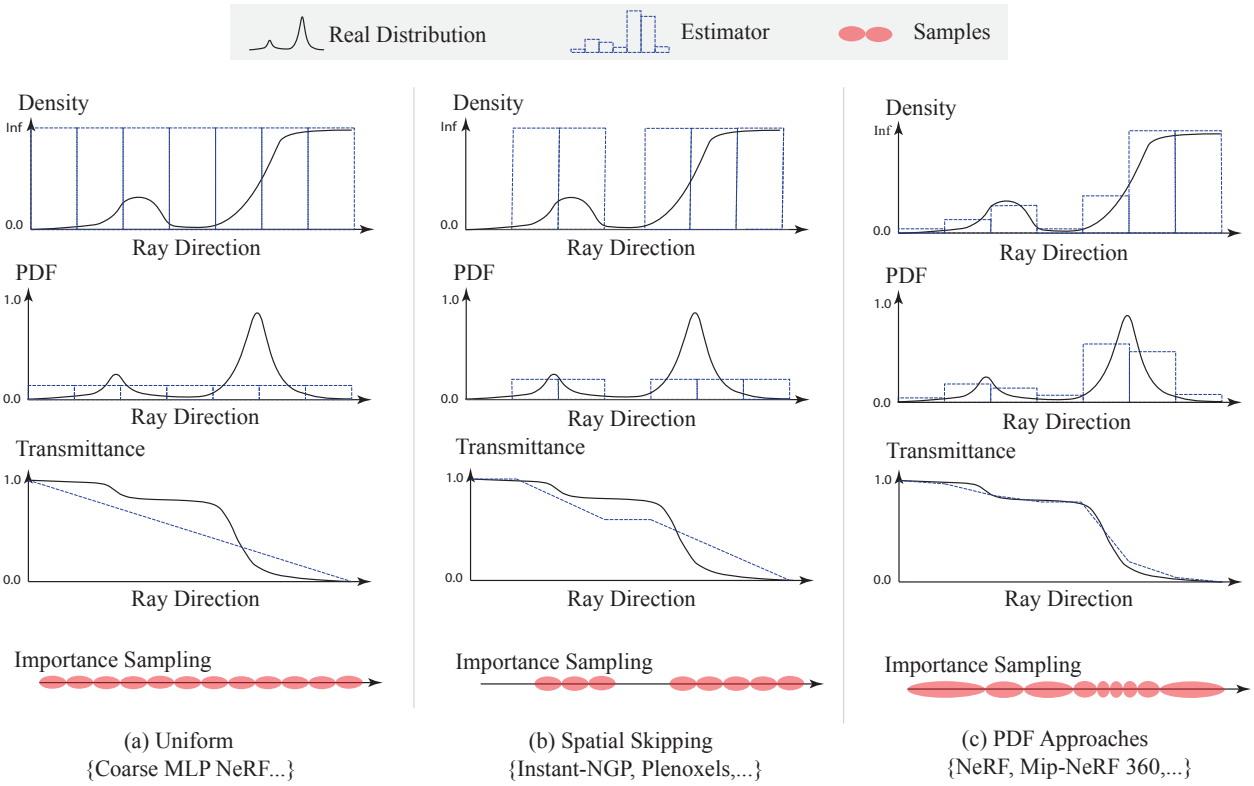


Figure 2.2: **Illustration of Sampling via Transmittance Estimator.** Although spatial skipping approaches (e.g., Occupancy Grid in Instant-NGP [63]) and PDF approaches (e.g., Proposal Network in Mip-NeRF 360 [2]) appear distinct from each other, they can both be viewed as constructing a transmittance estimator from which samples can be created via importance sampling.

2.3 Importance Sampling via Transmittance

Several advanced sampling approaches exist in the literature. For instance, Plenoxels [124] uses a sparse grid, Instant-NGP [63] uses an occupancy grid, NeRF [61] employs a coarse-to-fine strategy, Mip-NeRF 360 [2] proposes proposal networks. However, creating a toolbox that can support all of these approaches is not an easy task since they operate in completely different ways. In this section, we demonstrate that these approaches can all be viewed in a unified way of importance sampling. We also provide a mathematical proof that *transmittance is all you need for importance sampling*. Thus, each method essentially has its own way of creating an estimation of the transmittance along the ray, which we refer to as the *transmittance estimator*. This observation enables us to unify different types of sampling approaches under the same framework and leads to our NerfAcc toolbox.

Formulation

Efficient sampling is a well-explored problem in Graphics [24], wherein the emphasis is on identifying regions that make the most significant contribution to the final rendering. This objective is generally accomplished through importance sampling, which aims to distribute samples based on the probability density function (PDF), denoted as $p(t)$, between the range of $[t_n, t_f]$. By computing the cumulative distribution function (CDF) through integration, *i.e.*, $F(t) = \int_{t_n}^{t} p(v)\, dv$, samples are generated using the inverse transform sampling method:

$$t = F^{-1}(u) \quad \text{where} \quad u \sim \mathcal{U}[0, 1]. \tag{2.1}$$

In volumetric rendering, the contribution of each sample to the final rendering is expressed by the accumulation weights $T(t)\sigma(t)$:

$$\begin{aligned} C(\mathbf{r}) &= \int_{t_n}^{t_f} T(t)\, \sigma(t)\, c(t)\, dt \\ \text{where} \quad T(t) &= \exp\left(- \int_{t_n}^{t} \sigma(s)\, ds \right). \end{aligned} \tag{2.2}$$

Hence, the PDF for volumetric rendering is $p(t) = T(t)\sigma(t)$ and the CDF $F(t) = \int_{t_n}^{t} p(v)\, dv$ can be derived as a function of transmittance $T(t)$:

$$\begin{aligned} F(t) &= \int_{t_n}^{t} T(v)\sigma(v)\, dv \\ &= \int_{t_n}^{t} \frac{d}{dv} \left[- \exp\left(- \int_{t_n}^{v} \sigma(s) ds \right) \right] dv \\ &= 1 - \exp\left(- \int_{t_n}^{t} \sigma(s) ds \right) \\ &= 1 - T(t). \end{aligned} \tag{2.3}$$

Therefore, inverse sampling the CDF $F(t)$ is equivalent to inverse sampling the transmittance $T(t)$. Thus a transmittance estimator is sufficient to determine the optimal samples. Intuitively, this suggests to put more samples around regions where the transmittance changes rapidly – and that is exactly what happens when a ray hit a surface. Implementation wise, this observation enables us to compute the CDF directly using $1 - T(t)$, instead of the computationally expensive integral $\int_{t_n}^{t} T(v)\sigma(v)\, dv$, which is the standard implementation adopted by many popular codebases [61, 3, 2, 98, 25].

While advanced transmittance estimator techniques, such as delta tracking [106] with Monte Carlo sampling, are utilized in production-level volumetric rendering in Graphics [24], NeRFs operate in a distinct setting in which the scene geometry is not predefined but optimized on the fly. During NeRF optimization, the radiance field changes between iterations, necessitating the dynamic update of the transmittance estimator at each step k:

$$\mathcal{F} : T(t)^{k-1} \mapsto T(t)^{k}. \tag{2.4}$$

Methods	**PDF Estimator** $p(t_i)$	**Trans. Estimator** $T(t_i)$	**Update** $\mathcal{F}$	**Instantiation**
Uniform	Constant	$1 - \frac{t-t_n}{t_f-t_n}$	-	NeRF [61]
Spatial Skipping	$\frac{\mathbb{1}\left[\sigma(t_i)>\tau\right]}{\sum_{j=1}^{n}\mathbb{1}\left[\sigma(t_i)>\tau\right]}$	$1 - \frac{\sum_{j=1}^{i-1}\mathbb{1}\left[\sigma(t_i)>\tau\right]}{\sum_{j=1}^{n}\mathbb{1}\left[\sigma(t_i)>\tau\right]}$	EMA SGD	Instant-NGP [63] Plenoxels [124]
PDF Approaches	$\sigma(t_i)\exp(-\sigma(t_i)\,dt)$	$\exp(-\sum_{j=1}^{i-1}\sigma(t_i)\,dt)$	SGD SGD	NeRF [61] Mip-NeRF 360 [2]

Table 2.1: **Mathematical Formulations of Different Sampling Approaches.** We outlines the significant mathematical distinctions among each sampling approach, under the perspective of transmittance estimator. See Section 2.3 for notations.

This introduces additional challenges to efficient sampling because it becomes more difficult to accurately estimate the transmittance from a radiance field that is constantly changing. Current approaches employ either exponential moving average (EMA) or stochastic gradient descent (SGD) as the update function $\mathcal{F}$. However, we note that there may be other update functions that could be explored.

With these concepts in mind, let us now examine some of the existing approaches towards efficient sampling.

Uniform. If a transmittance estimator is not available, the only assumption we can make is that every point along the ray contributes equally to the final rendering. Mathematically, this assumption translates to a constant PDF and a linearly decaying transmittance $T(t) = 1 - (t - t_n)/(t_f - t_n)$. In this case, the sampling process is equivalent to uniformly sampling along the ray, i.e., $t_i = t_n + (t_f - t_n) \cdot u_i$. It is worth noting that every NeRF model that uses uniform sampling inherently assumes this linear transmittance decay, such as the coarse level in vanilla NeRF [61]. See Fig. 2.2(a) for the illustration.

Spatial Skipping. A more sophisticated approach to improve uniform sampling is to identify empty regions and skip them during sampling, as proposed in Instant-NGP's Occupancy Grid [63] and Plenoctrees' Sparse Grid [123]. This technique binarizes the density along the ray with a conservative threshold τ: $\hat{\sigma}(t_i) = \mathbb{1}\left[\sigma(t_i) > \tau\right]$. Consequently, the piece-wise constant PDF can be expressed as $p(t_i) = \hat{\sigma}(t_i)/\sum_{j=1}^{n}\hat{\sigma}(t_j)$, and the piece-wise linear transmittance estimator is $T(t_i) = 1 - \sum_{j=1}^{i-1}\hat{\sigma}(t_j)/\sum_{j=1}^{n}\hat{\sigma}(t_j)$. To update this estimator during optimization, Instant-NGP [63] directly updates the cached density with exponential moving average (EMA) over iteration k: $\sigma(t_i)^k = \gamma \cdot \sigma(t_i)^{k-1} + (1-\gamma) \cdot \sigma(t_i)^k$. Meanwhile, Plenoxels [124] updates the density via the gradient descent through the rendering loss. See Fig. 2.2(b) for an illustration.

PDF Approaches. Another type of approach is to directly estimate the PDF along the ray with discrete samples. In vanilla NeRF [61], the coarse MLP is trained using volumetric rendering loss to output a set of densities $\sigma(t_i)$. This allows for the creation of a piece-wise constant PDF: $p(t_i) = \sigma(t_i)\exp(-\sigma(t_i)\,dt)$, and a piece-wise linear transmittance estimator $T(t_i) = \exp(-\sum_{j=1}^{i-1}\sigma(t_i)\,dt)$. This approach was further improved in Mip-NeRF 360 [61] with a PDF matching loss, which allows for the use of a much smaller MLP in the coarse level, namely Proposal Network, to speedup the PDF construction. In both cases, the transmittance estimator is updated through gradient descent. See Fig. 2.2(c) for an illustration.

Table 2.1 provides a mathematical summary and comparison of these approaches. Additionally, we present an illustration in Figure 2.2 to provide an intuitive comparison of these approaches from the perspective of PDF (second row) and transmittance (third row), as well as how the samples can be created from the transmittance estimator via importance sampling (last row). This visualization also reveals some pros and cons for each approach, which we will discuss in in Section. 2.3.

Design Spaces

Choice of Representations. The transmittance estimator can use either an explicit voxel [63, 124, 10], an MLP [3, 2, 61], or a hybrid representation [98]. Depending on whether the estimator is explicit or not, it can be updated with either rule-based EMA [63, 10] or gradient descent with some supervision [124, 2, 61, 98]. Generally, voxel-based estimators are faster than implicit (e.g., MLP-based) estimators but suffer more from aliasing issues. It is worth noting that the transmittance estimator's representation can significantly benefit from advances in radiance field representation. For example, the Nerfacto model [98] uses the most recent hybird-representation HashEncoding [63] for both the radiance field and the sampling module, achieving the best quality-speed tradeoff in in-the-wild settings.

Handling Unbounded Scenes. So far, we have only discussed sampling within a bounded area $[t_n, t_f]$. For unbounded scenes, it is impossible to densely sample along the ray. Similar to the mipmaps used in graphics rendering, a general solution is to sample more coarsely as the ray goes further, as objects farther away appear in fewer pixels on the image plane. This can be achieved by creating a bijective mapping function $\Phi : s \in [s_n, s_f] \mapsto t \in [t_n, +\infty]$, and performing sampling in the s-space instead of the t-space. Several papers [2, 129, 79] that work on unbounded scenes have introduced different mapping functions Φ, to which we refer our readers for details.

Discussions

Pros and Cons. Sampling with uniform assumption is the easiest one to implement but with lowest efficiency in most cases. Spatial skipping is a more efficient technique since most of the 3D space is empty, but it still samples uniformly within occupied but occluded areas that contribute little to the final rendering (e.g., the last sample in Figure 2.2(b)). PDF-based estimators generally provide more accurate transmittance estimation, enabling samples to concentrate more on high-contribution areas (e.g., surfaces) and to be more spread out in both empty and occluded regions.

However, this also means that samples are always spread out throughout the entire space without any skipping, as shown in Figure 2.2(c). Moreover, the current approaches all introduce aliasing effects to the volumetric rendering due to either (1) the piece-wise linearity assumption for estimating transmittance along the ray [3, 2, 61, 124, 63] as illustrated in Figure 2.2, or (2) the underlying voxel representation for the transmittance estimator [124, 63] as discussed in Section 2.3. A recent work, Zip-NeRF [4], addresses the aliasing problem tied to these two exact issues (called "z-aliasing" and "xy-aliasing" in their work), which are naturally revealed under our unified framework.

Implementation Difficulties. The current implementations for efficient sampling are all highly customized and tightly integrated with the specific radiance field proposed in each paper. For instance, spatial skipping is implemented with customized CUDA kernels in Instant-NGP [63] and Plenoxels [124]. Mip-NeRF 360 [2], K-planes [25], and Nerfacto [98] have a proposal network implemented but it is closely integrated with their repositories and can only support limited types of radiance fields that come with the repository. However, as shown before, the sampling process is independent of the radiance field representation, thus it should be easily transferable across different NeRF variants. Due to the various implementation details, it typically requires significant effort to correctly implement an efficient sampling approach from scratch. Therefore, having an implementation that is easily transferable from repository to repository would be valuable in supporting future research on NeRF.

Insights from Unified Formulation. Comprehending the sampling spectrum through the lens of Transmittance Estimator paves the way for investigating novel sampling strategies. For example, our framework reveals that the Occupancy Grid from Instant-NGP [63] and the Proposal Network from Mip-NeRF 360 [2] are not mutually exclusive but complementary, as both aim to estimate the transmittance along the ray. Therefore, combining them becomes straight-forward: one can first compute the transmittance with the occupancy grid, and then refine the estimated transmittance with a proposal network. This enables both skipping on the empty space and concentrating the samples onto the surface. We explore this approach in Section 2.4 and demonstrate that it overcomes the limitation of the proposal network approach, which always samples the entire space. Furthermore, this formulation could potentially shed light on questions such as how to enhance the sampling procedure with depth information or other priors, which we encourage readers to investigate further.

2.4 NerfAcc Toolbox

In this paper, we present *NerfAcc* toolbox, designed for **Ne**ural **r**adiance **f**ield **Acc**eleration. It provides efficient sampling for volumetric rendering, that is universally applicable and easily integrable for a diverse range of radiance fields [61, 10, 63, 53, 108]. In this section, we first introduce the design principles of this toolbox, along with critical implementation details. To demonstrate its flexibility, we further show that it can significantly speedup the training of various NeRF-related papers by $1.5\times$ to $20\times$ with only minor modifications to existing codebases.

Algorithm 1: NerfAcc Rendering Pipeline. In NerfAcc, the samples are created from a transmittance estimator, which can be updated during the NeRF training. See Section 2.4 for details.

```
# nerf: a radiance field model.
# r_o: ray origins. (n_rays, 3)
# r_d: ray normalized directions. (n_rays, 3)

# e.g., Prop Net, Occ Grid.
estimator = nerfacc.TransmittanceEstimator()

def density_fn(t0, t1, r_id):
   """Query density from nerf."""
   return nerf.density(r_o[r_id], r_d[r_id], t0, t1)

def rgb_density_fn(t0, t1, r_id):
   """Query rgb and density from nerf."""
   return nerf.forward(r_o[r_id], r_d[r_id], t0, t1)

# Efficient sampling. (t0, t1, r_id): packed samples. (all_samples,)
t0, t1, r_id = nerfacc.sampling(r_o, r_d, estimator, density_fn=density_fn)
# Differentiable volumetric rendering.
color, opacity, depth, aux = nerfacc.rendering(t0, t1, r_id, rgb_density_fn=rgb_density_fn)
# Update the transmittance estimator.
estimator.update_every_n_steps(t0, t1, r_id, aux)
# nerf, r_o and r_d all receive gradients.
F.mse_loss(color, color_gt).backward()
```

Design Principles

This library is designed with these goals in mind:

- **Plug-and-play.** Our primary objective is to ease the challenges of implementing an efficient volumetric sampling technique for the research community. Therefore, NerfAcc is designed as a standalone library that can be easily installed from PyPI on both Windows and Linux platforms, and seamlessly integrated into any PyTorch codebase.
- **Efficiency & Flexibility.** To maximize the speed of the code, we fuse the operations into CUDA kernels as much as possible, while exposing flexible Python APIs to the users.
- **Radiance Field Complexity.** We target on supporting any radiance fields that are designed for per-scene optimization, including both density-based and SDF-based fields, for both static and dynamic scenes.

Implementation Details

NerfAcc incorporates two advanced sampling methods that can be decoupled from the radiance field representation, namely the Occupancy Grid from Instant-NGP [63] and the Proposal Network from Mip-NeRF 360 [2]. Algorithm 1 presents the pseudo code for volumetric rendering using the NerfAcc toolbox. In this section, we do not dive into the details of each algorithm, as we basically follow the original paper's implementation. Instead, we discuss the implementation designs that are crucial to maintain high efficiency and flexibility of this toolbox.

Methods	Dataset	Speedup	T (min) ↓	PSNR ↑	LPIPS ↓
TensoRF [10]	T&T	**1.5×**	18.3	28.13	0.143
+ nerfacc (occ)			12.6	28.10	0.150
TensoRF [10]	Syn.	**1.6×**	10.6	32.52	0.047
+ nerfacc (occ)			6.5	32.51	0.044
NeRF [61]	Syn.	**20×**	>1000	31.00	0.081
+ nerfacc (occ)†			52.0	31.55	0.072
Instant-NGP [63]	Syn.	**1.0×**	4.4	32.35	-
+ nerfacc (occ)†			4.4	32.55	0.056
+ nerfacc (prop)†			5.2	31.40	0.064
Instant-NGP [63]	360	**1.1×**	5.3	25.93	-
+ nerfacc (occ)†			5.0	26.41	0.353
+ nerfacc (prop)†			4.9	27.58	0.292

Table 2.2: **Improvements on Static NeRFs.** Experiments are conducted by replacing necessary code in the official repositories, except for those marked by † that are based on our re-implementations.

Methods	Dataset	Speedup	T (min) ↓	PSNR ↑	LPIPS ↓
TiNeuVox [66]	Hyper.	**1.7×**	56.3	24.19	0.425
+ nerfacc (occ)			33.0	24.19	0.434
+ nerfacc (prop)			34.3	24.26	0.398
TiNeuVox [22]	D-NeRF	**2.8×**	11.8	31.14	0.050
+ nerfacc (occ)			4.2	31.75	0.038
K-Planes [25]	D-NeRF	**1.6×**	63.9	30.28	0.043
+ nerfacc (occ)			38.8	30.35	0.042
T-NeRF [72]	D-NeRF	**20×**	>1000	28.78	0.069
+ nerfacc (occ)†			58.0	32.22	0.040

Table 2.3: **Improvements on Dynamic NeRFs.** Experiments are conducted by replacing necessary code in the official repositories, except for those marked by † that are based on our re-implementations.

Methods	Dataset	Speedup	T (min) ↓	PSNR ↑	LPIPS ↓	$\mathbf{E_R}$ / $\mathbf{E_T}$ ↓
BARF [66]	Syn.	**4.5×**	586	28.83	0.054	0.19 / 0.74
+ nerfacc (occ)			130	30.11	0.044	0.07 / 0.35

Table 2.4: **Improvements on NeRFs for Camera Optimization.** Rotation / transnation errors are denoted as E_R/E_T. E_T is scaled by 100.

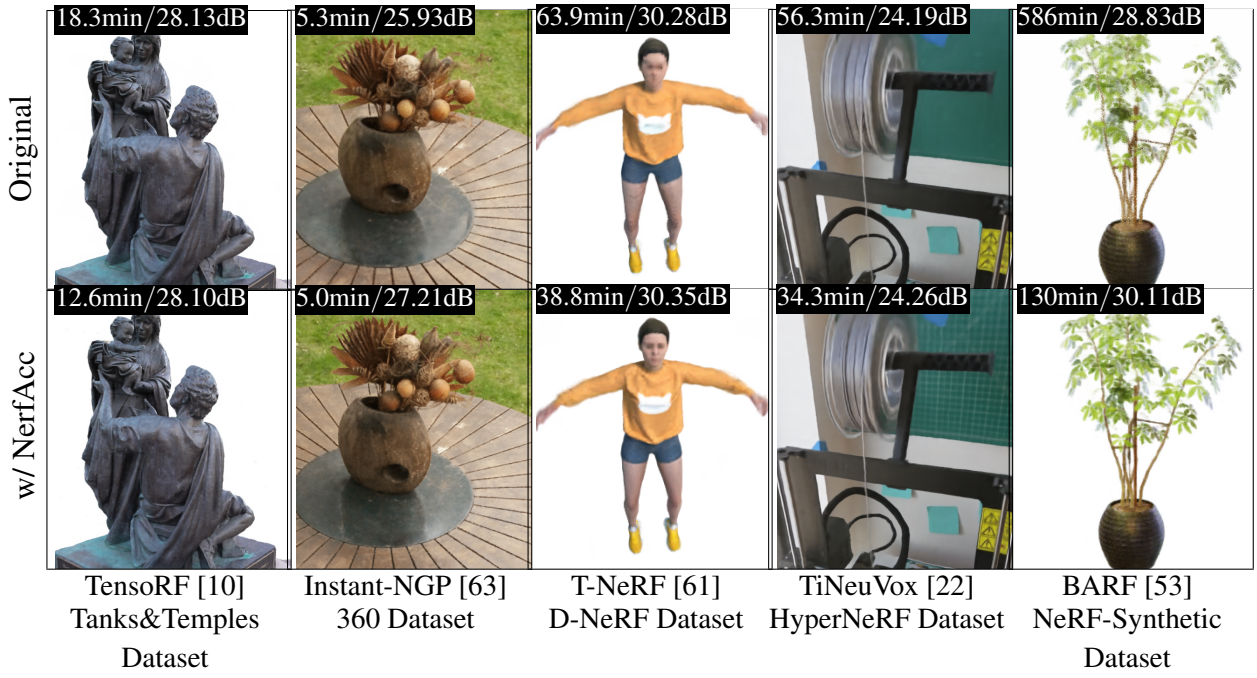


Figure 2.3: **Qualitative Results.** NerfAcc is able to significantly reduce the training time of various NeRF-related methods across multiple datasets, while also yielding slightly improved quality in the majority of cases. The training time and test PSNR are shown on the left corner of each image.

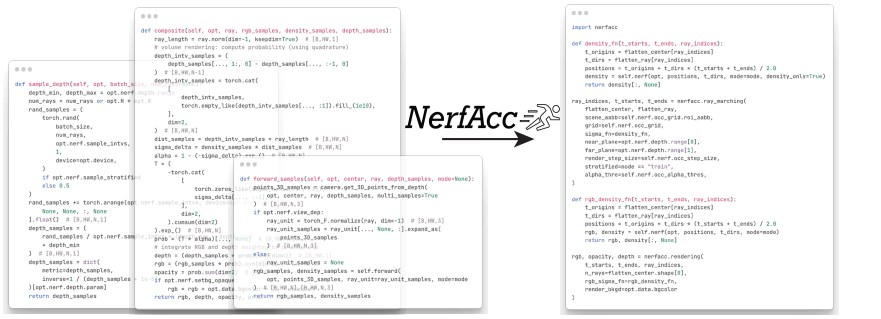


Figure 2.4: **Plug-and-play Example in BARF [53]'s Repository.** With around 50 lines of code change, both training speed and performance can be greatly improved with efficient sampling equipped in NerfAcc, as shown in Table 2.3.

Sample as Interval. In the NerfAcc toolbox, instead of representing each sample with a coordinate x, we represent the sample as an interval along the ray (t_0, t_1, r), where t_0 and t_1 are the start and end of the interval along the r-th ray. This interval-based representation offers three key advantages. Firstly, representing a sample as an interval instead of a single point allows us to support research

based on cone-based rays for anti-aliasing, such as Mip-NeRF [3] and Mip-NeRF 360 [2]. Secondly, since in almost all cases t_i does not require gradients, using (t_0, t_1, r) instead of $(\mathbf{x}_0, \mathbf{x}_1)$ to represent the interval allows for the detachment of the sampling process from the differentiable computational graph, thereby maximizing its speed. Lastly, the ray id r attached to each sample enables support for various numbers of samples across rays with a packed tensor, which we will discuss in the next paragraph. A similar representation has been adopted in Nerfstudio [98] to support various radiance fields.

Packed Tensor. To support sampling with spatial skipping, it is necessary to consider that each ray may result in a different number of valid samples. Storing the data as a tensor with shape `(n_rays, n_samples, ...)` and an extra mask with shape `(n_rays, n_samples)` to indicate which samples are valid leads to significant inefficient memory usage when large portions of space are empty. To address this, in NerfAcc, we represent samples as "packed tensors" with shape `(all_samples, ...)`, in which only valid samples are stored (see Algo. 1). To keep track of the associated rays for each sample, we also host an integer tensor with shape `(n_rays, 2)`, which stores the start index in the packed tensors and the number of samples on this ray. This approach is similar to that used in Instant-NGP [63] and PyTorch3D [76].

No Gradient Filtering. After importance sampling, inaccurate transmittance estimations can result in some samples lying in empty or occluded spaces, particularly in spatial skipping methods like Occupancy Grid. These samples can be filtered before being included in PyTorch's differentiable computation graph by evaluating their transmittance using the radiance field with gradients disabled. As backward passes are not required during filtering, this is much faster ($\sim 10\times$) than keeping all samples in the computation graph. In practice, samples with transmittance below 10^{-4} are disregarded during this process with almost no influence on the rendering quality. Note that this strategy is inspired from Instant-NGP [63]'s implementation.

Case Studies

We showcase the flexibility of NerfAcc on three types of NeRFs across seven papers: static NeRFs (NeRF [61], TensoRF [10], Instant-NGP [63]); dynamic NeRFs (D-NeRF [72], K-Planes [25] TiNeuVox [22]); and a NeRF variation for camera optimization (BARF [53]). Although many of these methods, *e.g.*, Instant-NGP, TensoRF, TiNeuVox and K-Planes, have already been highly optimized for efficiency, we are still able to accelerate their training by a large margin and achieve slightly better performance on nearly all cases. It is worth mentioning that experiments with TensoRF, TiNeuVox, K-Planes and BARF are conducted by integrating NerfAcc into the *official codebase* with around 100 lines of code change. The results of our experiments, including those of our baselines, are presented in Table 2.2,2.3, and2.4, all of which were conducted under the same physical environment using a single NVIDIA RTX A5000 GPU to facilitate comparison, as per [92]. Aside from the experiments reported in this paper, NerfAcc has also been integrated into

a few popular open-source projects such as `nerfstudio` [98] for density-based NeRFs, as well as `sdfstudio` [126] and `instant-nsr-pl` [32] for SDF-based NeRFs.

Static NeRFs. In this task, we experiment with three NeRF vaiants, including the original MLP-based NeRF [61], TensoRF [10] and Instant-NGP [63]. We show that NerfAcc works with both MLP-based and Voxel-based radiance fields, on both bounded (NeRF-Synthetic dataset [61], Tank&Template dataset [44]) and unbounded scenes (360 Dataset [2]). It is worth to note that with NerfAcc, one can train an Instant-NGP model with pure Python code and achieve slightly better performance than the official pure CUDA implementation, as shown in Table 2.2.

Dynamic NeRFs. In this task, we apply the NerfAcc toolbox to T-NeRF [72], K-Planes [25] and TiNeuVox [22], covering both the synthetic (D-NeRF [72]) and "in-the-wild" captures[1] (that accompany HyperNeRF [66]). When applying the occupancy grid approach to accelerate those dynamic methods, instead of representing a static scene with it, we share the occupancy grid across all frames. In other words, instead of using it to indicate the opacity of an area at a single timestamp, We use it to indicate the *maximum opacity at this area over all the timestamps*. This is not optimal but still makes the rendering very efficient, due to the fact there are limited movements in these datasets.

NeRFs for Camera Optimization. In this task, we employed the NerfAcc toolbox to BARF [53] on the NeRF-Synthetic dataset with perturbed cameras. The goal is to jointly optimize the radiance field and camera extrinsic for multi-view images. We observed that the spatial skipping sampling provided by NerfAcc facilitated faster training and significantly improved both image quality and camera pose reconstruction. These improvements could be attributed to the sparsity enforced in our sampling procedure. This finding may provide interesting avenues for future research.

Analysis of Different Sampling Approaches. Results in Table 2.2 show that the choice between occupancy grid and proposal network sampling can noticeably affect run-time and performance on different datasets. As each approach relies on a distinct set of hyperparameters, a systematic comparison between the two methods is crucial by sweeping the hyperparameter space. We varied the resolution and marching step size for occupancy grid and the number of samples and size of the proposal network for the proposal network approach. We plot pareto curves for each approach for both the NeRF-Synthetic and Mip-NeRF 360 datasets in Fig.2.5. This analysis indicates that occupancy grid sampling is suitable for the NeRF-Synthetic dataset, whereas the proposal network approach performs better on the 360 dataset. This is likely because the NeRF-Synthetic dataset contains more empty space that can be skipped effectively using the occupancy grid approach. However, in the case of real, unbounded data, the use of the occupancy grid approach is limited by the bounding box and the lack of empty space to skip, making the proposal network approach more effective. These experiments used the radiance field from Instant-NGP [63], with the same training recipes.

[1]These datasets teleport cameras and do not represent real captures [27].

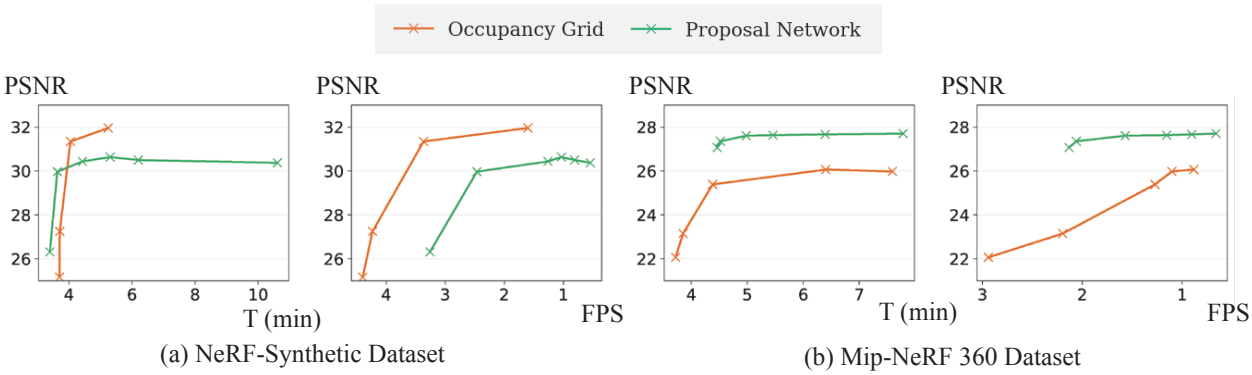


Figure 2.5: **Comparison between Different Sampling Approaches in terms of Training Time and Rendering FPS.** We sweep the hyper-parameters space for each sampling approach, and find out that occupancy grid approach gives the best performance-speed trade-off on the NeRF-Synthetic dataset, while the proposal network approach performs the best on the Mip-NeRF 360 dataset. Note T (min) denotes for *training* time and FPS is for *rendering* frames per second. All experiments use the HashEncoding from Instant-NGP [63] as the radiance field representation. Please see the supplementary materials for the hyperparameter space that we explored.

Combined Sampling

A benefit from the unified concept of transmittance estimator introduced in Section 2.3, is that it's straight-forward to combine the two distinct sampling approaches, as both of them essentially provide an estimation of the transmittance that can be used for importance sampling. For example, we find that simply stacking an occupancy grid on top of the proposal network, can significantly reduce the number of rays and shrink the near-far plane for the remaining rays on the NeRF-Synthetic dataset. This leads to improvements in quality, from 31.40dB to 32.35dB, and a reduction in training time, from 5.2min to 4.3min, compared to using only the proposal network for importance sampling. Figure 2.6 shows an example with the FICUS scene, where the floaters are cleaned out with the combined sampling. This experiment is conducted using the HashEncoding from Instant-NGP [63] as the radiance field representation.

2.5 Conclusions

In conclusion, this paper highlights the significant impact of advanced sampling approaches on improving the efficiency of Neural Radiance Fields (NeRF) optimization and rendering. We demonstrate that advanced sampling can significantly speed up the training of various recent NeRF papers, while maintaining high-quality results. The development of NerfAcc, a flexible Python toolbox, enables researchers to incorporate advanced sampling methods into NeRF-related methods easily. The exploration and comparison of advanced sampling methods are important steps towards developing more efficient and accessible NeRF-based methods. The presented results also

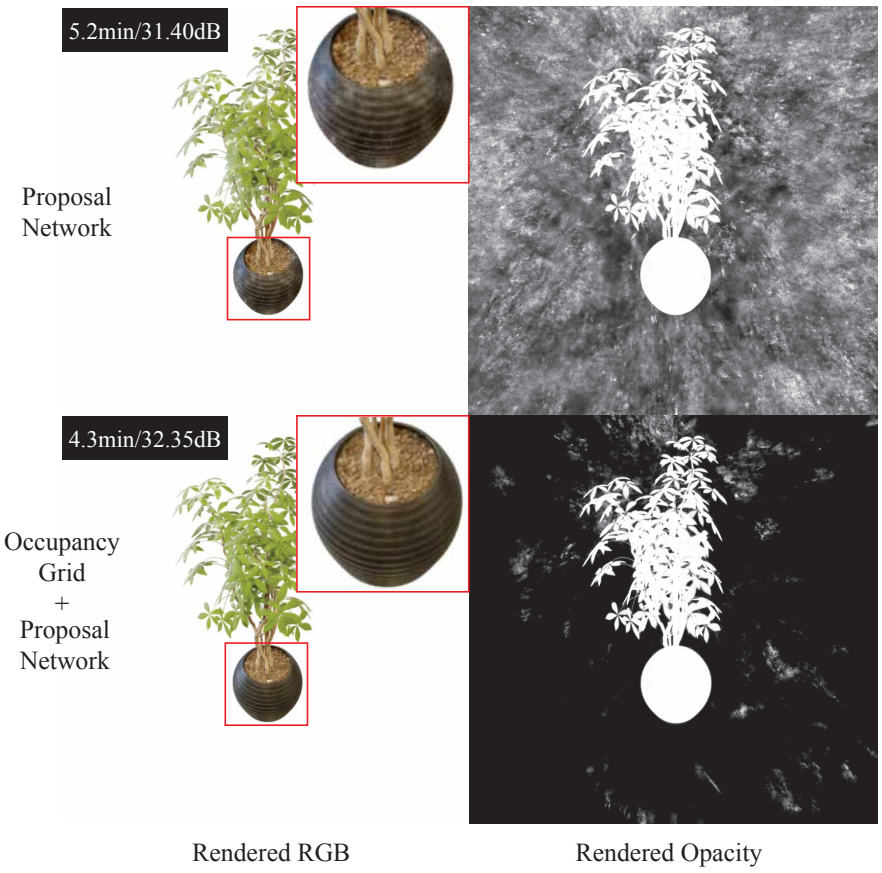


Figure 2.6: **Results of Combined Sampling on NeRF-Synthetic Dataset.** Benefit from our unified concept of transmittance estimator, we find it's straight-forward to combine the two distinct sampling approaches.

demonstrate the potential for further research to improve the performance of NeRF and other related techniques through advanced sampling strategies.

Chapter 3

NeRF-XL: A Efficient Multi-GPU Framework for NeRFs

We present NeRF-XL, a principled method for distributing Neural Radiance Fields (NeRFs) across multiple GPUs, thus enabling the training and rendering of NeRFs with an arbitrarily large capacity. We begin by revisiting existing multi-GPU approaches, which decompose large scenes into multiple independently trained NeRFs [96, 101, 58], and identify several fundamental issues with these methods that hinder improvements in reconstruction quality as additional computational resources (GPUs) are used in training. NeRF-XL remedies these issues and enables the training and rendering of NeRFs with an arbitrary number of parameters by simply using more hardware. At the core of our method lies a novel distributed training and rendering formulation, which is mathematically equivalent to the classic single-GPU case and minimizes communication between GPUs. By unlocking NeRFs with arbitrarily large parameter counts, our approach is the first to reveal multi-GPU scaling laws for NeRFs, showing improvements in reconstruction quality with larger parameter counts and speed improvements with more GPUs. We demonstrate the effectiveness of NeRF-XL on a wide variety of datasets, including the largest open-source dataset to date, MatrixCity [51], containing 258K images covering a 25km^2 city area. Visit our webpage at `https://research.nvidia.com/labs/toronto-ai/nerfxl/` for code and videos.

3.1 Introduction

Recent advances in novel view synthesis have greatly enhanced our ability to capture Neural Radiance Fields (NeRFs), making the process significantly more accessible. These advancements enable the reconstruction of both larger scenes and finer details within a scene. Expanding the scope of a captured scene, whether by increasing the spatial scale (e.g., capturing a multi-kilometer-long cityscape) or the level of detail (e.g., scanning the blades of grass in a field), involves incorporating a greater volume of information into the NeRF for accurate reconstruction. Consequently, for scenes with high information content, the number of trainable parameters required for reconstruction may exceed the memory capacity of a single GPU.

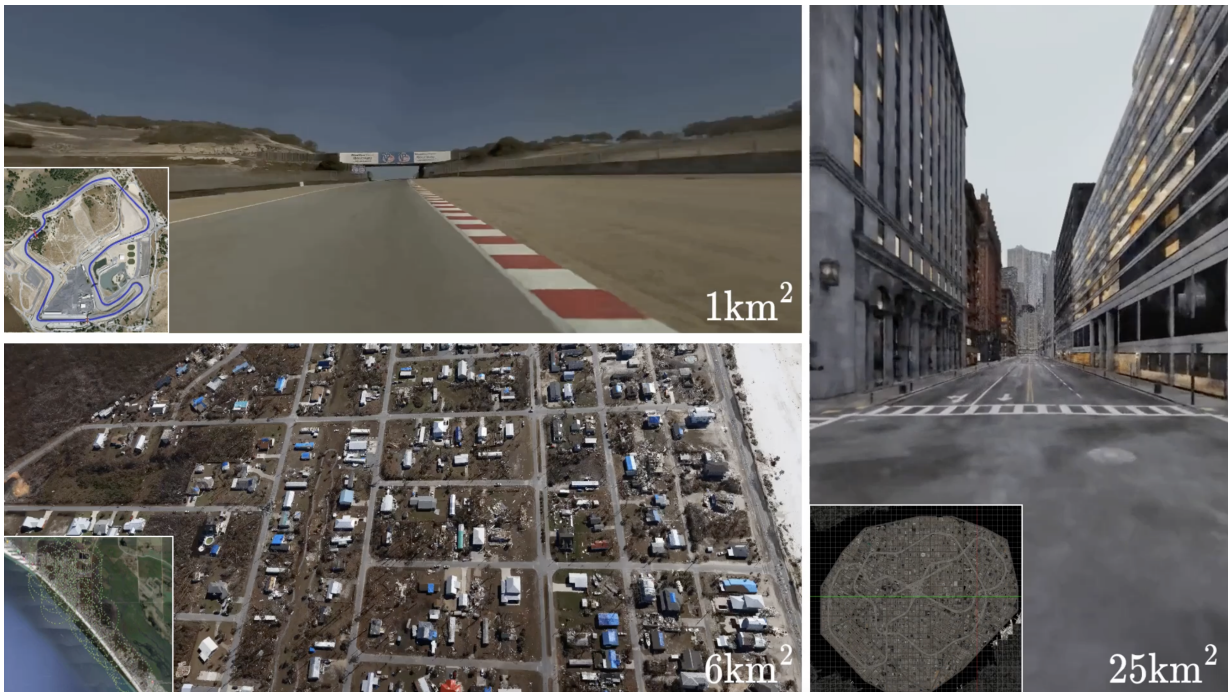


Figure 3.1: **Extremely large scale NeRFs reconstructed by our principled multi-GPU distributed algorithm.**

In this paper, we introduce NeRF-XL, a principled algorithm for efficiently distributing Neural Radiance Fields (NeRFs) across multiple GPUs. Our method enables the capture of high-information-content scenes, including those with large-scale and high-detail features, by simply adding more hardware resources. At its core, NeRF-XL allocates NeRF parameters across a disjoint set of spatial regions and trains them jointly across GPUs. Unlike conventional distributed training pipelines that synchronize gradients during the backward pass, our approach only requires information synchronization during the forward pass. Additionally, we drastically reduce the required data transfer between GPUs by carefully rewriting the volume rendering equation and relevant loss terms for the distributed setting. This novel rewriting enhances both training and rendering efficiency. The flexibility and scalability of our approach allows us to efficiently optimize NeRFs with an arbitrary number of parameters using multiple GPUs.

Our work contrasts with recent approaches that utilize multi-GPU algorithms to model large-scale scenes by training a set of independent NeRFs [96, 101, 58]. While these approaches require no communication between GPUs, each NeRF needs to model the entire space, including the background region. This leads to increased redundancy in the model's capacity as the number of GPUs grows. Additionally, these methods require blending NeRFs during rendering, which degrades visual quality and introduces artifacts in overlapping regions. Consequently, unlike NeRF-XL, these methods fail to achieve visual quality improvements as more model parameters (equivalent to more

GPUs) are used in training.

We demonstrate the effectiveness of our method across a diverse set of captures, including street scans, drone flyovers, and object-centric videos. These range from small scenes ($10m^2$) to entire cities ($25km^2$). Our experiments show that NeRF-XL consistently achieves improved visual quality (measured by PSNR) and rendering speed as we allocate more computational resources to the optimization process. Thus, NeRF-XL enables the training of NeRFs with arbitrarily large capacity on scenes of any spatial scale and detail.

3.2 Related Work

Single GPU NeRFs for Large-Scale Scenes Many prior works have adapted NeRF to large-scale outdoor scenes. For example, BungeeNeRF [114] uses a multi-scale, coarse-to-fine pipeline to address memory constraints; Grid-guided NeRF [117] uses multiple image planes for drone-captured footage; F2-NeRF [107] introduces a space warping algorithm for efficient level-of-detail handling in a free camera trajectory capture; and UrbanNeRF [80] leverages LiDAR and segmentation maps to improve in-the-wild captures. Despite their advancements, these prior works are bounded by the computational capacity of a single GPU.

NeRFs with Multiple GPUs An alternative approach for training NeRFs on large-scale scenes is to use multiple GPUs. BlockNeRF [96], MegaNeRF [102] and SNISR [113] partition a scene into overlapping NeRFs based on camera trajectory or spatial content, and optimize each NeRF independently (one per GPU). ProgressiveNeRF [58] adopts a similar strategy but recursively optimizes one NeRF at a time with overlapped blending. While these methods overcome the memory limitations of a single GPU, each independent NeRF has to model the entire scene within a spatial region, leading to increased redudancy (in the model's capacity) and decreased visual quality as more GPUs are used in training. Furthermore, these methods must rely on depth initialization for spatial partitioning [113], or introduce overlapping between NeRFs [96, 102, 58], which causes visual artifacts during rendering. We provide an in-depth analysis of the problems faced by prior multi-GPU methods in the next section.

3.3 Revisiting Existing Approaches: Independent Training

In leveraging multiple GPUs for large-scale captures, prior research [57, 96, 58] has consistently employed the approach of training multiple NeRFs focusing on different spatial regions, where each NeRF is trained independently on its own GPU. *However, independently training multiple NeRFs has fundamental issues that impede visual-quality improvements with the introduction of additional resources (GPUs).* This problem is caused by three main issues described below.

Model Capacity Redundancy. The objective of training multiple independent NeRFs is to allow each NeRF to focus on a different (local) region and achieve better quality within that region than a

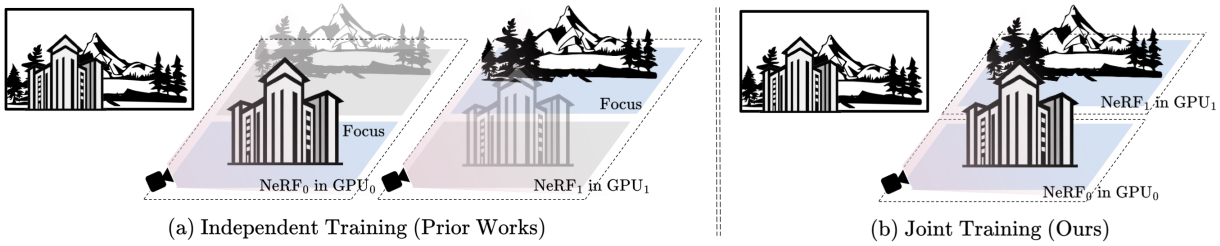


Figure 3.2: **Independent Training v.s. Joint Training with multi-GPU.** Training multiple NeRFs independently [96, 102, 58] requires each NeRF to model both the focused region and its surroundings, leading to redundancy in model's capacity. In contrast, our joint training approach utilizes non-overlapping NeRFs, thus without any redundancy.

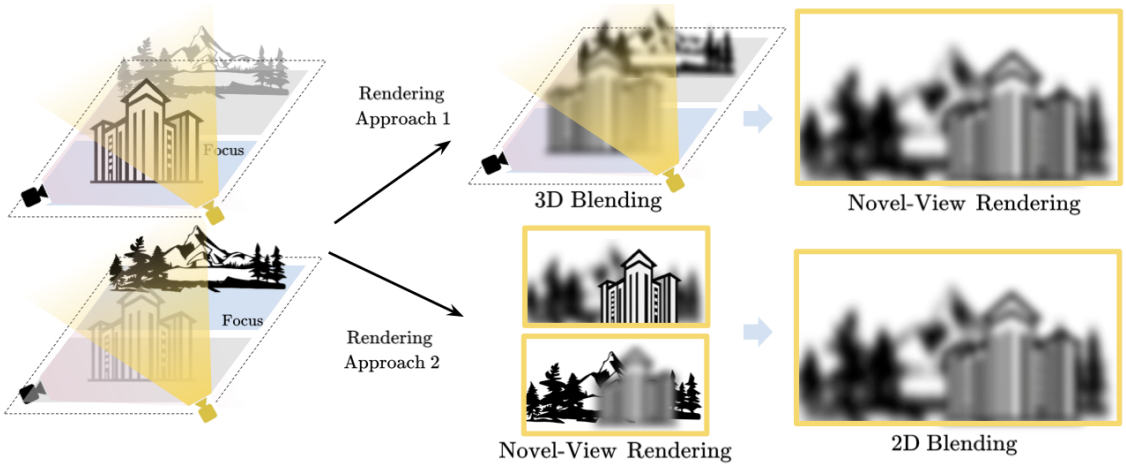


Figure 3.3: **Independent Training requires Blending for Novel-View Synthesis.** Either blending in 2D [96, 58] or 3D [102] introduces blurriness into the rendering.

single global model with the same capacity. Despite this intention, each NeRF is compelled to model not only its designated region but also the surrounding areas, since training rays often extend beyond the designated region as depicted in Figure 3.2(a). This leads to an inherent redundancy in the model's capacity since each NeRF must model both the local and surrounding regions. As a result, increasing the number of GPUs (and hence using smaller spatial regions per NeRF), increases the total redundancy in the model's capacity. For example, Mega-NeRF [102] exhibits 38%/56%/62% ray samples outside the tiled regions with $2\times/4\times/8\times$ tiles on the UNIVERSITY4 capture. In contrast, our proposed method of jointly training all tiles removes the need for surrounding region modeling in each NeRF, *thereby completely eliminating redundancy*, as shown in Figure 3.2(b)). This feature is crucial for efficiently leveraging additional computational resources.

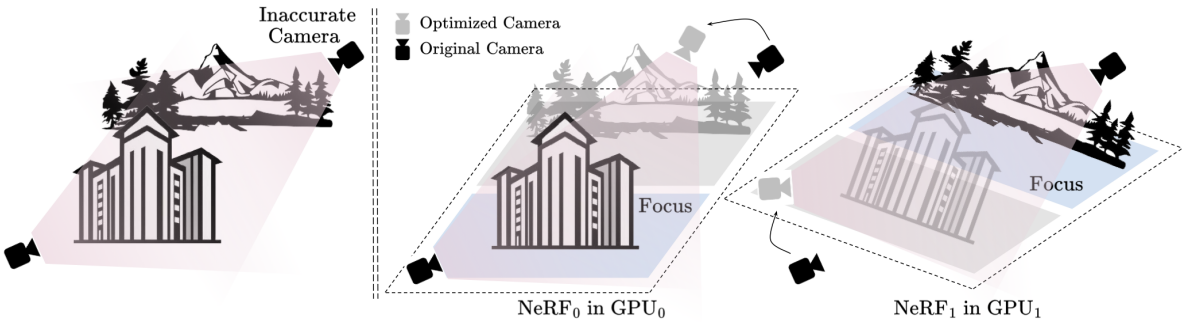


Figure 3.4: **Independent Training Creates Distinct Camera Optimizations.** Camera optimization in NeRF can be achieved by either transforming the inaccurate camera itself or all other cameras along with the underlying 3D scene. Thus, training multiple NeRFs independently with camera optimization may lead to inconsistencies in camera corrections and scene geometry, causing more difficulties for blended rendering.

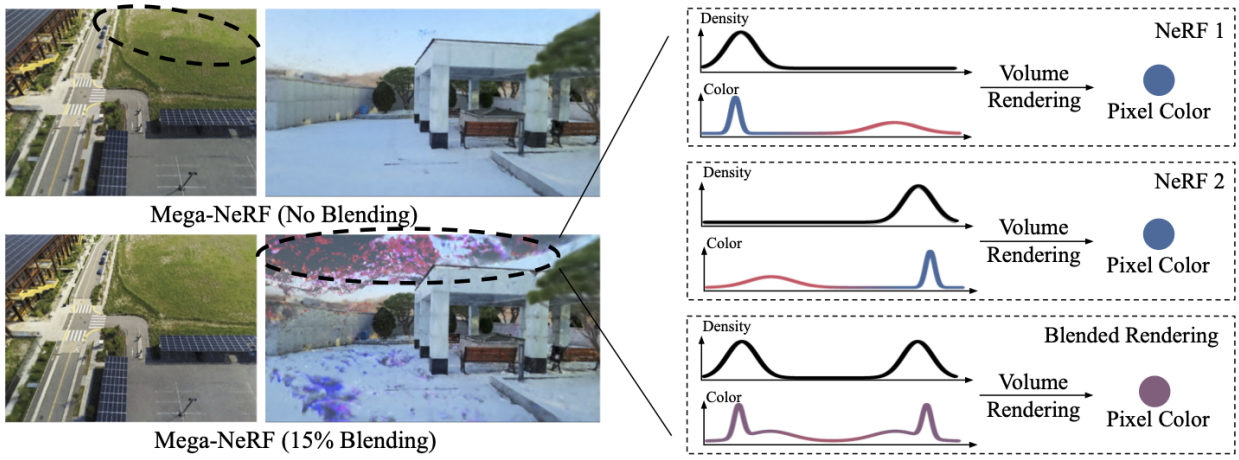


Figure 3.5: **Potential Artifacts Caused by 3D Blending.** On the left we show Mega-NeRF results trained with 2 GPUs. At 0% overlap, boundary artifacts appear due to independent training, while at 15% overlap, severe artifacts appear due to 3D blending. On the right we illustrate the reason behind this artifact: while each independently trained NeRF renders the correct color, the blended NeRF do not guarantee correct color rendering.

Blending for Rendering. When rendering independently trained NeRFs, it is often necessary to employ a blending strategy to merge the NeRFs and mitigate inconsistencies at the region boundaries. Past works typically choose local regions with a certain degree of overlap, such as 50% in Block-NeRF [96] and 15% in Mega-NeRF [102]. Two primary approaches exist for blending NeRFs during novel-view synthesis. One approach involves rendering each NeRF independently and then blending the resulting images when the camera is positioned within the overlapped region

(referred to as 2D blending) [96, 58]. The alternative approach is to blend the color and density in 3D for ray samples within the overlapped region (referred to as 3D blending) [102]. As illustrated in Figure 3.3, both approaches can introduce blur into the final rendering. Moreover, blending in 3D can lead to more pronounced artifacts in rendering, due to deviations in the volume rendering equation, as demonstrated in Figure 3.5. In contrast, our joint training approach does not rely on any blending for rendering. In fact, our method renders the scene in the exact same way during training and inference, thereby eliminating the train-test discrepancies introduced by past methods.

Inconsistent Per-camera Embedding. In many cases, we need to account for things like white balance, auto-exposure, or inaccurate camera poses in a capture. A common approach to model these factors is by optimizing an embedding for each camera during the training process, often referred to as appearance embedding or pose embedding [98, 56, 53]. However, when training multiple NeRFs independently, each on its own GPU, the optimization process leads to independent refinements of these embeddings. This can result in inconsistent camera embeddings due to the inherently ambiguous nature of the task, as demonstrated in Figure 3.4. Inconsistencies in appearance embeddings across NeRFs can result in disparate underlying scene colors, while inconsistencies in camera pose embeddings can lead to variations in camera corrections and the transformation of scene geometry. These disparities introduce further difficulties when merging the tiles from multiple GPUs for rendering. Conversely, our joint training approach allows optimizing a single set of per-camera embeddings (through multi-GPU synchronization), thus completely eliminating these issues.

Due to the issues listed above, prior works [96, 102] which train multiple independent NeRFs do not effectively harness the benefits of additional computational resources (GPUs) as they scale up, as evidenced in our experiments (§ 3.5). As a result, we advocate for a novel *joint* training approach. Without any heuristics, our approach gracefully enhances both visual quality and rendering speed as more GPUs are used in training, Moreover, our method reveals the multi-GPU scaling laws of NeRF for the first time.

3.4 Our Method: Joint Training

Background

Volume Rendering NeRFs [60] employ volume rendering to determine the pixel color through the integral equation:

$$\begin{aligned} C(t_n \to t_f) &= \int_{t_n}^{t_f} T(t_n \to t)\sigma(t)c(t)\,dt, \\ \text{where} \quad T(t_n \to t) &= \exp\left(-\int_{t_n}^{t} \sigma(s)\,ds\right). \end{aligned} \tag{3.1}$$

Here, $T(t_n \to t)$ represents transmittance, $\sigma(t)$ denotes density, and $c(t)$ signifies the color at position t along the ray.

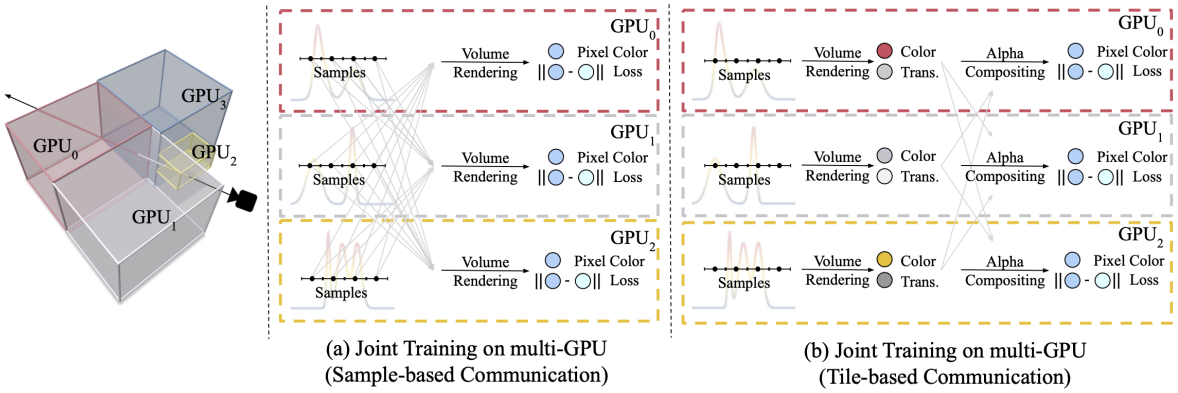


Figure 3.6: **Our Training Pipeline.** Our method jointly trains multiple NeRFs across all GPUs, each of which covers a disjoint spatial region. The communication across GPUs only happens in the forward pass but not the backward pass (shown in gray arrows). (a) We can train this system by evaluating each NeRF to get the sample color and density, then broadcast these values to all other GPUs for a global volume rendering (§ 3.4). (b) By rewriting volume rendering equation we can dramatically reduce the data transfer to one value per-ray, thus improving efficiency (§ 3.4).

Distortion Loss Initially introduced in Mip-NeRF 360 [2] and validated in subsequent works [4, 98, 49], this loss acts as a regularizer to address "floater" artifacts in NeRF reconstructions caused by limited training viewpoint coverage. It is calculated along a ray as

$$\mathcal{L}_{dist}(t_n \to t_f) = \int_{t_n}^{t_f} w(t_i) w(t_j) \left| t_i - t_j \right| \, dt_i \, dt_j, \tag{3.2}$$

where $w(t) = T(t_n \to t)\sigma(t)$ represents the volume rendering weight for each sample along the ray. Intuitively, it penalizes floaters by encouraging density concentration in minimal, compact regions. See [2] for more details.

Non-overlapped NeRFs

A straightforward strategy to increase model capacity with multiple GPUs is to partition 3D space into tiles and allocate a NeRF for each tile. But unlike prior works [102, 96, 58] that employ overlapped NeRFs to model both tiles and their surrounding regions, our method exclusively models *non-overlapped* tiles, with each NeRF assigned to a single tile. This distinction is illustrated in Figure 3.2.

To render our NeRFs across multiple GPUs, we first distribute ray samples among GPUs based on the bounding box of the tiles. Notably it's important to ensure that sample intervals do not extend beyond tile boundaries to prevent overlap between samples. We subsequently query sample attributes (i.e., color and density) on each respective GPU. Volume rendering is then performed through a global gather operation, consolidating information across all GPUs onto a single GPU

to compute the final pixel color. Since all sample intervals are non-overlapping, the scene can be rendered accurately following the volume rendering equation without the need for any blending.

Training proceeds in a similar fashion to rendering, except that during the forward pass *each* GPU performs the global gather operation (i.e., broadcast) to obtain the information (i.e., color and density) from all other GPUs (illustrated as gray lines in Figure 3.6(a)). Then, each GPU computes the loss locally and back-propagates the gradients to its own parameters. Notably the forward pass produces the exact same loss values on every GPU, but each loss lives in a different computational graph that only differentiates with respect to its own local parameters, thus no gradient communication are required across GPUs.

Such a naive scheme is extremely simple to implement, and mathematically identical to training and rendering a NeRF represented by multiple small NeRFs [78, 77] on a single large GPU. Distributing learnable parameters and computational cost across multiple GPUs allows scaling NeRF to scenes of any size, as well as making most parts of training fully parallel (*e.g.*, network evaluation, back-propagation). Despite its simplicity and scalability in comparison to blending overlapping NeRFs in prior works [96, 102, 58], this naive approach requires synchronizing $\mathcal{O}(SK^2)$ data across GPUs, where K is the number of GPUs, and S is the average number of samples per-ray per-GPU. As the number of GPUs increases or the ray step size decreases, synchronization across GPUs quickly becomes a bottleneck. Therefore, on top of this approach, we present a sophisticated solution that significantly alleviates the burden of multi-GPU synchronization in a principled manner.

Partition Based Volume Rendering

If we consider the near-far region $[t_n \to t_f]$ consisting of N segments $[t_1 \to t_2, t_2 \to t_3, ..., t_N \to t_{N+1}]$, we can rewrite the volume-rendering integral (3.1) into a sum of integrals for each segment along the ray:

$$C(t_1 \to t_{N+1}) = \int_{t_1}^{t_{N+1}} T(t_1 \to t)\sigma(t)c(t)dt = \sum_{k=1}^{N} T(t_1 \to t_k)C(t_k \to t_{k+1}) \tag{3.3}$$

in which the transmittance $T(t_1 \to t_k)$ can be written as:

$$T(t_1 \to t_k) = \prod_{i=1}^{k-1} T(t_i \to t_{i+1}) \tag{3.4}$$

The above equation states that volume rendering along an entire ray is equivalent to first performing volume rendering independently within each segment, then performing alpha compositing on all the segments. We can also rewrite the accumulated weights $A(t_1 \to t_{N+1})$ and depths $D(t_1 \to t_{N+1})$ in a similar way:

$$A(t_1 \to t_{N+1}) = \int_{t_1}^{t_{N+1}} T(t_1 \to t)\sigma(t)dt = \sum_{k=1}^{N} T(t_1 \to t_k)A(t_k \to t_{k+1}) \tag{3.5}$$

$$D(t_1 \to t_{N+1}) = \int_{t_1}^{t_{N+1}} T(t_1 \to t)\sigma(t)tdt = \sum_{k=1}^{N} T(t_1 \to t_k)D(t_k \to t_{k+1}) \tag{3.6}$$

We can further rewrite the point-based integral in the distortion loss as an accumulation across segments:

$$\begin{aligned}\mathcal{L}_{dist}(t_1 \to t_{N+1}) &= \int_{t_1}^{t_{N+1}} w(t_i)w(t_j)\left|t_i - t_j\right| dt_i dt_j \\ &= 2\sum_{k=1}^{N} T(t_1 \to t_k)S(t_1 \to t_k) + \sum_{k=1}^{N} T(t_1 \to t_k)^2 \mathcal{L}_{dist}(t_k \to t_{k+1})\end{aligned} \tag{3.7}$$

in which the $S(t_1 \to t_k)$ is defined as:

$$S(t_1 \to t_k) = D(t_k \to t_{k+1})A(t_1 \to t_k) - A(t_k \to t_{k+1})D(t_1 \to t_k) \tag{3.8}$$

Intuitively, the first term $S(t_1 \to t_k)$ penalizes multiple peaks across segments (zero if only one segment has non-zero values), while the second term $\mathcal{L}_{dist}(t_k \to t_{k+1})$ penalizes multiple peaks within the same segment. This transforms the pairwise loss on all samples into a hierarchy: pairwise losses within each segment, followed by a pairwise loss on all segments. Derivations for all the above formulae are given in the appendix.

Recall that the main drawback of our naive approach was an expensive per-sample data exchange across all GPUs. The above formulae convert sample-based composition to tile-based composition. This allows us to first reduce the per-sample data into per-tile data in parallel within each GPU and exchange only the per-tile data across all GPUs for alpha compositing. This operation is cost-effective, as now the data exchange is reduced from $O(KS^2)$ to $O(S^2)$ (each GPU contains a single tile). Figure 3.6(b) shows an overview of our approach. § 3.5 quantifies the improvement gained from this advanced approach compared to the naive version.

In addition to the volume rendering equation and distortion loss, a wide range of loss functions commonly used in NeRF literature can be similarly rewritten to suit our multi-GPU approach. For further details, we encourage readers to refer to the appendix.

Spatial Partitioning

Our multi-GPU strategy relies on spatial partitioning, raising the question of how to create these tiles effectively. Prior works [96, 102] opt for a straightforward division of space into uniform-sized blocks within a global bounding box. While suitable for near-rectangular regions, this method proves suboptimal for free camera trajectories and can lead to unbalanced compute assignment across GPUs. As noted in [107], a free camera trajectory involves uneven training view coverage, resulting in varying capacity needs across space (*e.g.*, the regions that are far away from any camera require less capacity than regions near a camera). To achieve balanced workload among GPUs, we want to ensure each GPU runs a similar number of network evaluations (i.e., has a similar number of

	Garden [2]	University4 [58]	Building [102]	Mexico Beach [14]	Laguna Seca	MatrixCity [51]
#Img	161	939	1940	2258	27695	258003
#Pix$_c$	175M	1947M	1920M	2840M	47294M	25800M
#Pix$_d$	0.84M	3.98M	-	9.63M	2819M	2007M

Table 3.1: **Data Statistics.** Our experiments are conducted on these captures from various sources, including street captures (UNIVERSITY4, MATRIXCITY, LAGUNA SECA), aerial captures (BUILDING, MEXICO BEACH) and an object-centric 360-degree capture (GARDEN). These data span a wide range of scales, enabling a comprehensive evaluation of the multi-GPU system. Pix$_c$ and Pix$_d$ are denoted for color pixels and depth pixels, respectively.

ray samples). This balance not only allocates compute resources evenly but also minimizes waiting time during multi-GPU synchronization for communicating the data, as unequal distribution can lead to suboptimal GPU utilization.

We propose an efficient partitioning scheme aimed at evenly distributing workload across GPUs. When a sparse point cloud is accessible (*e.g.*, obtained from SFM), we partition the space based on the point cloud to ensure that each tile contains a comparable number of points. This is achieved by recursively identifying the plane where the Cumulative Distribution Function (CDF) equals 0.5 for the 3D point distribution along each axis. As a result, this approach leads to approximately evenly distributed scene content across GPUs. In cases where a sparse point cloud is unavailable, indicating a lack of prior knowledge about the scene structure, we instead discretize randomly sampled training rays into 3D samples. This serves as an estimation of the scene content distribution based on the camera trajectory, enabling us to proceed with partitioning in a similar manner. This process is universally applicable to various types of captures, including street, aerial, and object-centric data, and runs very quickly in practice (typically within seconds). Please refer to the appendix for visualizations of partitioned tiles on different captures.

3.5 Experiments

Datasets. The crux of a multi-GPU strategy lies in its ability to consistently improve performance across all types of captures, regardless of scale, as additional resources are allocated. However, prior works typically evaluate their methods using only a single type of capture (*e.g.*, street captures in Block-NeRF, aerial captures in Mega-NeRF). In contrast, our experiments are conducted on diverse captures from various sources, including street captures (UNIVERSITY4 [58], MATRIXCITY [51], LAGUNA SECA[1]), aerial captures (BUILDING [102], MEXICO BEACH [14]) and an object-centric 360-degree capture (GARDEN [2]). These data also span a wide range of scales, from GARDEN with 161 images in a 10m^2 area, to MATRIXCITY with 258K images in a 25km^2 area, thereby offering a comprehensive evaluation of the multi-GPU system. Table 3.1 shows detailed statistics for each of these captures.

[1]Laguna Seca: An in-house capture of a 3.6km race track.

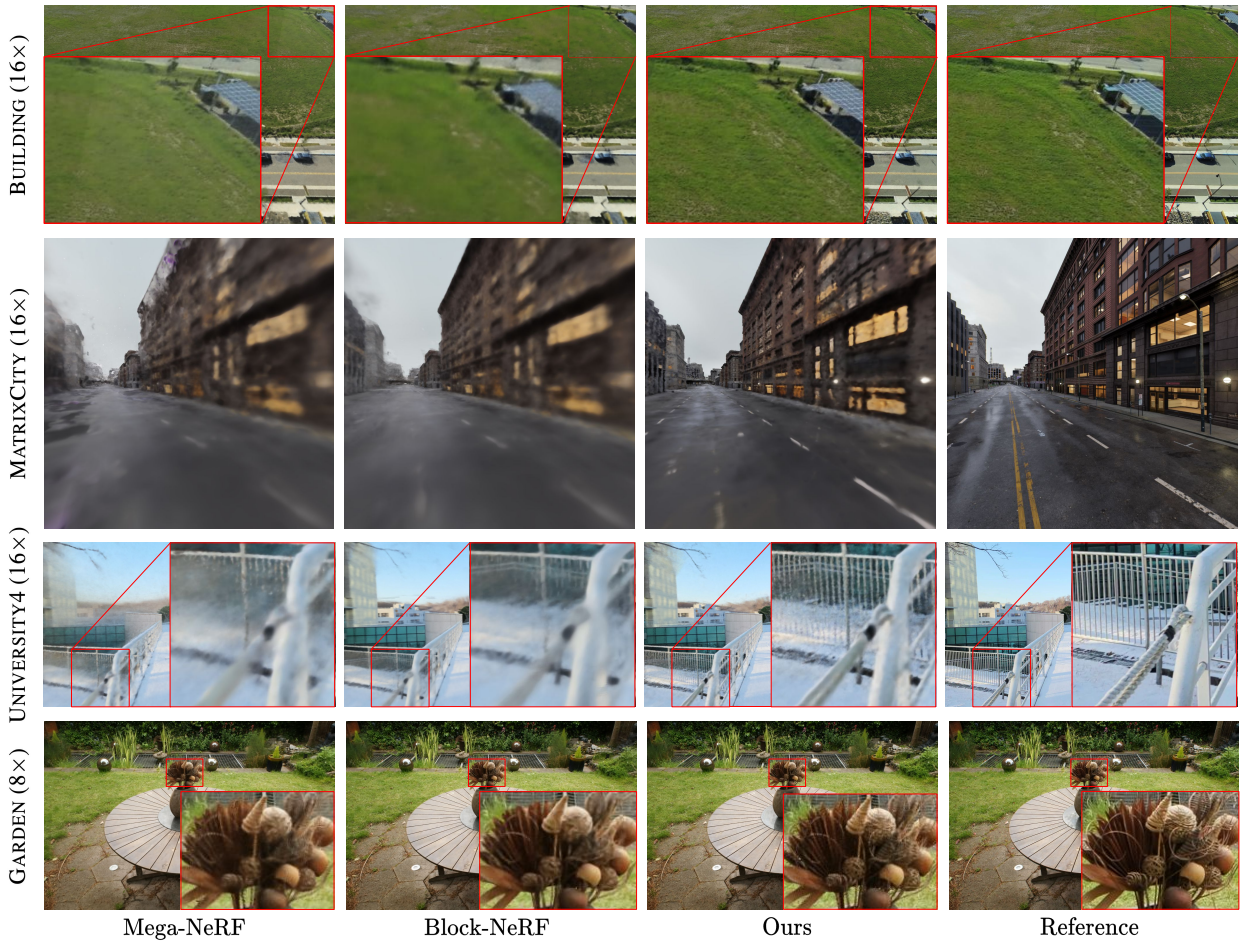


Figure 3.7: **Qualitative Comparison.** Comparing to prior works, our method efficiently harnesses multi-GPU setups for performance improvement on all types of data.

Joint Training v.s. Independent Training

In this section, we conduct a comparative analysis between our proposed approach and two prior works, Block-NeRF [96] and Mega-NeRF [102], all of which are aimed at scaling up NeRFs beyond the constraints of a single GPU. To ensure a fair evaluation solely on the multi-GPU strategies, we re-implemented each baseline alongside our method within a unified framework[2]. Each method is configured with the same NeRF representation (Instant-NGP [63]), spatial skipping acceleration structure (Occupancy Grid [63]), distortion loss [2], and multi-GPU parallel inference. This standardized setup allows us to focus on assessing the performance of different multi-GPU strategies in both training (i.e.,, joint vs. independent [96, 102]) and rendering (i.e.,, joint vs. 2D blending [96] vs. 3D blending [102]). For each baseline method, we adopt their default overlapping

[2]On BUILDING scene, our 8 GPU Mega-NeRF implementation achieves 20.8 PSNR comparing to 20.9 PSNR reported in the original paper.

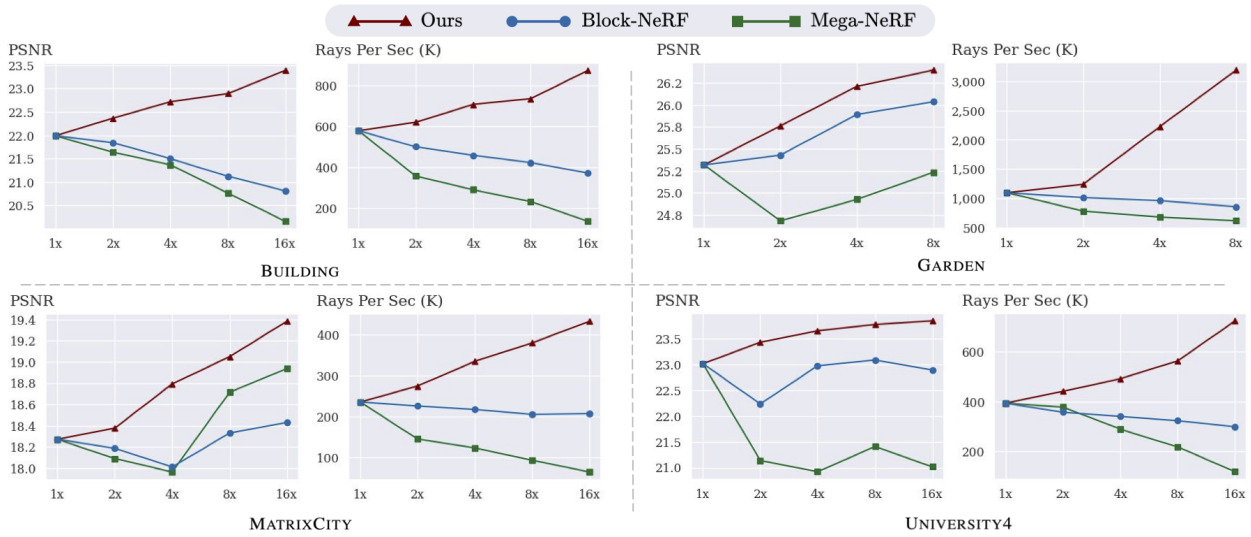


Figure 3.8: **Quantitative Comparison.** Prior works based on independent training fails to realize performance improvements with additional GPUs, while our method enjoys improved rendering quality and speed as more resources are added to training.

configurations, which is 15% for Mega-NeRF and 50% for Block-NeRF. All methods are trained for the same number of iterations (20K), with an equal number of total samples per iteration (effectively the batch size of the model). Please refer to the appendix for implementation details.

In this section we conduct experiments on four captures, including GARDEN [2], BUILDING [102], UNIVERSITY4 [58] and MATRIXCITY [51], with GPU configurations ranging from $1\times$ to $16\times$ (multi-node). We evaluate the scalability of each method using two key metrics: Peak Signal-to-Noise Ratio (PSNR) for quality assessment and Rays Per Second for rendering speed, on the respective test sets of each capture. As illustrated in Figure 3.8, baseline approaches struggle to improve rendering quality with an increase in the number of GPUs, largely due to the inherent issues associated with independent training outlined in § 3.3. Additionally, baseline methods also fails to achieve faster rendering with additional GPUs, as they either need to evaluate duplicate pixels for 2D blending [96] or duplicate 3D samples for 3D blending [102]. In contrast, our proposed approach, employing joint training and rendering, effectively eliminates model redundancy and train-test discrepancy. Thus, it gracefully benefits from increased parameters and parallelization with additional GPUs, resulting in nearly linear improvements in both quality and rendering speed. More qualitative comparisons are shown in Figure 3.7.

Robustness and Scalability

We further evaluate the robustness and scalability of our approach by testing it on larger scale captures with increased GPU resources. Specifically, Figure 3.10 showcases our novel-view rendering results on the 1km^2 LAGUNA SECA with 8 GPUs, the 6km^2 MEXICO BEACH [14] with 8

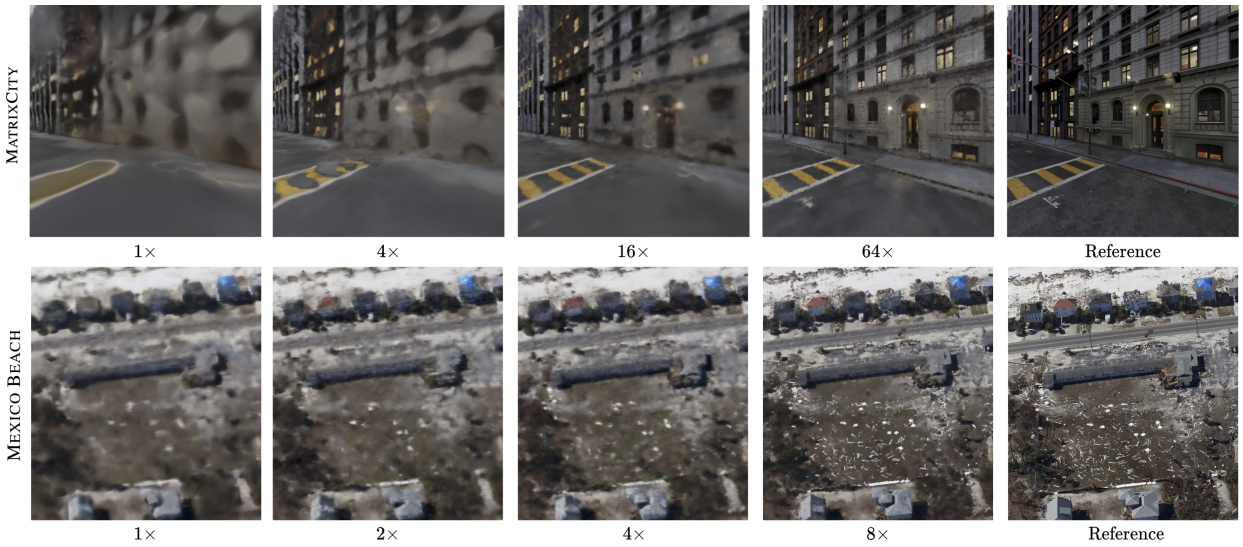


Figure 3.9: **Scalability of Our Approach.** More GPUs allow for more learnable parameters, leading to larger model capacity with better quality.

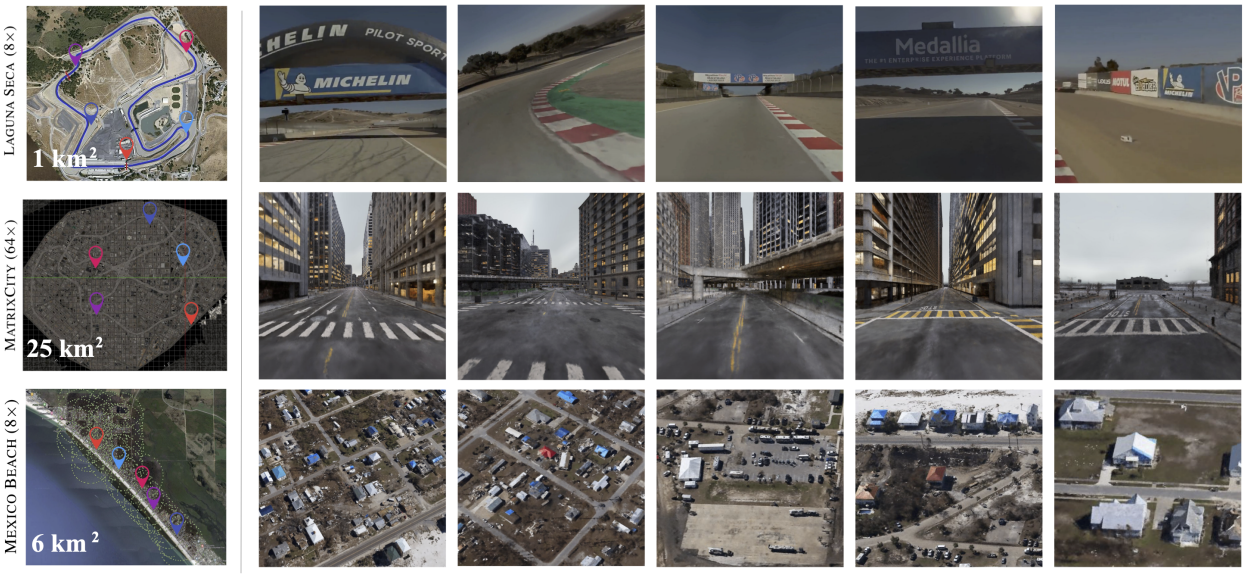


Figure 3.10: **More Rendering Results on Large Scale Captures.** We test the robustness of our approach on larger captures with more GPUs. Please refer to the our webpage for video tours on these data.

GPUs, and the 25km^2 MatrixCity [51] with 64 GPUs. It's noteworthy that each of these captures entails billions of pixels (see Table 3.1), posing a significant challenge to the NeRF model's capacity

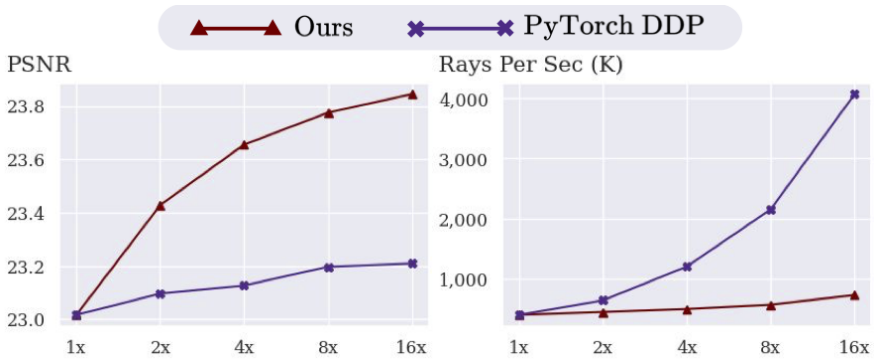


Figure 3.11: **Comparison with PyTorch DDP on UNIVERSITY4.** PyTorch Distributed Data Parallel (DDP) is designed for faster rendering by distributing rays across GPUs. In contrast, our approach distributes parameters across GPUs, scaling beyond the memory limits of single GPU in the cluster, and enabling larger model capacity for better quality.

due to the vast amount of information being processed.

Figure 3.9 presents qualitative results obtained using our approach, highlighting how the quality improves with the incorporation of more parameters through the utilization of additional GPUs. Please refer to our webpage for the video rendering.

Comparison with PyTorch DDP

Another common approach to utilize multi-GPU for NeRF is distributing rays across GPUs, *e.g.*, PyTorch's Distributed Data Parallel (DDP). This method typically allows for larger batch sizes during training or faster rendering through increased parallelization. However, DDP necessitates that all GPUs host *all* model parameters, thus limiting the model's capacity to the memory of a single GPU. In contrast, our approach assigns each GPU to handle a distinct 3D tiled region, aiming to alleviate memory constraints and ensure optimal quality even for large-scale scenes. Figure 3.11 illustrates a comparison between our method and DDP on the UNIVERSITY4 dataset. In this comparison, our method employs N$\times$ more parameters while DDP trains with N$\times$ more rays on N GPUs. The substantial improvement in PSNR indicates that large-scale NeRF benefits more from increased model capacity than from training more rays, a benefit uniquely enabled by our approach. However, DDP renders much faster than our approach due to the balanced workload created by parallelizing rays across GPUs. In contrast, our approach does not guarantee balanced workload distribution and consequently suffers from multi-GPU synchronization in run-time.

Multi-GPU Communication

We report the profiling results of multi-GPU communication costs on the UNIVERSITY4 capture in Figure 3.12. Despite achieving a reduction in communication costs by over 2$\times$ through partition-based volume rendering (tile-based vs. sample-based synchronization), multi-GPU communication

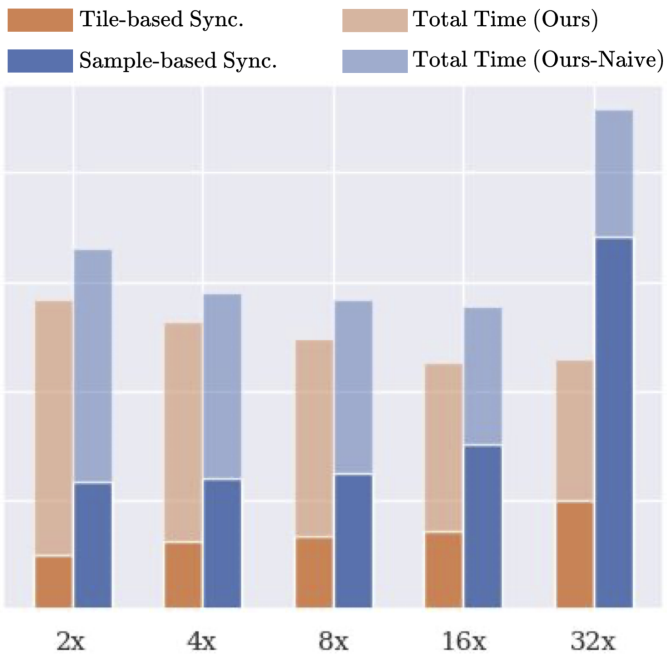


Figure 3.12: **Synchronization Cost on UNIVERSITY4**. Our partition-based volume rendering (§ 3.4) allows tile-based communication, which is much cheaper than the naive sample-based communication (§ 3.4), thus enabling faster rendering.

remains the primary bottleneck of our system. We attribute this to imbalanced workload distribution across GPUs, which could potentially be addressed through better spatial partitioning algorithms. We leave this optimization for future exploration.

3.6 Conclusion and Limitation

In conclusion, we revisited the existing approaches of decomposing large-scale scenes into independently trained NeRFs, and identified significant issues that impeded the effective utilization of additional computational resources (GPUs), thereby contradicting the core objective of leveraging multi-GPU setups to improve large-scale NeRF performance. Consequently, we introduced NeRF-XL, a principled algorithm to efficiently harness multi-GPU setups, and enhance NeRF performance at any scale by jointly training multiple non-overlapped NeRFs. Importantly, our method does not rely on any heuristics, and adheres to scaling laws for NeRF in the multi-GPU setting across various types of data.

However, our approach still has limitations. Similar to any other multi-GPU distributed setup, synchronization and communication overhead is inevitable in our joint training approach, which results in a slightly slower training speed ($1\times$-$1.5\times$) compared to baselines with independent training. Additionally, while our distributed approach is agnostic to NeRF representation in theory,

we have been only experimented with a popular choice, Instant-NGP [63], that equips with hash grids and MLPs. It will be an interesting future work to apply the framework to other representations, even beyond the task of static scene novel-view synthesis.

Chapter 4

gsplat: A Library for Efficient Gaussian Splatting

`gsplat` is an open-source library designed for training and developing Gaussian Splatting methods. It features a front-end with Python bindings compatible with the PyTorch library and a back-end with highly optimized CUDA kernels. `gsplat` offers numerous features that enhance the optimization of Gaussian Splatting models, which include optimization improvements for speed, memory, and convergence times. Experimental results demonstrate that `gsplat` achieves up to 10% less training time and $4\times$ less memory than the original [40] implementation. Utilized in several research projects, `gsplat` is actively maintained on GitHub. Source code is available at `https://github.com/nerfstudio-project/gsplat` under Apache License 2.0. We welcome contributions from the open-source community.

4.1 Introduction

Gaussian Splatting, a seminal work proposed by [40] is a rapidly developing area of research for high fidelity 3D scene reconstruction and novel view synthesis with wide interest in both academia and industry [23, 11, 1, 112]. It outperforms many of the previous NeRF-based [60] methods in several important areas, including i) computational efficiency for training and rendering, ii) ease of editing and post-processing, and iii) deployability on hardware-constrained devices and web-based technologies. In this paper, we introduce `gsplat`, an open-source project built around Gaussian Splatting that aims to be an efficient and user-friendly library. The underlying concept is to enable a simple and easily modifiable API for PyTorch-based projects developing Gaussian Splatting models. `gsplat` supports the latest research features and is developed with modern software engineering practices in mind. Since its initial release in October 2023, `gsplat` has garnered 67 contributors and over 2.5k stars on GitHub. `gsplat` is released under the Apache License 2.0. Documentation and further information are available on the website at:

`http://docs.gsplat.studio/`

The closest prior work implementing open-source Gaussian Splatting methods include GauStudio [120] which consolidates various research papers into a single code repository, several PyTorch-based reproductions [68, 35], and the more recent *Slang.D* [47] and *Brush* [8] reimplementations using the Slang.D and Rust programming languages, respectively. Unlike previous work, `gsplat` not only seeks to implement the original 3DGS work with performance improvements, but aims to provide an easy-to-use and modular API interface allowing for external extensions and modifications, promoting further research in Gaussian Splatting. We welcome contributions from students, researchers, and the open-source community.

4.2 Design

`gsplat` is a standalone library developed with efficiency and modularity in mind. It is installed from PyPI on both Windows and Linux platforms, and provides a PyTorch interface. For speed considerations, many operations are programmed into optimized CUDA kernels and exposed to the developer via Python bindings. In addition, a native PyTorch implementation is also carried in `gsplat` to support iteration on new research ideas. `gsplat` is designed to provide a simple interface that can be imported from external projects, allowing easy integration of the main Gaussian Splatting functionality as well as algorithmic customization based on the latest research. With well-documented examples, test cases verifying the correctness of CUDA operations, and further documentation hosted online, `gsplat` can also serve as an education resource for new researchers entering the field.

```python
import torch
from gsplat import rasterization
# Initialize a 3D Gaussian:
mean = torch.tensor([[0.,0.,0.01]], device="cuda")
quat = torch.tensor([[1.,0.,0.,0.]], device="cuda")
color = torch.rand((1, 3), device="cuda")
opac = torch.ones((1,), device="cuda")
scale = torch.rand((1, 3), device="cuda")
view = torch.eye(4, device="cuda")[None]
K = torch.tensor([[[1., 0., 120.], [0., 1., 120.], [0.,
    0., 1.]]], device="cuda") # camera intrinsics
# Render an image using gsplat:
rgb_image, alpha, metadata = rasterization(
    mean, quat, scale, opac, color, view, K, 240, 240)
```

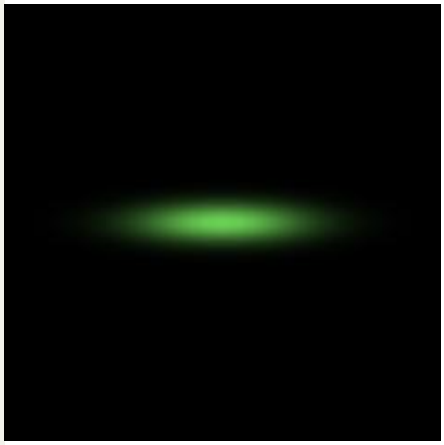


Figure 4.1: **Implementation of the main 3D Gaussian rendering process using the `gsplat` (v1.3.0) library with only 13 lines of code**. A single Gaussian is initialized (left codeblock) and rendered as an RGB image (right).

4.3 Features

The `gsplat` library consists of features and algorithmic implementations relating to Gaussian Splatting. With a modular interface, users can choose to enable features with simple API calls. Here, we briefly describe some of the algorithmic enhancements provided by `gsplat` which are not present in the original 3DGS implementation by [40].

Densification strategies. A key component of the Gaussian Splatting optimization procedure consists of densification and pruning of Gaussians in under- and over-reconstructed regions of the scene respectively. This has been an active area of research, and the `gsplat` library supports some of the latest densification strategies. These include the Adaptive Density Control (ADC) proposed by [40], the Absgrad method proposed in [121], and the Markov Chain Monte Carlo (MCMC) method proposed in [42]. `gsplat`'s modular API allows users to easily change between strategies.

```
from gsplat import MCMCStrategy, rasterization
strategy = MCMCStrategy() #Initialize the strategy
strategy_state = strategy.initialize_state()
for step in range(1000): # Training loop
    render_image, render_alpha, info = rasterization(...)
    strategy.step_pre_backward(...)# Pre-backward step
    loss = ... # Compute the loss
    loss.backward() # Backward pass
    strategy.step_post_backward(...) # Post-backward step

```

Figure 4.2: **Code-block for training a Gaussian model with a chosen densification strategy**.

Pose optimization. The Gaussian rendering process (seen in 4.1) in `gsplat` is fully differentiable, enabling gradient flow to Gaussian parameters $\mathcal{G}(c, \Sigma, \mu, o)$ and to other parameters such as the camera view matrices $\mathcal{P} = [\boldsymbol{R} \mid \boldsymbol{t}]$, which were not considered in the original work. This is crucial for mitigating pose uncertainty in datasets. Specifically, gradients of the reconstruction loss are computed with respect to the rotation and translation components of the camera view matrix, allowing for optimization of initial camera poses via gradient descent.

Depth rendering. Rendering depth maps from a Gaussian scene is important for applications such as regularization and meshing. `gsplat` supports rendering depth maps using an optimized RGB+Depth rasterizer that is also fully differentiable. `gsplat` supports rendering depth maps using the accumulated z-depth for each pixel and the alpha normalized expected depth.

N-Dimensional rasterization. In addition to rendering three-channel RGB images, `gsplat` also supports rendering higher-dimensional feature vectors. This is motivated by algorithms that combine learned feature maps with differentiable volume rendering [45, 41]. To accommodate the storage

needs of these features, the `gsplat` backend allows for adjustments to parameters affecting memory allocation during training, such as kernel block sizes.

Anti-aliasing. Viewing a 3D scene represented by Gaussians at varying resolutions can cause aliasing effects, as seen in prior 3D representations [6, 5]. When the resolution decreases or the scene is viewed from afar, individual Gaussians smaller than a pixel in size produce aliasing artifacts due to sampling below the Nyquist rate. Mip-Splatting [125] proposes a low pass filter on projected 2D Gaussian covariances, ensuring a Gaussian's extent always spans a pixel. `gsplat` supports rendering with the 2D anti-aliasing mode introduced in Yu et al.

4.4 Evaluation

Overall comparison. We compare the training performance and efficiency of `gsplat` training against the original implementation by Kerbl et al. on the MipNeRF360 dataset [5]. We use the standard ADC densification strategy and equivalent configuration settings for both. We report average results on novel-view synthesis, memory usage, and training time using an A100 GPU (`PyTorch` v2.1.2 and `cudatoolkit` v11.8) at 7k and 30k training iterations in 4.1.

Table 4.1: **Comparison of `gsplat` training performance with the original 3DGS (Kerbl et al.) implementation on the MipNeRF360 dataset**. Results are averaged over 7 scenes.

	PSNR ↑	SSIM ↑	LPIPS ↓	Memory (GB) ↓	Time (min) ↓
3DGS -7K	27.23	.8290	.2041	7.7	4.64
GSPLAT -7K	27.23	.8311	.2027	**4.3**	**3.36**
3DGS -30K	28.95	.8702	.1381	9.0	26.19
GSPLAT -30K	**29.00**	**.8715**	**.1357**	5.6	19.39

We achieve the same rendering performance as the original implementation whilst using less memory and significantly reducing training time.

Feature comparison. Furthermore, we analyze the impact of features provided in the `gsplat` library in 4.2.

Table 4.2: **`gsplat` feature comparison on the MipNeRF360 dataset averaged over 7 scenes**.

	PSNR↑	SSIM↑	LPIPS↓	Num GS	Mem (GB) ↓	Time (min) ↓
GSPLAT	29.00	.8715	.1357	3.24 M	5.62	19.39
w/ ABSGRAD	29.11	**.8778**	**.1241**	2.47 M	4.40	18.10
w/ MCMC	**29.18**	.8745	.1446	**1.00 M**	**1.98**	**15.42**
w/ ANTIALIASED	29.03	.8711	.1389	3.38 M	5.87	19.52

Chapter 5

PRoPE: Multi-view Attention Leads to Faster Convergence

As transformers become increasingly adopted for multi-view 3D perception tasks, a key challenge lies in effectively conditioning these models on camera parameters that ground visual observations in 3D space. We explore the generalization capabilities of these multi-view transformers, systematically analyzing how different approaches to camera conditioning affect model performance and robustness. Our analysis reveals that while existing attention-level conditioning methods show good generalization in terms of $SE(3)$ transformations, they struggle when faced with varying intrinsic parameters at test time. Through careful theoretical analysis, we identify that the key to robust generalization lies in encoding the complete projective relationship between cameras, which naturally encompasses both intrinsic and extrinsic parameters while maintaining invariance to the choice of global reference frame. This insight leads to Projective Positional Encoding (PRoPE), which emerges as a natural theoretical extension of existing approaches. We demonstrate the effectiveness of PRoPE across three diverse tasks: novel view synthesis, stereo depth estimation, and multi-view spatial cognition. Our method shows strong performance and robustness, particularly when handling varying focal lengths and number of views at test time. PRoPE integrates seamlessly into existing transformer architectures, including CAT3D and UniMatch, yielding consistent improvements with minimal computational overhead. The method is flash attention-friendly and is open-sourced at `https://github.com/liruilong940607/prope`.

5.1 Introduction

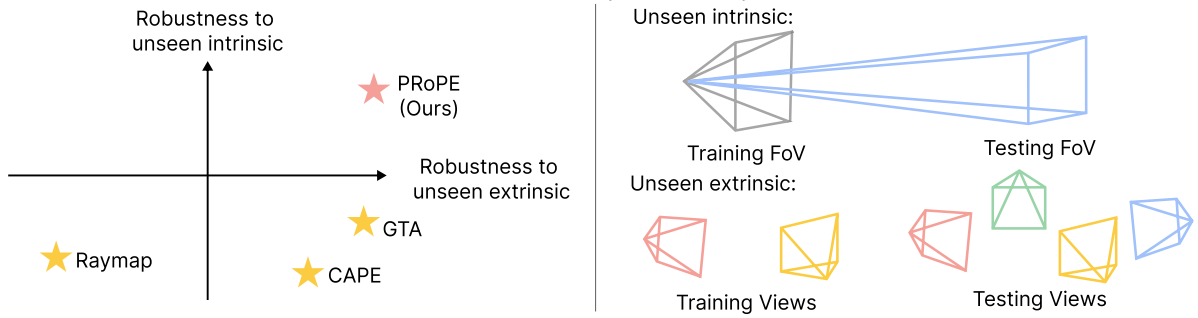
Images of our world exist in the context of the viewpoints they were captured from. The parameters of these viewpoints—extrinsics that provide position and orientation and intrinsics that provide focal lengths and field of view—ground visual observations in 3D space. The importance of leveraging this spatial grounding is becoming increasingly important, especially as advances in 3D vision and embodied agents make multiview images more ubiquitous.

To solve these tasks with transformers, models must take both image patches and camera parameters

as input. Image patches can be passed to models as standard visual tokens. Cameras, on the other hand, require special care. Just as naive position encoding techniques can hinder performance for learning in language models [85, 89], naive encodings of camera pose are also suboptimal for vision models [62, 46].

Method	Patch Coord Sys.	Intri Coord Sys.	Extri Coord Sys.
Plucker [128]	×	Abs.	Abs.
CAPE [46]	×	×	Rel.
GTA [62]	Rel.	×	Rel.
PRoPE	Rel.	Rel.	Rel.

(a) Coordinate System Analysis



(b) Robustness to Varying Camera Configurations

Figure 5.1: **Systematic Analysis of Camera Conditioning.** Our analysis reveals that existing methods either use absolute coordinate systems or encode incomplete camera information, limiting their robustness to varying camera configurations. PRoPE addresses both limitations by encoding complete camera information in a relative manner.

In this work, we explore the robustness of view-conditioned transformers, systematically analyzing how different approaches to camera conditioning affect model performance under varying camera configurations. We find that existing methods fall into two categories: token-level [128, 28, 37] and attention-level [62, 46] conditioning. Raymaps, which concatenate image tokens with an over-parameterized per-pixel representation of the ray and origin, naturally encode both intrinsics and extrinsics but are inherently sensitive to the choice of global frame. Attention-level conditioning, on the other hand, maintains invariance to global transformations by capturing pairwise relationships, similar to RoPE in sequence modeling [89]. However, these approaches have primarily focused on encoding extrinsics (SE(3)) without intrinsics.

Our systematic analysis reveals that while attention-level conditioning methods like GTA show good robustness to $SE(3)$ transformations, they struggle when faced with varying intrinsic parameters at test time. This limitation becomes particularly apparent in real-world scenarios where cameras may have different focal lengths or field of views. Through careful theoretical analysis, we identify that the key to robust performance lies in encoding the complete projective relationship between cameras, which naturally encompasses both intrinsic and extrinsic parameters.

This insight leads us to Projective Positional Encoding (PRoPE), which emerges as a natural theoretical extension of existing approaches. PRoPE encodes pairwise transformations between projective camera spaces, providing a unified framework that handles both intrinsic and extrinsic parameters while maintaining invariance to the choice of global frame. The projective transformation between cameras captures how 3D points map onto different image planes, encapsulating both the relative pose and the varying intrinsic parameters of the cameras. We inject these transformations into the attention mechanism, following the successful pattern established by GTA [62], but extending it to handle the full projective space.
We validate our approach through extensive experiments across three diverse tasks—novel view synthesis, stereo depth estimation, and multi-view spatial cognition—demonstrating significantly better robustness in practical settings, particularly when handling varying focal lengths and number of views at test time. The attention-level conditioning can be further enhanced by a token-level injection of intrinsics, CamRay, which preserves global frame invariance while improving performance. Our method consistently outperforms prior approaches across all settings, while remaining simple to implement (fewer than 30 lines of code) and compatible with modern transformer architectures. The code will be open-sourced upon publication.

5.2 Related Work

From absolute to relative positional encoding. Positional encoding is a fundamental technique that injects ordering information into the otherwise order-agnostic transformer architecture [104]. It has also been found critical in enabling neural scene representations to encode high-frequency information [61, 97]. How to encode the position information in sequence models has been an active area of research [89, 38, 69, 29, 84]. While early works [73, 17, 18, 9, 74, 81, 127, 43] focused on absolute positional encoding (APE), recent research has increasingly adopted relative positional encoding (RPE), particularly RoPE [33], as a standard across domains, including natural language processing [75, 100, 89, 31] and computer vision [55, 20, 48, 33, 99]. Notably, RPE offers key advantages over APE, including translation invariance, improved dependency modeling, and thus stronger generalization to long sequences [71, 89, 19]. In this work, we build on prior findings that shows RPE enhances camera conditioning in transformers [62, 46] and extend this idea by modeling pairwise camera relationships through projective transformations.

Multiview transformers. There is growing interest in applying transformers to 3D vision by conditioning multiple views on cameras. This approach has been explored in tasks such as view synthesis [90, 82, 110, 52], pose estimation [128], depth prediction [116, 118, 39], 3D scene understanding [34, 131], and robotics [86] We address transformers where inputs include N images from known cameras:

$$\{(\boldsymbol{I}_i, \boldsymbol{K}_i, \boldsymbol{T}_i^{cw})\}_{i=1}^{N} \tag{5.1}$$

where each $\boldsymbol{I}_i \in \mathbb{R}^{H\times W\times 3}$ is an image, $\boldsymbol{K}_i \in \mathbb{R}^{3\times 3}$ is the camera intrinsic matrix, and $\boldsymbol{T}_i^{cw} = (\mathbf{R}_i^{\text{cw}}, \mathbf{t}_i^{\text{cw}}) \in \text{SE}(3)$ is the rigid transform to camera coordinates from world coordinates.

Given the growing ubiquity of multiview image inputs, research on encoding camera information into transformers has gained traction, broadly falling into two categories (1) token-level and (2) attention-level conditioning:

Token-level raymap conditioning. Recent works show that over-parameterizing cameras as raymaps is effective in multiview transformers [128, 28, 37]. Each camera is encoded as a 6D pixel-aligned token, where each pixel stores either the ray origin and direction [61, 28], or Plücker coordinates [70, 128], providing a natural way to capture both intrinsics and extrinsics. It is then concatenated with corresponding patch tokens before entering the network.
While effective, raymaps share similarities with APE and inherit its limitations. They are sensitive to global $\mathrm{SE}(3)$ transformations applied to cameras and require defining a "canonical" coordinate system, forcing practitioners to choose a reference frame for camera normalization [59, 28, 30]—an often arbitrary and ambiguous decision that directly impacts performance. We overcome these limitations by encoding relative transformation between cameras in the projective coordinate space.

Attention-level $\mathrm{SE}(3)$-based conditioning. Just as NLP and vision models have transitioned from APE to RPE [71, 89], multiview transformers have similarly incorporated relative encodings for camera geometry by integrating them into the attention mechanism. Methods like GTA [62] and CAPE [46] encode pairwise $\mathrm{SE}(3)$ extrinsic relationships in attention, ensuring invariance to global transformations. Building on this line of work, we generalize relative transformations into projective coordinate spaces, which naturally encode both intrinsics and extrinsics. Empirically, we also find that injecting intrinsics at the token level in addition to attention-level conditioning further enhances performance while preserving global frame invariance.

5.3 Method

Our systematic analysis of existing camera conditioning methods reveals a fundamental connection between multi-view geometry and language modeling. Just as language models must capture relationships between tokens in a sequence, multi-view systems must model relationships between observations from different viewpoints. This insight leads us to Projective Positional Encoding (PRoPE), which extends the successful relative positional encoding paradigm from language models to encode complete projective relationships between cameras.
PRoPE is designed to address two key limitations we identified in existing approaches: (1) the sensitivity to global coordinate frame choice in token-level conditioning methods, and (2) the inability to handle varying intrinsic parameters in attention-level conditioning methods. By encoding pairwise transformations between projective camera spaces, PRoPE naturally encompasses both intrinsic and extrinsic parameters while maintaining invariance to the choice of global frame.
We additionally propose CamRay, a token-level geometry encoding technique that complements PRoPE by providing camera-frame ray directions. Unlike existing raymap approaches, CamRay is designed to be invariant to global frame choice while still capturing essential intrinsic information.

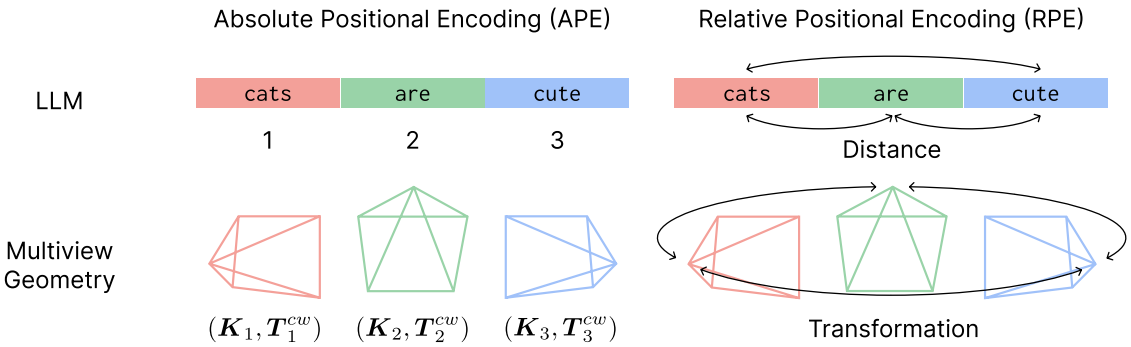


Figure 5.2: **Key Insights of Our Method.** Language-modeling and multi-view perception confront the same core problem: learning relationships among discrete observations. In an LLM, each token's position is its index in the sequence, and rotary positional encoding (RoPE) replaces absolute indices with relative offsets so attention can focus on token-to-token distances. In multi-view geometry, an observation's position is the camera that captured it—fully specified by its intrinsic and extrinsic parameters. Thus we could make this direct substitution – treating token offsets as relative camera transforms – laying the foundation for a geometry-aware attention mechanism that inherits RoPE's proven advantages.

Our experiments demonstrate that this combination provides the best of both worlds: the robustness of relative encoding with the expressiveness of token-level conditioning.

Preliminaries

PRoPE is based on frustums, which can be parameterized using standard camera intrinsic and extrinsic matrices $\boldsymbol{K}_i$ and $\boldsymbol{T}_i^{cw}$. The intrinsics capture the shape and field-of-view of the frustum, while the extrinsics capture its position and orientation. The geometry of frustums are encoded in the "world-to-image" projection matrix $\boldsymbol{P}_i \in \mathbb{R}^{3\times4}$:

$$\boldsymbol{P}_i = \begin{bmatrix}\boldsymbol{K}_i & \mathbf{0}^{3\times1}\end{bmatrix} \boldsymbol{T}_i^{cw} \tag{5.2}$$

This projective transformation can be used to compute 2D image coordinates from world coordinates,

$$\tilde{\mathbf{x}}_i \sim \boldsymbol{P}_i \tilde{\mathbf{X}}_{\text{world}}, \tag{5.3}$$

where $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{X}}_{\text{world}}$ are homogeneous coordinates in the image and world respectively.

Pairwise Projective Transforms

The core insight of PRoPE is that geometric relationships between camera frustums can be expressed as *pairwise* projective transforms, which are naturally invariant to the choice of global reference frame. This approach mirrors the success of relative positional encoding in language models, where pairwise relationships between tokens are encoded rather than absolute positions.

To compute these transforms, we begin by lifting our 3×4 world-to-image projection matrices $\boldsymbol{P}_i$ into 4×4 by augmenting with the standard basis vector $\mathbf{e}_4 = (0, 0, 0, 1)^\top$:

$$\tilde{\boldsymbol{P}}_i = \begin{bmatrix} \boldsymbol{P}_i \\ \mathbf{e}_4^\top \end{bmatrix} \tag{5.4}$$

This transformation maps homogeneous world coordinates to a projective space defined by the frustum of camera i. We can then relate frustum i and frustum j with the pairwise projective transformation:

$$\tilde{\boldsymbol{P}}_{ij} = \tilde{\boldsymbol{P}}_i \tilde{\boldsymbol{P}}_j^{-1}, \tag{5.5}$$

The 4×4 matrix $\tilde{\boldsymbol{P}}_{ij}$ encodes the complete geometric relationship between camera views, in a way that is invariant to the world coordinate frame. As we discuss next, it can also be efficiently injected directly into attention layers.

Attention Implementation

Following common patterns in relative position encoding [46, 62, 89], PRoPE injects pairwise frustum geometry directly into attention operations. The standard scaled dot product attention [104] function is formulated as:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V. \tag{5.6}$$

where $Q, K, V \in \mathbb{R}^{T \times d}$.We follow the GTA [62] formulation for injecting per-token transformations. Specifically, we use each token's geometric parameters to construct a block-diagonal transformation matrix $\mathbf{D}_t \in \mathbb{R}^{d \times d}$ that can be injected into attention blocks similar to RoPE [89]. We inject the pairwise relationships $\mathbf{D}_{t_1}\mathbf{D}_{t_2}^{-1}$ for all $t_1, t_2 \in 1 \dots T$ by replacing attention layers with:

$$\text{Attn}'(Q, K, V) = \mathbf{D} \circledcirc \text{Attn}(\mathbf{D}^\top \circledcirc Q, \mathbf{D}^{-1} \circledcirc K, \mathbf{D}^{-1} \circledcirc V) \tag{5.7}$$

where $\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 \dots \mathbf{D}_T \end{bmatrix} \in \mathbb{R}^{T \times d \times d}$ and $\circledcirc$ is a token-wise matrix multiplication $(\mathbf{D} \circledcirc Q)_{ni} = \sum_j \mathbf{D}_{nij} Q_{nj} \in \mathbb{R}^{T \times d}$. Importantly, this injection only requires transformations of the original attention operator's inputs and outputs; they therefore preserve compatibility with fused attention kernels such as FlashAttention [15].

To instantiate these operations for PRoPE, $\mathbf{D}_t$ must be specified from a large possible design space. We propose a simple but effective implementation, which aims to (1) encode frustum relationships *between* cameras—this uses the projective transformation proposed in Equation 5.5—and (2) encode relative patch positions *within* cameras.

We divide $\mathbf{D}_t$ into two equally sized block diagonal matrices. Each is with shape $\frac{d}{2} \times \frac{d}{2}$:

$$\mathbf{D}_t^{\text{Proj}} = \mathbf{I}_{d/8} \otimes \tilde{\boldsymbol{P}}_{i(t)} \tag{5.8}$$

$$\mathbf{D}_t^{\text{RoPE}} = \text{diag}\left(\text{RoPE}_{d/4}(x_t), \text{RoPE}_{d/4}(y_t)\right) \tag{5.9}$$

where $\mathbf{I}_{d/8}$ is the identity matrix, $\otimes$ is the Kronecker product, $i(t)$ is the camera index for token t, $\text{RoPE}_{d/4}(\cdot)$ constructs $\frac{d}{4} \times \frac{d}{4}$ rotary embeddings [89] for the input positions, and (x_t, y_t) are the patch coordinates for token t.
From these two matrices, we can write the final $\mathbf{D}_t$ matrix used by PRoPE as

$$\mathbf{D}_t = \text{diag}(\mathbf{D}_t^{\text{Proj}}, \mathbf{D}_t^{\text{RoPE}}). \tag{5.10}$$

This completes the definition of PRoPE.

Invariance and connections to existing methods. PRoPE has several additional properties, which become more evident when we unpack the transformations that make up $\tilde{\boldsymbol{P}}_{ij}$. We can express:

$$\tilde{\boldsymbol{P}}_{ij} = \begin{bmatrix} \boldsymbol{K}_i & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \boldsymbol{T}_i^{cw} (\boldsymbol{T}_j^{cw})^{-1} \begin{bmatrix} \boldsymbol{K}_j^{-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.11}$$

We observe: *(1) Global frame invariance.* Redefining the world frame is equivalent to right-multiplying both $\boldsymbol{T}^{cw}$ $SE(3)$ terms by a transformation matrix, which is algebraically eliminated in $\tilde{\boldsymbol{P}}_{ij}$. *(2) Reduction to GTA/CAPE.* For cameras with normalized intrinsics, $\boldsymbol{K} = \mathbf{I}_3$, $\tilde{\boldsymbol{P}}_{ij}$ simplifies to the relative $SE(3)$ transformations utilized in GTA [62] and CAPE [46]. These methods can therefore be interpreted as a special case of PRoPE where the intrinsic matrices are constrained to identity. *(3) Reduction to RoPE.* $\tilde{\boldsymbol{P}}_{ij}$ evaluates to identity for tokens derived from the same image. For such tokens, PRoPE simplifies to the remaining terms in $\mathbf{D}_t$, which correspond precisely to the RoPE terms employed in single-image vision transformers.

Token-level Conditioning

Camera-frame raymaps. As a complement to PRoPE, we also propose a token-level encoding of *camera-frame* ray directions, which are invariant to global frame and that we find empirically improves model results. We refer to this as CamRay in our experiments. Specifically, for each image $\boldsymbol{I}_i$, we compute a corresponding raymap $\mathbf{M}_{i,CamRay} \in \mathbb{R}^{H \times W \times 3}$ using ray directions in the camera's local frame. For each pixel coordinate (u, v), we compute:

$$\mathbf{M}_{i,CamRay}^{u,v} = \boldsymbol{K}_i^{-1} \begin{bmatrix} u & v & 1 \end{bmatrix}^\top \in \mathbb{R}^3 \tag{5.12}$$

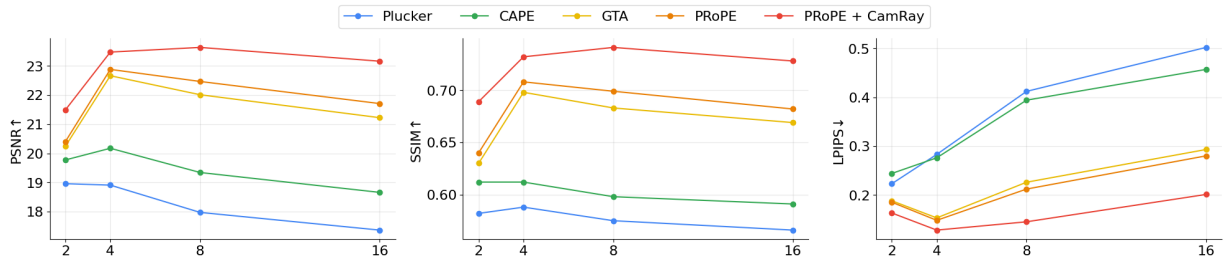
CamRay is concatenated with input images along the channel dimension, which expands input dimensions from $\mathbb{R}^{H \times W \times 3}$ to $\mathbb{R}^{H \times W \times 6}$.

Connections to prior work. CamRay is closely related to the raymaps used in existing works. Naive raymaps [28] are made up of origin and direction pairs and can be computed from CamRay:

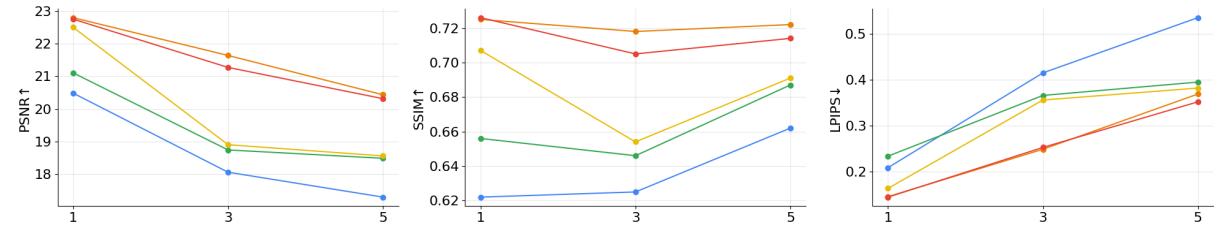
$$\boldsymbol{o}_i = -(\mathbf{R}_i^{\text{cw}})^\top \mathbf{t}_i^{\text{cw}} \tag{5.13}$$

$$\boldsymbol{d}_i^{u,v} = (\mathbf{R}_i^{\text{cw}})^\top \mathbf{M}_{i,CamRay}^{u,v} \tag{5.14}$$

$$\mathbf{M}_{i,\texttt{Naive}}^{u,v} = \begin{bmatrix} \boldsymbol{o}_i \\ \boldsymbol{d}_i^{u,v} \end{bmatrix} \in \mathbb{R}^6 \tag{5.15}$$



(a) Testing with {2, 4, 8, 16} input views.



(b) Testing with {1, 2, 3}× zooming-in.

Figure 5.3: **Evaluation on RealEstate10K** [130]. Our method demonstrates superior robustness when handling varying numbers of input views (a) and different focal lengths (b) at test time.

Plücker raymaps [128] replace the origin term with a moment term, which is computed as:

$$\mathbf{M}^{u,v}_{i,\texttt{Plücker}} = \begin{bmatrix} \boldsymbol{o}_i \times \boldsymbol{d}^{u,v}_i \\ \boldsymbol{d}^{u,v}_i \end{bmatrix} \in \mathbb{R}^6 \tag{5.16}$$

In contrast to raymaps used by prior work, CamRay does not encode camera extrinsics information.

5.4 Experiments

We evaluate the performance of existing camera conditioning methods – Plucker Raymap [128], CAPE [46], GTA [62][1], along with our proposed PRoPE on *three different tasks* with *six total datasets*: we evaluate novel view synthesis on Objaverse [16] and RealEstate10K [130], stereo depth estimation on RGBD [88], SUN3D [115], and Scenes11 [103], and multi-view perception on DL3DV [54]. While most experiments are under academic-scale compute to systematically study design choices instead of beating SOTA performance, we also include results from larger-scale training to validate our findings still hold on large models. All methods are implemented in the same codebase for fair comparison.

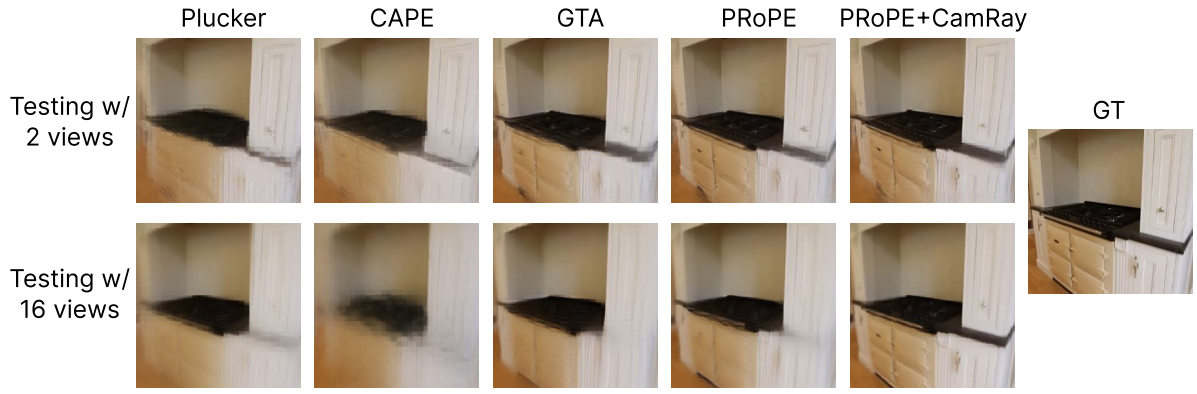


Figure 5.4: **Results of Longer Input Sequences at Test Time.** Our method demonstrates superior robustness when tested with varying numbers of input views, despite being trained with only 2 views.

Figure 5.5: **Results of Varying Focal Length at Test Time.** PRoPE explicitly models relative intrinsics, making it robust to zoomed-in test views. In contrast, CAPE and GTA have to implicitly understand intrinsic from pixels, often resulting in incorrect geometry.

Novel View Synthesis Task

Novel view synthesis requires rendering scenes from unseen viewpoints, given calibrated reference images and target camera parameters. This task demands precise modeling of pixel-camera relationships across multiple views. We reimplement LVSM [37], a recent state-of-the-art view synthesis method, as our base model for this task. LVSM extends the Vision Transformer [18] to multi-view inputs by encoding camera information as per-pixel Plücker rays, which are concatenated with image pixels. For further details, we refer readers to the original paper.

Baseline Methods. To compare various camera conditioning methods, we reimplement them into the same framework based on LVSM, with minimal changes to support different methods. (1) Plücker Raymap: Unmodified LVSM. (2) CAPE [46] / GTA [62] / PRoPE (ours): Replace Plücker with a shared fixed embedding and injects camera information into attention using respective formula. (3) PRoPE + CamRay (ours): Replace Plücker with CamRay, and injects camera information into

[1]The GTA baseline uses the $SE(3)$+$SO(2)$ variation of $\mathbf{D}_t$ studied by [62].

attention. To fully analyze the capability of each approach, in addition to evaluating the model on the test split which has the same distribution with the train split, we also evaluate the model in two practical settings, which could lead to out-of-distribution data at test time.

Setting 1. Longer Input Sequences at Test Time. The first setting deploys the model trained with a fixed number of input views (2 in our experiments) to significantly more views (up to 16) at test time. This is particularly important in real-world scenarios where the number of observations can vary substantially across different applications, and sometimes can be dynamically increasing – this poses a unique challenge for the model to handle out-of-distribution camera poses at test time. Note that a similar problem exists in the LLMs community known as "test-time context length extrapolation", where models are trained with a fixed context length but evaluated on longer sequences, requiring special handling of positional encodings.

Setting 2. Varying Focal Length at Test Time. The second setting evaluates the model's ability to handle varying focal lengths at test time. This is crucial as in practice, focal lengths can vary significantly across different cameras and zoom levels, and it is impractical to train a separate model for every possible focal length. We test our models with focal lengths ranging from $1\times$ to $3\times$ the training focal length, simulating scenarios where camera zooms in during deployment. This setting is particularly challenging for methods that don't explicitly encode camera intrinsics, as they may struggle to adapt to these variations in the camera's field of view.

Discussion. Results on the RealEstate10K [130] dataset are summarized in Figure 5.3 with visuals provided in Figure 5.4 and 5.5. Our analysis reveals three key findings. First, while Plücker Raymap encodes more complete camera information than methods such as CAPE and GTA, it consistently underperforms across all settings. This suggests that directly injecting pairwise camera information into attention mechanisms is more effective than relying solely on token-level conditioning. Second, PRoPE demonstrates superior performance and robustness compared to all other methods in both out-of-distribution settings, particularly when handling varying focal lengths. This indicates that explicitly modeling the relative *projective* relationship between cameras is more effective than modeling the relative $SE(3)$ relationship while inferring intrinsic relationship from pixels, as done in GTA and CAPE. Finally, we observe that additional conditioning with CamRay on top of PRoPE further enhances performance in certain cases. Since CamRay is designed to be invariant to the choice of world coordinate system, it preserves the robustness of PRoPE when handling out-of-distribution camera configurations.

Additional Results. We further validate the effectiveness of PRoPE through three additional experiments. First, we integrate PRoPE into CAT3D [28], a popular multi-view diffusion model finetuned from a video diffusion model, with the assistance of its authors on their original codebase. As shown in Table 5.1, this integration yields consistent improvements across all metrics while requiring zero additional model parameters and introducing negligible computational overhead. Second, on the Objaverse [16] dataset, integrating PRoPE into the LVSM model for novel-view synthesis

Method	3-view PSNR ↑	3-view SSIM ↑	3-view LPIPS ↓	6-view PSNR ↑	6-view SSIM ↑	6-view LPIPS ↓	9-view PSNR ↑	9-view SSIM ↑	9-view LPIPS ↓
Zip-NeRF [4]	12.77	0.271	0.705	13.61	0.284	0.663	14.30	0.312	0.633
ZeroNeRF [83]	14.44	0.316	0.680	15.51	0.337	0.663	15.99	0.350	0.655
ReconFusion [111]	15.50	0.358	0.585	16.93	0.401	0.544	18.19	0.432	0.511
CAT3D [28]	16.62	0.377	0.515	17.72	0.425	0.482	18.67	0.460	**0.460**
CAT3D [28] + PRoPE	**16.93**	**0.382**	**0.505**	**18.01**	**0.443**	**0.479**	**18.98**	**0.474**	0.461

Table 5.1: **Improving Multi-view Diffusion Model on the task of Few-view 3D Reconstruction.** Applying our PRoPE to CAT3D [28] yields noticeable improvements over the original model, with zero additional model parameters and negligible computational overhead.

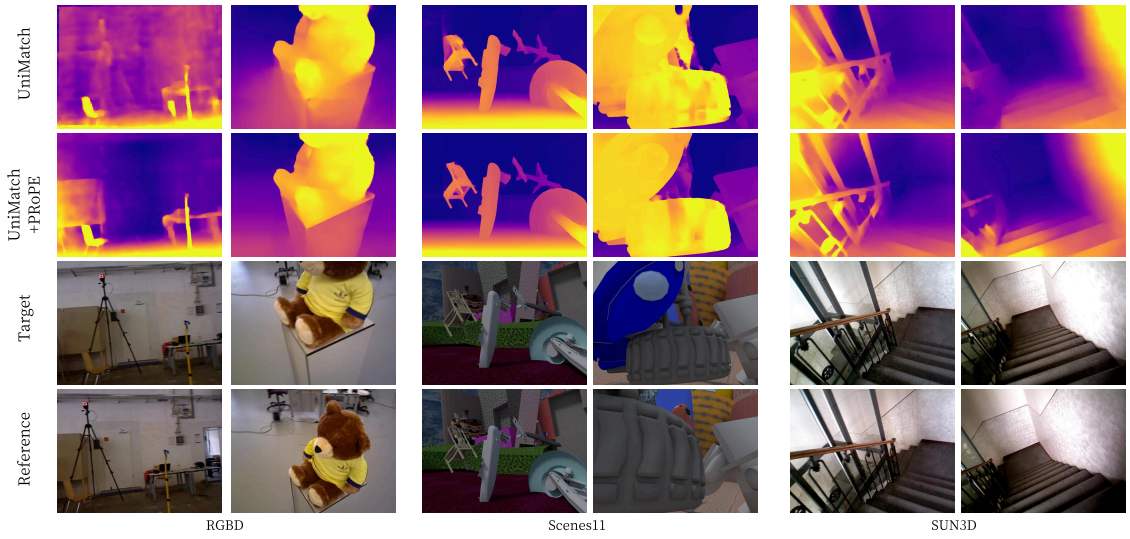


Figure 5.6: **Qualitative Results on Stereo Depth Estimation Task.** Injecting camera information into UniMatch [116] via our proposed PRoPE leads to significant improvements.

improves performance from 21.44/0.851/0.159 to 24.17/0.885/0.094 in terms of PSNR/SSIM/LPIPS metrics. Finally, when scaling up the LVSM training with approximately 150× more computational resources, we observe similar improvements, with metrics improving from 25.64/0.809/0.084 to 26.56/0.832/0.071, demonstrating that our method's advantages persist even at larger scales.

Stereo Depth Estimation Task

We set up this task using UniMatch [116], a popular pretrained multi-view transformer used in various downstream applications [12, 13, 87, 64]. UniMatch is originally trained on three different tasks. In this work, we focus on the stereo depth estimation task, which assumes known relative camera poses between input views.

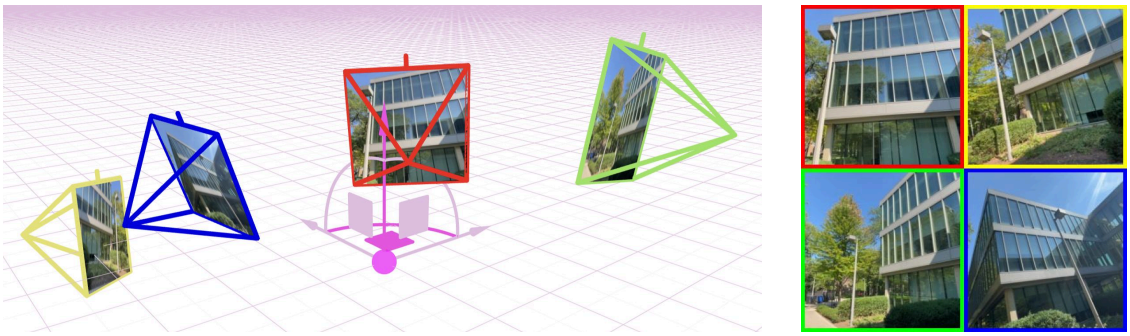


Figure 5.7: **The Spatial Cognition Task.** The model takes multi-view images and corresponding camera information as input and aims to identify inconsistent image-camera pairs (yellow here). Understanding the cross-view relationships between images and cameras is crucial for this task.

Dataset	Model	Abs Rel	Sq Rel	RMSE	RMSE log
RGBD [88]	UniMatch	0.123	0.175†	0.678	0.203
	UniMatch (Pretrain)	0.107	0.130†	0.606	**0.166**
	UniMatch + PRoPE	**0.105**	0.203†	**0.573**	0.181
SUN3D [115]	UniMatch	0.131	0.098	0.397	0.169
	UniMatch (Pretrain)	0.122	0.085	0.374	0.158
	UniMatch + PRoPE	**0.117**	**0.075**	**0.343**	**0.152**
Scenes11 [103]	UniMatch	0.065	0.085	0.575	0.126
	UniMatch (Pretrain)	0.054	0.067	0.511	0.108
	UniMatch + PRoPE	**0.049**	**0.063**	**0.474**	**0.104**

Table 5.2: **Performance Improvement on Stereo Depth Estimation Task with UniMatch [116]**. †The metric "Sq Rel" is less reliable on the RGBD dataset because its depth and camera pose are imperfect [103]. Models marked as "Pretrain" are initialized with optical flow pretraining, while others are trained from scratch.

Method	5 views	9 views	17 views
Plucker	69.1%	76.9%	74.6%
Plucker + PRoPE	81.1%	90.5%	91.8%
CamRay + PRoPE	**86.1%**	**93.0%**	**94.3%**

Table 5.3: **Results on the Spatial Cognition Task**. We report the accuracy of detecting inconsistent image-camera pairs on the DL3DV [54] dataset under varying numbers of input views. Both CamRay and PRoPE significantly helps with the performance without introducing additional model parameters. The illustration of this task can be found in Figure 5.7.

We incorporate camera information into UniMatch's cross-view attention mechanism using our proposed PRoPE, modifying only ∼50 lines of the official code. This modification incurs negligible computational overhead during training and inference while delivering significant performance improvements, as shown in Table 5.2. Notably, incorporating our PRoPE formulation and training from scratch achieves even superior performance compared to models initialized with optical flow pretraining (denoted as "Init" in the table). All models follow the exact same training protocols as described in the original paper.

Spatial Cognition Task

Inspired by the multi-image cognitive task proposed in [7], we design a spatial cognition task to assess a system's ability on understanding multi-view consistency. In this task, the multi-view system takes in multiple captured images (more than three) of the same environment from different viewpoints, along with their corresponding camera information. However, one of the image-camera pairs is deliberately mismatched. The system is then required to identify the incorrect pair among the inputs. Figure 5.7 provides two exemplars on the input and the desired output – the pair that is inconsistent with others.

This cognition task necessitates a faithful understanding of the relationship between images and cameras, as well as the geometric consistency across multiple viewpoints. The problem is carefully designed so that it cannot be solved by analyzing only the camera information, the images alone, or without reasoning about the multi-view relationships among all inputs. Despite its seemingly simple nature, the task is perfect to verify model's ability on multi-view image understanding.

The base model we used in this task is a transformer-based architecture, similar to the one employed in our NVS task, but adapted with a classification head (see details in the supplemental material). We evaluate two approaches for incorporating camera information: (1) injecting it as a Plücker Raymap concatenated with image pixels, and (2) using our PRoPE formulation. As shown in Table 5.3, PRoPE significantly improves multi-view understanding. Notably, performance with PRoPE continues to improve as the number of views increases during testing, whereas the Plücker Raymap approach does not exhibit the same trend. This suggests that encoding camera information solely through the Plücker Raymap is not an efficient way for the model to process and utilize the camera effectively.

5.5 Conclusion and Future Work

In this paper, we introduce PRoPE, a simple but effective method for incorporating camera information into multi-view transformer architectures. PRoPE offers several advantages: (1) it unifies both intrinsic and extrinsic camera conditioning, (2) it operates between pairs, ensuring invariance to world coordinate choice, and (3) it is compatible with FlashAttention [15] and thus induces minimal computational overhead.

However, PRoPE still has some limitations that warrant future investigation. Direct matrix multiplication of the projective matrix with Q/K/V vectors may introduce numerical instability when the matrix is ill-conditioned. Furthermore, incorporating "multi-frequency" encoding for the projective matrix—beneficial for positional encoding [33]—remains nontrivial. We also see potential in extending PRoPE to handle dynamic scenes and exploring its application in video-based tasks. We leave these challenges for future work.

Chapter 6

Conclusion

6.1 The Central Theme: Leveraging 3D Knowledge for Efficiency

A central theme of this thesis is that efficiency in 3D perception is not just a matter of faster computation, but is fundamentally enabled by the unique structure and properties of 3D data. The key factors that made the advances in this work possible stem from leveraging the inherent spatial and geometric structure of 3D data, which offers additional levers for algorithmic innovation.
First, *sparsity in 3D space* means that most of the volume is empty, and only a small fraction contains meaningful content. Methods like NerfAcc and gsplat exploit this property, focusing computation where it matters most and skipping redundant work. Second, the *volume rendering equation* provides a mathematical bridge between 2D images and 3D structure, as leveraged in NeRF-XL to enable efficient distributed rendering and training. Third, *multi-view geometry*—the relationships between images captured from different viewpoints—remains a powerful source of structure, as demonstrated by PRoPE's use of projective geometry to guide learning and inference. These are not new ideas; they are rooted in the foundations of computer vision and graphics. Yet, as this thesis shows, they remain vital tools for designing efficient algorithms in the era of deep learning. By combining classic insights with modern computational techniques, we can build systems that are not only more powerful, but also more practical and accessible.

6.2 Looking Ahead: The Future of 3D Perception

In this thesis, I have discussed two main approaches to 3D perception: optimization-based and learning-based methods. Each has its own strengths and limitations. Ideally, we would like 3D perception systems to be both accurate—with quantifiable error bounds, as in optimization-based solutions—and able to leverage prior knowledge from large datasets, as in learning-based approaches. In practice, learning-based methods can provide an initial estimate of 3D information—such as depth, camera pose, or even a NeRF or Gaussian Splatting model—which, even if imperfect, can

significantly accelerate subsequent optimization by providing a strong starting point rather than beginning from scratch.

This raises an important question: what form should this initial knowledge from learning-based models take? Recent trends point to two directions: feed-forward prediction models and generative models. In my view, these are not fundamentally different; both can be seen as forms of *conditional generative models*. Feed-forward models simply predict the mean or most likely outcome, ignoring the uncertainty inherent in the prediction.

From another perspective, conditional generative models also mirror how humans perceive the world. When we estimate the distance to an object, for example, we rarely know the exact value, but instead form a probabilistic sense—there is always some uncertainty in our perception.

Ideally, then, the learning-based component should provide a distribution over possible 3D interpretations, rather than a single deterministic guess. This leads to a new question: how can optimization-based methods refine an initial estimate when that estimate is itself a distribution, rather than a single prediction?

This is a central question I am constantly thinking about. Most existing optimization-based methods—such as NeRF, Gaussian Splatting, or SLAM—begin from a single estimate of the 3D structure. For example, InstantSplat [21] uses multi-view depth predictions to initialize a Gaussian Splatting model for rapid 3D reconstruction. But what if the depth prediction were a distribution, or included explicit uncertainty? Could this richer information be leveraged to further improve optimization?

A related challenge also arises recently in novel view synthesis with video models, which can generate high-fidelity images from limited observations. However, when attempting to recover 3D structure from these generated images, multi-view consistency often breaks down, since each view is generated independently as a sample from its own distribution. Some researchers address this by modeling the joint distribution of multiple views by generating all views at once, but this approach is computationally expensive. The solution that I'm imagining should be to develop optimization-based methods that accept distributions over pixels from each viewpoint, rather than single samples, and output something also as a distribution, instead of a deterministic 3D structure. This could reduce the need to model the full joint distribution in the learned model, saving computational resources and simplifying the learning problem.

Of course, these ideas remain open questions, and I do not yet have a concrete solution. However, I believe that the future of 3D perception lies in the integration of conditional generative models with optimization-based approaches that can refine and build upon an initial distributional estimate of 3D structure, and that the outputs of the optimization-based methods should still remain a distribution. Achieving this may require a fundamental rethinking of current 3D representations and a tighter coupling between learning-based and optimization-based methods.

Bibliography

[1] Yanqi Bao et al. “3D Gaussian Splatting: Survey, Technologies, Challenges, and Opportunities”. In: *arXiv preprint arXiv:2407.17418* abs/2407.17418 (2024). URL: https://api.semanticscholar.org/CorpusID:271404782.

[2] Jonathan T Barron et al. “Mip-nerf 360: Unbounded anti-aliased neural radiance fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5470–5479.

[3] Jonathan T Barron et al. “Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5855–5864.

[4] Jonathan T Barron et al. “Zip-NeRF: Anti-aliased grid-based neural radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 19697–19705.

[5] Jonathan T. Barron et al. “Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[6] Jonathan T. Barron et al. “Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields”. In: *International Conference on Computer Vision (ICCV)* (2021).

[7] Tyler Bonnen et al. “Evaluating multiview object consistency in humans and image models”. In: *Advances in Neural Information Processing Systems* 37 (2025), pp. 43533–43548.

[8] Arthur Brussee. *Brush - universal splats*. https://github.com/ArthurBrussee/brush. 2024.

[9] Mathilde Caron et al. “Emerging properties in self-supervised vision transformers”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 9650–9660.

[10] Anpei Chen et al. “Tensorf: Tensorial radiance fields”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 333–350.

[11] Guikun Chen and Wenguan Wang. “A survey on 3d gaussian splatting”. In: *arXiv preprint arXiv:2401.03890* (2024).

[12] Yuedong Chen et al. “Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images”. In: *European Conference on Computer Vision*. Springer. 2024, pp. 370–386.

[13] Yuedong Chen et al. “Mvsplat360: Feed-forward 360 scene synthesis from sparse views”. In: *Advances in Neural Information Processing Systems* 37 (2025), pp. 107064–107086.

[14] CivilAirPatrol. *Hurrican Michael Imageries*. `http://fema-cap-imagery.s3-website-us-east-1.amazonaws.com/Others/2018_10_FL_Hurricane-Michael/`.

[15] Tri Dao. “Flashattention-2: Faster attention with better parallelism and work partitioning”. In: *arXiv preprint arXiv:2307.08691* (2023).

[16] Matt Deitke et al. “Objaverse: A universe of annotated 3d objects”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 13142–13153.

[17] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186.

[18] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[19] Abhimanyu Dubey et al. “The llama 3 herd of models”. In: *arXiv preprint arXiv:2407.21783* (2024).

[20] Patrick Esser et al. “Scaling rectified flow transformers for high-resolution image synthesis”. In: *Forty-first international conference on machine learning*. 2024.

[21] Zhiwen Fan et al. “Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds”. In: *arXiv preprint arXiv:2403.20309* 2.3 (2024), p. 4.

[22] Jiemin Fang et al. “Fast Dynamic Radiance Fields with Time-Aware Neural Voxels”. In: *SIGGRAPH Asia 2022 Conference Papers*. 2022.

[23] Ben Fei et al. “3D Gaussian Splatting as New Era: A Survey”. In: *IEEE Transactions on Visualization and Computer Graphics* (2024), pp. 1–20. DOI: `10.1109/TVCG.2024.3397828`.

[24] Julian Fong et al. “Production volume rendering: Siggraph 2017 course”. In: *ACM SIGGRAPH 2017 Courses*. 2017, pp. 1–79.

[25] Sara Fridovich-Keil et al. “K-planes: Explicit radiance fields in space, time, and appearance”. In: *arXiv preprint arXiv:2301.10241* (2023).

[26] Clement Fuji Tsang et al. *Kaolin: A Pytorch Library for Accelerating 3D Deep Learning Research*. `https://github.com/NVIDIAGameWorks/kaolin`. 2022.

[27] Hang Gao et al. “Monocular dynamic view synthesis: A reality check”. In: *Advances in Neural Information Processing Systems*. 2022.

[28] Ruiqi Gao et al. “Cat3d: Create anything in 3d with multi-view diffusion models”. In: *arXiv preprint arXiv:2405.10314* (2024).

[29] Olga Golovneva et al. “Contextual Position Encoding: Learning to Count What’s Important”. In: *arXiv preprint arXiv:2405.18719* (2024).

[30] Vitor Guizilini et al. “Zero-Shot Novel View and Depth Synthesis with Multi-View Geometric Diffusion”. In: *arXiv preprint arXiv:2501.18804* (2025).

[31] Daya Guo et al. “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning”. In: *arXiv preprint arXiv:2501.12948* (2025).

[32] Yuan-Chen Guo. *Instant Neural Surface Reconstruction*. 2022. URL: `https://github.com/bennyguo/instant-nsr-pl`.

[33] Byeongho Heo et al. “Rotary position embedding for vision transformer”. In: *European Conference on Computer Vision*. Springer. 2024, pp. 289–305.

[34] Yining Hong et al. “3d concept learning and reasoning from multi-view images”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 9202–9212.

[35] Binbin Huang. *torch-splatting*. `https://github.com/hbb1/torch-splatting`. 2023.

[36] Yoonwoo Jeong, Seungjoo Shin, and Kibaek Park. *NeRF-Factory: An awesome PyTorch NeRF collection*. 2022. URL: `https://github.com/kakaobrain/NeRF-Factory/`.

[37] Haian Jin et al. “Lvsm: A large view synthesis model with minimal 3d inductive bias”. In: *arXiv preprint arXiv:2410.17242* (2024).

[38] Amirhossein Kazemnejad et al. “The impact of positional encoding on length generalization in transformers”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 24892–24928.

[39] Bingxin Ke et al. “Repurposing diffusion-based image generators for monocular depth estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 9492–9502.

[40] Bernhard Kerbl et al. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics* 42.4 (July 2023).

[41] Justin Kerr et al. “LERF: Language Embedded Radiance Fields”. In: *International Conference on Computer Vision (ICCV)*. 2023.

[42] Shakiba Kheradmand et al. “3D Gaussian Splatting as Markov Chain Monte Carlo”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2024.

[43] Alexander Kirillov et al. “Segment anything”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 4015–4026.

[44] Arno Knapitsch et al. “Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction”. In: *ACM Transactions on Graphics* 36.4 (2017).

[45] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. “Decomposing NeRF for Editing via Feature Field Distillation”. In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022. URL: `https://arxiv.org/pdf/2205.15585.pdf`.

[46] Xin Kong et al. “Eschernet: A generative model for scalable view synthesis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 9503–9513.

[47] George Kopanas. *Slang.D Gaussian Splatting Rasterizer*. `https://github.com/google/slang-gaussian-rasterization`. 2024.

[48] Black Forest Labs. *FLUX*. `https://github.com/black-forest-labs/flux`. 2024.

[49] Ruilong Li et al. “Nerfacc: Efficient sampling accelerates Nerfs”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 18537–18546.

[50] Ruilong Li et al. “Tava: Template-free animatable volumetric actors”. In: *arXiv preprint arXiv:2206.08929* (2022).

[51] Yixuan Li et al. “Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 3205–3215.

[52] Hanxue Liang et al. “Feed-Forward Bullet-Time Reconstruction of Dynamic Scenes from Monocular Videos”. In: *arXiv preprint arXiv:2412.03526* (2024).

[53] Chen-Hsuan Lin et al. “Barf: Bundle-adjusting neural radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5741–5751.

[54] Lu Ling et al. “Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 22160–22169.

[55] Haotian Liu et al. “Visual instruction tuning”. In: *Advances in neural information processing systems* 36 (2023), pp. 34892–34916.

[56] Ricardo Martin-Brualla et al. “Nerf in the wild: Neural radiance fields for unconstrained photo collections”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 7210–7219.

[57] Quan Meng et al. “Gnerf: Gan-based neural radiance field without posed camera”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6351–6361.

[58] Andreas Meuleman et al. “Progressively optimized local radiance fields for robust view synthesis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16539–16548.

[59] Ben Mildenhall et al. “Local light field fusion: Practical view synthesis with prescriptive sampling guidelines”. In: *ACM Transactions on Graphics (ToG)* 38.4 (2019), pp. 1–14.

[60] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *European Conference on Computer Vision ECCV*. 2020.

[61] Ben Mildenhall et al. “Nerf: Representing scenes as neural radiance fields for view synthesis”. In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[62] Takeru Miyato et al. “Gta: A geometry-aware attention mechanism for multi-view transformers”. In: *arXiv preprint arXiv:2310.10375* (2023).

[63] Thomas Müller et al. “Instant neural graphics primitives with a multiresolution hash encoding”. In: *arXiv preprint arXiv:2201.05989* (2022).

[64] Muyao Niu et al. “Mofa-video: Controllable image animation via generative motion field adaptions in frozen image-to-video diffusion model”. In: *European Conference on Computer Vision*. Springer. 2024, pp. 111–128.

[65] Michael Oechsle, Songyou Peng, and Andreas Geiger. “Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5589–5599.

[66] Keunhong Park et al. “Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields”. In: *arXiv preprint arXiv:2106.13228* (2021).

[67] Keunhong Park et al. “Nerfies: Deformable neural radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5865–5874.

[68] Janusch Patas. *gaussian splatting cuda*. `https://github.com/MrNeRF/gaussian-splatting-cuda`. 2023.

[69] Bowen Peng et al. “Yarn: Efficient context window extension of large language models”. In: *arXiv preprint arXiv:2309.00071* (2023).

[70] Julius Plucker. “Xvii. on a new geometry of space”. In: *Philosophical Transactions of the Royal Society of London* 155 (1865), pp. 725–791.

[71] Ofir Press, Noah A Smith, and Mike Lewis. “Train short, test long: Attention with linear biases enables input length extrapolation”. In: *arXiv preprint arXiv:2108.12409* (2021).

[72] Albert Pumarola et al. “D-nerf: Neural radiance fields for dynamic scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10318–10327.

[73] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[74] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PmLR. 2021, pp. 8748–8763.

[75] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *Journal of machine learning research* 21.140 (2020), pp. 1–67.

[76] Nikhila Ravi et al. “Accelerating 3d deep learning with pytorch3d”. In: *arXiv preprint arXiv:2007.08501* (2020).

[77] Daniel Rebain et al. “Derf: Decomposed radiance fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14153–14161.

[78] Christian Reiser et al. “Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 14335–14345.

[79] Christian Reiser et al. “MERF: Memory-Efficient Radiance Fields for Real-time View Synthesis in Unbounded Scenes”. In: *arXiv preprint arXiv:2302.12249* (2023).

[80] Konstantinos Rematas et al. “Urban radiance fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12932–12942.

[81] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[82] Mehdi SM Sajjadi et al. “Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 6229–6238.

[83] Kyle Sargent et al. “Zeronvs: Zero-shot 360-degree view synthesis from a single image”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 9420–9429.

[84] Connor Schenck et al. “Learning the RoPEs: Better 2D and 3D Position Encodings with STRING”. In: *arXiv preprint arXiv:2502.02562* (2025).

[85] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. “Self-attention with relative position representations”. In: *arXiv preprint arXiv:1803.02155* (2018).

[86] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. “Perceiver-actor: A multi-task transformer for robotic manipulation”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 785–799.

[87] Cameron Smith et al. “Flowmap: High-quality camera poses, intrinsics, and depth via gradient descent”. In: *arXiv preprint arXiv:2404.15259* (2024).

[88] Jürgen Sturm et al. “A benchmark for the evaluation of RGB-D SLAM systems”. In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pp. 573–580.

[89] Jianlin Su et al. “Roformer: Enhanced transformer with rotary position embedding”. In: *Neurocomputing* 568 (2024), p. 127063.

[90] Mohammed Suhail et al. “Light field neural rendering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8269–8279.

[91] Cheng Sun, Min Sun, and Hwann-Tzong Chen. “Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5459–5469.

[92] Cheng Sun, Min Sun, and Hwann-Tzong Chen. “Improved direct voxel grid optimization for radiance fields reconstruction”. In: *arXiv preprint arXiv:2206.05085* (2022).

[93] Towaki Takikawa et al. *Kaolin Wisp: A PyTorch Library and Engine for Neural Fields Research*. `https://github.com/NVIDIAGameWorks/kaolin-wisp`. 2022.

[94] Towaki Takikawa et al. "Neural geometric level of detail: Real-time rendering with implicit 3D shapes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11358–11367.

[95] Towaki Takikawa et al. "Variable bitrate neural fields". In: *ACM SIGGRAPH 2022 Conference Proceedings*. 2022, pp. 1–9.

[96] Matthew Tancik et al. "Block-NeRF: Scalable Large Scene Neural View Synthesis". In: *arXiv* (2022).

[97] Matthew Tancik et al. "Fourier features let networks learn high frequency functions in low dimensional domains". In: *Advances in neural information processing systems* 33 (2020), pp. 7537–7547.

[98] Matthew Tancik et al. "Nerfstudio: A modular framework for neural radiance field development". In: *ACM SIGGRAPH 2023 Conference Proceedings*. 2023, pp. 1–12.

[99] Genmo Team. *Mochi 1*. `https://github.com/genmoai/models`. 2024.

[100] Hugo Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[101] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. "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 (CVPR)*. June 2022, pp. 12922–12931.

[102] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. "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*. 2022, pp. 12922–12931.

[103] Benjamin Ummenhofer et al. "Demon: Depth and motion network for learning monocular stereo". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5038–5047.

[104] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[105] Dor Verbin et al. "Ref-nerf: Structured view-dependent appearance for neural radiance fields". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2022, pp. 5481–5490.

[106] John Von Neumann. "13. various techniques used in connection with random digits". In: *Appl. Math Ser* 12.36-38 (1951), p. 3.

[107] Peng Wang et al. "F2-NeRF: Fast Neural Radiance Field Training with Free Camera Trajectories". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4150–4159.

[108] Peng Wang et al. “Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction”. In: *arXiv preprint arXiv:2106.10689* (2021).

[109] Zirui Wang et al. “NeRF–: Neural radiance fields without known camera parameters”. In: *arXiv preprint arXiv:2102.07064* (2021).

[110] Rundi Wu et al. “CAT4D: Create Anything in 4D with Multi-View Video Diffusion Models”. In: *arXiv:2411.18613* (2024).

[111] Rundi Wu et al. “Reconfusion: 3d reconstruction with diffusion priors”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024, pp. 21551–21561.

[112] Tong Wu et al. “Recent advances in 3d gaussian splatting”. In: *Computational Visual Media* 10.4 (2024), pp. 613–642.

[113] Xiuchao Wu et al. “Scalable neural indoor scene rendering”. In: *ACM transactions on graphics* 41.4 (2022).

[114] Yuanbo Xiangli et al. “Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering”. In: *European conference on computer vision*. Springer. 2022, pp. 106–122.

[115] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. “Sun3d: A database of big spaces reconstructed using sfm and object labels”. In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1625–1632.

[116] Haofei Xu et al. “Unifying Flow, Stereo and Depth Estimation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[117] Linning Xu et al. “Grid-guided Neural Radiance Fields for Large Urban Scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 8296–8306.

[118] Lihe Yang et al. “Depth anything: Unleashing the power of large-scale unlabeled data”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 10371–10381.

[119] Lior Yariv et al. “Volume rendering of neural implicit surfaces”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 4805–4815.

[120] Chongjie Ye et al. “GauStudio: A Modular Framework for 3D Gaussian Splatting and Beyond”. In: *arXiv preprint arXiv:2403.19632* (2024).

[121] Zongxin Ye et al. “AbsGS: Recovering Fine Details for 3D Gaussian Splatting”. In: *arXiv preprint arXiv:2404.10484* (2024). arXiv: 2404.10484 [cs.CV].

[122] Alex Yu et al. “pixelnerf: Neural radiance fields from one or few images”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4578–4587.

[123] Alex Yu et al. “Plenoctrees for real-time rendering of neural radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5752–5761.

[124] Alex Yu et al. “Plenoxels: Radiance fields without neural networks”. In: *arXiv preprint arXiv:2112.05131* (2021).

[125] Zehao Yu et al. “Mip-Splatting: Alias-free 3D Gaussian Splatting”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (2024).

[126] Zehao Yu et al. *SDFStudio: A Unified Framework for Surface Reconstruction*. 2022. URL: `https://github.com/autonomousvision/sdfstudio`.

[127] Xiaohua Zhai et al. “Sigmoid loss for language image pre-training”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 11975–11986.

[128] Jason Y Zhang et al. “Cameras as rays: Pose estimation via ray diffusion”. In: *arXiv preprint arXiv:2402.14817* (2024).

[129] Kai Zhang et al. “Nerf++: Analyzing and improving neural radiance fields”. In: *arXiv preprint arXiv:2010.07492* (2020).

[130] Tinghui Zhou et al. “Stereo magnification: learning view synthesis using multiplane images”. In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), pp. 1–12.

[131] Chenming Zhu et al. “Llava-3d: A simple yet effective pathway to empowering lmms with 3d-awareness”. In: *arXiv preprint arXiv:2409.18125* (2024).