THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Neural Rendering for Autonomous Driving

CARL LINDSTRÖM

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2026

Neural Rendering for Autonomous Driving

CARL LINDSTRÖM
ISBN 978-91-8103-408-0

Acknowledgements, dedications, and similar personal statements in this thesis, reflect the author's own views.

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5865
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000

Cover:
Reconstruction of a driving scene using SplatAD.

Printed by Chalmers Digital Printing
Gothenburg, Sweden, May 2026

Neural Rendering for Autonomous Driving

CARL LINDSTRÖM
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Autonomous driving holds the potential to fundamentally transform transportation, but realizing that potential requires rigorous testing across a vast and diverse range of scenarios. Exhaustive real-world testing is neither safe for exploring edge cases nor practical at the scale required for adequate coverage, making high-fidelity simulation an essential component of the development and validation pipeline. This thesis addresses the challenge of building digital twins for autonomous driving: data-driven virtual replicas of real-world scenes that faithfully reproduce sensor observations while remaining editable for counterfactual testing. The first contribution is NeuRAD, a neural simulator that jointly reconstructs camera and lidar data from recorded driving sequences. By explicitly modeling key sensor characteristics, including rolling shutter effects, beam divergence, and non-returning lidar rays, NeuRAD achieves state-of-the-art rendering fidelity across multiple autonomous driving datasets. The second contribution is SplatAD, which replaces the volumetric representation of NeuRAD with 3D Gaussian primitives and purpose-built rasterization algorithms. This yields real-time rendering of both camera and lidar data at competitive fidelity, while reducing training and inference times by an order of magnitude, making large-scale simulation substantially more practical. The third contribution is IDSplat, which eliminates the reliance on human-annotated 3D bounding boxes required by both NeuRAD and SplatAD. By combining vision foundation models with classical matching and estimation techniques, IDSplat achieves annotation-free dynamic scene reconstruction that generalizes zero-shot to new datasets. The fourth contribution is a highly parallelizable GPU algorithm for constructing large-scale 3D Voronoi and power diagrams, addressing a key computational bottleneck in mesh-based neural rendering and enabling larger, higher-fidelity scene representations. Finally, the thesis investigates the real-to-sim gap: the discrepancy between how autonomous systems perceive real and rendered sensor data. Through large-scale evaluation, correlations between rendering quality and downstream perception performance are identified, and fine-tuning strategies are developed that improve model robustness to simulation artifacts without compromising real-world performance. Together, these contributions advance the state of neural simulation for autonomous driving, bringing scalable, high-fidelity virtual testing closer to practical deployment.

Keywords: Autonomous driving, sensor simulation, neural rendering, novel view synthesis, neural radiance fields, 3D Gaussian splatting

List of Publications

This thesis is based on the following publications:

[A] Adam Tonderski†, **Carl Lindström**†, Georg Hess†, William Ljungbergh, Lennart Svensson, Christoffer Petersson, "NeuRAD: Neural Rendering for Autonomous Driving". *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[B] Georg Hess†, **Carl Lindström**†, Maryam Fatemi, Christoffer Petersson, Lennart Svensson, "SplatAD: Real-Time Lidar and Camera Rendering with 3D Gaussian Splatting for Autonomous Driving". *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[C] **Carl Lindström**†, Mahan Rafidashti†, Maryam Fatemi, Lars Hammarstrand, Martin R. Oswald, Lennart Svensson, "IDSplat: Instance-Decomposed 3D Gaussian Splatting for Driving Scenes". To be published in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Findings (CVPRF)*, 2026.

[D] Bernardo Taveira†, **Carl Lindström**†, Maryam Fatemi, Lars Hammarstrand, Fredrik Kahl, "Scalable GPU Construction of 3D Voronoi and Power Diagrams". To be published in *ACM SIGGRAPH 2026 Conference Proceedings (SIGGRAPH)*, 2026.

[E] **Carl Lindström**†, Georg Hess†, Adam Lilja, Maryam Fatemi, Lars Hammarstrand, Christoffer Petersson, Lennart Svensson, "Are NeRFs ready for autonomous driving? Towards closing the real-to-simulation gap". *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024.

Other publications by the author, not included in this thesis, are:

[F] J. Strietzel, T. Fischer, **C. Lindström**, M. Pollefeys, X. Wang, "FiG3D: Fisher-Guided Generative Priors for 3DGS Optimization". In submission.

[G] W. Ljungbergh, A. Lilja, A. Tonderski, A. Laveno, **C. Lindström**, W. Verbeke, J. Fu, C. Petersson, L. Hammarstrand, M. Felsberg, "GASP: Unifying geometric and semantic self-supervised pre-training for autonomous driving". *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2026.

†These authors contributed equally to this work.

[H] T. Li, Y. Qiu, Z. Wu, **C. Lindström**, P. Su, M. Nießner, H. Li, "MTGS: Multi-Traversal Gaussian Splatting". In submission.

[I] D. Hagerman, R. Naeem, J. Lindqvist, **C. Lindström**, F. Kahl, L. Svensson, "SwInception - Local Attention Meets Convolutions". *International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI)*, 2024.

[J] M. Alibeigi, W. Ljungbergh, A. Tonderski, G. Hess, A. Lilja, **C. Lindström**, D. Motorniuk, J. Fu, J. Widahl, C. Petersson, "Zenseact Open Dataset: A large-scale and diverse multimodal dataset for autonomous driving". *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[K] E. Jernheden[†], **C. Lindström**[†], R. Persson[†], M. Wedenmark[†], E. Erős, S. Roselli, K. Åkesson, "Comparison of exact and approximate methods for the vehicle routing problem with time windows". *IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020.

Contents

Acknowledgments

This thesis was far from a solo effort. I have been fortunate to be surrounded by people who not only made this possible but also made the journey genuinely fun.

First, I would like to thank my supervisors. Thank you, Lennart, Lars, and Maryam, for your excellent advice throughout the years and for providing an environment that encouraged exploration and collaboration. You have shown me tremendous support and encouragement, and probably also a great deal of patience.

I would also like to thank all the great people I have met along the way, at conferences, courses, and study trips, as well as my colleagues at Chalmers and Zenseact. Thank you to everyone in the AGP team, and to Carl, Mats, Jonas, and Gabriel for supporting and coaching the group during these years.

A special thank you, of course, to William, Georg, Tonderski, and Lilja, with whom I have shared most of this journey. From bug hunting, deadline all-nighters, and complaining about R2, to celebration beers, Tokyo karaoke, and traveling the world. Your company made this journey far more fun than I thought a PhD was allowed to be.

I also want to express my gratitude to all my co-authors, for the collaborations and contributions that ultimately shaped this thesis. A special thank you to Martin, for your invaluable support and guidance, and for at times taking on an unofficial supervisor role. And to Mahan and Bernardo, for excellent collaborations and great company during crunch times.

Last but not least, I want to thank my family and friends for all the support during these years. To BeachBoys, for putting up with paper writing during ski trips. To Verkstadsvägen 4 and Granshult, for being a second home. To my siblings Moa and August, for your encouragement and patience during deadline times. To my parents, for being an endless source of support and encouragement, and for raising me to have the courage and curiosity to try new things. And finally to you, Wilma. Thank you for supporting me through the most stressful of times, for always staying positive and bringing out the laughter in me, and most of all, for filling these years with joy and love.

Carl Lindström
Gothenburg, May 2026

Funding

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and by Zenseact AB through their industrial PhD program.

Acronyms

AD:	Autonomous Driving
ADAS:	Advanced Driver Assistance System
CNN:	Convolutional Neural Network
DL:	Deep Learning
GPS:	Global Positioning System
IMU:	Inertial Measurement Unit
MLP:	Multi-Layer Perceptron
NeRF:	Neural Radiance Field
NVS:	Novel View Synthesis

Part I

Overview

CHAPTER 1

Introduction

The future of transportation might not be quite as exciting as the flying cars depicted in sci-fi movies, but self-driving cars come pretty close. Autonomous driving (AD), also known as self-driving or driverless technology, represents one of the most transformative promises in modern transportation history. At its core, AD refers to vehicles that can perceive their environment, plan ahead, and make driving decisions with minimal or no human intervention. In its essence, it removes the task of driving from the human driver, in any weather, time of day, or traffic situation.

The potential benefits are easy to imagine: from saving lives through fewer road accidents, to more efficient, sustainable use of urban space. Early robotaxi deployments have already demonstrated safety improvements within restricted operational design domains—predefined sets of operating conditions such as location, weather, and traffic complexity [1]—and the number of consumer vehicles equipped with advanced driver assistance (ADAS) and AD capabilities continues to grow [2].

However, deploying autonomous systems without human supervision requires an exceptionally high degree of safety and reliability. The systems must be rigorously tested and validated to ensure that they can safely handle any scenario they might encounter in the real world. Such validation is difficult and complex, and is one of the limiting factors in deploying autonomous systems to wider operational design domains.

This thesis addresses this validation challenge by exploring how *digital twins*, high-fidelity virtual replicas of real-world environments, can be created from real-world data and used to enable more comprehensive testing in simulation. The work spans multiple aspects of this problem, including scene representation, sensor modeling, computational efficiency, scalability, and realism. Across these contributions, the unifying objective is the development of digital twins that help advance the development and deployment of safe autonomous vehicles. Although the primary focus is AD, many of the proposed methods generalize naturally to related domains that require reconstruction of virtual 3D environments for realistic sensor simulation, such as robotics, augmented reality, and embodied AI systems.

The remainder of this introduction outlines the specific challenges addressed in this thesis and concludes with an overview of the remaining chapters.

1.1 Challenges addressed by the thesis

This thesis addresses the following key challenges in building digital twins for AD:

Building high-fidelity twins from real-world data Validating AD systems requires access to a large and diverse set of scenarios to ensure safe operation in all settings prior to deployment. In practice, however, collecting safety-critical events in the real world scales poorly and offers limited controllability, particularly as the intended operating conditions expand and scenario coverage requirements increase. Controlled testing on proving grounds can partially alleviate this limitation through the use of inflatable vehicles and crash dummies. Nevertheless, such setups lack the richness and diversity of real-world environments and are both costly and time-consuming to scale.

This thesis addresses the challenge of constructing high-fidelity digital twins directly from real-world data. The goal is not only to faithfully replay recorded scenarios, but also to enable controlled modifications that support (1) closed-loop simulation of collected data and (2) transformation of nominal situations into safety-critical ones.

Traditionally, controllability has been achieved using fully synthetic data generated by simulation engines. In these approaches, scenes are assembled from manually created assets whose behavior is governed by predefined rules or data-driven models. Planning and control systems can often be validated at an abstract level using object-level interfaces alone. However, validating perception systems, or the full AD system end-to-end, requires realistic simulation of raw sensor data, which is typically provided by physics-based game engines [3], [4].

Achieving sufficiently high sensor realism with such engines requires substantial engineering effort, deep domain expertise, and detailed knowledge of proprietary sensor specifications. Furthermore, engine-based simulation pipelines depend heavily on artist-designed assets, which limits scalability and constrains the diversity of generated scenarios.

Recent advances in differentiable scene representations for novel view synthesis, often referred to as *neural rendering* [5], [6], offer a compelling alternative. Rather than manually constructing environments, 3D scenes can be learned directly from real-world sensor data. These learned representations, coupled with differentiable rendering, enable photorealistic synthesis from novel viewpoints. As a result, recorded real-world data can be explored from alternative perspectives to support closed-loop simulation. Moreover, with additional instance-level scene decomposition dynamic objects can be explicitly represented, allowing controlled manipulation of individual objects and their trajectories.

Despite their promising fidelity, existing approaches face important limitations in the context of AD. First, neural rendering methods are generally not designed primarily for outward-looking scenes, multi-camera rigs, or multi-modal sensor setups that combine cameras with lidar, which limits their practical utility for AD simulation. Second, they struggle with the large-scale environments and high-speed dynamic characteristics of AD scenes.

To address these challenges, we introduce new techniques for neural rendering that jointly model camera and lidar data in the large-scale, dynamic settings typical of AD. In Paper A [7], we present NeuRAD, a neural simulator that achieves state-of-the-art performance in both camera and lidar rendering. By explicitly modeling key sensor characteristics, including rolling shutter effects, beam divergence, camera-specific exposure, and non-returning lidar rays, NeuRAD captures phenomena that prior methods ignore, leading to substantial gains in rendering quality across multiple datasets. Papers B [8] and C [9] build on these capabilities while tackling the further challenges of computational efficiency and scalable scene reconstruction, as discussed next.

Scaling neural rendering for autonomous driving simulation The techniques proposed in Paper A [7] enable high-fidelity simulation of both camera and lidar data. As demonstrated in [10], NeuRAD enables the transformation of recorded nominal driving scenarios into safety-critical ones that force emergency maneuvers, providing valuable test cases for AD validation. However, although the renderings are sufficiently realistic for sensor-level closed-loop simulation, the underlying represen-

tation limits efficiency in terms of training and inference time. This, in turn, restricts the number of scenarios that can be reconstructed and simulated within a given time budget.

To address this limitation, we develop SplatAD (Paper B [8]), which instead uses 3D Gaussian primitives as its scene representation and employs rasterization for efficient rendering. Through novel rasterization techniques and sensor modeling tailored to the explicit Gaussian representation, SplatAD enables real-time, high-fidelity rendering of both camera and lidar data. Notably, the method achieves state-of-the-art performance while being an order of magnitude faster than NeuRAD, and can achieve competitive fidelity in a fraction of the training time. These improvements make neural simulators substantially more practical for autonomous driving applications.

While both NeuRAD and SplatAD demonstrate photorealistic rendering, with the latter providing substantial efficiency gains, both approaches share an important limitation when scaling to large scenario sets. To model dynamic actors as separate instances, they rely on human-annotated object trajectories and 3D bounding boxes. Although common in the field, such annotations are expensive and time-consuming to obtain at scale.

In Paper C [9], we address this challenge by introducing IDSplat, a framework for reconstructing and rendering dynamic scenes without requiring human annotations. By combining vision foundation models as data priors with classical matching and estimation techniques, IDSplat achieves performance competitive with SplatAD while eliminating manual labeling and enabling zero-shot generalization to new datasets. Together, these advances enable the construction of digital twins that are not only high-fidelity but also scalable to large simulation workloads.

Both SplatAD and IDSplat highlight the advantages of explicit scene representations, particularly in terms of inference efficiency enabled by purpose-built rasterization techniques. However, rasterization also introduces limitations. The approximations that enable high speed also make accurate modeling of light transport phenomena more challenging, complicating the synthesis of realistic reflections, refractions, and shadows, and limiting support for more complex sensor models.

Recent research has explored mesh-based neural rendering using polyhedral meshes derived from Voronoi diagrams, enabling explicit representations that can be ray traced with efficiency close to rasterization-based methods [11]. However, ray tracing in this setting requires knowledge of mesh connectivity, which necessitates frequent updates of the dual Delaunay tetrahedralization during training when the mesh is optimized. Consequently, both training efficiency and scalability to a higher number of primitives are limited by the computational cost of maintaining this connectivity.

To address this bottleneck, Paper D introduces a highly parallelizable GPU algorithm for fast construction of large-scale 3D Voronoi and power diagrams. The method builds on parallel individual cell construction via iterative clipping against bisecting planes from carefully selected candidate neighbors. Beyond matching state-of-the-art performance for Delaunay tetrahedralization on small problem sizes, the proposed approach scales robustly to large point sets and diverse spatial distributions, with increased efficiency and succeeding where previous methods fail. Moreover, the method naturally generalizes to power diagrams, which are not supported by many existing methods. This has a direct practical impact on mesh-based neural rendering by enabling larger scene representations and, consequently, improved reconstruction quality. In the context of AD, where scenes are large and rendering efficiency is critical, this brings mesh-based representations closer to a viable alternative to rasterization, offering a path toward accurate modeling of reflections, refractions, shadows, and complex camera models without sacrificing efficiency.

Simulation validity and the real-to-sim gap Papers A–D focus on constructing digital twins that are both high-fidelity and computationally scalable. However, for simulation results to be meaningful, it is essential to verify that the AD system perceives real and rendered data in a consistent manner. To this end, Paper E [12] investigates the *real-to-sim gap* for neural simulators in the context of AD.

We identify meaningful correlations between image reconstruction metrics and downstream model performance, providing insights into the expected performance of the neural simulator under different conditions. In addition, we study fine-tuning strategies in which rendered data is systematically leveraged to improve real-world performance.

Furthermore, we introduce a complementary perspective on the real-to-sim gap that focuses on improving model robustness to simulation artifacts, rather than solely increasing rendering fidelity. Together, these findings offer practical guidance on the reliability and effective use of neural simulators for both validation and training augmentation.

1.2 Outline

The remainder of this thesis is organized as follows. Chapter 2 situates the work within the broader context of AD. It reviews the structure of modern AD systems, discusses the sensor modalities that simulation must reproduce, and examines current validation strategies and their limitations. The chapter closes by surveying the land-

scape of neural simulators for AD and identifying the open challenges that motivate the contributions of this thesis.

Chapter 3 develops the technical foundations of neural rendering. It introduces the novel view synthesis problem, the two fundamental rendering paradigms used throughout the thesis, and the evaluation metrics employed in subsequent chapters. It then traces the progression of scene representations from implicit volumetric fields through explicit Gaussian primitives to mesh-based formulations, providing the background needed to understand the design choices made in Papers A, B, and D.

Chapter 4 turns from general-purpose neural rendering to the specific demands of AD. It discusses how large unbounded environments, fast ego-motion, and dynamic actors complicate scene parameterization, how multi-sensor rigs require dedicated modeling of camera and lidar characteristics, and how training and inference efficiency, together with annotation requirements, determine whether a method can scale to the scenario volumes required for validation. Papers A, B, C, and D are positioned within this discussion as concrete responses to these challenges.

Chapter 5 addresses the question of simulation validity. It defines the real-to-sim gap as a complementary problem to the more familiar sim-to-real gap, examines the relationship between rendering quality metrics and downstream perception performance, and presents practical strategies for mitigating the gap through both data augmentation and model robustness. Paper E is discussed within this chapter.

Chapter 6 summarizes the included papers and their contributions, and Chapter 7 finally concludes the thesis by summarizing the contributions, reflecting on the broader validation challenge, and outlining promising directions for future research.

CHAPTER 2

The role of simulation in autonomous driving

While driving along a highway, a vehicle in the adjacent lane suddenly swerves into your path. You brake sharply, avoiding a collision but unsettling your passengers. In the aftermath, a natural set of counterfactual questions arises: What if the other vehicle had accelerated more aggressively? What if the braking had occurred earlier? What if a lane change had been executed instead? For an autonomous vehicle, the ability to systematically revisit and vary such scenarios, generating new sensor observations for each hypothetical, would be enormously valuable for development and validation.

In practice, however, this capability remains limited. Contemporary validation pipelines rely predominantly on two approaches: replaying previously recorded sensor logs without modification, or accumulating additional real-world driving miles. Log replay enables repeatable evaluation but is inherently open-loop; the autonomous system cannot influence the environment, because no new sensor observations are generated in response to its actions. Real-world testing, in contrast, provides closed-loop interaction but is costly, time-consuming, and constrained by safety requirements, thereby limiting controllability and ultimately scenario coverage. As a result, neither approach alone scales to the breadth and diversity of situations required for robust validation.

These limitations have motivated a growing interest in high-fidelity closed-loop simulation, and in particular in digital twins constructed directly from real-world data.

In the context of AD, a digital twin is a virtual replica of a recorded scene in which static structure, dynamic actors, and sensor behavior are faithfully reproduced. Such a representation supports controlled modifications of the environment, inserting or removing traffic participants, altering trajectories, or executing alternative ego-vehicle behaviors, while continuously generating sensor observations consistent with the modified scene.

By enabling the autonomy stack to interact with a realistic yet fully controllable virtual environment, digital twins offer a promising pathway toward scalable and safe system evaluation. Compared to manually authored virtual worlds, data-driven digital twins have the potential to better capture real-world complexity and rare edge cases while reducing content creation effort. At the same time, achieving the required levels of geometric fidelity, photorealism, physical consistency, and computational efficiency remains an open research challenge. This chapter reviews the fundamentals of AD, prevailing testing and validation methodologies, and the role of simulation within them, before surveying the current digital twin landscape and positioning this thesis within it.

2.1 Autonomous driving systems

Autonomous driving is fundamentally a robotics problem. The vehicle must perceive its environment from a set of sensors, plan a path to reach a target destination, and execute the plan while ensuring safety for itself and other road users. While this might seem like a simple task at first glance, the environment is complex and dynamic, with unpredictable road users, varying weather conditions, and a large set of traffic rules, both formal and normative, that varies between countries and regions. To tackle this complexity, the task has historically been divided into a series of sub-tasks, each handled by a specialized module. A perception system detects relevant objects in the environment, including other road users, lane markings, and traffic lights; a prediction system anticipates how those objects will behave; a planning system computes safe paths; and a control system executes the planned path [13]. This modular approach has the advantage that each component can be developed, optimized, and tested in isolation, with clear interfaces between them. Today, many of these components are implemented as neural networks trained on large datasets for their specific tasks [14]–[17].

While modular systems allow for decoupled development, the interfaces between sub-systems introduce information bottlenecks. The planning module can reason only about what is passed across the defined interface, not about the overall context of the

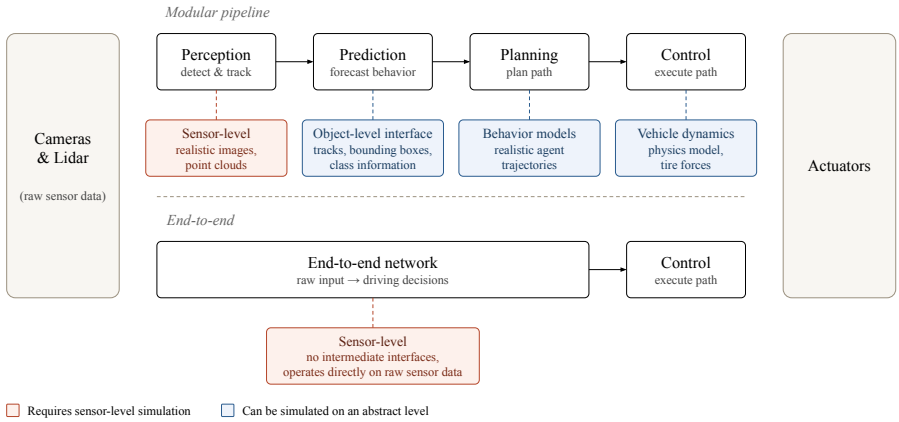


Figure 2.1: Validation requirements across pipeline architectures. In a modular pipeline (top), each module can in principle be validated using simulation at the appropriate level of abstraction: planning and control require only behavioral and dynamic models, while prediction can operate on object-level interfaces. Only perception validation demands sensor-level simulation. In an end-to-end system (bottom), no intermediate interface exists; the network consumes raw sensor data directly, so sensor-level simulation is required throughout. Digital twins address this most demanding requirement.

scene. As the diversity and complexity of the operational domain grow, so does the difficulty of defining interfaces that transfer all decision-relevant information without loss.

An alternative that has gained considerable traction is to reduce the number of sub-systems by training neural networks to directly output driving decisions from raw sensor data [18]–[21]. This end-to-end approach allows the system to reason about the full context of a situation without human-defined information bottlenecks. However, it presents its own challenges: the absence of isolated components makes debugging harder and does not allow for decoupled optimization or testing of individual capabilities. Hybrid approaches also exist, such as architectures in which perception and planning are trained jointly but control remains a separate sub-system, or fully end-to-end models that retain intermediate outputs for auxiliary supervision and easier diagnostics. Which paradigm is best suited for AD remains an active area of research and industry discussion.

These architectural differences have direct consequences for simulation. As Fig-

ure 2.1 illustrates, modular systems permit partial validation of individual components at lower levels of abstraction: prediction, planning, and control can be tested without sensor-level simulation. Full-system validation, however, demands sensor-level simulation in both paradigms, and in end-to-end architectures there is no fallback for component-level testing.

Despite their architectural differences, all contemporary AD systems share a critical dependency on data. Much of the recent progress in the field has been driven by advances in deep learning, where multi-layered neural networks extract patterns from data and generalize them to previously unseen situations. Achieving robust generalization to a vehicle's intended operational domain therefore requires training datasets that are both extensive and diverse, covering a broad range of scenarios representative of real-world conditions. Crucially, this dependency does not end at training. The same diversity requirement applies to validation: a system must be tested across a wide range of conditions to provide credible evidence of safety before deployment. Meeting both demands calls for large volumes of real-world data as well as the ability to generate and simulate virtual scenarios at scale, which is the central challenge this thesis addresses.

2.2 Sensor modalities and simulation requirements

An autonomous vehicle perceives its environment through a suite of complementary sensors. Cameras capture rich visual information about the scene, lidar provides accurate depth measurements of the surroundings, and radar provides robust velocity estimates and range measurements even in adverse weather. Beyond these primary perception sensors, vehicles incorporate GPS, IMU, and odometry sensors for localization, and often consume high-definition map information as a structured data source. Exact sensor configurations vary considerably between manufacturers and operational domains and remain a topic of ongoing industry discussion.

What matters for simulation, however, is not the diversity of configurations per se, but what each sensor type requires in order to be faithfully reproduced.

Cameras Cameras form images by projecting a three-dimensional scene through a lens onto a two-dimensional image plane, recording each pixel as the integral of incoming light over an exposure interval. Faithful simulation of a camera, therefore, requires an accurate geometric model of the lens and projection, correct handling of

(a) (b) (c)

Figure 2.2: Camera effects relevant to AD simulation. (a) Rolling shutter distortion causes vertical structures to appear skewed when the vehicle is in motion, as the sensor reads pixel rows sequentially rather than simultaneously. (b–c) The same scene captured by two different cameras in a multi-camera rig, showing noticeable differences in color and brightness arising from per-camera exposure and imaging characteristics. Red dashed rectangles highlight the affected regions.

exposure, and realistic rendering of scene appearance, including lighting, material reflectance, and shadows. Two effects deserve particular attention in the context of autonomous vehicles. First, because many cameras use a rolling shutter, which reads pixel rows sequentially rather than simultaneously, each row of an image is captured at a slightly different moment in time. When the vehicle or scene objects are in motion, this produces characteristic distortions: structures appear skewed or leaning, as illustrated in Figure 2.2a. Second, in a multi-camera setup, individual cameras may exhibit different appearances of the same scene due to differences in exposure settings and imaging characteristics. As shown in Figures 2.2b–2.2c, the same sun shade captured by two different cameras of the same sensor rig can differ noticeably in color and brightness. Faithful simulation must account for both effects to generate images that are consistent with what each specific camera observes.

Lidar Lidar sensors emit laser pulses and measure the time of flight of returning echoes to produce dense point clouds of the environment. Automotive lidar sensors are broadly categorized into spinning mechanical designs, which sweep a 360-degree field of view and are the dominant type in public research datasets [22]–[27], and solid-state designs with a fixed field of view that are increasingly prevalent in production vehicles due to their lower cost and compact form factor [28]. Faithful simulation requires accurate modeling of the beam geometry, including the angular pattern of

emitters and the divergence of individual beams; the intensity of returned pulses, which depends on surface reflectance and incidence angle; and the behavior of rays that return no echo, due to specular surfaces, transparent materials, or range limits. As with cameras, motion introduces distortions during a scan: since scanning lidars acquire points sequentially, each return is measured at a slightly different instant, meaning both ego-vehicle motion and moving objects in the scene can cause geometric inconsistencies in the resulting point cloud.

Radar Radar sensors measure reflected electromagnetic waves to estimate range and radial velocity of surrounding objects. They are notably robust to adverse weather conditions and darkness, which makes them a common element of production sensor suites. However, faithful simulation of radar at the raw signal level is substantially more complex than for cameras or lidar, as it requires modeling wave propagation, multi-path reflections, and clutter in a physically accurate way. Existing public automotive datasets typically provide post-processed radar point clouds or object-level detections rather than raw waveforms [26], [27], limiting the signal-level information available for simulation development.

For these reasons, raw-signal radar simulation has received comparatively little attention in the AD context. The scarcity of raw radar waveforms in public automotive datasets has limited the development of data-driven reconstruction methods. Works such as Radar Fields [29] and DART [30] have demonstrated neural rendering of raw radar signals, but do so using custom sensor rigs rather than automotive platforms. Simulation of post-processed automotive radar point clouds has been explored more recently [31], [32], but the field remains in its early stages compared to cameras and lidar.

Consequently, radar simulation falls outside the scope of this thesis, which focuses on cameras and lidar as the primary high-bandwidth perception sensors covered by the public automotive datasets considered. Accurate simulation of the full sensor suite is, in principle, required for complete closed-loop testing of an autonomy stack. The camera and lidar modalities addressed in this thesis represent the core of that requirement, and the framework developed can, in principle, be extended to accommodate additional sensor types as simulation methods for them mature.

2.3 Validation strategies and their limitations

Validating an AD system is, at its core, a coverage problem. The system must perform safely across an enormous space of possible situations: varying weather, lighting, road geometries, traffic densities, and the unpredictable behavior of other road users. Rare but safety-critical events, such as a pedestrian stepping into the road from behind an occluding object, or a vehicle running a red light, may be encountered only once in millions of kilometers of driving yet must be handled correctly every time. Exhaustively testing an AD system in the real world to achieve such coverage would require billions of miles of driving [33], an undertaking that is not feasible within any practical timeframe. In practice, validation is therefore distributed across multiple complementary strategies, each offering a trade-off between realism, cost, and scenario coverage, and each leaving a specific gap that the next tries to fill.

Open-loop testing Open-loop testing represents the most fundamental form of validation. It refers to an evaluation setting in which the system processes predefined inputs without receiving feedback from its own outputs; effectively a replay of recorded scenarios. The autonomous system consumes sensor data from previously recorded drives, and its outputs, such as perceived objects, predicted trajectories, or planned maneuvers, are assessed offline against annotated ground truth or expert driver actions. Because these outputs do not influence the environment or alter subsequent inputs, evaluation is limited primarily by computational resources and can be automated and repeated efficiently across large datasets and multiple software versions. This scalability is valuable, but it comes with a fundamental limitation: open-loop testing does not capture the consequences of the system's decisions. The evaluation indicates what the system would decide in a frozen snapshot of the scenario, but not how that decision would affect subsequent events. A planned lane change may trigger reactions from surrounding traffic or introduce new occlusions for the perception system; such interaction-dependent effects remain invisible to open-loop evaluation.

Shadow testing A variant of open-loop evaluation is shadow deployment, in which the autonomous system operates in parallel with a live production or human-driven vehicle, receiving identical sensor inputs while its outputs are recorded for offline analysis rather than actuated. This enables direct comparison between the autonomous system's decisions and those of the human driver or production system, and can be distributed across an entire fleet for large-scale exposure to real-world diversity. Shadow deployment therefore retains the scalability advantage of open-loop

testing while providing naturally diverse data collection. However, it shares the same core limitation: because the system's decisions are never actuated, interaction effects and error compounding over time remain unobserved.

Real-world testing Despite its costs and practical limitations, real-world testing remains an indispensable component of system validation. Experiments in controlled environments such as closed test tracks enable repeatable evaluation of specific scenarios, including emergency braking responses, obstacle avoidance maneuvers, or performance under varying road surface conditions. Scenarios can be executed without risk to public road users, and conditions can be standardized across multiple test runs. Controlled testing is also a key instrument for standardized safety certification processes such as Euro NCAP [34].

Beyond controlled settings, real-world validation also occurs through supervised operation in public traffic, where a trained safety driver monitors the autonomous system and is prepared to intervene at any moment. This exposes the system to authentic traffic dynamics and genuine interactions among road users but is resource-intensive and provides limited control over scenario coverage. Safety-critical events are rare by nature and cannot be reliably encountered on demand; achieving meaningful coverage demands large fleets, substantial personnel, and enormous investment in time. Real-world testing is necessary but insufficient on its own. The scale of coverage required for robust validation and efficient iterative development cannot be reached without simulation.

Simulation-based testing Simulation addresses the coverage and scalability problems that real-world testing cannot overcome on its own. It enables precise control of environmental conditions and allows safety-critical scenarios to be generated and executed repeatedly without endangering road users. Regression testing following software updates becomes practical, as large scenario suites can be run automatically and consistently without physical vehicles or drivers.

The validity of simulation-based results, however, depends critically on fidelity. For simulation to produce meaningful evidence, the autonomous system must behave in the virtual environment as it would under real-world conditions, requiring accurate reproduction of sensor inputs, realistic modeling of other traffic participants, and credible vehicle dynamics. Insufficient fidelity risks producing misleading conclusions: a system may appear robust in simulation yet fail in the real world, or may be prematurely penalized due to unrealistic simulation artifacts.

Traditionally, high-fidelity simulators have been built using game engines and

manually authored 3D assets [3], [4]. These environments offer controllability and physical consistency, but their fidelity depends on the quality and diversity of hand-crafted content, which is expensive to produce and may not capture the richness and statistical diversity of real-world driving. The discrepancy between the system's behavior in simulation and in the real world, commonly referred to as the *sim-to-real gap*, is precisely the limitation that motivates a data-driven alternative. A second line of recent work pursues data-driven simulation through generative *world models* that synthesize sensor data, typically video, directly from learned distributions of real driving [35]–[39]. While such models offer impressive visual realism and scale naturally with data, the controllability and 3D consistency required for closed-loop sensor simulation remain open challenges, and these approaches are discussed in more detail in Chapter 4. Digital twins constructed directly from real-world sensor data address the sim-to-real gap by learning scene representations from recorded drives, inheriting the diversity and complexity of real traffic while retaining the controllability that simulation requires.

2.4 Neural simulation: landscape and open challenges

The past few years have seen rapid development of a family of techniques, collectively referred to as *neural rendering*, in which a scene representation is learned directly from captured sensor data so that novel observations can be synthesized from viewpoints that were never recorded. Neural Radiance Fields (NeRFs) [5] and 3D Gaussian Splatting (3DGS) [40] are the two dominant paradigms, and the technical details of both are the subject of Chapter 3. For the purposes of the present chapter, what matters is the capability these methods provide: given a set of posed sensor observations, they produce an optimized scene representation that can be rendered from alternative viewpoints with a fidelity that approaches the original recording. Applied to AD data, this opens a path toward the data-driven digital twins introduced earlier, in which the static structure and dynamic contents of a recorded scene are reconstructed from the ego vehicle's own sensors and can subsequently be re-rendered along modified trajectories or with altered actor configurations.

The result is a simulation paradigm that is conceptually distinct from game-engine-based alternatives. Rather than authoring a virtual world from predefined assets, the world is inferred from data and inherits whatever richness, variability, and distributional bias the recorded drives contain. This shift, from manual content creation to

automated reconstruction, is the principal appeal of neural simulation for AD and is also the source of its specific open challenges.

Current landscape Early adaptations of NeRF to automotive scenes [41]–[44] established that volumetric representations could be trained on outward-looking driving data and extended to handle moving objects by decomposing the scene into a static background and a set of dynamic actors, typically localized using annotated 3D bounding boxes. UniSim [45] and MARS [46] subsequently demonstrated that such decompositions could support closed-loop sensor simulation on full automotive datasets, and NeuRAD [7], presented as Paper A of this thesis, showed that joint camera and lidar simulation is feasible within a unified NeRF-based framework across multiple public datasets.

Subsequently, the introduction of 3D Gaussian Splatting [40] triggered a rapid wave of adaptations to driving scenarios [47]–[51]. The main attraction is rendering speed: explicit primitives are markedly faster to rasterize than the dense ray sampling required by volumetric fields, which is of direct practical importance when the same scene must be rendered many times during closed-loop evaluation. Paper B [8] extends this line of work by simulating cameras and lidar jointly within a Gaussian-based framework. A complementary strand of research focuses specifically on high-fidelity lidar simulation [52]–[54], treating the sensor's characteristic beam geometry and ray-drop behavior as a modeling target in its own right.

A separate line of work addresses the assumption that dynamic actors are known in advance. Methods such as SUDS [55], EmerNeRF [56], PVG [57], 4DGS [58], Desire-GS [59], and AD-GS [60] learn to separate static and dynamic content from the sensor data itself, either by time-conditioning the scene representation or by splitting it into explicit static and dynamic components using photometric or geometric consistency cues. Paper C [9] belongs to this family and targets the specific case where actor instances must be identified and tracked across time without relying on human annotations.

Finally, a growing body of work applies neural simulation as a tool for evaluating downstream autonomy. NeuroNCAP [10] is an illustrative example, using a data-driven digital twin as the substrate for safety-relevant closed-loop benchmarking of end-to-end driving models. Efforts along these lines make it tangible that the ambition of the field is not photorealistic rendering in isolation, but the construction of simulators whose outputs can be consumed by an AD stack as a stand-in for real sensor data.

Open challenges Despite this progress, several open challenges remain before neural simulation can serve as a dependable substitute for real-world testing. The remainder of this thesis is organized around three of them.

Fidelity under autonomous driving conditions. Most foundational work in neural rendering has been developed and benchmarked on object-centric scenes with dense, well-distributed camera coverage. Automotive recordings violate essentially all of these assumptions. Sensors are outward-looking rather than inward-looking; scenes extend far beyond what can be densely sampled, camera trajectories are close to linear, sensors vary in capture settings and modalities, and the ego vehicle moves at speeds that introduce the sensor-motion artifacts described in Section 2.2. Addressing these factors requires dedicated design choices for how the scene is parameterized across spatial scales, how far-field structure is represented, how dynamic actors are modeled, and how rolling-shutter and multi-sensor characteristics are handled during rendering. Papers A [7] and B [8] target precisely this setting, jointly modeling camera and lidar, explicitly accounting for key sensor characteristics such as rolling shutter, beam divergence, and ray drop, and introducing scene parameterizations suited to the large spatial extents encountered in driving data. Chapters 3 and 4 survey the technical foundations of neural rendering and the specific adaptations required to obtain faithful multi-sensor simulation under AD conditions.

Scaling from single scenes to scenario suites. A second challenge concerns scalability. The methods primarily discussed and studied in this thesis are *per-scene* optimization procedures: each recorded sequence yields an independently trained representation, and each additional scene requires additional optimization time. Meaningful validation of an autonomy stack involves not one scene but thousands, ideally covering a diverse spectrum of environments, traffic patterns, and rare events. Two bottlenecks limit this progression. The first is computational: per-scene optimization can take hours and rendering costs vary widely across methods, both of which constrain how many scenes can be processed within a realistic budget. The second is supervisory: most high-fidelity methods rely on accurate 3D bounding-box annotations for dynamic actors, and such annotations are costly to produce at the scale implied by fleet-wide data collection. Progress on either front is an active area of research, with accelerated scene representations and rendering algorithms, annotation-free methods, and feed-forward alternatives to per-scene optimization representing promising directions. Chapter 4 discusses these issues in detail. Paper C [9] targets the supervisory bottleneck by reconstructing dynamic scenes without relying on human-annotated actor trajectories, while Paper D [11] contributes a computational building block for scaling mesh-based representations, a promising foundation for scene reconstruction

that combines explicit geometry with principled yet efficient rendering.

Simulation validity. A neural simulator that produces visually convincing renderings is not automatically a simulator whose outputs can be trusted as a proxy for real data. Perceptually minor artifacts can bias a downstream perception model in ways that are difficult to anticipate from rendering metrics alone, and whether higher rendering fidelity translates into more valid simulation conclusions is itself an open question. This concern has been termed the *real-to-sim gap*, and it coexists with the more familiar *sim-to-real gap* rather than replacing it. The two describe distinct transfer directions and can both apply to the same simulator. How to quantify the real-to-sim gap in practice, how it affects downstream models, and what can be done to mitigate it are the subject of Paper E [12] and Chapter 5.

These three challenges structure the remainder of the thesis. Chapter 3 covers the technical foundations of neural scene representations and differentiable rendering. Chapter 4 examines how these methods are adapted to the conditions and scale of AD. Chapter 5 addresses the validity of the resulting simulations.

CHAPTER 3

Neural scene representations and differentiable rendering

Consider the challenge of synthesizing what a camera would see from a position that was never actually occupied during a recorded drive. The question is deceptively simple to state: given a set of images taken along a fixed trajectory, produce a new image from a viewpoint nearby but not among those observed. This problem, known as novel view synthesis, sits at the heart of data-driven digital twin construction. If it can be solved with sufficient fidelity, then the recorded trajectory of an ego vehicle becomes not a single fixed sequence but an explorable environment, one that can be traversed from alternative paths and from which alternative sensor observations can be synthesized on demand.

This chapter builds the technical foundation needed to understand how such synthesis is achieved in the context of AD. Section 3.1 establishes the novel view synthesis task, introduces the two fundamental rendering paradigms that make it tractable, and defines the metrics used to evaluate rendering quality throughout this thesis. Section 3.2 surveys the landscape of scene representations for neural rendering, tracing a progression from implicit volumetric fields through explicit Gaussian primitives to mesh-based representations.

3.1 Novel view synthesis and neural rendering

Novel view synthesis (NVS) is the task of synthesizing images of a scene from specific points of view, given only images from different viewpoints. A natural approach would be to recover the physical properties of the scene and re-render it from the desired viewpoint using classical computer graphics. In practice, however, physical parameters such as geometry, material properties, and illumination are rarely available in real-world captures, and estimating them from images alone, a problem known as inverse rendering, remains highly challenging.

The field of *neural rendering* has emerged as an alternative, combining classical rendering ideas from computer graphics with deep learning to directly learn a scene's underlying geometry and appearance from observed data, without explicitly recovering its physical parameters [6]. These methods optimize a compact scene representation, typically parameterized by neural networks, using a differentiable rendering function as supervision, with the goal of synthesizing photorealistic images from novel viewpoints. By separating scene modeling from rendering in this way, the learned representation can be queried at viewpoints that were never observed during optimization. Applied to AD, the input is typically a set of timestamped, posed images and lidar scans collected along a recorded drive, and the goal is to synthesize new observations from nearby viewpoints or timesteps across both camera and lidar modalities. Rather than replaying fixed sensor logs, a scene model learned via neural rendering can generate observations for alternative ego trajectories and modified actor configurations, enabling closed-loop testing beyond the original recording.

Rendering functions

Central to neural rendering is the choice of rendering function, which determines both what the scene representation must encode and how well real sensor physics can be modeled. Two dominant rendering paradigms exist, illustrated in Figure 3.1, each with a distinct trade-off between speed and physical accuracy.

Rasterization is the process of converting 3D geometric primitives into fragments that form a 2D image. Given scene primitives with known 3D positions, typically polygons, lines, or points, each primitive is projected onto the image plane under the camera model. A depth buffer tracks the closest primitive at each pixel location to resolve visibility, and a programmable shader determines the final pixel color from the fragment's interpolated attributes together with material and lighting data. Rasterization maps well onto GPUs because primitive and fragment operations are highly parallelizable and execute the same program across many fragments simultaneously,

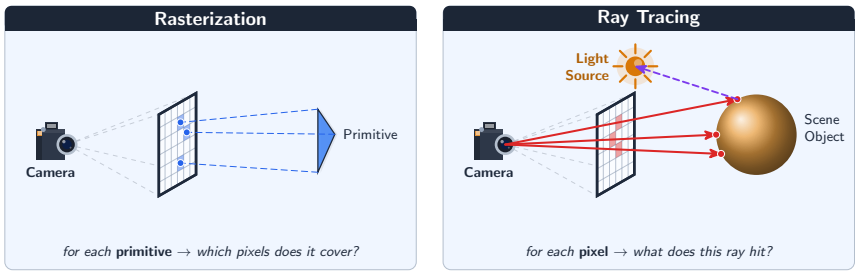


Figure 3.1: Rasterization vs. ray tracing. Rasterization (*left*) iterates over scene primitives and projects each onto the image plane, using a depth buffer to resolve visibility. Ray tracing (*right*) iterates over pixels, casting a ray per pixel to find the nearest intersection with scene geometry; secondary rays for shadows and reflections emerge naturally from the same mechanism.

requiring limited synchronization. Its principal limitation is that while rasterization solves primary visibility accurately, it models light transport only approximately; effects such as reflections, refractions, and soft shadows typically require additional approximations rather than emerging naturally from the algorithm.

Ray tracing, in contrast, determines pixel colors by casting rays from the sensor into the scene and computing their interactions with surfaces as they propagate. Because ray paths can follow specular and diffuse light transport more directly, the method naturally supports complex optical effects such as reflections, refractions, and soft shadows. Primary-ray visibility alone is typically more expensive than rasterization because each ray must search the scene for intersections and often accesses memory irregularly; the cost increases substantially with additional light bounces, although dedicated GPU ray-tracing hardware has narrowed the performance gap in recent years. Beyond surface interactions, ray tracing generalizes naturally to *volume rendering*, where rays do not terminate at a surface but instead march through a medium, accumulating properties such as opacity and color along their path.

This trade-off between the two paradigms is directly consequential for neural rendering method design. Methods that rely on rasterization can be made extremely fast but model light transport and sensor physics only approximately. Methods based on ray tracing offer a closer correspondence to physical image formation, making them naturally more flexible in this regard. The sensor model reduces to a ray generation function, so arbitrary sensor geometries (*e.g.* non-pinhole cameras or the spherical

parameterization of lidar) and per-ray poses (*e.g.* rolling shutter) follow without any changes to the core algorithm. This flexibility, however, comes at the cost of higher computational demand. As will become clear in Section 3.2, different scene representations are closely coupled to one or the other paradigm.

Learning through differentiable rendering

All neural rendering methods discussed in this chapter share a common training loop. A scene representation is initialized and a rendering function produces a synthetic image from a known viewpoint. This synthetic image is compared to the corresponding observed image using a photometric loss, typically an L1 or L2 pixel error combined with a perceptual term, and gradients are propagated back through the rendering function into the representation parameters via gradient descent. By iterating this process across all available viewpoints, the representation is gradually shaped to reproduce the observed data.

The key requirement that makes this possible is differentiability. The rendering function must be differentiable with respect to the scene representation parameters so that gradients can flow back into the representation. Volume rendering lends itself naturally to differentiation because pixel colors are computed as weighted integrals along a ray, and when the underlying representation is continuous, the pixel colors vary smoothly with its parameters. Rasterization is not inherently differentiable due to the discrete nature of primitive coverage, and achieving differentiability requires a custom backward pass that approximates gradients through the visibility discontinuities.

Regardless of how these challenges are resolved, every method must encode two fundamental quantities in its scene representation. The first is geometry, which determines what is visible from a given viewpoint. The second is appearance, which determines the color of visible surfaces, including view-dependent effects such as specular highlights. The choice of parameterization and its associated rendering function is what most distinguishes the methods discussed in the following subsections.

Evaluation metrics

Measuring the quality of novel view synthesis requires comparing rendered outputs to held-out sensor observations. A subset of frames is held out as a validation set and the remainder used for optimization, directly analogous to the train/test split in supervised learning. The held-out views are not truly independent, they are drawn from the same scene and typically the same trajectory as the training frames, so nearby viewpoints share significant visual overlap. The implications of this evaluation

protocol for simulation utility are examined in Chapter 5.

For camera rendering, three complementary metrics are standard. Peak signal-to-noise ratio (PSNR) measures pixel-level reconstruction accuracy in decibels; higher values indicate closer agreement between rendered and observed images, but the metric weights all pixels equally, so errors in large flat regions can mask poor reconstruction of fine detail. The structural similarity index (SSIM) [61] captures perceptual similarity in terms of luminance, contrast, and structure, and is more aligned with human visual judgment than raw pixel error. Learned Perceptual Image Patch Similarity (LPIPS) [62] uses features from a pretrained network to measure perceptual distance and tends to better capture differences that are visually salient but poorly reflected in pixel-wise metrics. Together these three metrics provide complementary perspectives on rendering quality. In settings where no ground-truth images exist, for example for viewpoints outside the collected trajectory, Fréchet Inception Distance (FID) [63] is sometimes used as a proxy. Rather than comparing individual image pairs, it measures the distance between the distributions of generated and real image features, computed over a collection of images. None of these metrics directly measures simulation utility for downstream AD tasks, though they serve as useful proxies for developing and comparing neural rendering methods, a distinction that will become central in Chapter 5.

For lidar rendering, the relevant quantities differ. Geometry is typically evaluated by measuring the depth error between synthesized and observed returns along the known ray directions. Chamfer distance is also commonly used and measures the geometric accuracy of the rendered point cloud relative to the observed one by summing nearest-neighbor distances in both directions, penalizing both missing points and incorrectly placed ones. Intensity error captures the accuracy of the predicted reflectance for each beam, and ray drop accuracy measures how well the simulator predicts which beams fail to return a measurement.

3.2 Scene representations for neural rendering

A scene representation for neural rendering encodes the geometry and appearance of a scene in a form that supports both optimization from sensor observations and synthesis of novel views via a rendering function. Such representations make fundamentally different choices about where scene information is stored, how the rendering function operates on it, and what rendering paradigm it is coupled to. These choices, in turn, have cascading consequences for reconstruction quality, rendering speed, training efficiency, editability, and memory consumption. Each of the representations surveyed

below was partly motivated by limitations of earlier approaches, but they are best understood as complementary design points rather than a strictly linear progression.

Implicit representations and neural radiance fields

In early works, novel view synthesis was approached through classical light field methods [64], [65], which capture the full plenoptic function by densely sampling rays and interpolating between them at render time. While effective for constrained capture setups, these methods require impractically dense sampling to generalize to new viewpoints. Neural network-based approaches subsequently emerged with the rise of deep learning [66], [67], predicting blending weights using CNNs [68], refining the texture space [69], [70], and coupling the learning with volumetric ray-marching [71], [72]. Neural Radiance Fields (NeRF) [5] improved the efficiency and quality of volumetric ray-marching by introducing importance sampling and positional encoding, and its success resulted in a new paradigm with an explosion of follow-up works.

Neural radiance fields A scene is modeled as a continuous volumetric function that maps a 3D spatial position $\mathbf{x} = (x, y, z)$ and a viewing direction $\mathbf{d} = (\theta, \phi)$ to a density $\sigma(\mathbf{x})$ and an emitted color $\mathbf{c}(\mathbf{x}, \mathbf{d}) = (r, g, b)$. In practice, this function is implemented as a multilayer perceptron (MLP) $F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$, whose weights Θ are optimized to satisfy this mapping from the observed data. Inputs are commonly encoded using positional encodings to enable the network to learn high-frequency functions [73].

To render a pixel from this volumetric representation, principles from classical volume rendering are applied [74]. A ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is cast from camera origin $\mathbf{o}$ with direction $\mathbf{d}$ and sampled at multiple distances t along the ray. For each sample, the MLP is queried to produce a density and color. The expected color of the ray over bounds t_n and t_f is obtained by integrating these contributions weighted by accumulated transmittance:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \tag{3.1}$$

where $T(t)$ denotes the accumulated transmittance along the ray, *i.e.* the probability that the ray travels from t_n to t without being absorbed:

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right). \tag{3.2}$$

Rendering an image from the neural radiance field thus requires estimating the integral in (3.1) for every pixel, or more precisely, for the camera ray passing through each pixel. In practice, this integral is approximated numerically using quadrature:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i, \tag{3.3}$$

where δ_i is the distance between adjacent samples along the ray, σ_i and $\mathbf{c}_i$ are the density and color at sample i, and T_i is the transmittance at that sample:

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right). \tag{3.4}$$

Optimizing the scene representation reduces to finding weights Θ such that the synthesized colors match the observed colors, typically via gradient descent under the loss:

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left\| C(\mathbf{r}) - \hat{C}(\mathbf{r}) \right\|_2^2, \tag{3.5}$$

where $\mathcal{R}$ is the set of observed rays, $C(\mathbf{r})$ the observed color, and $\hat{C}(\mathbf{r})$ the synthesized color. With sufficiently many rays observed from diverse viewpoints, the optimization is constrained enough to recover both the geometry and appearance of the scene, enabling high-quality novel view synthesis.

The method's key strengths are its continuity and expressive view-dependent appearance modeling. The implicit representation is continuous and not bound to any fixed grid resolution, naturally interpolating between observed viewpoints, while directional dependence allows it to reproduce specular and other angle-dependent effects. Its primary limitation is computational cost: rendering each ray requires querying the MLP at many sample points, making both training and inference slow.

The field of implicit representations Several lines of work have followed since the introduction of NeRF. Several methods improve rendering efficiency [75]–[81] by either precomputing the expensive neural field into representations that are faster to render at inference time, or reparameterizing the scene to accelerate training. Instant-NGP (iNGP) [75] replaced positional encodings with a learnable multi-resolution hash grid, enabling fast spatial feature lookups while permitting a much smaller MLP. This reduced training times from hours to minutes.

A separate line of work [82]–[85] targets aliasing, seeking alternatives to naive supersampling of multiple rays per pixel. Mip-NeRF [82] renders conical frustums

defined by each camera pixel rather than infinitesimal rays, constructing an approximation of the positional encoding over the volume of each frustum. Mip-NeRF 360 [83] extends this with a scene parameterization better suited to large, unbounded scenes, and Zip-NeRF [84] combines the anti-aliasing framework of Mip-NeRF 360 with the grid-based representation of iNGP.

A further direction replaces volumetric density with signed distance functions (SDFs) to obtain cleaner surface geometry. NeuS [86] and VolSDF [87] both reformulate the volume rendering integral in terms of an SDF, producing well-defined surface normals and enabling mesh extraction without the ambiguous density boundaries of standard NeRF.

Explicit representations and 3D Gaussian splatting

In contrast to continuous representations and volumetric rendering, explicit 3D representations such as meshes and points can be rendered very quickly with GPU-based rasterization. However, as discussed in Section 3.1, rasterization does not naturally lend itself to differentiable rendering due to the discrete nature of primitive coverage.

3D Gaussian Splatting 3D Gaussian Splatting (3DGS) [40] resolves this by representing the scene as a set of 3D Gaussians. The Gaussian distribution of these primitives creates a differentiable volumetric representation, while closed-form solutions for affine transformation of Gaussians allow for efficient projection to 2D where they can be rasterized in a highly parallelized manner. The result is a method that trains and renders at speeds far exceeding NeRF-based methods, with image quality competitive on the scenes it was designed for.

Instead of learning a neural network to represent the geometry and appearance of the scene, 3DGS parameterizes the scene as a set of 3D Gaussians, where the mean $\mu = (x, y, z)$ represents the 3D position of a primitive and its covariance $\mathbf{\Sigma}$ represents its extent and orientation, commonly factorized into a scaling matrix $\mathbf{S}$ and a rotation matrix $\mathbf{R}$:

$$\mathbf{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T. \tag{3.6}$$

In practice, these factors are commonly represented by a 3D vector $\mathbf{s}$ for the scale and a quaternion $\mathbf{q}$ for the rotation, which can be converted into the corresponding matrices. In addition to its spatial parameters, each 3D Gaussian has an associated opacity $o \in (0, 1)$ to represent how much radiance the primitive should absorb, and a set of spherical harmonics (SH) coefficients representing its color. Similar to NeRF, these parameters are optimized to minimize a photometric loss. Due to the more

outlier-prone nature of an explicit representation, the more robust L1 loss is typically favored over the L2 loss used in NeRF.

To render an image, the 3D Gaussians are first projected to 2D splats. Given a posed camera, each mean and covariance are transformed from world to camera coordinates, giving μ^C and $\mathbf{\Sigma}^C$. The mean is projected to image space $\mu^I \in \mathbb{R}^2$ via perspective projection. The covariance is propagated to image space using a first-order approximation [88], linearizing the projection around the Gaussian mean:

$$\mathbf{\Sigma}^I = \mathbf{J}^I \mathbf{\Sigma}^C \left(\mathbf{J}^I\right)^T \in \mathbb{R}^{2\times 2}, \tag{3.7}$$

where $\mathbf{J}^I$ is the two upper rows of the Jacobian of the projective transform. The color of a pixel $\mathbf{p} \in \mathbb{R}^2$ is then computed using point-based α-blending:

$$C = \sum_{i \in N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{3.8}$$

where N is the ordered set of Gaussians that intersect the pixel, $\mathbf{c}_i$ is the color of Gaussian i, and α_i is obtained by multiplying the associated opacity with the evaluated 2D Gaussian:

$$\alpha_i = o_i \exp\left(-\frac{1}{2}(\mathbf{p} - \mu^I)^T (\mathbf{\Sigma}^I)^{-1} (\mathbf{p} - \mu^I)\right). \tag{3.9}$$

This formulation allows gradients from the photometric loss to propagate to all parameters of every Gaussian intersecting a given pixel, enabling gradient-based optimization of the explicit representation.

To make this efficient, the rasterizer in [40] divides the image into tiles of 16×16 pixels and assigns each Gaussian to the tile or tiles it overlaps, determined by computing axis-aligned bounding rectangles from the eigenvalues of the 2D covariance. Gaussians are sorted globally by depth in view space, before being divided into per-tile thread-blocks launching one thread per pixel, where each tile independently blends its Gaussians front-to-back. Threads within the same tile collaboratively load Gaussian data into shared memory before traversing the sorted list, yielding high parallelism for both data loading and processing.

During optimization, Gaussians may be incorrectly placed or scaled due to ambiguities in the 3D-to-2D projection, forming local optima that gradient updates alone cannot resolve. Initializing from a sparse point cloud, typically obtained via Structure-from-Motion (SfM) [89], mitigates this to some extent, but the narrow basin of attraction of the photometric loss means positional gradients are only informative

for small displacements, limiting the ability to relocate Gaussians over larger distances to explain unmodeled geometry or correct misplaced primitives. To address this, 3DGS introduces heuristics for adaptive density control. At regular intervals, primitives are cloned in under-reconstructed areas, split in over-reconstructed areas, and removed if transparent.

The field of explicit representations Since its introduction, 3DGS has inspired a wide range of follow-up works addressing its limitations along several directions. Mip-Splatting [90] addresses aliasing, analogous to Mip-NeRF, by introducing 3D smoothing filters to regularize the maximum frequency of primitives during optimization and a 2D Mip filter in image space to handle aliasing when rendering at lower sampling rates. 2DGS [91] replaces volumetric 3D Gaussians with flat 2D Gaussians embedded in 3D space, which better align with surfaces. Gaussian Opacity Fields [92] define a volumetric opacity field by casting rays through the scene and computing opacity via explicit ray-Gaussian intersections, replacing the projection-based rendering of standard 3DGS. This enables surface extraction directly from the level sets of the opacity field, without resorting to Poisson reconstruction or TSDF fusion. The adaptive density control heuristics have also received attention. 3DGS-MCMC [93] reformulates densification as Markov Chain Monte Carlo (MCMC) sampling, demonstrating increased robustness to initialization. Robustness to real-world capture conditions has been addressed by [94], which compensates for motion blur and rolling shutter caused by natural camera motion directly during optimization. On the efficiency side, LightGaussian [95] reduces the storage and rendering cost of trained scenes through pruning and knowledge distillation. Several works also address approximations in the rasterization pipeline. StopThePop [96] corrects sorting approximations in the tile-based rasterizer that cause popping artifacts during camera motion. 3DGRT [97] and EVER [98] replace rasterization with ray tracing through Gaussian volumes, inheriting the flexibility of ray-tracing frameworks and opening the door to effects that are difficult or impossible to achieve with rasterization, such as secondary rays. 3DGUT [99] replaces the projection model with an unscented transform, enabling accurate projection under nonlinear camera models such as wide-angle and fisheye lenses.

Neural rendering with explicit representations is not limited to 3D Gaussian-based methods. A recent line of work has explored methods closer to mesh-based optimization [11], [100], [101]. Triangle Splatting [100] uses triangle primitives as its scene representation, with a window function to allow smooth, differentiable rasterization. Since the optimized triangle soup can be rendered in a traditional game engine, it

achieves very fast rendering speeds. Radiant Foam [11] and Radiance Meshes [101] instead represent the scene using polyhedral and tetrahedral meshes, respectively. Radiant Foam partitions the space using a 3D Voronoi diagram and forms images by ray-tracing through the cells, striking an attractive balance between rendering speed and the physical accuracy that ray tracing enables. Radiance Meshes take the dual of the Voronoi diagram, a Delaunay tetrahedralization, to partition the scene, and introduce a method for rasterizing the tetrahedral volume to achieve higher rendering speeds. While both methods show promising results for mesh-based rendering, they share a common bottleneck: maintaining mesh connectivity during optimization incurs a significant computational cost, and this recomputation currently scales poorly to large scene representations.

CHAPTER 4

Applying neural rendering to autonomous driving

Neural rendering offers a promising foundation for AD simulation, but the methods introduced in Chapter 3 were primarily designed for object-centric benchmarks with bounded scenes, dense viewpoint coverage, and controlled capture conditions. AD data violates most of these assumptions. A recorded drive traverses a large, unbounded outdoor environment along a single outward-looking trajectory, with dynamic actors breaking the static-scene assumption that underlies most reconstruction pipelines. The sensor suite is heterogeneous, typically combining multiple cameras with different lenses and exposure settings alongside lidar, and both modalities exhibit artifacts from fast ego motion. This chapter examines how methods in the field address these challenges. Section 4.1 concerns the scene itself, discussing how representations are parameterized to accommodate unbounded geometry, dynamic actors, and the heterogeneous sensor configurations typical of AD capture. Section 4.2 turns to the sensors, examining the characteristics of real cameras and lidars that must be modeled for rendered observations to be faithful substitutes for recorded ones. Section 4.3 then considers the cost of applying these methods at fleet scale, where the computational burden of per-scene optimization and the reliance on manual annotations become the dominant obstacles.

4.1 Scene parameterization

As discussed in Section 3.2, the choice of how to parameterize the scene determines the representation's expressiveness, rendering speed, and training efficiency. In this section, we discuss the implications that AD scenes impose on scene representations and review common strategies to address these challenges.

Large and unbounded scenes

The neural rendering methods discussed in Chapter 3 were largely developed on object-centric datasets [5], [83], where a camera captures dense views around a central object of interest. AD scenes break this assumption in several ways. They are inherently outward-looking, reconstructed from a sensor rig traveling through the middle of the environment rather than orbiting around it. A typical scene consists of 10–20 seconds of temporally collected sensor data, during which the sensor rig may traverse more than 500 meters depending on the scenario and vehicle speed. The spatial extent to which reconstruction must be performed is considerably larger still, since geometry must be modeled out to the sensor horizon, which can involve buildings and structures kilometers away.

Implicit representations of unbounded space Representing large scenes using NeRFs comes with several challenges. Unbounded scenes can occupy arbitrarily large regions of Euclidean space, while many methods require a bounded domain, for example the hash grid used in iNGP [75]. Allocating representational capacity uniformly across the scene is rarely desirable. Details in the far distance are typically less important than the immediate surroundings of the ego vehicle, and a uniform allocation would require feature grids or MLPs large enough to be memory-intensive and slow to query.

In UniSim [45], the memory demands of such scenes are addressed by using sparse voxel grids to represent and optimize features only near observed surfaces. This approach requires a good initial estimate of surface locations. When lidar measurements exist, they provide a strong prior, but their capture angle and range are limited. A complementary strategy separates the modeling of distant regions using dedicated radiance fields [45] or environment maps [46], [55], which represents far-field objects and the sky as a 2D texture parameterized over a sphere at infinity.

Mip-NeRF 360 [83] presents a contracted scene parameterization: a mapping from world space to a parameterization domain that acts as the identity in a region around the scene center and becomes linear in disparity further out. The scene itself

remains unbounded, but the mapped space fits inside a finite domain in all directions, making it compatible with bounded representations such as feature grids. Maintaining a volumetric representation for distant content, rather than collapsing it to a 2D environment map, retains parallax in the far field as the ego vehicle moves. Distant structures shift relative to one another in the way that real geometry at finite depth does, which an environment map cannot reproduce.

These strategies address where representational capacity is placed but not how rays are evaluated through it. Naive uniform sampling along rays in large scenes wastes nearly all samples on empty space, and reducing the sample count uniformly degrades quality wherever surfaces actually exist. Proposal sampling addresses this by using smaller auxiliary MLPs to predict promising intervals along each ray, followed by denser sampling with a more expressive network only in those intervals [83], [84].

Primitive-based representations at scale Explicit representations face analogous challenges to those of implicit ones, but the relevant resource is the number of primitives rather than the capacity of a feature grid or MLP. The two questions that drove the implicit discussion, where to place representational capacity and how to sample it, return in different forms: how to represent far-field content, where placing primitives directly is difficult, and how to manage the primitive count itself in scenes where adequate fidelity requires very many of them.

Far-field handling typically takes one of two forms. The most common choice in AD-targeted 3DGS methods is to delegate sky and distant background to a separate environment map [47], [49], [57], since both the gradient-based position updates and densification heuristics of standard 3DGS struggle to place primitives at extreme distances. As in the implicit case, this removes the ability to model far-field parallax, but remains a practical choice, particularly if the far-field background is of little interest in downstream applications. An alternative is to represent the far field with primitives whose initial positions are placed on a disparity-spaced grid beyond the lidar range, as in Paper B [8], retaining a single representation across near- and far-field at the cost of additional primitives.

Primitive count is a defining property of the representation, with direct consequences for memory, rendering cost, and the level of geometric detail that can be captured. Large, unbounded AD scenes typically require many primitives to represent near and far geometry simultaneously, and the standard gradient-based densification strategy in 3DGS adaptively increases and prunes primitives based on optimization signals, leading to a scene- and hyperparameter-dependent model size with only indirect control over the final memory footprint. The MCMC-based formulation of [93]

replaces these heuristics with state transitions consistent with stochastic-gradient Langevin dynamics. The primitive count is then a directly controllable parameter, useful when the memory footprint must be predictable.

Beyond the count itself, the per-primitive cost of view-dependent appearance can be reduced by shrinking the spherical-harmonic parameterization or replacing it with a small learnable feature decoded by a shared network. Orthogonal to both, post-hoc pruning strategies based on importance [95] or Fisher-information sensitivity [102] reduce primitive counts after optimization without changing the training procedure.

Mesh-based neural rendering approaches face the same primitive-count constraint, with an additional bottleneck specific to their formulation: the algorithms that maintain primitive connectivity scale poorly with the number of cells, which is particularly limiting in AD scenes where large spatial extent and fine geometric detail jointly demand high primitive counts. This setting remains comparatively underexplored for AD data, and removing the connectivity bottleneck is a prerequisite for evaluating mesh-based methods at AD scale.

Finally, extending to larger city-scale environments can be achieved by composing independently trained scene blocks with careful handling of overlaps and transitions [103]–[105]. Such block-based or hierarchical strategies are often necessary to keep memory usage and optimization time tractable at urban scale.

Modeling dynamic actors

A core assumption of vanilla NeRF and 3DGS is that the scene is static: each point in space has fixed appearance and geometry throughout all observations. This assumption is routinely violated in AD data. Vehicles, cyclists, and pedestrians move continuously through the scene, appearing at different positions in images and lidar scans captured at different times. Without explicit treatment, dynamic objects create cross-view inconsistencies that a static representation cannot reconcile, producing ghosted or blurred artifacts that degrade reconstruction quality and prevent the controllable actor manipulation that simulation requires.

Explicit instance decomposition The standard approach is scene decomposition. The scene is partitioned into a static background and a set of rigid dynamic actors, each represented in a canonical coordinate frame. A separate representation is learned for each actor; at render time, actors are transformed into the world frame using their pose at the queried timestamp. This formulation, introduced in Neural Scene Graphs [41] and adopted broadly in subsequent works [7], [8], [45]–[49], enables

Figure 4.1: Trajectory editing through explicit scene decomposition. Three scenes reconstructed with NeuRAD [7], showing original sensor data (left) alongside renders with edited actor trajectories (right). Because each actor is represented in a canonical frame and transformed into the world via a per-timestamp pose, recorded trajectories can be replaced with arbitrary ones, enabling counterfactual scenarios.

controlled trajectory editing by replacing a recorded trajectory with a modified one. Figure 4.1 illustrates this capability on scenes reconstructed with NeuRAD, where actor poses have been modified to produce counterfactual trajectories.

The rigid-body assumption underlying this decomposition fits many actor classes well but does not hold universally. For vehicles, which are kinematically rigid, the assumption is appropriate: the entire actor can be described by a single time-varying SE(3) pose, and trajectory editing is straightforward. The rigid assumption holds reasonably well for cyclists, where pedaling introduces only minor non-rigidity. Pedestrians are more problematic, as the articulated motion of swinging arms and legs produces deformations that a single rigid transform cannot capture, leading to blurry

reconstructions [7]. Extending the bounding-box paradigm to handle this would in principle require annotating individual body parts, which is not feasible at scale.

OmniRe [49] relaxes this limitation by retaining rigid bounding boxes for vehicles while modeling pedestrians and cyclists with SMPL-based [106] body pose parameters estimated from video. This richer representation captures articulated motion more faithfully. However, it comes with additional preprocessing, where human body poses must be tracked and aligned, which relies on specialized models and is sensitive to occlusion and viewpoint coverage. Moreover, SMPL is limited to standard human body shapes and typical gaits, so actors with unusual configurations or non-standard classes still fall back to less structured representations. Generalizing this approach to the full diversity of actors encountered in real-world driving therefore remains non-trivial.

Time-varying representations A different direction avoids explicit instance decomposition altogether by making all scene representation parameters functions of time. EmerNeRF [56] learns separate static and dynamic hash grid fields and routes queries based on predicted scene flow. Periodic Vibration Gaussians [57] and subsequent methods such as 4DGS [58], DeSiRe-GS [59], and AD-GS [60] extend 3DGS by making Gaussian attributes time-dependent, adding degrees of freedom to the optimization that allow the representation to change over time. Being learned solely from data, these methods can reconstruct more general non-rigid dynamics without relying on pre-defined motion models or pose parameters. However, this parameterization does not necessarily recover the true underlying motion, since both appearance and opacity can be altered over time to satisfy the photometric consistency objective. The result is often a temporally varying representation that faithfully renders the training views but struggles with larger interpolations and the harder case of extrapolation, as illustrated in Figure 4.2. Because temporal change is modeled at the level of individual primitives rather than coherent instances, such representations also provide no means to move, remove, or replace individual actors. The representation reproduces observed motion but does not support arbitrary trajectory editing.

Discussion The choice of dynamic actor representation defines a trade-off between representational flexibility, the freedom to fit observed motion and appearance, and structural editability, the degree to which the representation exposes coherent instances that can be manipulated independently. The two are in tension, since flexibility is typically gained by relaxing the structural constraints that make editing possible. Rigid bounding-box decomposition sits at one end, imposing strong structural con-

Figure 4.2: Comparison of dynamic scene modeling strategies under sparse supervision. A held-out frame from a scene reconstructed using every fourth frame of a 10 Hz sequence during training. Left: ground-truth image. Middle: an explicit-trajectory representation, where actors are modeled as rigid instances with parameterized motion (IDSplat [9]). Right: a time-dependent parameterization, where scene dynamics are modeled by Gaussian attributes that vary with time (DeSiRe-GS [59]). The time-dependent parameterization tends to struggle with recovering the true underlying motion, resulting in degraded performance in novel views.

straints that enable precise trajectory manipulation but cannot represent non-rigid dynamics. The works in this thesis adopt this design, with NeuRAD, SplatAD, and IDSplat modeling dynamic actors as rigid instances within a scene-decomposition framework. Time-varying primitive methods sit at the other, fitting articulated motion in the training views by absorbing dynamics into appearance and opacity variations, but without recovering the underlying motion or supporting instance-level editing. Hybrid approaches such as OmniRe occupy intermediate positions, gaining non-rigid expressiveness for specific actor classes while retaining instance-level structure. A more general hybrid that goes beyond class-specific extensions could combine motion modeling at multiple levels of granularity. The coarse rigid motion of an actor would be captured through scene decomposition, faithfully recovering the kinematics of the object as a whole, while finer non-rigid effects, such as articulated limbs or opening car doors, would be modeled by time-dependent parameters that only need to represent small deltas from the rigid motion. Such a decomposition would retain the controllability of explicit trajectories while restoring the expressiveness needed for non-rigid actors, making it a promising direction for future work. Underlying these modeling choices is a more fundamental question: how faithfully does motion need to be reconstructed for AD simulation to be useful? It is not obvious that capturing every articulated detail of a pedestrian's gait is necessary for the downstream perception or planning models that the simulation is meant to evaluate. Answering this requires empirical study of the real-to-sim gap as a function of motion fidelity, a direction Chapter 5 returns to from a broader perspective.

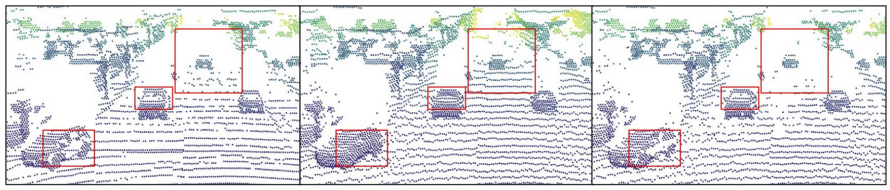
4.2 Sensor-realistic rendering

Geometric accuracy alone is insufficient for AD simulation. Even a representation that perfectly reconstructs scene geometry can yield unrealistic sensor observations if it fails to model the physical processes underlying each sensor's measurements. Moreover, accurate modeling of these processes is essential for reliable supervision during optimization; otherwise, discrepancies and approximations must be absorbed by the representation itself. This section reviews key sensor characteristics relevant in the AD context and discusses how the choice of rendering paradigm constrains what can be modeled faithfully.

Lidar properties

Lidar measures the world by emitting laser pulses and timing the returns. The basic output is a sparse set of range samples, but several further properties of the measurement process matter for downstream perception. Returns carry a reflectance intensity that depends on the surface, the angle of incidence, and the range, and which is the primary signal for tasks where target surfaces are geometrically indistinguishable from their surroundings, such as lane and road-marking detection [107]. A laser pulse may also fail to produce a return when its power is deflected away from the sensor by a specular surface or when the returned signal falls below the detection threshold, an effect commonly called *ray drop*. A simulator that emits a return for every beam direction produces point clouds whose sparsity pattern does not match that of the real sensor, constituting a domain shift that propagates into downstream perception [12], [108], as illustrated in Figure 4.3. Because lidar is a light-based sensor, the volume-rendering principles developed for cameras can be extended to it with little modification: the same neural field that produces per-location density and color can be augmented with a per-location reflectance and ray-drop output, accumulated along the ray with the same alpha compositing as for color.

Beyond range, intensity, and ray drop, lidar measurements have further characteristics whose detailed modeling lies outside the scope of this thesis: each pulse has a finite beam footprint that grows with range and can produce multiple returns when its footprint covers surfaces at different ranges, and as an active sensor incurs a round-trip transmittance that the standard volume-rendering integral does not account for. Dedicated lidar-simulation work has examined these effects in depth [52], [108], [109].



(a) Ground truth (b) Without ray drop, CD: 22.6 (c) With ray drop, CD: 4.1

Figure 4.3: Effect of modeling ray drop on simulated lidar point clouds. Highlighted regions show areas where the absence of returns from specular or low-reflectance surfaces is most pronounced. Without explicit modeling, the simulator emits returns for every beam direction and produces point clouds that diverge in sparsity pattern from the real sensor. Modeling ray drop probability recovers the characteristic gaps and brings the simulated cloud substantially closer to ground truth. Reported as Chamfer distance (CD) normalized by the number of ground-truth points (lower is better).

Rolling shutter

Rolling shutter distortion is present in virtually all consumer and automotive cameras, but its severity scales with the speed of relative motion between the camera and the scene. In object-centric NVS settings, where the camera moves slowly around a stationary subject, rolling shutter is rarely consequential. In AD, particularly for side-facing cameras whose optical axis is perpendicular to the direction of travel, the sensor can translate by tens of centimeters between the readout of the first and last rows of a single frame. Treating the resulting image as captured from a single pose introduces a temporal inconsistency that manifests as visible geometric distortion, illustrated in Figure 4.4a.

The lidar sensor is subject to the same effect, often more severely. Many automotive lidars are spinning sensors that complete a full rotation in roughly 100 ms, during which the ego vehicle may travel several meters at highway speeds. The resulting motion distortion of the point cloud is well known and is conventionally addressed by ego-motion compensation, in which each return is transformed into a common reference frame using its individual timestamp. However, ego-motion compensation alone does not resolve the issue for training. A return that was visible from the sensor pose at its emission time may not be visible from the reference pose used after compensation. When such returns are used as supervision for a single-origin ray model, the implied ray can pass through other geometry, producing line-of-sight errors that yield inconsistent gradient updates. Neglecting these effects in neural rendering

creates inconsistencies between views, resulting in inaccurate geometry and blurry renderings, as shown in Figure 4.4b.

The appropriate remedy depends on the rendering paradigm. In volume rendering, each ray can be assigned its own origin and direction, derived from the sensor pose at the instant the corresponding row or lidar point was captured. This makes rolling shutter compensation natural and exact within the ray-casting model and straightforward to implement when per-row or per-point timestamps are available, with the camera shutter time approximated when not directly provided. Figures 4.4c and 4.4d illustrate the effect of such modeling, adopted in Paper A. Modeling the lidar effect alone already shows some improvement (Figure 4.4c), but inconsistencies with the camera produce conflicting gradients, and the renderings remain blurry. Modeling the effect for both sensors (Figure 4.4d) recovers the geometry faithfully and yields clean, realistic renderings.

Handling this effect with rasterization is more difficult, as all Gaussians are projected to a single camera origin per frame. A direct correction would require reprojecting every Gaussian against every row-level pose, which is computationally prohibitive. A more tractable approach, explored in Paper B following [94], is to approximate the effect directly in 2D image space. Given each Gaussian's velocity, composed from the ego vehicle and actor velocities, its velocity relative to the camera is projected to the image plane, yielding an approximate pixel velocity that is used to shift the Gaussian's projected mean according to the capture time of each row:

$$\mathbf{v}^I = \mathbf{J}^I(-\omega_C \times \mu^C - \mathbf{v}_C + \mathbf{v}_{\text{dyn}}) \in \mathbb{R}^2. \tag{4.1}$$

Here, $\mathbf{v}^I$ is the Gaussian's approximated pixel velocity, $\mathbf{J}^I$ is the Jacobian of the projection, μ^C is the Gaussian's mean in the camera coordinate system, ω_C and $\mathbf{v}_C$ are the angular and linear velocities of the camera expressed in the camera coordinate system, and $\mathbf{v}_{\text{dyn}}$ is the velocity contribution from an associated dynamic actor, formed from its angular and linear velocities transformed into the camera coordinate system. In practice, this approximation works well, as shown both qualitatively and quantitatively in Paper B, without adding significant runtime. The approximation reuses the Jacobian of the nonlinear projective transform that 3DGS already evaluates at each Gaussian's mean, and inherits the validity regime of that linearization. Accuracy, therefore, degrades where the projection is strongly nonlinear, including when image-space displacements during the exposure are large enough to leave the neighborhood in which the linearization is accurate. The formulation additionally assumes constant velocity over the exposure interval, neglecting acceleration and other higher-order effects.

(a) original **(b)** no modeling **(c)** modeling lidar only **(d)** modeling lidar + camera

Figure 4.4: Impact of modeling rolling shutter in a high-speed scenario, with inset PSNR. (a) The original side-camera image. (b) Without any rolling shutter modeling, renderings are blurry and the recovered geometry is unrealistic, particularly for thin structures such as the pole. (c) Modeling only the lidar rolling shutter improves quality but leaves cross-modal inconsistencies that keep the renderings blurry. (d) Modeling the effect for both lidar and camera recovers sharp renderings and accurate geometry.

Sensor footprint and aliasing

Both camera pixels and lidar beams have a non-trivial spatial footprint. A camera pixel integrates radiance over a finite solid angle determined by the lens and sensor geometry, and a lidar beam diverges into a cone whose cross-section grows with range. Treating either as an infinitesimally thin ray amounts to point-sampling a continuous signal, which produces aliasing whenever the scene contains detail finer than the pixel footprint at the rendered scale [82], [90]. This is particularly relevant for AD data, where the same surfaces are observed across a wide range of distances as the vehicle moves through the scene. Without scale-aware modeling, training observations of the same structure at different ranges impose mutually inconsistent constraints, and the resulting representation generalizes poorly to novel views at any specific scale.

In volume-rendering, this problem is addressed by methods such as Mip-NeRF [82], which approximates the integral of positional encodings over the full footprint, a conical frustum along the ray. For grid-based representations [75], where such integrals cannot be evaluated analytically, the footprint can instead be approximated through multisampling combined with resolution-dependent downweighting [84]. While effective at reducing aliasing, these approaches incur a significant computational overhead due to the increased number of samples.

The rasterization analogue is provided by Mip-Splatting [90], which introduces a 3D smoothing filter that bounds each Gaussian's spatial extent by the maximum

sampling rate at which it was observed in training, and a 2D filter applied at render time that approximates the box-integration a real camera pixel performs over its footprint.

Per-camera appearance variation

Multi-camera rigs on production vehicles are heterogeneous. Individual cameras may have different lens types, exposure curves, white balance settings, and post-processing pipelines, so the same scene content appears with different brightness and color depending on which sensor captures it. A representation that treats all cameras as identical absorbs these systematic differences into the scene appearance itself, producing inconsistent novel-view renderings and biased gradients during optimization. The standard remedy in the AD setting is a per-sensor appearance embedding, learned jointly with the scene and applied as a correction at render time. Anchoring the embedding to the sensor rather than to the individual image, as is sometimes done in static-scene NeRF work [110], is what makes the correction generalize to novel views from the same sensor instead of overfitting to a specific observed image. A useful side effect is that the embedding can be swapped at inference, allowing any view to be rendered with the appearance of any sensor in the rig.

4.3 Scaling to large scenario sets

High-fidelity neural simulation of a single scene is a necessary but insufficient condition for practical AD validation. A meaningful test suite requires hundreds or thousands of reconstructed scenarios, covering diverse traffic situations, road geometries, and environmental conditions. Achieving this at scale exposes two practical bottlenecks: the compute required to optimize per-scene representations and the manual effort required to produce the annotations they depend on, alongside a more fundamental question about whether per-scene optimization is the right paradigm at all. This section examines each in turn.

Computational cost of per-scene optimization

Per-scene optimization requires fitting a separate scene representation for every recorded sequence to be reconstructed. Although the cost of doing so determines how many sequences can be reconstructed within a given compute budget, the relationship

to scenario throughput is more nuanced. A single reconstruction can be replayed against many ego policies, edited actor trajectories, and AD system versions, so the amortized cost per scenario decreases with scene reuse. Training cost governs how quickly the library of reconstructions can grow; inference cost governs how many scenarios can be simulated against it.

Furthermore, the two costs are coupled. Because optimization in neural rendering depends on gradients computed from synthesized observations, every training step pays the cost of an inference pass, so improvements to rendering speed translate directly into faster optimization. Total training time additionally depends on the number of steps to convergence, which informative initialization, for example from lidar point clouds or from dense image matching [111], can substantially reduce.

The two rendering paradigms differ markedly on both axes. Volume-rendering methods [7], [45], [56] require dense ray sampling and many forward passes through neural fields per pixel, keeping both training and inference times high. Rasterization-based methods [8], [47], [49] replace dense sampling with primitive projection and tile-based depth-ordered accumulation, evaluating decoders on the rasterized feature image rather than per sample. The combined effect of explicit initialization, rasterized rendering, and dedicated handling of AD sensor characteristics is, as Paper B demonstrates, sufficient to bring training and inference costs an order of magnitude below volume-rendering baselines at comparable quality. This combination of advantages is broadly why rasterization has come to dominate AD-targeted neural rendering, despite the additional engineering required to handle distorted lenses, lidar projection, and rolling shutter, all of which fall out naturally in the volume-rendering formulation but require dedicated mechanisms in rasterization (Section 4.2).

Mesh-based neural rendering approaches such as Radiant Foam [11] face a different scaling bottleneck: training requires frequent recomputation of cell adjacencies via Delaunay tetrahedralization as the mesh is optimized, which becomes the dominant cost at large primitive counts. While Paper D proposes an algorithm that significantly alleviates this bottleneck, the application to AD data remains future work. Consequently, it is still unclear whether mesh-based methods can match both the quality and efficiency of 3DGS-based approaches in AD-specific settings.

The annotation bottleneck

While the scene decomposition strategy discussed in Section 4.1 is effective for modeling most dynamic actors in AD scenes, it depends on a costly input: 3D bounding boxes with temporal associations across frames, commonly obtained by

manual annotations. Most major AD datasets provide such labels for benchmarking purposes, but producing them is a labor-intensive process that does not scale gracefully to the volumes of data required for industrial deployment. Even with semi-automated annotation pipelines, the cost of producing, verifying, and maintaining these labels remains substantial.

A common partial substitute is to replace manual labels with predictions from 3D object trackers, but this trades one dependency for another rather than removing it. High-performing trackers are trained on specific datasets with specific annotation taxonomies, requiring dataset-specific fine-tuning when applied to new sensor configurations or object categories. Their predictions also tend to require post-hoc correction of identity switches and state-estimation errors before they can be used effectively for neural rendering.

An attractive annotation-free alternative is to rely on general-purpose vision foundation models to obtain a self-supervised instance decomposition that can be applied to any sequence without manual labor. Promptable 2D video segmentation models such as SAM 2 [112] produce temporally coherent masks for arbitrary objects, and have been trained on large enough corpora to generalize zero-shot across domains. Language-grounded variants [113], [114] additionally enable querying by class, for example *car* or *pedestrian*, removing the dependence on a fixed taxonomy. The challenge is that these models operate in 2D image space, while the scene representation requires per-actor 3D geometry and trajectories.

This is the direction explored in Paper C [9]. Figure 4.5 illustrates the resulting pipeline: per-frame 2D instance masks and image features from visual foundation models are lifted to 3D using corresponding lidar point clouds, producing feature- and instance-painted points. Per-instance poses are then estimated by matching feature correspondences across time, and the resulting trajectories are smoothed and jointly refined with the Gaussian scene representation during optimization. The approach does not rely on dataset-specific retraining and can be applied across different object categories. In experiments, it achieves rendering quality comparable to annotation-dependent baselines, suggesting that priors from visual foundation models may be sufficient to reduce reliance on manual annotations. Further reducing the reliance on lidar by leveraging monocular depth estimation models [115] is a promising direction for future work.

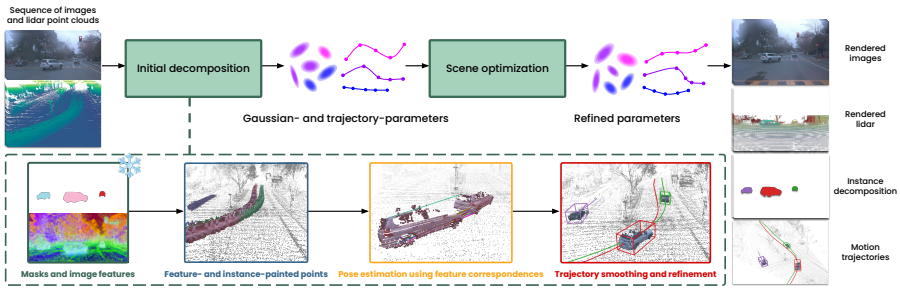


Figure 4.5: Annotation-free instance decomposition in Paper C. A sequence of images is processed by vision foundation models into 2D instance masks (Grounded-SAM-2 [113]) and image features (DINOv3 [116]), which are lifted to 3D using corresponding lidar point clouds to produce feature- and instance-painted points. Per-instance poses are estimated by feature correspondence matching across time, and the resulting trajectories are smoothed and refined jointly with the Gaussian parameters during scene optimization. The optimized representation supports rendering of camera, lidar, instance masks, and motion trajectories.

Alternatives to per-scene optimization

Per-scene optimization remains the dominant paradigm in neural rendering for AD, but its cost has motivated several alternative directions, each making a different trade-off between fidelity, controllability, and the cost of producing a new scene.

Feed-forward reconstruction Feed-forward methods learn a generalizable model that maps input observations directly to a scene representation in a single forward pass, eliminating per-scene optimization entirely. Recent surveys provide comprehensive reviews of the broader field [117], [118]; here we restrict attention to AD-relevant developments. Several methods predict Gaussian primitives directly from sparse surround-view input, either under a static-scene assumption [119] or with explicit handling of dynamics through predicted scene flow [120], [121]. A complementary line operates on pointmaps rather than Gaussians, extending DUSt3R-style regression [122] to the AD setting [123]. Other methods couple feed-forward reconstruction with generative components, using HD-map-conditioned voxel diffusion to support scene synthesis and extrapolation [124], [125]. Current AD-targeted feed-forward methods do not consistently match per-scene optimization in rendering fidelity, but the gap is narrowing and the throughput advantage is structural. Hybrid designs that use feed-forward output as initialization for a short per-scene refinement, or

that iteratively refine feed-forward predictions through residual feedback [126], [127], offer a plausible route to combining the throughput of the feed-forward paradigm with the fidelity ceiling of per-scene optimization.

Generative world models A more radical alternative abandons explicit scene reconstruction and replaces it with a generative model that synthesizes plausible sensor observations conditioned on driving context. Models such as the GAIA series [35], [36], Cosmos [37] and its driving-specific variant [38], and the Waymo World Model [39] take video, action, and structured conditioning as input and produce future sensor observations as output, without an explicit 3D scene at any intermediate stage. Their defining capability is the synthesis of scenes that were never recorded, including traffic scenarios, weather conditions, and times of day that are absent or rare in the training distribution but can be produced through conditioning. The strengths and limitations are largely complementary to those of reconstruction-based methods. The marginal cost of producing a new scenario is the cost of inference rather than the cost of recording, annotating, and reconstructing it, and variation along axes such as illumination and weather is comparatively easy to produce. Conversely, conditioning is not the same as precise control: actor placement, sensor characteristics of the kind discussed in Section 4.2, and geometric consistency across views and time are not enforced by construction and are difficult to verify at the precision that perception evaluation requires. Lidar output is largely absent from the paradigm altogether. Training such models also requires data and compute resources on a scale only available among a small number of industrial actors, which is reflected in the origin of the most prominent results. Reconstruction and generation are likely to remain complementary in the near term, and an active line of work increasingly combines the two by using reconstructed scenes as conditioning for generative variation, or generative outputs as conditioning for reconstruction.

Actor compositing A third direction sidesteps full-scene reconstruction by compositing pre-built actor assets into a separately reconstructed static background. An actor library is populated with neural representations of individual vehicles, pedestrians, and other dynamic objects, optimized once per asset and inserted at arbitrary positions and orientations into any background scene. Recent work has extended this paradigm using diffusion-based refinement to improve realism and scene integration of inserted actors [128], and generating new assets directly in a learned latent space rather than relying on a fixed library [129]. The approach offers strong scenario controllability and amortizes asset cost across scenes, but the static background is

typically still produced by per-scene reconstruction.

Discussion These directions occupy different positions on a fidelity-controllability-cost surface rather than ranking on a single axis. Per-scene optimization currently sets the bar for reconstruction fidelity and for the precise reproducibility required when an AD system must be evaluated against a specific recorded scenario. Actor compositing extends this paradigm by amortizing actor models across scenes, raising the utility of each reconstruction. Feed-forward methods aim to retain the geometric grounding of optimization-based approaches while removing the per-scene cost. Generative world models trade explicit reconstruction for breadth and ease of producing new scenarios, with consistency and physical plausibility as the open issues. The latter two directions have received limited attention addressing the physical sensor processes needed for high-fidelity simulation, as discussed in Section 4.2, suggesting a promising avenue for future research.

The work in this thesis sits within the per-scene optimization paradigm and contributes to relaxing its main costs: Paper B reduces optimization and rendering time by an order of magnitude relative to volume-rendering predecessors, Paper C removes the dependence on manual 3D annotation, and Paper D extends the primitive-count ceiling for mesh-based representations that may eventually serve as alternatives to Gaussian primitives in this setting.

CHAPTER 5

Simulation validity and the real-to-sim gap

Chapters 3 and 4 address how to use neural rendering to build digital twins for autonomous driving that are high-fidelity and computationally scalable. This chapter asks a qualitatively different question: once a digital twin has been built, how reliably can conclusions drawn from simulated data be transferred to real data?

A digital twin that produces photorealistic images but introduces structured visual artifacts not captured by image quality metrics can produce misleading inputs to an AD system in ways that are difficult to detect without direct comparison to real data. If a simulator produces artifacts that make it easier for a perception model to detect objects, then the test results in simulation will overestimate the performance of the model. Conversely, if the simulator produces artifacts that make it harder for the model to detect objects, equally wrong conclusions will be drawn about the model's performance.

This chapter examines the real-to-sim gap, its relationship to rendering quality and downstream performance, and what can be done to make simulation results more reliable in practice.

5.1 The real-to-sim gap

Sim-to-real Two distinct but related simulation gaps arise in autonomous driving development. The *sim-to-real gap* is the long-studied problem of transferring knowledge acquired in simulation to real-world deployment. A model trained on synthetic data often generalizes poorly when applied to real sensors because the simulator's appearance and physics differ from reality. In the AD context, this gap remains actively relevant. It arises when using game-engine-based simulators to generate perception training data, when training planners or reinforcement learning agents in synthetic environments, or when augmenting real datasets with synthetic sensor observations. Extensive work in robotics and machine learning has developed strategies for bridging this gap, including domain randomization, adversarial feature alignment, and image-to-image stylization [3], [4], [130].

Real-to-sim The complementary problem, and the focus of this chapter, is the *real-to-sim gap*: the tendency of a system trained entirely on real data to behave differently when evaluated on simulated data. This gap is specific to simulation-based validation, where an AD system that has never been exposed to rendered images is asked to perform on them as a proxy for real-world testing. Unlike the sim-to-real setting, no domain adaptation has been applied during training, and the gap arises purely from the discrepancy between real and rendered sensor observations. The two gaps are not mutually exclusive. A development pipeline typically uses simulated data in both settings: augmenting training with targeted synthetic data, and validating on simulated high-fidelity renderings. Each setting carries its own dominant gap, and warrants different mitigation strategies. Despite the growing interest in neural rendering for AD, the real-to-sim direction has received considerably less systematic attention than its sim-to-real counterpart, and large-scale studies across models, scenes, and rendering qualities remain scarce.

Measuring the gap Measuring the gap on interpolated views, where each rendered image has a corresponding real image, is the most direct setting. Two complementary quantities are informative. The first is the performance drop, defined as how much the model's standard metric (such as object detection accuracy) deteriorates for a given perception task when the input is rendered rather than real-world data. The second is detection agreement, which measures the fraction of predictions that the model makes consistently across the two modalities, regardless of whether those predictions are correct with respect to ground truth. Performance drop quantifies

how much the simulation under- or overestimates model capability, while detection agreement quantifies whether the model is responding to the same features in both settings, and is therefore the more direct proxy for simulation validity. As shown in Paper E, the detection agreement can vary drastically between models evaluated on the same rendered scenes, highlighting the need for investigating the gap for the specific simulation use case.

Prior work Few works on neural rendering for AD address the real-to-sim gap directly. Where it appears, it is most often as supplementary downstream experiments accompanying a method whose main contribution is rendering quality. UniSim [45], DyNFL [53], and NeuRAD [7] each report how various downstream models, spanning object detection, semantic segmentation, motion forecasting, planning, and depth estimation, perform on rendered sensor observations relative to real ones, reporting performance drop and detection agreement as measures of the real-to-sim gap. While these experiments provide valuable quantitative results on the gap, they are run on small evaluation sets relative to the scale at which AD perception is normally benchmarked and offer little practical guidance on how to characterize, predict, or mitigate the gap going forward, with more extensive studies so far limited to fully synthetic setups [131]. Paper E [12] takes up the real-to-sim question as the object of study rather than as a side check on a rendering method.

5.2 Limitations of image quality metrics

Progress in neural rendering for AD is conventionally tracked through the image quality metrics discussed in Section 3.1, which quantify the pixel- or perceptual-level distance between rendered and real images. The implicit assumption is that these metrics serve as a proxy for the real-to-sim gap. If rendered images are perceptually indistinguishable from real ones, then a perception model evaluated on them should respond as if they were real. This assumption is partially correct, as shown by the real-to-sim experiments in prior work where methods with higher rendering fidelity also perform better when it comes to agreement with real data on downstream tasks [7], [45]. But the relationship between image-space metrics and downstream task behavior is more nuanced.

Image quality metrics vs. downstream performance Image quality metrics are natural objectives for a rendering method since they directly quantify what is being optimized. However, while serving as useful proxies for the real-to-sim gap, they do

(a) Original **(b)** NeuRAD rendering **(c)** Original with color shift

Figure 5.1: Image quality metrics can be misleading proxies for downstream performance. Detections from an object detector are overlaid as 2D bounding boxes (blue: pedestrian, orange: car). (a) The original image, on which the perception model correctly detects the pedestrian. (b) A NeuRAD rendering of the same scene: actor blurring causes the detection to be missed, despite the rendering attaining a relatively high PSNR. (c) The original image with a uniform color shift applied: PSNR is lower than in (b), yet the detection is preserved exactly because the spatial structure the model relies on is intact. The example illustrates how rendering errors that are spatially concentrated on semantically loaded regions can affect downstream performance more than larger, more uniformly distributed pixel-level errors.

not tell the full picture. A method with better PSNR may still produce artifacts that are more consequential for tasks such as object detection. A rendering method may distribute its error across many pixels in ways that are imperceptible to a perception model, or concentrate it in small regions that happen to alter a single bounding box prediction. Reported failure modes in neural rendering for AD include blurry reconstructions of non-rigid actors such as pedestrians, thin structures appearing semi-transparent under long night-time exposures, and floaters from strong blooming and lens-flare [7]. As illustrated in Figure 5.1, blurry actor reconstructions can cause missing detections from downstream models, despite achieving better PSNR than the color-distorted version with zero real-to-sim gap. Sensor effects such as rolling shutter and motion blur similarly have a small impact on image quality metrics when unmodeled, yet their omission introduces a visible gap to real data that a perception model may be sensitive to [7].

Image quality metrics still serve as useful proxies for the gap, in particular when complemented with experiments to verify their correlation to downstream performance, a topic further studied and discussed in Paper E.

Downstream performance in extrapolated views Extrapolated views, produced by rendering from poses that deviate from the training trajectory, are the setting most relevant for closed-loop simulation and targeted scenario generation. They are also the setting in which the standard evaluation setup breaks down. No corresponding real image exists, which rules out both image quality metrics as well as any direct measurement of detection agreement. Distribution-based metrics such as FID remain computable, since it compares distributions rather than paired images, but its scores are difficult to interpret directly. However, correlations can first be established between FID and detection agreement on interpolated views, and then be used as guidance in the extrapolated setting. In practical terms, FID can be used to calibrate how far a given rendering method can be extrapolated before the expected real-to-sim gap exceeds an acceptable threshold.

A second, and in some ways more fundamental, difficulty is that extrapolated evaluations combine two distinct effects. The first is a genuine degradation of the renderings at poses far from the training trajectory. The second is that the downstream model itself may perform worse in extrapolated scenarios, which differ from those it was exposed to during training. For instance, a laterally shifted camera places the vehicle between lanes or close to road boundaries, configurations that are rare in the real training distribution. A measured performance drop on a downstream task under extrapolation, therefore, cannot be cleanly separated without controlled real-world data from the same shifted poses. Closed-loop simulation is consequently the setting that most justifies neural rendering, and the one in which the real-to-sim gap is hardest to measure.

5.3 Closing the gap

The preceding sections have aimed to give a more nuanced picture of what closing the real-to-sim gap entails. Improving rendering fidelity remains a productive direction, and image quality metrics remain useful benchmarks for tracking it, but neither on its own gives a complete account of simulation validity for a given downstream task. This section turns to what can be done about the residual gap in practice. Two directions are discussed here: model-side strategies that adjust the perception model so that whatever artifacts remain matter less for its predictions, and rendering-side strategies that use generative priors to reduce the artifacts themselves in regimes where the reconstruction signal is weak. Each addresses a different part of the problem, and each carries its own caveats, which the section also examines.

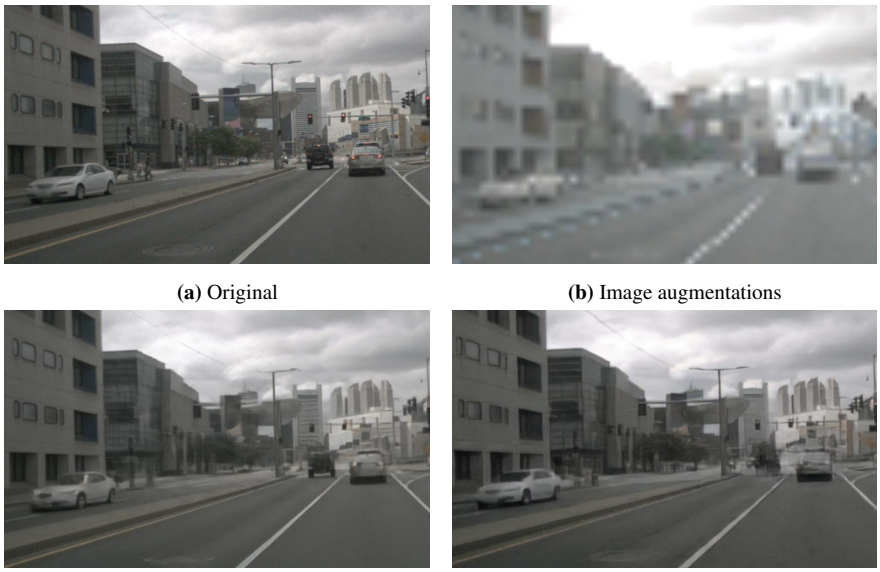
Robustness to rendering artifacts A complementary direction to improving simulator fidelity is to address the gap from the model side. Rather than making the simulator more realistic, the downstream model is made more tolerant of simulation artifacts. The most direct model-side intervention is to fine-tune the downstream model on data that expose it to the artifacts simulation will introduce. Strategies span a spectrum from agnostic to artifact-specific, with corresponding trade-offs in cost and effectiveness.

At the simulator-agnostic end, classical image augmentations applied during fine-tuning, such as Gaussian blur, photometric jitter, additive noise, and downsampling followed by upsampling, broaden the distribution the model is exposed to without requiring any rendered data and therefore scale to full perception datasets at negligible cost. These are common augmentation strategies when training deep learning models on computer vision tasks [132], and are likely already included in the model's original training. It is therefore not obvious that further fine-tuning with similar augmentations provides additional value. A separate question is whether these generic augmentations can be tuned to match the specific artifacts produced by neural rendering simulators, rather than just broadening the input distribution in the abstract.

At the simulator-specific end, fine-tuning on a mixture of real and rendered images from the simulator directly exposes the model to the same artifact structure it will be evaluated against. This comes at the cost of running the rendering pipeline at training scale: every scene to be used as augmentation must first be reconstructed, which can incur substantial computational cost at the scale of typical perception training datasets.

An intermediate strategy is to learn an image-to-image model [133] that maps real images to the rendered domain and use its output as augmentation. Such models can be trained once on a relatively small set of rendered target images, and then applied at inference cost to a much larger dataset of real images, producing render-like augmentations at the scale of the perception training set. This strategy avoids per-scene reconstruction while still exposing the model to renderer-like artifacts, subject to how well the image-to-image model captures the renderer's output distribution. Figure 5.2 illustrates examples of these different strategies applied to the same image. Paper E discusses the approaches and their relative performance across downstream models in detail.

When robustness becomes adaptation Fine-tuning the downstream model to be more robust against rendering artifacts can affect its performance on real data, and it is important to verify that real-data performance is not sacrificed in the process. As



(a) Original **(b)** Image augmentations

(c) Rendered image **(d)** Image-to-image

Figure 5.2: The three model-side augmentation strategies discussed in Section 5.3, applied to the same source image. (a) Original real image. (b) Classical image augmentations (Gaussian blur, photometric jitter, additive noise, downsampling-then-upsampling), which broaden the input distribution without targeting specific simulator artifacts. (c) A NeuRAD rendering of the same scene, which carries the artifact structure of the rendering pipeline directly but requires per-scene reconstruction. (d) An image-to-image translation of the original toward the rendered domain, which approximates the rendered artifact distribution at inference cost without per-scene reconstruction.

discussed in Paper E [12], this varies considerably across augmentation strategies and models, where some configurations leave real-data performance untouched, others degrade it slightly, and, more interestingly, certain augmentations actually improve it for some models. The robustness gained depends on the specific combination of renderer and downstream model, since the artifact structure changes with the rendering method, the reconstruction quality, and the model's own sensitivities. A reliable understanding of how a given fine-tuning strategy affects a given (renderer, downstream model) pair therefore has to be established per case rather than assumed from a general result.

Augmenting real-world training with rendered views The fine-tuning machinery using rendered views can be turned to a different purpose: making downstream models more robust on real data by training on rendered views that expose them to configurations that are rare or absent in the original training distribution. Driving data is heavily concentrated near the lane center, since vehicles spend most of their time there, and downstream models trained on such data tend to perform poorly when confronted with laterally shifted, off-axis, or steep cut-in configurations at test time [12]. Synthesizing training views that fill these gaps has been a long-standing goal. The early end-to-end driving work of Bojarski et al. [18] simulated lane-shift and rotation perturbations through approximate viewpoint warping precisely to teach the network to recover from off-center positions. Neural rendering offers a substantially better tool for this, since it produces geometrically and photometrically faithful renderings rather than warped approximations, and labels from the original views can in many cases be transferred to the synthesized poses through the known 3D transformation between them. Several works have reported real-data improvements from training on rendered novel views. UC-NeRF [134] augments monocular-depth training with NeRF-rendered views and reports sharper, more accurate depth predictions on real data, and S-NeRF++ [135] reports performance gains on multiple downstream perception tasks when training is supplemented with NeRF-simulated scenes. The caveat for this use of rendered data is that geometrically valid relabeling assumes that the new viewpoint sees the same scene content as the original, which may break under occlusion. A laterally shifted camera can reveal previously occluded surfaces or hide previously visible ones, and labels copied directly from the original pose will be wrong in those regions.

Generative priors for residual artifacts Model-side robustness reduces sensitivity to artifacts the model has been exposed to during fine-tuning. It does not,

on its own, address the more severe quality degradation at extrapolated viewpoints, where the rendering method itself is operating outside its reliable regime, and where augmentation strategies calibrated to interpolated-view artifacts are not guaranteed to cover the different failure modes that appear under large pose shifts.

A complementary direction, pursued independently of model-robustness work, is to improve the rendering method's own behavior in these regimes through generative priors. Diffusion models trained on large corpora of real driving imagery encode rich knowledge of plausible scene content, which can be used to fill in regions of an extrapolated rendering that the underlying reconstruction has left underdetermined.

The diffusion model can be used as a post-hoc image enhancer that operates on rendered outputs without modifying the underlying reconstruction [128]. Alternatively, generated views can be distilled back into the 3D representation during reconstruction, training it on a mixture of observed and generated views so that the geometry and appearance optimization is aware of the prior [136]. General-purpose work on diffusion priors for novel-view synthesis, including ReconFusion [137] and CAT3D [138], has demonstrated high-quality 3D reconstructions from very few input views, and is a promising route to improved extrapolation.

Taken together, the strategies discussed in this section do not eliminate the real-to-sim gap, but they make it more visible, at least partially quantifiable, and bounded, which is the precondition for using simulation results to draw trustworthy conclusions about real-world system behavior.

CHAPTER 6

Summary of included papers

This chapter provides a summary of the included papers.

6.1 Paper A

Adam Tonderski[†], **Carl Lindström**[†], Georg Hess[†], William Ljungbergh, Lennart Svensson, Christoffer Petersson
NeuRAD: Neural Rendering for Autonomous Driving
Published in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),
pp. 14895–14904, 2024.
DOI: 10.1109/CVPR52733.2024.01411

We propose NeuRAD, a novel view synthesis method for dynamic autonomous driving scenes. Our method is the first to jointly model full 360° multi-camera rigs and lidar data, and the first to emphasize sensor-accurate modeling of rolling shutter

effects, lidar beam divergence, and ray dropping for improved realism. Together, these contributions set a new state of the art across five popular autonomous driving benchmarks.

AT, CL and GH share first authorship. They jointly developed the method, carried out the experiments, and wrote the manuscript. The implementation was done by AT, CL, and GH, with assistance from WL. LS and CP provided valuable feedback on the method and manuscript.

6.2 Paper B

Georg Hess†, **Carl Lindström**†, Maryam Fatemi, Christoffer Petersson, Lennart Svensson
SplatAD: Real-Time Lidar and Camera Rendering with 3D Gaussian Splatting for Autonomous Driving
Published in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11982–11992, 2025.
DOI: 10.1109/CVPR52734.2025.01119

NeuRAD produces high-fidelity reconstructions but training takes several hours, and rendering is too slow for real-time simulation. We propose SplatAD, which reformulates the problem using 3D Gaussian Splatting to enable real-time rendering and training times measured in minutes rather than hours. While existing 3DGS methods are limited to camera data, SplatAD is the first to jointly render both camera and lidar from a single unified representation. This is achieved through purpose-built rasterization algorithms paired with sensor modeling designed to fit the rasterization paradigm. SplatAD reaches competitive performance within minutes of training and surpasses the previous state of the art when matching their training times.

GH and CL share first authorship. They collaboratively developed and implemented the method, performed the experiments, and wrote the manuscript. MF, CP, and LS provided valuable feedback on the experimental setup and the manuscript.

6.3 Paper C

Carl Lindström[†], Mahan Rafidashti[†], Maryam Fatemi, Lars Hammarstrand, Martin R. Oswald, Lennart Svensson
IDSplat: Instance-Decomposed 3D Gaussian Splatting for Driving Scenes
To be published in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Findings (CVPRF)*, 2026

Existing work on dynamic scene reconstruction rely on costly human-annotated object trajectories or model dynamic content as unstructured time-varying primitives, preventing decomposition and manipulation of individual objects. We propose IDSplat, which instead reconstructs dynamic scenes with explicit instance-level decomposition and learnable motion trajectories, without requiring any human annotations. By combining vision foundation models with classical matching and estimation techniques, IDSplat achieves competitive reconstruction quality while enabling practical scene manipulation such as actor removal and trajectory editing.

CL and MR share first authorship. CL proposed the original project idea and concept in discussions with LS. CL led the development and implementation of the proposed method, with MR contributing to the design and refinement of the approach. MR took primary responsibility for setting up and running baseline comparisons. CL and MR jointly carried out the experiments and led the writing of the manuscript. MF, LH and MO provided input on the method and experiments, and all authors contributed valuable feedback on the manuscript.

6.4 Paper D

Bernardo Taveira[†], **Carl Lindström**[†], Maryam Fatemi, Lars Hammarstrand, Fredrik Kahl
Scalable GPU Construction of 3D Voronoi and Power Diagrams
To be published in *ACM SIGGRAPH 2026 Conference Proceedings (SIGGRAPH)*, 2026

Mesh-based neural rendering methods represent scenes as volumetric meshes derived from Voronoi or Delaunay topology, but repeatedly recomputing these structures during training creates a computational bottleneck that limits scalability. We introduce a highly parallelizable GPU algorithm for constructing large-scale 3D Voronoi and power diagrams, where each cell is computed independently through iterative cell-clipping, using a directional culling criterion and bounding volume hierarchy for

efficient neighbor search. The method scales robustly to tens of millions of points where existing methods fail, and naturally generalizes to power diagrams. Applied to mesh-based neural rendering, removing this bottleneck directly enables larger scene representations and improved reconstruction quality.

BT and CL share first authorship. They jointly conceptualized the work, developed and implemented the method, designed and conducted the experiments, and took the lead on writing the manuscript. MF, LH, and FK contributed through discussions on problem formulation and evaluation and provided substantial feedback on the experiments and the manuscript.

6.5 Paper E

Carl Lindström[†], Georg Hess[†], Adam Lilja, Maryam Fatemi, Lars Hammarstrand, Christoffer Petersson, Lennart Svensson

Are NeRFs ready for autonomous driving? Towards closing the real-to-simulation gap

Published in IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),
pp. 4461–4471, 2024.
DOI: 10.1109/CVPRW63382.2024.00449

As neural simulators improve, a critical question emerges: do perception models interpret simulated data the same way as real sensor data? We conduct the first large-scale study of the real-to-simulation gap in an autonomous driving context, evaluating object detectors and an online mapping model on both real and neural-rendered data. Rather than exclusively targeting rendering fidelity, we propose fine-tuning strategies that improve model robustness to simulation artifacts, yielding notable gains and in some cases also improving real-world performance. We further identify FID and LPIPS as strong predictors of the real-to-simulation gap, offering practical guidance on when neural simulators can be trusted.

CL and GH share first authorship. They developed the overall methodology, ran the experiments, and analyzed the results, with AL providing additional support in

implementation and experimentation. CL and GH led the writing of the manuscript, with all co-authors providing valuable feedback on the manuscript.

CHAPTER 7

Concluding remarks

In this thesis, we have studied how neural rendering can be used to construct digital twins of recorded driving data, with the goal of creating virtual environments that are sufficiently realistic and controllable to support the validation of AD systems. Our contributions include methods for jointly rendering camera and lidar data, modeling key sensor characteristics, reducing the cost and annotation requirements needed to scale these approaches, and systematically evaluating how well simulated data reflects real-world system behavior. While these results represent meaningful progress, several open challenges remain before neural simulators can serve as a primary tool for AD validation.

Toward the full automotive sensor suite This thesis focuses on the joint treatment of camera and lidar modalities, while radar remains largely outside the scope of current neural rendering approaches. Yet radar is a core component in many AD systems and is essential to simulate for comprehensive system validation. Developing methods that can render camera, lidar, and radar from a unified learned representation, ensuring cross-modal consistency by construction, would help bridge a significant gap between existing neural simulators and real-world sensor configurations. A notable step in this direction is NeuRadar [31], which demonstrates such unified multi-modal rendering, but this line of work remains relatively underexplored and would benefit

from further attention. Progress in this direction is currently limited by the scarcity of datasets that include radar, particularly those that capture all three modalities simultaneously. Expanding such datasets would likely play a key role in advancing the field.

Richer models of dynamic actors The works in this thesis model dynamic scenes as collections of rigid instances, enabling precise trajectory editing but limiting the representation of non-rigid motion in pedestrians, cyclists, and articulated vehicle parts. At the other extreme, time-varying primitive methods capture such dynamics by absorbing them into position, appearance, and opacity variations, but at the cost of instance-level controllability. A promising direction is to bridge these paradigms through multi-scale representations, where coarse rigid motion captures object-level kinematics and finer non-rigid effects are modeled as local, time-dependent deviations, balancing control and expressiveness. More fundamentally, it remains an open question how accurately such motion must be reconstructed for the downstream tasks the simulation is meant to support. Addressing this requires systematic study of the real-to-sim gap as a function of motion fidelity.

Reconstruction and generation in combination The methods proposed in this thesis achieve high-fidelity renderings and accurate geometric representations, but they remain fundamentally limited by the coverage of the original training views. For AD simulation, it is desirable to deviate substantially from the recorded trajectories in order to explore a wider range of scenarios. Enabling this requires neural rendering approaches to go beyond faithful reconstruction: in addition to stronger regularization, they must be able to plausibly infer and synthesize content in regions that were not observed during training. Recent advances in generative models point to a promising direction for extending neural scene representations [128], [136]–[138]. Key open challenges include extending these methods to support a broader range of camera models, as well as generalizing them to lidar and radar modalities. Another important research direction is the benchmarking of such generative approaches in the absence of ground-truth data, an issue that will shape future progress and, if addressed effectively, could accelerate development across the field.

Rendering paradigms and explicit representations The rasterization paradigm has significantly accelerated the development and practical applicability of neural rendering for AD, with SplatAD contributing along this line. This efficiency, however, comes with approximations: sensor effects that are straightforward to model under ray

tracing often require careful workarounds in a rasterization setting. Recent advances in mesh-based neural rendering offer a promising alternative, providing explicit representations that more naturally support both rasterization and ray tracing. Paper D contributes in this direction by enabling such methods to scale to the complexity required for AD scenes. Their adoption in AD, however, remains largely unexplored, making this an interesting avenue for future research.

References

[1] L. Di Lillo, T. Gode, X. Zhou, J. M. Scanlon, R. Chen, and T. Victor, "Do autonomous vehicles outperform latest-generation human-driven vehicles? a comparison to waymo's auto liability insurance claims at 25.3m miles," Waymo Research, Tech. Rep., 2024.

[2] D. Hepp, M. Kellner, S. Jautelat, M. Guggenheimer, and T. Aloise, *The automotive software and electronics market through 2035*, `https://www.mckinsey.com/features/mckinsey-center-for-future-mobility/our-insights/mapping-the-automotive-software-and-electronics-landscape`, Accessed: 2026-02-19, Jan. 2026.

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proceedings of the Conference on Robot Learning (CORL)*, PMLR, 2017, pp. 1–16.

[4] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*, Springer, 2018, pp. 621–635.

[5] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Cham: Springer International Publishing, 2020, pp. 405–421, ISBN: 978-3-030-58452-8.

[6] A. Tewari, J. Thies, B. Mildenhall, *et al.*, "Advances in neural rendering," in *Computer Graphics Forum*, Wiley Online Library, vol. 41, 2022, pp. 703–735.

[7] A. Tonderski, C. Lindström, G. Hess, W. Ljungbergh, L. Svensson, and C. Petersson, "Neurad: Neural rendering for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 14 895–14 904.

[8] G. Hess, C. Lindström, M. Fatemi, C. Petersson, and L. Svensson, "SplatAD: Real-time lidar and camera rendering with 3D Gaussian splatting for autonomous driving," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 11 982–11 992.

[9] C. Lindström, M. Rafidashti, M. Fatemi, L. Hammarstrand, M. R. Oswald, and L. Svensson, "Idsplat: Instance-decomposed 3d gaussian splatting for driving scenes," *arXiv preprint arXiv:2511.19235*, 2025.

[10] W. Ljungbergh, A. Tonderski, J. Johnander, *et al.*, "NeuroNCAP: Photorealistic closed-loop safety testing for autonomous driving," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2025, pp. 161–177.

[11] S. Govindarajan, D. Rebain, K. M. Yi, and A. Tagliasacchi, "Radiant foam: Real-time differentiable ray tracing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025, pp. 4135–4145.

[12] C. Lindström, G. Hess, A. Lilja, *et al.*, "Are NeRFs ready for autonomous driving? towards closing the real-to-simulation gap," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024, pp. 4461–4471.

[13] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[14] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023.

[15] Z. Li, W. Wang, H. Li, *et al.*, "Bevformer: Learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[16] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8844–8854.

[17] J. Wang, T. Ye, Z. Gu, and J. Chen, "Ltp: Lane-based trajectory prediction for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 17 134–17 142.

[18] M. Bojarski, D. Del Testa, D. Dworakowski, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[19] Y. Hu, J. Yang, L. Chen, *et al.*, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 17 853–17 862.

[20] B. Jiang, S. Chen, Q. Xu, *et al.*, "Vad: Vectorized scene representation for efficient autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 8340–8350.

[21] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[22] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[23] P. Xiao, Z. Shao, S. Hao, *et al.*, "Pandaset: Advanced sensor suite dataset for autonomous driving," in *IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 3095–3101.

[24] B. Wilson, W. Qi, T. Agarwal, *et al.*, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[25] P. Sun, H. Kretzschmar, X. Dotiwalla, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2446–2454.

[26] H. Caesar, V. Bankiti, A. H. Lang, *et al.*, "nuScenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 621–11 631.

[27] M. Alibeigi, W. Ljungbergh, A. Tonderski, *et al.*, "Zenseact open dataset: A large-scale and diverse multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 20 178–20 188.

[28] R. Roriz, J. Cabral, and T. Gomes, "Automotive lidar technology: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6282–6297, 2022.

[29] D. Borts, E. Liang, T. Broedermann, *et al.*, "Radar fields: Frequency-space neural scene representations for fmcw radar," in *ACM SIGGRAPH 2024 Conference Papers*, 2024, pp. 1–10.

[30] T. Huang, J. Miller, A. Prabhakara, *et al.*, "Dart: Implicit doppler tomography for radar novel view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 24 118–24 129.

[31] M. Rafidashti, J. Lan, M. Fatemi, J. Fu, L. Hammarstrand, and L. Svensson, "NeuRadar: Neural radiance fields for automotive radar point clouds," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2025, pp. 2479–2489.

[32] P.-C. Kung, S. Harisha, R. Vasudevan, A. Eid, and K. A. Skinner, "Radarsplat: Radar gaussian splatting for high-fidelity data synthesis and 3d reconstruction of autonomous driving scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025, pp. 27 596–27 606.

[33] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016, ISSN: 0965-8564.

[34] European New Car Assessment Programme (Euro NCAP), *Euro ncap test and assessment protocols*, Accessed: 2025-02-21, 2025.

[35] A. Hu, L. Russell, H. Yeo, *et al.*, "Gaia-1: A generative world model for autonomous driving," *arXiv preprint arXiv:2309.17080*, 2023.

[36] L. Russell, A. Hu, L. Bertoni, *et al.*, "Gaia-2: A controllable multi-view generative world model for autonomous driving," *arXiv preprint arXiv:2503.20523*, 2025.

[37] N. Agarwal, A. Ali, M. Bala, *et al.*, "Cosmos world foundation model platform for physical ai," *arXiv preprint arXiv:2501.03575*, 2025.

[38] X. Ren, Y. Lu, T. Cao, *et al.*, "Cosmos-drive-dreams: Scalable synthetic driving data generation with world foundation models," *arXiv preprint arXiv:2506.09042*, 2025.

[39] Waymo, *The Waymo world model: A new frontier for autonomous driving simulation*, Waypoint: The official Waymo blog, Accessed: 2026-05-02, Feb. 2026.

[40] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–14, 2023.

[41] J. Ost, F. Mannan, N. Thuerey, J. Knodt, and F. Heide, "Neural scene graphs for dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 2856–2865.

[42] A. Kundu, K. Genova, X. Yin, *et al.*, "Panoptic neural fields: A semantic object-aware neural scene representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 871–12 881.

[43] X. Fu, S. Zhang, T. Chen, *et al.*, "Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation," in *International Conference on 3D Vision (3DV)*, IEEE, 2022, pp. 1–11.

[44] Z. Xie, J. Zhang, W. Li, F. Zhang, and L. Zhang, "S-neRF: Neural radiance fields for street views," in *International Conference on Learning Representations (ICLR)*, 2023.

[45] Z. Yang, Y. Chen, J. Wang, *et al.*, "Unisim: A neural closed-loop sensor simulator," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 1389–1399.

[46] Z. Wu, T. Liu, L. Luo, *et al.*, "MARS: An instance-aware, modular and realistic simulator for autonomous driving," *CAAI International Conference on Artificial Intelligence*, 2023.

[47] Y. Yan, H. Lin, C. Zhou, *et al.*, "Street gaussians for modeling dynamic urban scenes," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[48] X. Zhou, Z. Lin, X. Shan, Y. Wang, D. Sun, and M.-H. Yang, "Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 21 634–21 643.

[49] Z. Chen, J. Yang, J. Huang, *et al.*, "Omnire: Omni urban scene reconstruction," *arXiv preprint arXiv:2408.16760*, 2024.

[50] M. Khan, H. Fazlali, D. Sharma, *et al.*, *Autosplat: Constrained gaussian splatting for autonomous driving scene reconstruction*, 2024.

[51] H. Zhou, J. Shao, L. Xu, *et al.*, "Hugs: Holistic urban 3d scene understanding via gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 21 336–21 345.

[52] S. Huang, Z. Gojcic, Z. Wang, *et al.*, "Neural lidar fields for novel view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[53] H. Wu, X. Zuo, S. Leutenegger, O. Litany, K. Schindler, and S. Huang, "Dynamic lidar re-simulation using compositional neural fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 19 988–19 998.

[54] Z. Zheng, F. Lu, W. Xue, G. Chen, and C. Jiang, "Lidar4d: Dynamic neural fields for novel space-time view lidar synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 5145–5154.

[55] H. Turki, J. Y. Zhang, F. Ferroni, and D. Ramanan, "SUDS: Scalable urban dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 12 375–12 385.

[56] J. Yang, B. Ivanovic, O. Litany, *et al.*, "EmerneRF: Emergent spatial-temporal scene decomposition via self-supervision," in *International Conference on Learning Representations (ICLR)*, 2024.

[57] Y. Chen, C. Gu, J. Jiang, X. Zhu, and L. Zhang, "Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering," *arXiv preprint arXiv:2311.18561*, 2023.

[58] G. Wu, T. Yi, J. Fang, *et al.*, "4D Gaussian splatting for real-time dynamic scene rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 20 310–20 320.

[59] C. Peng, C. Zhang, Y. Wang, *et al.*, "DeSiRe-GS: 4D street gaussians for static-dynamic decomposition and surface reconstruction for urban driving scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 6782–6791.

[60] X. Jiawei, D. Kai, F. Zexin, W. Shenlong, X. Jin, and Y. Jian, "AD-GS: Object-aware B-Spline Gaussian splatting for self-supervised autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025, pp. 24 770–24 779.

[61] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[62] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 586–595.

[63] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[64] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96, New York, NY, USA: Association for Computing Machinery, 1996, pp. 31–42, ISBN: 0897917464.

[65] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96, New York, NY, USA: Association for Computing Machinery, 1996, pp. 43–54, ISBN: 0897917464.

[66] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "Deepstereo: Learning to predict new views from the world's imagery," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2016, pp. 5515–5524.

[67] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2016, pp. 286–301.

[68] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," *ACM Transactions on Graphics (ToG)*, vol. 37, no. 6, pp. 1–15, 2018.

[69] G. Riegler and V. Koltun, "Free view synthesis," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 623–640.

[70] J. Thies, M. Zollhöfer, and M. Nießner, "Deferred neural rendering: Image synthesis using neural textures," *Acm Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[71] P. Henzler, N. J. Mitra, and T. Ritschel, "Escaping plato's cave: 3d shape from adversarial rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9984–9993.

[72] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer, "Deepvoxels: Learning persistent 3d feature embeddings," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2437–2446.

[73] M. Tancik, P. Srinivasan, B. Mildenhall, *et al.*, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 7537–7547, 2020.

[74] J. T. Kajiya and B. P. Von Herzen, "Ray tracing volume densities," *ACM SIGGRAPH computer graphics*, vol. 18, no. 3, pp. 165–174, 1984.

[75] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–15, 2022.

[76] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 5501–5510.

[77] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 5752–5761.

[78] S. Fridovich-Keil, G. Meanti, F. R. Warburg, B. Recht, and A. Kanazawa, "K-planes: Explicit radiance fields in space, time, and appearance," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 12 479–12 488.

[79] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2022, pp. 333–350.

[80] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi, "Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 16 569–16 578.

[81] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking neural radiance fields for real-time view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 5875–5884.

[82] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5855–5864.

[83] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded anti-aliased neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 5470–5479.

[84] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Zip-NeRF: Anti-aliased grid-based neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 19 697–19 705.

[85] W. Hu, Y. Wang, L. Ma, *et al.*, "Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 19 774–19 783.

[86] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 27 171–27 183.

[87] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 4805–4815, 2021.

[88] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, "Ewa volume splatting," in *Proceedings Visualization, 2001. VIS'01.*, IEEE, 2001, pp. 29–538.

[89] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.

[90] Z. Yu, A. Chen, B. Huang, T. Sattler, and A. Geiger, "Mip-splatting: Alias-free 3d gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 19 447–19 456.

[91] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," in *ACM SIGGRAPH 2024 Conference Papers*, ser. SIGGRAPH '24, Denver, CO, USA: Association for Computing Machinery, 2024, ISBN: 9798400705250.

[92] Z. Yu, T. Sattler, and A. Geiger, "Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 6, pp. 1–13, 2024.

[93] S. Kheradmand, D. Rebain, G. Sharma, *et al.*, "3d gaussian splatting as markov chain monte carlo," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[94] O. Seiskari, J. Ylilammi, V. Kaatrasalo, *et al.*, "Gaussian splatting on the move: Blur and rolling shutter compensation for natural camera motion," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2024, pp. 160–177.

[95] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, and Z. Wang, "Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, pp. 140 138–140 158, 2024.

[96] L. Radl, M. Steiner, M. Parger, A. Weinrauch, B. Kerbl, and M. Steinberger, "Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 4, pp. 1–17, 2024.

[97] N. Moenne-Loccoz, A. Mirzaei, O. Perel, *et al.*, "3d gaussian ray tracing: Fast tracing of particle scenes," *ACM Transactions on Graphics*, vol. 43, no. 6, pp. 1–19, 2024.

[98] A. Mai, P. Hedman, G. Kopanas, *et al.*, "Ever: Exact volumetric ellipsoid rendering for real-time view synthesis," *arXiv preprint arXiv:2410.01804*, 2024.

[99] Q. Wu, J. M. Esturo, A. Mirzaei, N. Moenne-Loccoz, and Z. Gojcic, "3dgut: Enabling distorted cameras and secondary rays in gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 26 036–26 046.

[100] J. Held, R. Vandeghen, A. Deliege, *et al.*, "Triangle splatting for real-time radiance field rendering," in *Thirteenth International Conference on 3D Vision*, 2025.

[101] A. Mai, T. Hedstrom, G. Kopanas, J. Kontkanen, F. Kuester, and J. T. Barron, "Radiance meshes for volumetric reconstruction," *arXiv preprint arXiv:2512.04076*, 2025.

[102] A. Hanson, A. Tu, V. Singla, M. Jayawardhana, M. Zwicker, and T. Goldstein, "Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5949–5958.

[103] M. Tancik, V. Casser, X. Yan, *et al.*, "Block-nerf: Scalable large scene neural view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8248–8258.

[104] J. Chen, W. Ye, Y. Wang, *et al.*, "Gigags: Scaling up planar-based 3d gaussians for large scene surface reconstruction," *arXiv preprint arXiv:2409.06685*, 2024.

[105] B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis, "A hierarchical 3d gaussian representation for real-time rendering of very large datasets," *ACM Transactions On Graphics (TOG)*, vol. 43, no. 4, pp. 1–15, 2024.

[106] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 34, no. 6, 248:1–248:16, Oct. 2015.

[107] A. Hata and D. Wolf, "Road marking detection using lidar reflective intensity data and its application to vehicle localization," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2014, pp. 584–589.

[108] S. Manivasagam, I. A. Bârsan, J. Wang, Z. Yang, and R. Urtasun, "Towards zero domain gap: A comprehensive study of realistic lidar simulation for autonomy testing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 8272–8282.

[109] S. Manivasagam, S. Wang, K. Wong, *et al.*, "Lidarsim: Realistic lidar simulation by leveraging the real world," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 167–11 176.

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

[111] D. Kotovenko, O. Grebenkova, and B. Ommer, "Edgs: Eliminating densification for efficient convergence of 3dgs," *arXiv preprint arXiv:2504.13204*, 2025.

[112] N. Ravi, V. Gabeur, Y.-T. Hu, *et al.*, "SAM 2: Segment anything in images and videos," *International Conference on Learning Representations (ICLR)*, 2025.

[113] T. Ren and S. Shen, *Grounded-SAM-2*, `https://github.com/IDEA-Research/Grounded-SAM-2`, 2024.

[114] N. Carion, L. Gustafson, Y.-T. Hu, *et al.*, *Sam 3: Segment anything with concepts*, 2025.

[115] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything: Unleashing the power of large-scale unlabeled data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 10 371–10 381.

[116] O. Siméoni, H. V. Vo, M. Seitzer, *et al.*, "DINOv3," *arXiv preprint arXiv:2508.10104*, 2025.

[117] J. Zhang, Y. Li, A. Chen, *et al.*, "Advances in feed-forward 3d reconstruction and view synthesis: A survey," *arXiv preprint arXiv:2507.14501*, 2025.

[118] W. Wang, Q. Cao, S. Gao, *et al.*, "Feed-forward 3d scene modeling: A problem-driven perspective," *arXiv preprint arXiv:2604.14025*, 2026.

[119] Q. Tian, X. Tan, Y. Xie, and L. Ma, "Drivingforward: Feed-forward 3d gaussian splatting for driving scene reconstruction from flexible surround-view input," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, 2025, pp. 7374–7382.

[120] H. Lu, T. Xu, W. Zheng, *et al.*, "Drivingrecon: Large 4d gaussian reconstruction model for autonomous driving," *arXiv preprint arXiv:2412.09043*, 2024.

[121] J. Yang, J. Huang, Y. Chen, *et al.*, "STORM: Spatio-temporal reconstruction model for large-scale outdoor scenes," *arXiv preprint arXiv:2501.00602*, 2025.

[122] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 20 697–20 709.

[123] X. Fei, W. Zheng, Y. Duan, *et al.*, "Driv3r: Learning dense 4d reconstruction for autonomous driving," *arXiv preprint arXiv:2412.06777*, 2024.

[124] X. Ren, Y. Lu, H. Liang, *et al.*, "Scube: Instant large-scale scene reconstruction using voxsplats," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, pp. 97 670–97 698, 2024.

[125] Y. Lu, X. Ren, J. Yang, *et al.*, "Infinicube: Unbounded and controllable dynamic 3d driving scene generation with world-guided video models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025, pp. 27 272–27 283.

[126] S. Tang, W. Ye, P. Ye, *et al.*, "Hisplat: Hierarchical 3d gaussian splatting for generalizable sparse-view reconstruction," in *International Conference on Learning Representations (ICLR)*, 2025, pp. 48 436–48 449.

[127] Y. Wang, T. Huang, H. Chen, and G. H. Lee, "Freesplat: Generalizable 3d gaussian splatting towards free view synthesis of indoor scenes," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, pp. 107 326–107 349, 2024.

[128] W. Ljungbergh, B. Taveira, W. Zheng, *et al.*, "R3d2: Realistic 3d asset insertion via diffusion for autonomous driving simulation," *arXiv preprint arXiv:2506.07826*, 2025.

[129] Z. Yang, J. Wang, H. Zhang, S. Manivasagam, Y. Chen, and R. Urtasun, "Genassets: Generating in-the-wild 3d assets in latent space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 22 392–22 403.

[130] S. C. Lambertenghi and A. Stocco, "Assessing quality metrics for neural reality gap input mitigation in autonomous driving testing," in *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, IEEE, 2024, pp. 173–184.

[131] T. Klinghoffer, J. Philion, W. Chen, *et al.*, "Towards viewpoint robustness in bird's eye view segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 8515–8524.

[132] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[133] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8798–8807.

[134] K. Cheng, X. Long, W. Yin, *et al.*, "Uc-nerf: Neural radiance field for under-calibrated multi-view cameras in autonomous driving," *arXiv preprint arXiv:2311.16945*, 2023.

[135] Y. Chen, J. Zhang, Z. Xie, *et al.*, "S-nerf++: Autonomous driving simulation via neural reconstruction and generation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[136] J. Z. Wu, Y. Zhang, H. Turki, *et al.*, "Difix3d+: Improving 3d reconstructions with single-step diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025, pp. 26 024–26 035.

[137] R. Wu, B. Mildenhall, P. Henzler, *et al.*, "Reconfusion: 3d reconstruction with diffusion priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 21 551–21 561.

[138] R. Gao, A. Holynski, P. Henzler, *et al.*, "CAT3d: Create anything in 3d with multi-view diffusion models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.