ABSTRACT

Title of Thesis: Illumination-Aware
3D Gaussian Splatting
for Spacecraft Reconstruction
and Scene Composition

Sean McCurry
Master of Science, 2026

Thesis Directed by: Professor John R. Martin
Department of Aerospace Engineering

Radiance fields have recently emerged as a powerful tool for learning continuous, three-dimensional representations of complex scenes from collections of discrete image observations. Neural Radiance Fields and 3D Gaussian Splatting have demonstrated high-quality novel view synthesis across a wide range of terrestrial applications; however, their use in spacecraft modeling introduces unique challenges that cannot be fully represented with standard static-scene formulations. Spacecraft are often observed under variable illumination conditions, with large changes in sun angle and stark shadows. These conditions make it difficult for conventional radiance field models to separate persistent spacecraft structure from lighting-dependent appearance, limiting traditional implementations' usefulness for creating accurate reconstructions for spacecraft inspection, mapping, and proximity operations.

This thesis investigates radiance field modeling for spacecraft scenes, with a particular focus on 3D Gaussian Splatting for dynamic relighting, shadowing, and multi-object scene composition.

Building on the modular radiance field training and evaluation framework *Nerfstudio*, this work introduces an interactive pipeline for rendering multiple independently trained radiance field models within a shared scene, develops sun-angle-conditioned relighting and shadowing extensions for spacecraft modeling, and explores training strategies intended to improve reconstruction quality and learning stability in spacecraft-imaging domains. The proposed methods are evaluated across simulated, synthetic, and real-world datasets, demonstrating their promise and practical limitations for spacecraft scene reconstruction over a spectrum of use cases. Together, these contributions move radiance fields toward more flexible, illumination-aware representations for spacecraft modeling, enabling realistic visualization and analysis under the dynamic lighting, occlusion, and relative positioning conditions of space operations.

Illumination-Aware 3D Gaussian Splatting for Spacecraft Reconstruction and Scene Composition

by

Sean McCurry

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2026

Advisory Committee:
Professor John R. Martin, Chair/Advisor
Professor Tam Nguyen
Professor Michael Otte

Acknowledgements

For the completion of this thesis, I would first like to thank my family for their unfaltering support over the course of my academic career. My dad, whose own passion for space was the kindling for the path I've chosen as an aerospace engineer; my mom, whose understanding and confidence in me carries me forward still; and my brother, AJ, whose witty remarks have me catching stares as I chuckle to myself in public days later, and whom I love more than he knows. You all are my home base, and I can never thank you enough for giving me the foundation from which I have been able to build myself from.

Thank you to my friends in both New York and Maryland, whose persistent companionship and ever-surprising introduction of some absurdity or another into my daily life has brightened more days than I can count. For my NY friends, you guys have been with me through some of the most formative years of my life, and I truly would not be the person I am today without every single one of you. and I can't thank you all more for the wealth of memories you've given me. For my MD friends, you all have supported me since we started together in freshman year, and I could not have asked for a more endearing and down-to-earth group of people to share my college experience with.

I would like to thank Dr. Martin, for believing in my capability to grow as a researcher, and for always providing the assistance to do so. I would also like to thank the members of the Machine Learning for Dynamical Systems Lab, who made my time in graduate school more enjoyable than I could have hoped for. I will truly miss our group meetings.

Lastly, I would like to acknowledge the Johns Hopkins University Applied Physics Laboratory and the SMURFs IRAD team for funding this research. I am incredibly thankful for the opportunity to work on such an interesting project, and appreciate the support immensely.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

LEO	Low Earth Orbit
SSA	Space Situational Awareness
SfM	Structure-from-Motion
MVS	Multi-View Stereo
3DGS	3D Gaussian Splatting
NeRF	Neural Radiance Field
MLP	Multi-Layer Perceptron
GRAF	Generative Radiance Field
BRDF	Bidirectional Reflectance Distribution Function
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity Index
LPIPS	Learned Perceptual Image Patch Similarity
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
BER	Binary Error Rate
IoU	Intersection over Union
EMA	Exponential Moving Average

Chapter 1: Introduction

1.1 Motivation and Objectives

With an uptick in interest in the cislunar region of space and operations in low Earth orbit (LEO) for commercial, civil, and defense applications, safety when navigating these regions is of the utmost importance. The rapid growth in satellite constellations, ride-share launches, and space-based infrastructure has led to an increasingly congested orbital environment. As the population of resident space objects continues to grow, collision threats are increasing in both frequency and complexity, necessitating onboard optical navigation pipelines that are not only accurate, but also fast enough to handle time-sensitive decision-making.

Beyond collision avoidance alone, the ability to recover accurate three-dimensional models of objects in space underpins a broad class of emerging spaceflight capabilities. These include small-body mapping and characterization, optical navigation for rendezvous and proximity operations, high-fidelity synthetic imagery generation for autonomy testing and mission rehearsal, space situational awareness (SSA) for cataloging and monitoring resident space objects, and debris characterization and removal for long-term orbital sustainability [4, 5].

Current approaches for resolving three-dimensional models of objects in space from collections of images rely primarily on classical geometric reconstruction pipelines, including Structure-from-Motion (SfM), multi-view stereo (MVS), and mesh or voxel-based modeling techniques. While these methods have demonstrated success in terrestrial and laboratory settings, they often rely on consistent lighting and strong geometric textures to reliably establish correspondences across viewing directions. However, in space, objects are frequently observed under harsh lighting conditions stemming from extreme specular reflections and deep shadowing that can severely degrade feature matching and

reconstruction fidelity.

Figure 1.1: ADRAS-J RPO mission. Image retrieved from Astroscale's ADRAS-J mission page [6]. © Astroscale.

Learned radiance field methods, and in particular 3D Gaussian Splatting (3DGS), present a promising alternative for generating high-fidelity, continuous 3D scene representations directly from 2D image data. By optimizing a compact set of explicit 3D Gaussian primitives from a collection of images, these models can efficiently render the desired scene from novel viewpoints, bypassing explicit correspondence matching and expensive per-ray evaluations while retaining high visual quality. Importantly, radiance field methods can be modified to accommodate environment-dependent appearance effects, making them well-suited to the challenging illumination regimes encountered in space.

As such, the utility of 3DGS methods can be extended to space-based image collections

in which the objects to be learned—such as satellites, debris fragments, or other spacecraft—are observed sparsely or under rapidly changing lighting conditions. Beyond direct use in optical navigation pipelines, these representations offer a mechanism for generating photorealistic synthetic imagery for pre-flight evaluation, enabling stress-testing of perception and guidance algorithms across lighting, geometry, and viewpoint variations that may be difficult or unsafe to reproduce in-situ.

However, while prior work has focused primarily on relighting in terrestrial scenes, the capacity of radiance-field representations to model space-specific inter- and intra-model lighting interactions remains largely unexplored. This limitation is particularly relevant for space applications, in which deep shadows and harsh illumination play a central role in proximity operations, inspection, and debris removal. This thesis works to demonstrate the feasibility of composing multiple space-image-based radiance field models within a shared three-dimensional scene, and seeks to establish a principled framework for reproducing inter-spacecraft lighting and shadowing effects. By developing a simple, efficient, and interactive pipeline for 3DGS-based scene composition in space environments, this research aims to enable more realistic scene understanding and visualization for downstream tasks such as relative navigation, proximity operations, and situational awareness in orbital regimes.

1.2 Background

1.2.1 Neural Radiance Fields

Neural Radiance Fields (NeRFs) [1] presented a fundamental shift in learned 3D scene representation by modeling a continuous scene as a 5D vector-valued function that directly maps 3D spatial positions (x, y, z) and 2D viewing directions (θ, ϕ) to emitted color values $\mathbf{c} = (r, g, b)$ and volumetric density (σ). In the original formulation, a multi-layer perceptron (MLP) takes in

the viewing direction and spatial location as inputs and is trained to predict the density and view-dependent radiance at that point. Novel views of a scene are synthesized by sampling many points along a camera ray, querying the network for each sampled point, and integrating the predicted color and density values using differentiable volumetric rendering techniques to produce a final image.

With this architecture, NeRFs learn continuous, differentiable volumetric representations of scenes directly from posed images alone, without requiring explicit geometric supervision or pre-computed surface proxies. The original NeRF formulation demonstrated that implicit, learned representations can achieve photorealistic novel-view synthesis, accurately modeling occlusions and view-dependent appearance effects.

1.2.2 3D Gaussian Splatting

As neural radiance field methods matured, researchers began exploring alternative scene representations that could retain high visual fidelity while addressing the practical limitations of NeRF-style implicit models for real-time rendering. A notable advance in this realm, and the primary method used throughout this thesis, is 3D Gaussian Splatting [2], which replaces the implicit volumetric function of NeRFs with a set of explicit 3D Gaussian primitives that are optimized during training to collectively approximate the scene's radiance field.

In the 3DGS paradigm, the scene is modeled as a cloud of anisotropic 3D Gaussians, initialized with either a specific or randomized point cloud input, which possess explicit position, covariance, orientation, color, and opacity parameters. During training, these parameters are jointly optimized to minimize a photometric reconstruction loss over the input images. Parallel to this process exists a density control strategy that duplicates, splits, or culls Gaussians based on how well they represent

certain spatial regions of the scene. Finally, 3DGS renders scenes using a visibility-aware, GPU-accelerated, differentiable rasterization pipeline that projects each anisotropic 3D Gaussian into a screen-space elliptical footprint, compositing overlapping primitives using front-to-back alpha blending. By exploiting the explicit nature of these volumetric primitives, this splatting formulation avoids dense ray marching and repeated neural network evaluations, enabling orders-of-magnitude acceleration in both rendering and training compared to NeRF-style pipelines.

1.2.3 Appearance Modulation, Relighting, and Shadowing Techniques

As the focus of this thesis is on neural radiance field and 3D Gaussian Splatting techniques, the methods of performing relighting and appearance modulation will also remain largely within these two paradigms.

Early neural view synthesis work largely targeted *novel viewpoint rendering*, with lighting conditions that were static or "baked into" the collection of images. In this nascent NeRF framework, the goal was only to reproduce what the cameras saw, rather than intentionally separate geometry and appearance into different components.

As NeRF-style models matured, efforts to push beyond pure view synthesis and towards appearance modulation emerged. A major practical step in this direction was handling unconstrained capture: real photo collections contain exposure shifts, time-of-day changes, weather, and transient occluders. Methods such as NeRF-in-the-Wild [7] introduced explicit mechanisms (e.g., latent embeddings) to account for per-image appearance variation, allowing a single underlying scene representation to remain stable even when the input images are not. The result of this work was a decoupling of object geometry and appearance, enabling control over a broader range of lighting

effects that contribute to the visual appearance of the modeled geometry and leading to more reliable and realistic scene representations.

Simultaneously, branches of research focused on directly decomposing scene representations by using physically-grounded lighting terms. Neural Reflectance Fields, for example, explicitly represent not just density and color, but also surface-oriented properties such as normals and reflectance terms, and combine these with differentiable rendering so that illumination can be changed after training [8]. Related efforts like Ref-NeRF improve the handling of glossy, view-dependent effects by structuring the appearance model around reflected radiance rather than purely emitted color, making the learned representation more compatible with reflective materials [9]. Neural Incident Light Fields (NeILF) continues this trajectory by treating illumination itself as a learnable field and jointly optimizing lighting and material to support relighting in complex scenes [10]. Together, these works marked a shift from direct reconstruction to more complex recombination under various lighting conditions.

More recently, the growth in popularity of 3D Gaussian Splatting has reshaped approaches to relighting and appearance modeling, with occasional similarities to NeRF-style methods as well. While splatting-based representations trade away parts of the implicit nature of MLP-based NeRF models, they provide a foundation for physically grounded lighting and appearance modifications due to the nature of their explicitly learned primitives. This motivated a cohort of methods that augment Gaussians with additional appearance parameters, such as learned reflectance or lighting functions, normals, or material properties, such that modifying illumination during inference is possible without significant performance drops. Examples include approaches that decompose a scene's appearance into these physically inspired components and use visibility reasoning schemes to produce plausible shadowing and relighting effects [11–13]. Recent 3DGS work has also begun adapting shadowing methods to more specialized settings. Aira et al. introduce an Earth-observation Gaussian Splatting

framework that uses a shadow-mapping formulation based on comparing satellite-view elevation with the corresponding sun-camera elevation, allowing geometrically consistent cast shadows to be rendered efficiently for satellite imagery [14]. Mir et al. introduce Deep Gaussian Shadow Maps, which precompute Gaussian light transmittance over concentric shells and store the result in octahedral atlases for efficient shadow queries during rendering [15]. GS^3 further emphasizes efficiency by introducing a representation and "triple splatting" process geared toward fast novel lighting-and-view synthesis, taking advantage of splatting techniques to produce relighting and shadowing effects while preserving the rapid rendering and interactive nature of Gaussian Splatting pipelines [16]. GS^3's "shadow splatting" method of generating plausible shadowing effects through re-use of the 3DGS rasterizer is particularly relevant, as it is one of the shadowing formulations adapted for this thesis.

Alternatively, approaches more akin to those developed for NeRFs began applying *appearance conditioning* techniques to 3D Gaussian Splatting. As 3DGS methods moved towards more realistic, "in-the-wild" settings, these analogous conditioning mechanisms emerged to model appearance changes in unconstrained image sets and transient effects in the scene using learned per-image and per-Gaussian latent embeddings [17–19]. Conceptually, this line of research involves models that separate geometry and appearance in the same way that approaches like NeRF-W [7] do, enabling rapid relighting and inference-time appearance modulation when rendering.

The arc of these methods inspires the approaches of this thesis, particularly for producing shadowing effects using 3D Gaussian Splatting models, as well as providing background context against which to compare and contrast the contributions of this research.

1.2.4 Scene Composition and Multi-Model Representation

While the first iterations of neural scene representation gained traction for novel view synthesis of a single captured scene, these earlier instantiations implicitly assume that the world is static, making it difficult to support capabilities such as rearranging objects, inserting new objects, or composing scenes from independently learned models (and having them interact faithfully).

One of the first lines of work addressing this limitation sought to make neural rendering more *object aware* and editable. Object-Centric Neural Scene Rendering introduces neural representations for individual objects so that scenes can be rendered as compositions of objects rather than as a single field, enabling arbitrary rearrangement without retraining [20]. Similarly, Learning Object-Compositional Neural Radiance Fields proposes mechanisms to explicitly model object identity inside a neural scene representation, enabling higher-level edits such as moving or adding elements while still producing plausible visibility and shading effects [21]. Moreover, the introduction of Neural Depth Fields (NeDFs) shows how separately trained neural objects can be composed together with dynamic, ray-based shadowing interactions between them, further bringing the idea of composition forward as a priority in neural rendering pipelines [22]. Instead of treating the scene being modeled as a monolith, these works intentionally center the rendering process around a composition of interacting components, bringing the use of these models closer to traditional graphics workflows.

A second major branch addresses scaling and efficiency, where "multi-model rendering" comes through as a practical requirement instead of an editing feature. Large-scale scene reconstructions—spanning building, city blocks, or large swaths of land—often exceed the capacity of a single model trained on one GPU. This motivated decomposition strategies that partition the scene, training multiple models on chunks or aspects of the world that can be rendered together [23, 24]. In these paradigms,

composition introduces its own challenges: that adjacent sub-models must appear coherent across boundaries, and the system needs to remain robust to real-world variation in captures. Block-NeRF emphasizes that this includes not only geometric continuity but also appearance alignment across different captures and time periods, reflecting the reality that large scenes are typically collected over long durations under changing conditions [24]. Complementary work on the efficiency side, such as KiloNeRF, addresses the inference speed bottleneck by replacing a single global network with many smaller local ones, trading "one big model" for "many specialized models" in order to achieve practical rendering performance [25]. More recently, NeRF-XL further formalizes scalability as a primary goal by distributing both training *and* rendering across multiple GPUs in an efficient manner, enabling substantially greater capacities for faster and more accurate scene learning and rendering [26]. These works reinforce the trend of scene decomposition and distributed processing methods to enhance the flexibility and rendering capacities of neural scene representations.

The emergence of 3D Gaussian Splatting reframes composition yet again. The explicit nature of the Gaussian primitives makes rigid transforms and duplication more natural, but also highlights the challenges, particularly with lighting/shadowing and object grouping, that complicate the scene composition problem. Recent Gaussian-focused work begins to directly address these gaps. Gaussian Grouping introduces mechanisms for grouping Gaussians into object-level components, enabling segmentation-driven editing operations such as removal, inpainting, and recomposition, taking steps towards a true Gaussian-based scene editor [11]. Building on this direction, ComGS targets the specific problem of object-scene composition in Gaussian-based rendering, emphasizing the difficulty of producing visually coherent lighting and shadows. The authors introduce Surface Octahedral Probes (SOPs) to store lighting and occlusion information, achieving high-quality, real-time rendering with vivid shadowing [27]. Together, these works suggest that Gaussian splatting is evolving from "fast

reconstruction and view synthesis" toward a broader interactive scene representation that supports grouping, editing, and composition, converging on many of the same user-facing goals that motivated object-compositional NeRF systems.

Viewed together, the literature traces a consistent arc as it shifts from static neural representations to dynamic and scalable reconstructions, moving further towards explicit methods that support multi-model and multi-object composition. This context motivates the approaches explored in this thesis, where composition and interaction between multiple, independently trained models are key focuses.

1.2.5 Radiance Fields for Space Environments

While neural radiance fields and Gaussian splatting methods were developed with terrestrial datasets in mind, a growing body of work has begun adapting these representations to the unique constraints of space environments. Unlike traditional scenes, orbital and deep-space imagery often introduces extreme illumination cases, high-contrast specular reflections, limited viewpoints, and deep shadowing. These conditions require additional care, and in some cases, modifications to existing frameworks, to adapt them for use in space.

One of the first investigations into neural scene representations for spacecraft appeared in Vision-Based Neural Scene Representations for Spacecraft [28], which examines the performance of NeRF and Generative Radiance Field (GRAF) models for representing spacecraft. With the goal of examining 3D model reconstruction from both methods on densely and sparsely-sampled scenes, they find that NeRF models benefit from posed imagery and better match lighting/view-dependent effects, while GRAFs can operate without pose supervision, which is attractive when metadata is scarce or unreliable. Similarly, Spacecraft-NeRF [29] is later introduced by Liu et al. as a high-

fidelity spacecraft reconstruction method. Trained on a synthetic dataset, Spacecraft-NeRF improves foreground model detail by masking complex background content and improves reconstruction quality via a hybrid encoding strategy, outperforming several baseline methods.

An adjacent line of work uses radiance fields to remove the dependency on known target models and obtain prefilter state-observation measurements of objects for state estimation pipelines. Spacecraft State Estimation Using Neural Radiance Fields [30] employs NeRFs to construct an implicit map of the operational domain, reducing onboard storage and memory requirements, and eliminating the need to perform up/downlink with a ground station to infer a target's state information. Similarly, Legrand et al. address the issue of conventional pose estimators' need for a CAD model to generate synthetic training data, a requirement that inhibits operating on debris or unknown space objects. From a set of sparse imagery, they train NeRFs with learnable appearance embeddings to generate a larger dataset with greater viewpoint and illumination diversity [31]. This dataset is used to train an off-the-shelf spacecraft pose estimation network, and they find competitive performance when compared with CAD-model-trained networks.

Following the emergence of 3D Gaussian splatting [2], a thread of research employing 3DGS models for space-based applications has flourished. Nguyen et al. present an approach for training and rendering 3D Gaussian splatting models that can run onboard current spaceflight hardware, outperforming existing NeRF methods and paving the way for greater interest in these models' use in space contexts [32]. Subsequent work highlights the efficacy of introducing environmental knowledge and training modifications into the learning process of space-based 3D Gaussian splatting models. Examples include introduction of appearance embedding models and lighting mask filtering mechanisms to more accurately reconstruct non-cooperative space targets (NCTs) [33] and adaptive Gaussian pruning strategies and geometric regularization techniques to enhance surface detail of

asteroid models [34]. A work of particular relevance to this thesis is conducted by Park et al., where incorporation of Sun position priors into the photometric optimization pipeline and use of GS^3's "shadow splatting" is a key focus. They utilize a positionally-encoded sun direction prior to improve training performance of unknown spacecraft structures, and introduce a Gaussian isotropy-emphasizing loss term to improve shadow sharpness in space environments [35]. Their implementation is used in Chapter 4 of this thesis as an evaluation reference for spacecraft shadowing performance.

Looking past pure model reconstruction cases, radiance field methods have also shown promise for contributing to autonomous small body mapping and exploration cases. Wei et al. introduce a NeRF variant built into the decision loop of a reinforcement learning agent for instant 3D mapping and quantitative uncertainty estimation of the learned map. Using this uncertainty estimate, they optimize a reinforcement learning policy to predict the next best state to minimize uncertainty in surface mapping. In simulation across multiple real small body models, their framework outperforms circular-orbit and randomized baselines in both reconstruction completeness and precision, and ablation studies demonstrate that removing the uncertainty-driven reward collapses performance. The work establishes a concrete and efficient framework for coupling radiance fields and reinforcement learning, opening the door to a new line of cooperative-model mapping research [36].

Collectively, these works underscore the utility of radiance field methods for space environments; by tuning the training procedure and taking advantage of environmental features, both NeRF and 3DGS methods are demonstrated to be promising alternatives to classical 3D representation schemes. This trajectory of research strongly motivates further exploration of optimization and culling strategy modification, Sun-direction encoding, and exploitation of environment knowledge — primary aspects of the work accomplished in this thesis.

1.2.6 Physics-based Rendering Software

Physics-based rendering (PBR) refers to a class of rendering techniques that revolve around accurate simulation of light transport using physically grounded models of radiative transfer. These systems either approximate or solve the rendering equation and utilize physically meaningful bidirectional reflectance distribution functions (BRDFs), affording consistent modeling of illumination, material appearance, and view-dependent effects. Prior to the emergence of neural radiance field and Gaussian splatting methods, physics-based renderers were the dominant paradigm for producing high-fidelity images in both research and production environments. Now, many rendering frameworks have been extended to accommodate use of radiance field models, allowing users to import and render their learned scene representation into the rendering environment.

The *Physically Based Rendering Toolkit* (PBRT) [37] is widely regarded as the canonical academic reference implementation of physically based light transport simulation. Developed alongside the textbook *Physically Based Rendering: From Theory to Implementation* [38], PBRT provides a rigorous implementation of Monte Carlo path tracing, bidirectional scattering distribution functions (BSDFs), spectral rendering, volumetric light transport, and importance sampling strategies. Primarily used in research and education, PBRT serves as a reference platform for evaluating new techniques, material models, and volumetric transport algorithms.

Mitsuba 3 is a modular, reconfigurable rendering system designed for extensibility and experimental light transport research [39]. The framework supports a wide range of rendering algorithms, including Monte Carlo path tracing, bidirectional path tracing, and advanced light transport techniques. A distinguishing feature of Mitsuba 3 is its integration with differentiable rendering pipelines, enabling gradient-based optimization of scene parameters such as geometry,

material properties, and illumination. Unlike production renderers that prioritize throughput and robustness for content creation, Mitsuba 3 is explicitly engineered for algorithmic experimentation. Its architecture allows interchangeable integrators, sampling strategies, and material models, making it well-suited for research in inverse rendering, material estimation, and computational imaging. The renderer supports spectral rendering, polarization effects, and physically grounded bidirectional scattering distribution functions (BSDFs), thereby enabling high-fidelity simulation of complex optical phenomena.

Blender's Cycles engine [40] is an open-source path-tracing renderer integrated into the Blender software suite. Cycles implements physically based shading models and global illumination via Monte Carlo integration, supporting both CPU and GPU execution. Its node-based material system incorporates energy-conserving BRDFs and physically meaningful parameters such as albedo and roughness. In contrast to Cycles' path-tracing approach, Blender's EEVEE [41] engine implements real-time rasterization augmented with physically based shading models. Using techniques like image-based lighting and approximation of global illumination, EEVEE excels at achieving interactive frame rates while maintaining physically plausible behavior.

LuxCoreRender is an open-source physically based rendering engine emphasizing unbiased light transport simulation [42]. It supports path tracing, bidirectional path tracing, and hybrid rendering techniques. It is commonly used in architectural visualization and product rendering applications where photorealistic global illumination is required, and its ease of access makes it a useful tool for independent users.

Vira is a rendering and ray-tracing library developed by the NASA Goddard Space Flight Center for space exploration and mission design purposes [43]. Engineered with scientific visualization and mission-relevant optical simulation in mind, its rendering capabilities support both path-tracing and

rasterization methods, as well as modeling of indirect lighting and other effects under physically meaningful environmental conditions.

These systems share several defining characteristics: explicit modeling of light transport, physically meaningful material parameterization, global illumination effects, and an overarching emphasis on physical correctness. While their individual use cases may differ slightly, they represent the primary frameworks in which photorealistic rendering is achieved outside of radiance field implementations.

1.2.7 Radiance Field Training Frameworks

As radiance field methods have matured from single-paper reference implementations into reusable research and production tools, several training ecosystems emerged to package data ingestion, optimization and training, rendering functionality, evaluation, and interactive visualization into cohesive software stacks.

NVIDIA's Instant-NGP [44] presents a systems-driven approach to radiance field training that addresses key bottlenecks in encoding and rendering. In addition to their model training pipeline, Instant-NGP's key contribution is a multi-resolution hash-grid encoding that replaces large MLP capacities with compact, learnable feature lattices, enabling near-instant optimization of neural graphics primitives, including NeRFs, on a single GPU.

MultiNeRF is a consolidated reference training codebase for three specific CVPR 2022 papers: Mip-NeRF 360 [45], Ref-NeRF [9], and RawNeRF [46]. While centered around these specific methods, MultiNeRF enables training, evaluation, and rendering of the aforementioned methods via a unified pipeline. MultiNeRF is less a "jack-of-all-trades" framework and more so a research-

reproduction tool for examining a small family of NeRF variants, which can be particularly useful for benchmarking against precise implementations.

JaxNeRF [47] refers to a JAX-based NeRF training codebase used primarily as an early performance and scalability baseline. Benefiting from JAX's XLA compilation and multi-device execution model, it serves as a solid foundation for work targeting faster training or alternative sampling strategies due to its relative efficiency when compared to early TensorFlow or PyTorch implementations.

Nerfstudio [48] is explicitly designed as a modular PyTorch framework to streamline end-to-end NeRF development. It handles dataset ingestion and preprocessing, plug-and-play methods, training and evaluation scripts, and integrated visualization and export tools. Of the frameworks mentioned, Nerfstudio is the most completely integrated pipeline, with an architectural emphasis on ease-of-use and rapid experimentation. As such, it is the framework of choice for the work presented in this thesis.

Lastly, NerfAcc [49] is a NeRF acceleration toolbox that targets efficient sampling and evaluation in the volumetric rendering pipeline. Designed to be compatible with a wide swath of existing codebases, NerfAcc provides speedups for both training and inference through a Python-facing API implementation. Building on ideas popularized by Instant-NGP [44], NerfAcc extends these concepts to broader regimes, positioning it as commonly-used infrastructure in higher-level radiance field training stacks.

1.3 Research Gap

Despite rapid progress in neural radiance fields and 3D Gaussian Splatting, much of the existing work remains centered on single-scene reconstruction, terrestrial relighting, or controlled object-

level editing. Emergent space-focused radiance field work has demonstrated the promise of these representations for spacecraft reconstruction, pose estimation, and onboard rendering. However, there remains a gap between high-quality reconstruction of an isolated object and the broader needs of space operations, where multiple independently modeled objects may need to be rendered together, illuminated by a shared Sun direction, and analyzed under harsh shadowing. In particular, inter-object lighting and shadow interaction, space-specific shadow behavior, and training strategies that account for the geometric and photometric difficulties of spacecraft imagery remain underexplored.

This thesis addresses this gap by extending 3D Gaussian Splatting from a fast single-object reconstruction method toward a more interactive and space-aware scene representation. The work develops tools for composing and rendering multiple independently trained radiance field models in a shared scene, introduces Sun-conditioned appearance modulation and shadowing methods for inter- and intra-model lighting interaction, and explores informed training strategies that encourage Gaussians to remain in geometrically meaningful regions. These methods are then evaluated across simulation, synthetic benchmark, and real-world orbital imagery to identify where the pipeline is robust, where it depends on favorable metadata, and what failure modes remain. In doing so, this thesis makes progress toward radiance field representations that are not only visually accurate, but also more useful for spacecraft inspection, proximity operations, and space situational awareness.

1.4 Contributions

The primary contributions of this thesis are as follows:

1. **Multi-Model Rendering and Interactive Viewing Pipeline:** An extension of *Nerfstudio*'s rendering capabilities is developed to accommodate analysis and viewing of multiple, separately

trained radiance field models. Algorithms for transforming and rendering either multiple NeRF or 3DGS models in a shared world scene are introduced and paired with *Nerfstudio*'s interactive viewer, *Viser*, to allow for live rendering and scene composition.

2. **Sun-Conditioned Dynamic Relighting and Shadowing for 3D Gaussian Splatting Models:** Sun illumination-conditioned dynamic relighting of spacecraft is achieved via learned per-Gaussian embeddings and a neural appearance field, allowing models to modulate their appearance based on where the sun is positioned in the scene. Furthermore, complex shadowing effects for 3DGS-modeled spacecraft are achieved using shadow mapping and shadow splatting. A per-Gaussian depth bias is developed to reduce the effect of superficial self-shadowing artifacts under harsh lighting, and an adaptive light source placement algorithm is constructed to pair with the interactive rendering pipeline for multi-model scene consistency.

3. **Informed Training for Space-based 3D Gaussian Splatting Models:** Two informed culling and densification strategies for 3D Gaussian Splatting are created, Mesh-Aware and Support-Aware, with the goal of improving spacecraft reconstruction, and localizing Gaussians to meaningful regions of the scene. The mesh-aware strategy takes advantage of having a ground-truth or approximate mesh available to provide a volumetric guide for where Gaussians are allowed to exist, and the support-aware strategy associates a "usefulness" score per-Gaussian to veto culling decisions for contributing scene components. Strategy performance is evaluated and recommendations for future development are offered. Additionally, an early-training warm-start procedure is developed to draw Gaussians towards foreground-likely locations to improve results when using random box initializations instead of a point cloud.

4. **Case Studies and Operation on Real-World Data:** The aforementioned 3D Gaussian

Splatting methods are deployed on a spectrum of datasets increasing in difficulty: simulation data, synthetic laboratory data, and real-world space imagery. Performance on each dataset is analyzed, failure modes are identified, and training recommendations and considerations are provided for each training regime.

1.5 Outline of Thesis

With the foundation for understanding the contributions of this work formalized, the structure of this thesis is as follows: Chapter 2 introduces the two overarching classes of radiance fields discussed in this thesis and lists the evaluation metrics used in the subsequent chapters. Chapter 3 details the process for modifying *Nerfstudio*'s rendering architecture to have the capability for rendering multiple, separately trained radiance field models in the same world scene. Chapter 4 discusses the primary methods of creating inter- and intra-model shadows and dynamic relighting. Chapter 5 presents the mesh-aware and support-aware training strategies, as well as the foreground attraction warm-start. Chapter 6 compares performance of 3D Gaussian splatting models on datasets with different levels of information availability, including real-world space imagery. Finally, Chapter 7 offers a summary of research contributions, and discusses avenues for future work.

Chapter 2: Methods

2.1 Neural Radiance Fields

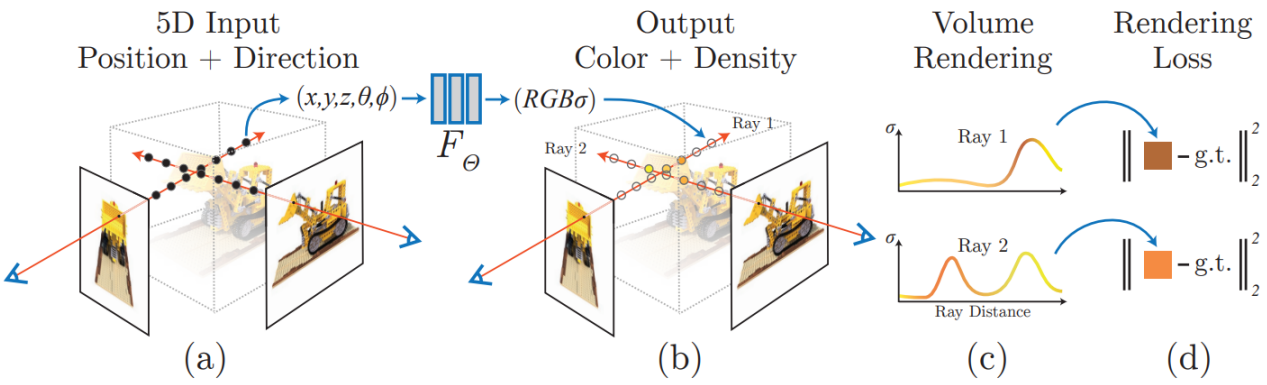


Figure 2.1: Training process of a Neural Radiance Field model. Reproduced from Mildenhall et al. [1].

As mentioned in Section 1.2.1, Neural Radiance Fields [1] represent a three-dimensional scene as a continuous, vector-valued, 5D function that predicts emitted radiance and density from a 3D position and 2D viewing direction — enabling high-fidelity, novel view synthesis by modeling the volumetric light transport of a scene using a neural network. This function is parameterized by a neural network F_θ, which maps the position and viewing direction to a color and volume density:

$$F_\theta(\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma) \tag{2.1}$$

where $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ is a spatial coordinate, $\mathbf{d} = (\theta, \phi) \in \mathbb{R}^2$ is the viewing direction, $\mathbf{c} \in \mathbb{R}^3$ is the emitted color, and $\sigma \in \mathbb{R}$ is the volume density.

The volume density $\sigma(\mathbf{x})$ is interpreted to be the differential probability of a given camera ray terminating at the spatial location $\mathbf{x}$, and the final pixel color is then computed as a weighted sum of the predicted colors along the ray.

Pixel colors are computed by integrating the radiance along camera rays using the volume

rendering equation. Given a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with near and far bounds t_n and t_f, where $\mathbf{o}$ is the origin of the camera and $\mathbf{d}$ is the ray-direction unit vector, the rendered color is:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t), \mathbf{d})dt \tag{2.2}$$

where $T(t)$ denotes the accumulated transmittance along the ray from t_n to t, and is given by:

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds.\right) \tag{2.3}$$

In practice, this integral is approximated using discrete samples along the ray:

$$C(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i \tag{2.4}$$

where

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i) \quad \text{and} \quad T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \tag{2.5}$$

with $\delta_i = t_{i+1} - t_i$ representing the distance between adjacent samples.

With this formulation, a NeRF is trained by minimizing the difference between rendered pixel colors and ground truth images captured from known camera viewpoints, with a training loss defined as the squared error between the rendered and true pixel colors:

$$L = \sum_{\mathbf{r}\in R} \|C(\mathbf{r}) - C_{gt}(\mathbf{r})\|^2 \tag{2.6}$$

2.2 3D Gaussian Splatting

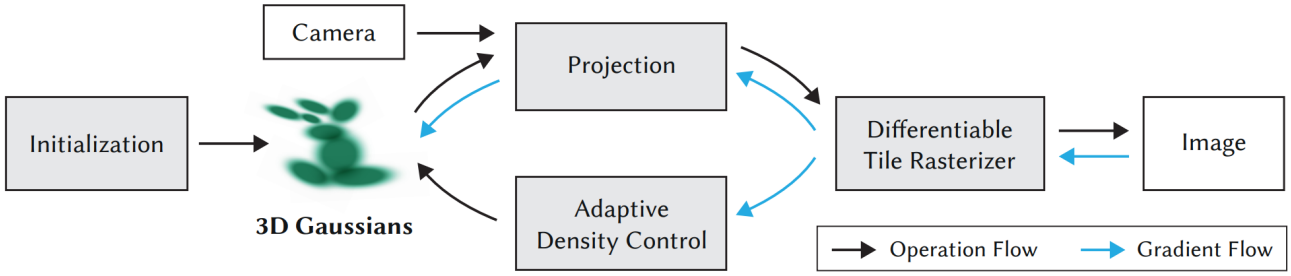


Figure 2.2: Training process of a 3D Gaussian Splatting model. Reproduced from Kerbl et al. [2].

As introduced in Section 1.2.2, 3D Gaussian splatting [2] is an alternative form of scene representation to NeRFs in which a cloud of anisotropic 3D Gaussians with parameters that are optimized over the course of training to volumetrically reproduce the scene. Each 3D Gaussian is represented as:

$$G(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right) \tag{2.7}$$

where $\mu \in \mathbb{R}^3$ is the Gaussian center (mean), and $\mathbf{\Sigma} \in \mathbb{R}^{3\times3}$ is the 3D covariance matrix defined in the world frame. During training, additional parameters such as opacity and spherical harmonic-represented color are optimized as well. Interwoven with the optimization of these explicit parameters is a Gaussian culling and densification strategy that controls the duplication, splitting, and pruning of Gaussians to better represent the scene. To render an image, 3D Gaussians are first projected into screen space before a GPU-accelerated, tile-based rasterization pass is performed, efficiently approximating volumetric light transport via front-to-back alpha composition and producing the final image.

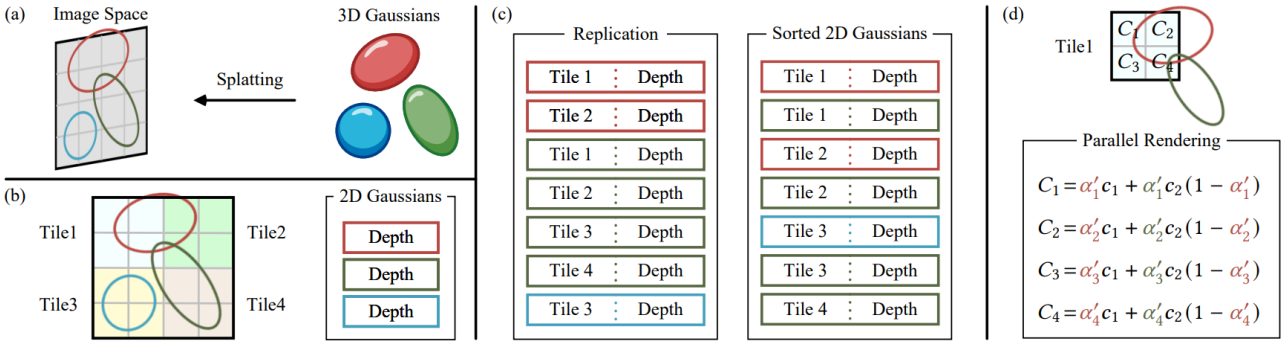


Figure 2.3: Illustration of the rasterization process. Reproduced from Chen and Wang [3].

Over the course of training, the parameters of each Gaussian are optimized using gradient-based learning. The rasterization process, shown in Figure 2.3, is differentiable, allowing gradients of the rendered image with respect to Gaussian primitives to be computed via the rendering pipeline. The training loss combines the mean absolute error between rendered and ground-truth images with a structural dissimilarity (D-SSIM) term:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}, \quad \lambda = 0.2. \tag{2.8}$$

The projection of 3D Gaussians into the 2D screen space requires use of the extrinsic and intrinsic matrices of the rendering camera, which will be denoted by $\mathcal{P}$ and $\mathcal{K}$:

$$\mathcal{P} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}, \qquad \mathcal{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

where $R \in \mathbb{R}^{3\times3}$ is the camera rotation, $t \in \mathbb{R}^3$ is the camera translation, f_x, f_y are the focal lengths, and c_x, c_y are the 2D screen-space coordinates of the image center.

First, the 3D Gaussian in world space is mapped to 3D camera space by:

$$\hat{\boldsymbol{\mu}}_i = R^\top(\boldsymbol{\mu}_i - t), \qquad \hat{\boldsymbol{\Sigma}}_i = R^\top \boldsymbol{\Sigma}_i R. \tag{2.10}$$

Next, the 3D camera-space Gaussian center is projected to 2D screen space with a z-depth (into the image) via:

$$\boldsymbol{\mu}_i' = \begin{bmatrix} \hat{\mu}_x/\hat{\mu}_z \\ \hat{\mu}_y/\hat{\mu}_z \end{bmatrix}. \tag{2.11}$$

Furthermore, since perspective projection is nonlinear, the 3D camera-space covariance $\hat{\boldsymbol{\Sigma}}_i$ cannot be projected exactly using a single linear transform. Using *gsplat*'s (*Nerfstudio*'s rasterization pipeline) [50] convention, this is approximated with a first-order Taylor expansion evaluated at the Gaussian center. The Jacobian associated with this linearization is:

$$J = \frac{1}{\hat{\mu}_z} \begin{bmatrix} 1 & 0 & -\hat{\mu}_x/\hat{\mu}_z \\ 0 & 1 & -\hat{\mu}_y/\hat{\mu}_z \end{bmatrix}. \tag{2.12}$$

Finally, we compute the 2D screen-space covariance matrix as:

$$\boldsymbol{\Sigma}_i^{2D} = J\hat{\boldsymbol{\Sigma}}_i J^\top \in \mathbb{R}^{2\times 2} \tag{2.13}$$

Once the Gaussians have been projected to screen space, the screen is split into 16×16 tiles, noting each of the tiles that a given Gaussian overlaps and assigning it to that tile. Gaussians are then sorted based on their depth into the scene from the camera's perspective. Taking advantage of GPU parallelization, one parallel computing thread is launched for each tile; for a given pixel, color and opacity values are accumulated front-to-back, taking into account the Gaussian's screen-space position

with respect to that pixel. For a pixel $p_{(x,y)}$, let i index the N total Gaussians that contribute to that pixel.

$$\alpha_i = o_i \exp\left(-\frac{1}{2}\left(p_{(x,y)} - \mu_i\right)^\top \left(\mathbf{\Sigma}_i^{2D}\right)^{-1}\left(p_{(x,y)} - \mu_i\right)\right) \tag{2.14}$$

where o_i is the opacity parameter of Gaussian i. Iterating front to back, the transmittance of Gaussian i, T_i, is computed as

$$T_i = \prod_{j=1}^{i-1}(1 - \alpha_j), \tag{2.15}$$

and the final pixel color is given by

$$\hat{C}_{p_{(x,y)}} = \sum_{i \in N} \mathbf{c}_i \alpha_i T_i, \tag{2.16}$$

where $\mathbf{c}_i$ is the color of Gaussian i. When a target saturation of α is met for a given pixel, accumulation stops and processing terminates, resulting in a final image once all pixels have been completed.

In 3DGS, color is represented via spherical harmonic coefficients, which are converted into direct color values when evaluated with a view direction. Each Gaussian i optimizes a coefficient vector $\hat{\mathbf{a}}_i$, which parameterizes a directional function over viewing directions. Given the view direction $\mathbf{d}_i$ from the camera towards the Gaussian, the rendered color is obtained by evaluating the spherical harmonic basis in that direction and weighting the basis functions by the predicted coefficients.

$$\mathbf{c}_i(\mathbf{d}_i) = \sum_{\ell=0}^{L_c} \sum_{m=-\ell}^{\ell} \hat{\mathbf{a}}_{i,\ell m} Y_{\ell m}(\mathbf{d}_i), \tag{2.17}$$

where L_c is the color spherical harmonic degree, $Y_{\ell m}$ is the spherical harmonic basis function of

degree ℓ and order m, and $\hat{\mathbf{a}}_{i,\ell m}$ is the predicted coefficient associated with that basis function. These coefficients are fed through the rasterization engine, and evaluated prior to the alpha composition step.

2.3 Evaluation Metrics

To evaluate the quantitative performance of the models in this thesis, we consider two metric regimes: 2D image-space metrics, which compare ground-truth and predicted images, and 3D mesh or point cloud-based metrics, which compare a ground-truth mesh to a learned model's mesh or point cloud output.

The 2D image-space metrics evaluated are:

- **Peak Signal-to-Noise Ratio (PSNR)** Measures the ratio between the maximum possible signal intensity and the reconstruction error, expressed on a logarithmic scale. It evaluates overall pixel fidelity between the rendered image and ground truth. PSNR is reported in decibels (dB), with values ranging from 0 to ∞ in theory. Higher values indicate better reconstruction quality and lower pixel-wise error.

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{MAX^2}{\mathrm{MSE}} \right) \tag{2.18}$$

  where MAX is the maximum possible pixel value and MSE is the mean squared error.

- **Structural Similarity Index (SSIM)** Evaluates perceptual similarity by comparing local luminance, contrast, and structural information between a predicted image I^{pred} and ground truth image I^{gt}. SSIM is typically reported in the range $[0, 1]$, where 1 indicates perfect structural

similarity and values closer to 0 indicate lower similarity.

$$\mathrm{SSIM}(I^{\text{pred}}, I^{\text{gt}}) = \frac{(2\mu_{\text{pred}}\mu_{\text{gt}} + C_1)(2\sigma_{\text{pred,gt}} + C_2)}{(\mu_{\text{pred}}^2 + \mu_{\text{gt}}^2 + C_1)(\sigma_{\text{pred}}^2 + \sigma_{\text{gt}}^2 + C_2)} \tag{2.19}$$

where μ_{pred} and μ_{gt} denote local means, σ_{pred}^2 and σ_{gt}^2 denote local variances, $\sigma_{\text{pred,gt}}$ denotes the local covariance, and C_1 and C_2 are small positive constants used to stabilize the division.

- **Learned Perceptual Image Patch Similarity (LPIPS)** Measures perceptual similarity using deep feature activations from a pretrained neural network. It captures high-level visual differences beyond pixel-wise comparison. In practice, LPIPS is often reported as a nonnegative value in the range $[0, 1]$, with values closer to 0 indicating greater perceptual similarity.

$$\mathrm{LPIPS}(I^{\text{pred}}, I^{\text{gt}}) = \sum_l w_l \left\| \phi_l(I^{\text{pred}}) - \phi_l(I^{\text{gt}}) \right\|_2^2 \tag{2.20}$$

where $\phi_l(\cdot)$ denotes the feature map extracted from layer l of a pretrained network (AlexNet in this thesis), and w_l are learned per-layer weights.

- **Mean Absolute Error (MAE)** Computes the average absolute difference between predicted and ground-truth pixel values, treating all errors equally. MAE is reported in the same units as the image intensity values, typically in the range $[0, 1]$, where 0 indicates perfect agreement.

$$\mathrm{MAE} = \frac{1}{N} \sum_{i=1}^{N} \left| I_i^{\text{pred}} - I_i^{\text{gt}} \right| \tag{2.21}$$

- **Root Mean Squared Error (RMSE)** Measures the square root of the average squared pixel differences, penalizing larger errors more heavily than MAE. RMSE is also reported in the

same units as the image intensity values, typically in the range $[0, 1]$, with 0 indicating perfect agreement.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(I_i^{\text{pred}} - I_i^{\text{gt}}\right)^2} \tag{2.22}$$

- **Binary Error Rate (BER)** Measures the fraction of incorrectly classified pixels for binary classification tasks such as shadow versus non-shadow segmentation. In this work, predicted and ground-truth shadow masks are compared pixel-wise after thresholding, and pixels are counted as errors when the predicted binary label does not match the ground-truth binary label. BER is reported in the range $[0, 1]$, where 0 indicates perfect classification and 1 indicates that every pixel is classified incorrectly.

$$\mathrm{BER} = \frac{FP + FN}{TP + TN + FP + FN} \tag{2.23}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

- **Shadow Region Signed Bias** Measures the average signed luminance difference over ground-truth shadow pixels, indicating whether predicted shadows are systematically too bright or too dark. When luminance values are normalized to $[0, 1]$, this metric is reported in the range $[-1, 1]$, where 0 indicates no average signed bias. Positive values indicate that predicted shadows are too bright, while negative values indicate that predicted shadows are too dark.

$$\mathrm{Bias}_{\text{shadow}} = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\left(L_i^{\text{pred}} - L_i^{\text{gt}}\right) \tag{2.24}$$

where $\mathcal{S}$ denotes the set of ground-truth shadow pixels.

- **Shadow Boundary Luminance MAE** Computes the mean absolute luminance error within a narrow band around the ground-truth shadow boundary, capturing inaccuracies in edge placement and transition. When luminance values are normalized to $[0, 1]$, this metric is reported in the range $[0, 1]$, where 0 indicates perfect boundary-region luminance agreement.

$$\mathrm{MAE}_{\text{boundary}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left| L_i^{\text{pred}} - L_i^{\text{gt}} \right| \tag{2.25}$$

where $\mathcal{B}$ denotes the set of pixels near the shadow boundary.

- **Boundary Sharpness Ratio** Compares gradient magnitudes at shadow boundaries between prediction and ground truth to quantify whether edges are too diffuse or too sharp. This metric is reported in the range $[0, \infty)$, where a value of 1 indicates matching boundary sharpness. Values below 1 indicate that predicted boundaries are more diffuse than the ground truth, while values above 1 indicate that predicted boundaries are sharper.

$$\text{Sharpness Ratio} = \frac{\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left\| \nabla L_i^{\text{pred}} \right\|}{\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left\| \nabla L_i^{\text{gt}} \right\|} \tag{2.26}$$

For binary error rate, shadow region signed bias, shadow boundary luminance MAE, and boundary sharpness ratio, a luminance value below **0.1** is considered shadowed.

The 3D metrics evaluated are:

- **Chamfer Distance** Measures the average closest-point distance between a predicted point cloud set $\mathcal{P}^{\text{pred}}$ and a ground truth point cloud set $\mathcal{P}^{\text{gt}}$. It captures overall geometric similarity while

being robust to small misalignments. Chamfer Distance is reported in the squared units of the coordinate system when squared Euclidean distances are used, and its range is $[0, \infty)$. Lower values indicate closer geometric agreement between the two point sets, with 0 indicating perfect correspondence.

$$\mathrm{CD}(\mathcal{P}^{\text{pred}}, \mathcal{P}^{\text{gt}}) = \frac{1}{|\mathcal{P}^{\text{pred}}|} \sum_{p \in \mathcal{P}^{\text{pred}}} \min_{q \in \mathcal{P}^{\text{gt}}} \|p - q\|_2^2 + \frac{1}{|\mathcal{P}^{\text{gt}}|} \sum_{q \in \mathcal{P}^{\text{gt}}} \min_{p \in \mathcal{P}^{\text{pred}}} \|q - p\|_2^2 \tag{2.27}$$

- **Hausdorff Distance** Measures the maximum deviation between two point sets by considering the worst-case nearest-neighbor distance. It is sensitive to outliers and captures the largest geometric discrepancy. Hausdorff Distance is reported in the same units as the coordinate system and has a range of $[0, \infty)$. Lower values indicate better alignment, while higher values reveal large local deviations.

$$\mathrm{HD}(\mathcal{P}^{\text{pred}}, \mathcal{P}^{\text{gt}}) = \max \left\{ \sup_{p \in \mathcal{P}^{\text{pred}}} \inf_{q \in \mathcal{P}^{\text{gt}}} \|p - q\|_2, \sup_{q \in \mathcal{P}^{\text{gt}}} \inf_{p \in \mathcal{P}^{\text{pred}}} \|q - p\|_2 \right\} \tag{2.28}$$

- **Intersection over Union (IoU)** Measures the volumetric overlap between predicted geometry $\mathcal{V}^{\text{pred}}$ and ground truth geometry $\mathcal{V}^{\text{gt}}$, computed on a shared world-aligned voxel occupancy grid. IoU is reported in the range $[0, 1]$, where 1 indicates perfect overlap and 0 indicates no overlap.

$$\mathrm{IoU}(\mathcal{V}^{\text{pred}}, \mathcal{V}^{\text{gt}}) = \frac{|\mathcal{V}^{\text{pred}} \cap \mathcal{V}^{\text{gt}}|}{|\mathcal{V}^{\text{pred}} \cup \mathcal{V}^{\text{gt}}|} \tag{2.29}$$

Chapter 3: Multi-Model Rendering and Interactive Viewing Pipeline

Radiance field frameworks, including *Nerfstudio* [48], are primarily designed for rendering a single trained scene representation at a time. While this is often sufficient for terrestrial imagery or isolated reconstruction cases, it does not support compositing of multiple, ***independently trained*** models. While previous lines of work have explored composition of multiple radiance field objects [20–25, 27, 51], explicit combination of independently trained models (either NeRF or 3DGS-based) in the context of spacecraft scenes has yet to be developed. The capability to render multiple models together at a level lower than an image-space overlay affords greater scene control, and allows for exploration of more complicated inter-model effects, detailed in Chapter 4. Furthermore, it provides the user with control over object location, rotation, and scale, leading to greater ease of use in customizing a scene before rendering images. Therefore, rendering procedures for both Neural Radiance Field and 3D Gaussian Splatting models in *Nerfstudio* that enable joint querying and direct model interaction are required.

Ideally, these rendering procedures abide by the following guidelines for rendering accuracy, as well as ease of implementation and downstream use:

1. All models should be queried using a single rendering camera so that they share a common rendered viewpoint.

2. Each independently trained model should be able to be translated, rotated, and scaled via a rigid transform for custom positioning within a scene.

3. Composition must occur *below* the image level to allow for complex lighting interactions between models.

4. Multi-model rendering procedures should operate alongside existing *Nerfstudio* infrastructure

where possible to minimize downstream conflicts.

The following sections detail the separate implementations and rendering procedures for both NeRF and 3D Gaussian Splatting models.

3.1 Rendering Multiple NeRF Models

Rendering multiple separately trained NeRF models in a single scene has unique challenges. Each model has a separate density field, which is queried at different learned locations according to a *proposal sampling network*, a small MLP that learns where the largest density contributions occur along each camera ray to more intelligently place samples. Since each individual model is trained to represent its own scene, they have no way of interpreting the difference between local and world coordinates; however, when compositing multiple models with unique rigid transforms, these two coordinate frames must be decoupled. Furthermore, if two models are in close proximity, or overlapping in any way, their ray samples must be combined and disentangled to allow for proper sample weighting during volumetric rendering.

3.1.1 System Level Rendering Pipeline

For both NeRF and 3DGS-based models, the multi-model rendering pipeline begins with instantiating a *Nerfstudio* `Cameras` object to be shared as the rendering camera for all models in the scene. From this camera, rays are shot into the scene in the form of a `RayBundle`. The `RayBundle` is cloned, transformed into that model's posed scene frame, and passed through its scene box, proposal sampler network, and density field. Per-model sampled intervals, densities, and color channels are collected and merged into a common ray partition before being reweighted and composited into a final

image.

3.1.2 NeRF Model Transformations

Each independently trained NeRF model represents geometry and appearance within its own local coordinate system. That is, during training, it learns a density field defined in a normalized spatial domain corresponding to the scene it was trained on. When multiple models are placed within a shared scene, these local coordinate systems must be mapped into a common world-space coordinate frame.

To accomplish this, each model is assigned a rigid transformation consisting of a rotation matrix R, translation vector t, and scale factor s. This transformation is then applied to the cloned camera rays before querying the model. Applying the transformation to the camera rays instead of each individual model's learned density field avoids warping the implicit scene representation, which can negatively affect accurate reconstruction.

For a ray with origin $\mathbf{o}$ and direction $\mathbf{d}$ generated by the shared rendering camera, the transformed ray used to query a particular model is:

$$\begin{aligned} \mathbf{o}' &= s(R\mathbf{o}) + \mathbf{t} \\ \mathbf{d}' &= R\mathbf{d} \end{aligned} \tag{3.1}$$

This transformation maps the ray from the global world coordinate frame into the local coordinate frame of the model. After the transformation is applied, the `RayBundle` is passed through the model's collider, which updates the near and far ray bounds according to the model's scene bounding box. The bounds of this scene box are set to ensure that ray sampling occurs only within spatial regions occupied by the model. This focuses the locations of ray samples to spots along a ray that are the most important

for representing the scene, and improve reconstruction.

3.1.3 Per-Model Ray Sampling

After transforming the rays into an individual model's coordinate frame, the renderer performs the standard NeRF sampling procedure independently for each model. Each model uses its proposal sampler network to determine where along the ray samples should be placed. The proposal sampler iteratively refines sample positions along the ray according to the predicted density distribution of the model.

For each sampled interval along a ray, the NeRF field produces two quantities: a density value σ, representing the volumetric opacity of the field at that location, and a color or channel value $\mathbf{c}$, representing the radiance emitted toward the camera. Since each model performs sampling independently, the resulting ray partitions often differ between models. As a result of the individual transforms for each model, one model may place samples in regions where another model does not. Furthermore, ray samples generated by each model still exist separately from one another. In order to correctly combine the contributions of multiple models, these per-model ray samples must first be merged into a common ray partition.

3.1.4 Ray Partition Merging

To construct a shared ray partition to represent the combined contribution of each model being displayed in a scene, the renderer first collects all sampled interval boundaries produced by each model along the ray. Each model provides a set of interval "start" and "end" values corresponding to the sampled segments of that ray. These endpoints are concatenated and sorted to form a unified set of

depth coordinates along the ray.

Let the set of all sample starts and ends be:

$$\mathcal{P} = \{t^{(m)}_{\text{start},i}\} \cup \{t^{(m)}_{\text{end},i}\}, \tag{3.2}$$

for all models m and all sample intervals i. After sorting these boundary points by distance along each ray, consecutive duplicate values are removed to avoid degenerate and potentially numerically unstable intervals. The resulting ordered set of depth values defines a new partition of the ray into contiguous segments:

$$[t_0, t_1], [t_1, t_2], \ldots, [t_{K-1}, t_K]. \tag{3.3}$$

For each merged segment, a midpoint value is computed:

$$t_{\text{mid},k} = \frac{t_k + t_{k+1}}{2} \tag{3.4}$$

along with the segment length:

$$\Delta_k = t_{k+1} - t_k. \tag{3.5}$$

These merged segments form a common sampled structure across all models. However, the possibility of samples from individual models overlapping must be accounted for in edge cases. For each merged segment midpoint, the renderer locates which original interval from each model contains that midpoint. The density and color values associated with that interval are then assigned to the merged segment. This allows each model to contribute a density and color estimate for every merged segment along the ray when an original sample coincides with a merged one. The densities

of all models are summed over each merged ray segment so that volumetric extinction accumulates consistently across overlapping fields:

$$\sigma_k = \sum_{m=1}^{M} \sigma_{m,k}. \tag{3.6}$$

The segment color is then approximated as a density-weighted average of the per-model color predictions, producing an effective local radiance value for the merged segment. This approximation is exact only under a piecewise-constant assumption within each merged interval, but in practice provides a stable and physically interpretable rule for composite volumetric rendering.

3.1.5 Volumetric Compositing of Combined Rays

Once the merged ray segments have been constructed and their density and color values determined, the final rendering step follows the standard NeRF volumetric compositing formulation. For each segment k, an opacity value is computed as

$$\alpha_k = 1 - \exp(-\sigma_k \Delta_k) \tag{3.7}$$

and the transmittance of the ray up to that segment is defined as:

$$T_k = \exp\left(-\sum_{j<k} \sigma_j \Delta_j\right). \tag{3.8}$$

The segment weight used during color accumulation is then:

$$w_k = T_k \alpha_k. \tag{3.9}$$

The final color for the pixel corresponding to the ray is obtained by accumulating the weighted color contributions of each segment:

$$\mathbf{C} = \sum_k w_k \mathbf{c}_k. \tag{3.10}$$

Depth and total opacity accumulation values are computed using the same volumetric weights. By recomputing volumetric weights over the merged ray partition, the renderer ensures that the densities of multiple models compete faithfully for visibility along the ray, enabling physically consistent occlusion to occur between the separate models in a scene.

3.1.6 Multiple NeRF Model Rendering Algorithm

Algorithm 1 Multi-Model NeRF Ray Composition

Input: Shared camera C, NeRF models $\{M_1, \ldots, M_N\}$, transforms $\{T_1, \ldots, T_N\}$
Output: Rendered image
Generate a `RayBundle` from camera C
for each chunk of rays **do**
    for each model M_i **do**
        Clone the ray bundle
        Transform rays using T_i
        $\mathbf{o}' = s(R\mathbf{o}) + \mathbf{t}$
        $\mathbf{d}' = R\mathbf{d}$
        Apply model collider to update near and far bounds
        Run proposal sampler to generate ray samples
        Evaluate the NeRF field to obtain densities $\sigma_{i,k}$ and channels $\mathbf{c}_{i,k}$
        Store interval boundaries, densities, and channels
    end for
    Merge all interval boundaries across models
    for each merged segment k **do**
        Assign density and channel contributions from each model
        Compute combined density $\sigma_k = \sum_{i=1}^{N} \sigma_{i,k}$
        Compute density-weighted color $\mathbf{c}_k = \frac{\sum_{i=1}^{N} \sigma_{i,k} \mathbf{c}_{i,k}}{\sum_{i=1}^{N} \sigma_{i,k}}$
    end for
    Compute volumetric weights
    $\alpha_k = 1 - \exp(-\sigma_k \Delta_k)$
    $T_k = \exp\left(-\sum_{j<k} \sigma_j \Delta_j\right)$
    $w_k = T_k \alpha_k$
    Composite final color $\mathbf{C} = \sum_k w_k \mathbf{c}_k$
end for
Reshape outputs into image dimensions

3.2 Rendering Multiple 3DGS Models

Rendering 3D Gaussian Splatting models is a fundamentally different paradigm from rendering NeRF or any ray-based methods. While NeRFs exist as implicit neural fields over the entire scene, 3DGS models maintain their representation of the scene as a set of explicit, anisotropic Gaussian

primitives that are optimized to best reconstruct the scene during training. As a result, the Gaussian primitives from each model are transformed into a shared world coordinate system and rasterized together as a single unified set of splats, significantly streamlining the multi-model composition and rendering process.

3.2.1 System Level Rendering Pipeline

As with the NeRF-based models, the pipeline begins by instantiating a shared *Nerfstudio* `Cameras` object to serve as the rendering camera. For each model present in the scene, its Gaussian primitives are transformed according to a user-defined rigid transform and concatenated into a single set of primitives. With all primitives collected, the renderer performs a single rasterization pass over the combined set of Gaussians, leveraging the CUDA-accelerated rasterization engine to produce the final image.

3.2.2 3DGS Model Transformations

Each independently trained 3DGS model represents geometry within its own local coordinate frame, centered about the origin. In order to place multiple models within a shared scene, each model must undergo a rigid transformation that maps it into the global world frame. As before the rigid transform consists of rotation matrix R, translation vector t, and scale factor s, which are applied directly to the Gaussian primitives of each model before rasterization.

The center of each Gaussian is transformed according to:

$$\mu_i' = s(R\mu_i) + t, \tag{3.11}$$

where μ_i is the original Gaussian center and μ_i' is the transformed center in world coordinates. Because Gaussian splats represent anisotropic volumes with orientation, the quaternion orientation of each must also be transformed. The global rotation matrix R is thus converted to quaternion form and multiplied with each Gaussian's original quaternion:

$$q_i' = q_R \otimes q_i. \tag{3.12}$$

where q_R represents the quaternion form of the rotation matrix and $\otimes$ denotes quaternion multiplication.

Lastly, the Gaussian scale parameters must be adjusted via the scale factor s. Since the Gaussian scales are stored in log-space for numerical stability, the scaling operation becomes additive:

$$\log \sigma_i' = \log \sigma_i + \log s. \tag{3.13}$$

3.2.3 Combining 3DGS Models

After applying the spatial transformations to each model, the renderer aggregates the Gaussian primitives from all models into a single unified set. Unlike multi-model NeRF rendering, which combines the ray samples of each model along each ray into a single merged partition, 3DGS models can be combined simply by concatenating their primitive parameter tensors.

For each model, the renderer extracts Gaussian centers, orientations, scales, opacities, and appearance parameters, which are then concatenated to produce a unified set of primitives representing the entire scene. This aggregation step effectively treats all Gaussian primitives as belonging to a single composite model.

3.2.4 Rasterization of Combined Gaussian Set

Once the Gaussian primitives of all models have been transformed and concatenated, the renderer performs a single rasterization pass to produce the final image, as detailed in Section 2.2.

3.2.5 Multiraster Algorithm

Algorithm 2 Multiraster

Input: Rendering camera C, trained Gaussian models $M_1, \ldots, M_N$, transforms $T_1, \ldots, T_N$
Output: Rendered image I
Initialize camera intrinsics and view matrix from C
Initialize empty tensors for Gaussian parameters
for each model M_i **do**
    Extract Gaussian parameters: centers μ_i, scales σ_i, orientations q_i, opacities α_i, colors c_i
    Retrieve model transform $T_i = (R_i, t_i, s_i)$
    Transform Gaussian centers: $\mu_i' = s_i(R_i\mu_i) + t_i$
    Update Gaussian orientations via quaternion multiplication: $q_i' = q_{R_i} \otimes q_i$
    Update Gaussian scales: $\log \sigma_i' = \log \sigma_i + \log s_i$
    Append transformed parameters to global Gaussian tensors
end for
Concatenate Gaussian parameters from all models into a unified Gaussian set
Perform shadow computation (see Chap. 4)
Project Gaussians into screen space using camera intrinsics and view matrix
Rasterize Gaussian splats and compute per-pixel contributions
Output final rendered image I

3.3 Live Rendering of Multiple Models

As part of *Nerfstudio*'s full-stack radiance field training and rendering pipeline, a live, web-based viewer known as *Viser* [52] is used. *Viser* enables real-time rendering and novel view synthesis of *Nerfstudio*'s NeRF and 3DGS models, as well as selection of various rendering modes to view depth and accumulation in addition to traditional RGB output.

A core component of the work towards rendering multiple separately trained NeRF or 3DGS

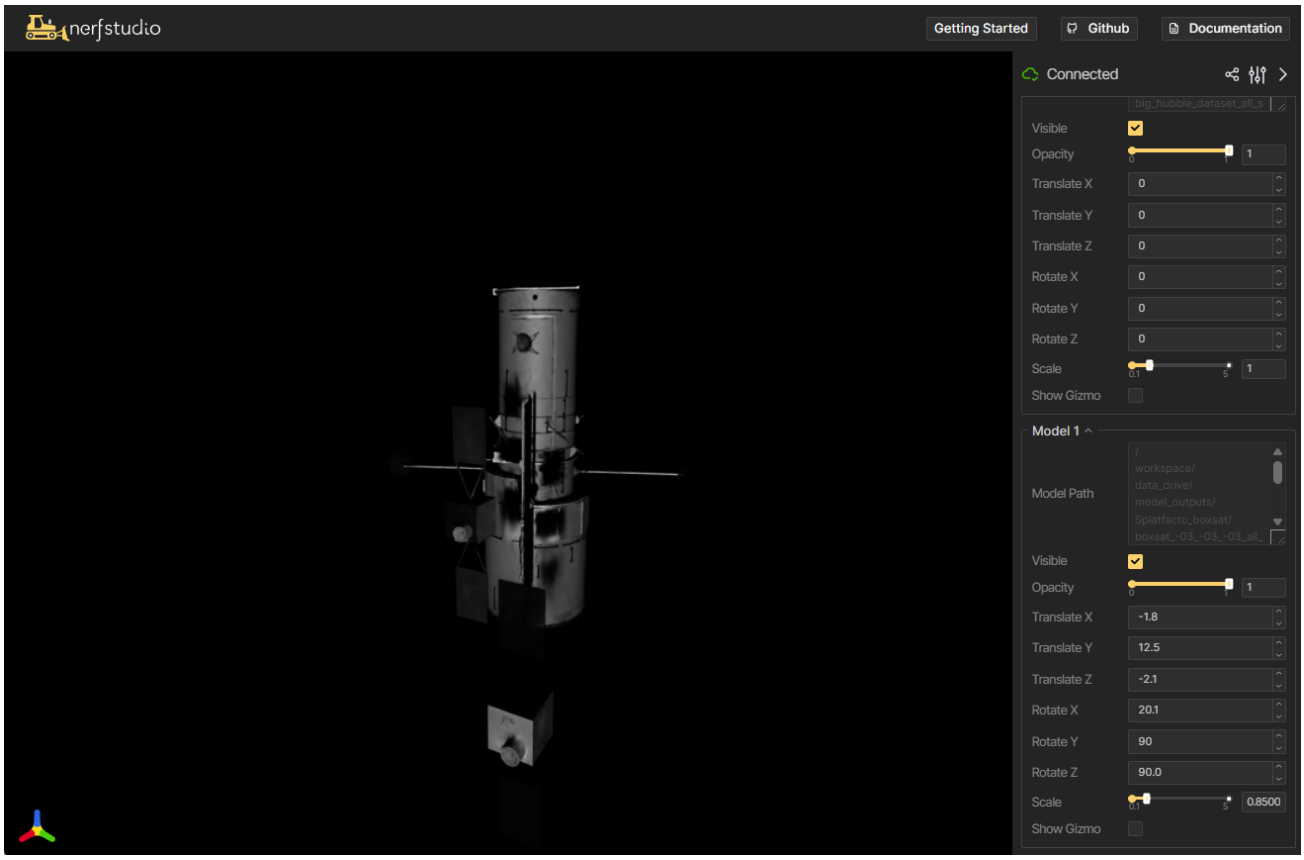


Figure 3.1: Two 3DGS boxsat models casting shadows onto a separately-trained 3DGS Hubble model in *Viser*.

models in a single scene was integration of this new rendering pipeline with *Viser*'s live rendering capabilities for easier analysis and improved user experience. In addition to wiring client-side rendering calls with the NeRF and 3DGS multi-model renderers described in Sections 3.1 and 3.2, users can translate, rotate, and scale each model in real-time via interactive transformation gizmos and per-model scale settings.

These capabilities allow for seamless examination of multiple independently trained models, streamlining the comparison process and making it easier to use dynamic relighting features, discussed in Chapter 4.

3.4 Advantages of 3D Gaussian Splatting

While both NeRF and 3DGS models have the ability to reconstruct 3D models from 2D image datasets, the explicit primitives maintained by 3D Gaussian splatting and the speed of the rasterization process allow 3DGS models to be rendered orders of magnitude faster than NeRF models. This effect is exacerbated when more than one model is composed in a given scene, shown in Figure 3.2. Due to the expensive ray-sampling and repeated neural network evaluations, NeRF models render around *400x* slower than 3DGS models, motivating a shift to 3D Gaussian splatting as the primary method in this thesis.

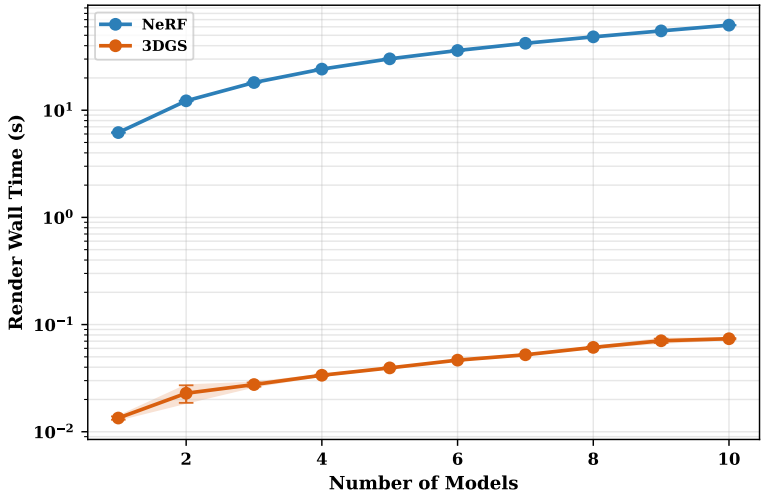


Figure 3.2: Rendering time comparison between NeRF and 3DGS methods.

Chapter 4: Sun-Conditioned Dynamic Relighting and Shadowing for 3D Gaussian Splatting Models

The environment of space is characterized by harsh lighting and shadowing conditions that complicate learning accurate scene representations via learned methods. Sharp specular reflections off solar panels and foil materials or deep shadows caused by self-occluding geometry or other objects can confuse NeRFs or 3DGS models, making it difficult to capture a broad sweep of lighting conditions with vanilla implementations. Furthermore, to retain a sense of realism in multi-model scenes, complex shadowing effects occurring between models must also be taken into account.

4.1 Learned Appearance and Feature Embeddings

A fundamental limitation of learned scene representations is the assumption that a single, static radiance function can encode all possible observations of a scene. However, collections of both terrestrial and space-based images often exhibit significant photometric variation due to changes in illumination or temporary occlusions. Therefore, enforcing a single consistent appearance across all views can result in warped geometries or improper coloring when lighting conditions change. This issue arises across both neural radiance field and 3D Gaussian splatting representation families, since both methods regress appearance from a heterogeneous dataset using shared parameters.

Learned appearance embeddings were introduced to address this limitation by explicitly modeling appearance variation as a latent conditioning variable. For neural radiance field methods, NeRF-W (or NeRF-in-the-Wild) [7] introduced per-image embeddings to capture view-dependent photometric effects, allowing the core scene geometry to remain stable while appearance varies across poses. This formulation effectively separates geometry from variations in lighting, decoupling the

learned structure and its appearance under a particular observation condition. More recently, works like Splatfacto-W and WildGaussians [18, 19] have incorporated these techniques into 3D Gaussian splatting models, showing that combining per-image appearance embeddings with per-Gaussian neural color features improves the performance of 3DGS models under unconstrained image collections, where lighting and appearance differ substantially across views.

For 3D Gaussian splatting models, this allows each Gaussian to retain its optimized position, orientation, anisotropic scale, and opacity primitives, but its effective color representation is no longer represented directly from a single set of spherical harmonic coefficients. Instead, a learned function implicitly maps conditioning variables — such as local Gaussian latent features, sun direction, and view direction — to the color values used for rendering. This is especially useful in scenes where the same physical structure is viewed under different lighting conditions, as it simultaneously preserves consistent geometry while varying appearance behavior. Per-Gaussian feature embeddings further increase the expressivity of the scene representation by learning an additional per-primitive feature vector. These feature vectors can then be used as inputs to an appearance network, serving as local latent descriptors that can capture information beyond that of spherical harmonics, such as local appearance biases or dynamic changes.

In the presented system, these ideas are incorporated directly into the Gaussian representation. Each Gaussian present in the scene stores its geometric parameters and opacity, while its appearance is encoded via a learned *appearance field* that predicts a view- and sun-conditioned color value at render time.

4.2 Dynamic Relighting

In this section, the aforementioned appearance modulation techniques are employed on *Nerfstudio*'s core 3DGS model, Splatfacto, to perform dynamic relighting of learned spacecraft scenes based on the incoming sun direction. While appearance or feature embeddings can account for image-specific photometric variation, they do not by themselves provide an explicit mechanism for controllable relighting. For applications involving spacecraft and other objects observed under changing solar conditions or at different times, absorbing global illumination changes purely into latent appearance codes is insufficient. To retain appropriate control of relighting, appearance must be modeled as a function of incident sunlight direction, motivating the dynamic relighting component of the rendering pipeline.

The key idea is to condition Gaussian appearance on the sun direction. Instead of predicting a single canonical appearance for each Gaussian in the scene, the model predicts per-Gaussian color conditioned on an illumination encoding derived from the current solar incidence angle and the viewing direction. In the presented implementation, this functionality is combined with per-Gaussian feature embeddings. When enabled, the renderer extracts sun-angle metadata associated with the current camera and converts it into a 3D sun-direction unit vector. Given sun azimuth ϕ and elevation θ, the unit sun direction (pointing away from the origin of the scene, towards the sun) is:

$$\mathbf{s} = \begin{bmatrix} \cos\theta\cos\phi \\ \cos\theta\sin\phi \\ \sin\theta \end{bmatrix}. \tag{4.1}$$

This produces a physically interpretable directional descriptor that can be used to modulate appearance

prediction.

To make use of the latent per-Gaussian variables and the sun direction, the model instantiates a dedicated appearance prediction module, `SplatfactoAppearanceField`, whenever Gaussian feature embeddings or dynamic relighting are enabled. This field functions as a neural decoder from conditioning space to renderable Gaussian appearance parameters. The network then outputs a direct color value for each Gaussian, allowing the rendered appearance to be determined at inference time from the active conditioning signals rather than from a single static set of stored coefficients. With all three enabled, the appearance field prediction is as follows:

$$\mathbf{c}_i = f_\theta\left(\mathbf{z}_i^{(g)}, \psi(\mathbf{s}), \psi(\mathbf{d}_i)\right), \tag{4.2}$$

where $\mathbf{z}_i^{(g)} \in \mathbb{R}^{128}$ is the learned appearance vector for Gaussian i, $\mathbf{s}$ is the sun direction vector, $\mathbf{d}_i$ is the view direction from the camera to Gaussian i, and f_θ is the learned appearance field function. When enabled, f_θ is trained concurrently with the optimization of the remaining Gaussian primitives. This design combines the standard 3D Gaussian splatting appearance modeling with more physically meaningful illumination variations present in space-based imagery, utilizing an explicitly controllable lighting representation while retaining the efficiency of Gaussian splatting. Table 4.1 details the appearance field parameters used in the subsequent experiments.

Parameter	Value
Gaussian appearance feature dimension	128
Appearance network depth	3 layers
Appearance network width	256 hidden units
Sun-direction SH encoding degree	4

Table 4.1: Appearance field configuration used for dynamic relighting.

Figure 4.2 shows the relighting performance over a sweep of sun angles using a vanilla 3DGS

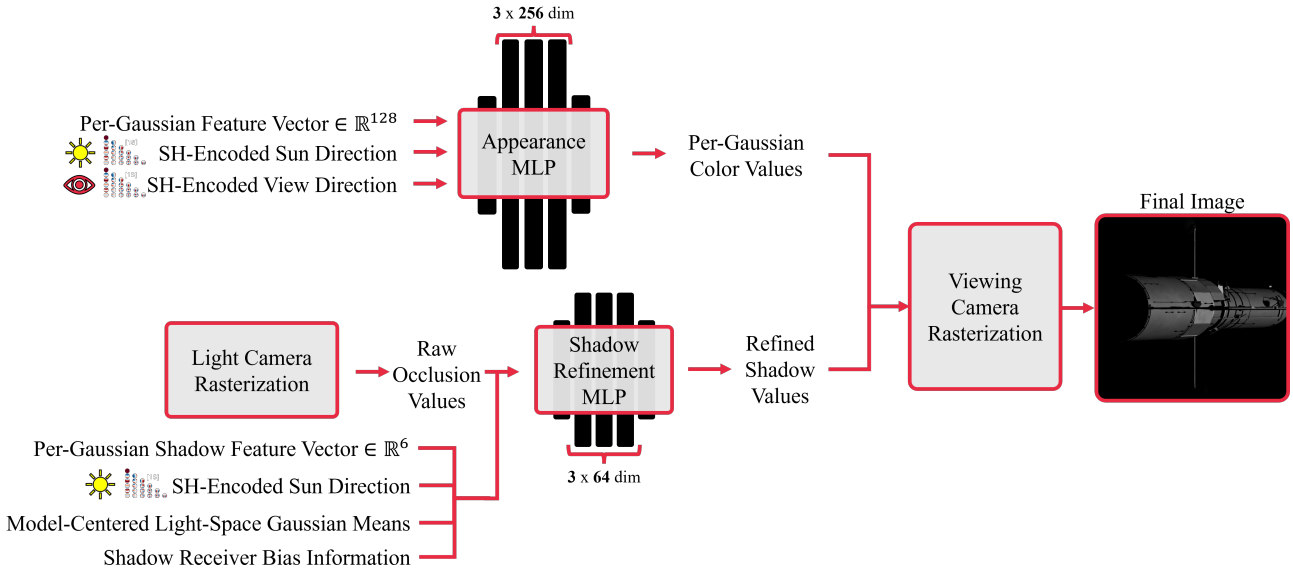


Figure 4.1: Illustration of the full appearance module pipeline.

implementation and a dynamically relightable 3DGS model. The "Vanilla" and "Relightable" models were trained on a dataset with 100 sun angles (1000 images per sun angle) distributed using a spherical Fibonacci mapping, and the "Vanilla (fully-lit only)" was trained on the same data, but filtered to only use training images within azimuth/elevation tolerances of $(30°, 15°)$ from each incident sun direction. As expected, the relightable model is able to successfully ingest and encode the sun direction into its appearance, using the diverse illumination cases present in the data to enable dynamic relighting. In contrast, the Vanilla methods are unable to perform fixed-view relighting, as their appearance only changes when the view direction changes. Of particular note is the regular "Vanilla" case in the second row — with no mechanism for disentangling the conflicting lighting conditions present in the dataset from its appearance, the model collapses, culling the majority of the scene's Gaussians altogether.

The relightable model trades per-condition specialization for cross-condition generality. Single-illumination models can bake a fixed lighting state directly into Gaussian color, opacity, and geometry, while the relightable model must explain multiple illumination states using one shared spatial

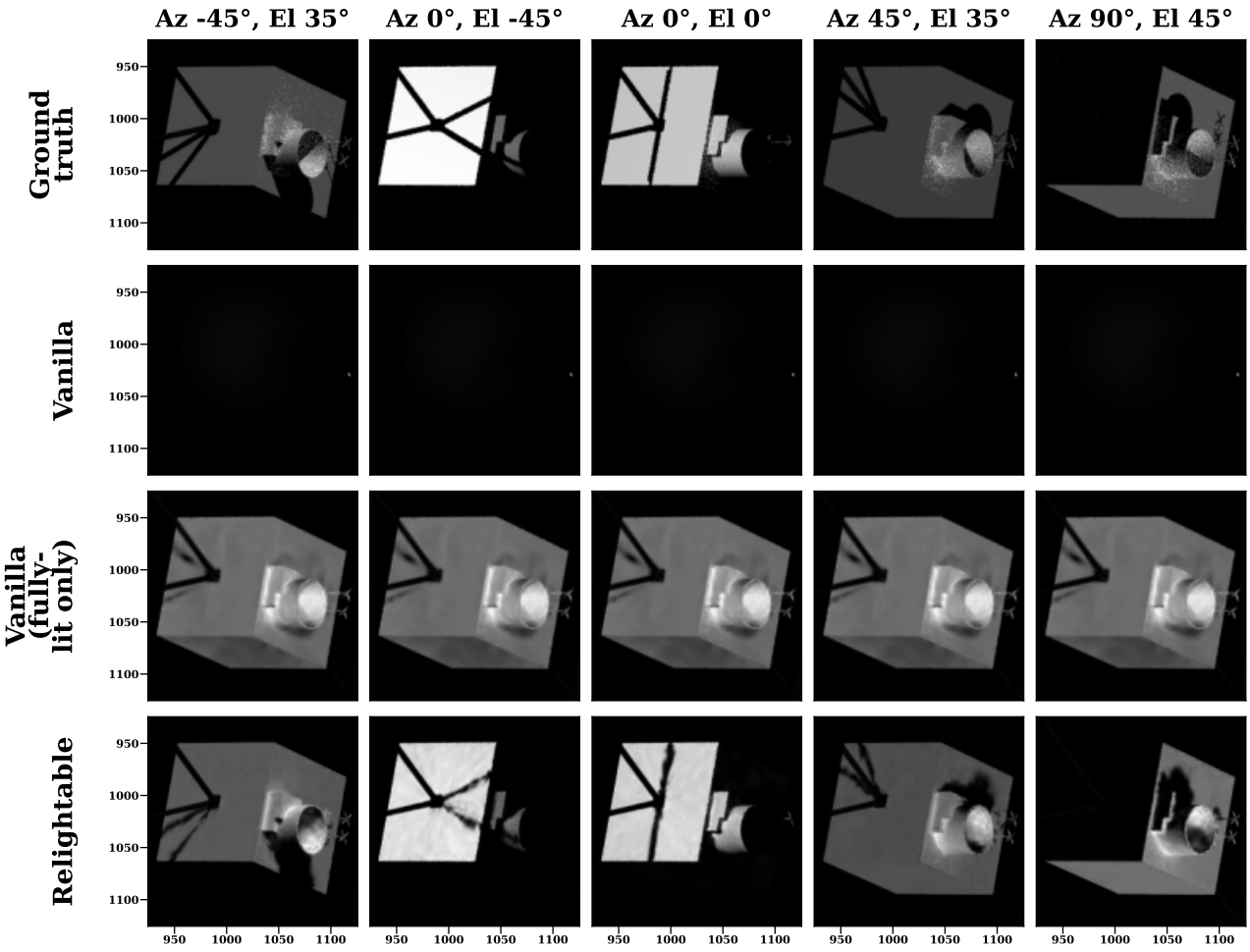


Figure 4.2: Appearance field-based dynamic relighting with a single 3DGS model.

representation and a conditioned appearance function. As a result, gradients from different sun angles impose competing constraints, especially in shadowed or specular regions. Shown in Figure 4.3, this leads to modestly reduced performance on any one illumination condition despite improved flexibility across lighting conditions.

However, the single-illumination case is not necessarily preferable from a geometric standpoint. Since surfaces that remain shadowed under the fixed lighting condition contribute limited image evidence, they are often weakly supervised during optimization. Consequently, the model can display strong photometric performance for the observed illumination while still producing incomplete or

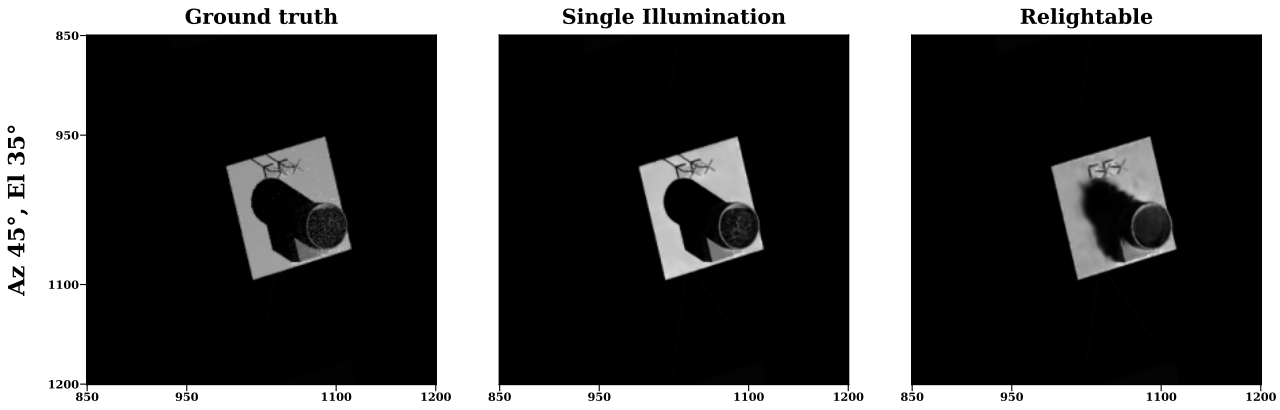


Figure 4.3: Photometric reconstruction comparison between fixed-illumination-trained and relightable 3DGS models.

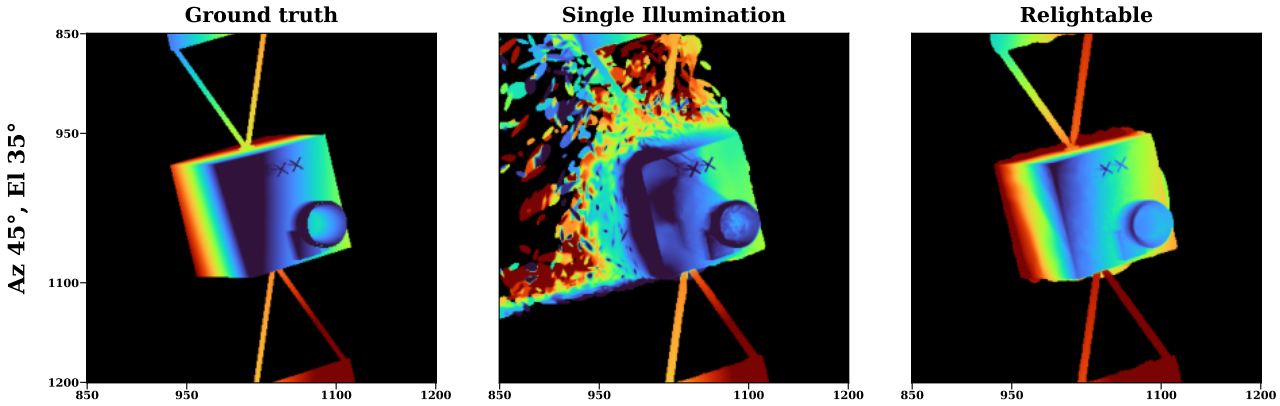


Figure 4.4: Spatial reconstruction comparison between fixed-illumination-trained and relightable 3DGS models.

poorly formed geometry in persistently shadowed regions. This behavior is highlighted by the depth visualization in Figure 4.4, which demonstrates that per-condition photometric specialization can come at the cost of spatial completeness.

4.3 Inter-Model Shadowing

Accurate shadow modeling is particularly important in the context of space-based scenes, where illumination cases are dominated by a single, distant, high-intensity light source—the sun. Unlike terrestrial environments, space often lacks light scattering effects from the atmosphere or indirect

global illumination. As a result, lighting interactions are characterized by extreme contrast, with sharply defined shadow boundaries, deep occlusion regions, and strong view-dependent reflectance from metallic and other specular surfaces. In multi-spacecraft scenes, explored in Chapter 3, these effects are critical for retaining realism and scene coherence, and play a large role in the accurate reconstruction of space-based models.

To address this, an explicit shadowing pipeline is incorporated into the rendering system. Two approaches are explored: Gaussian shadow mapping, inspired by traditional computer graphics rendering systems, and shadow splatting, a method tailored to 3D Gaussian splatting that re-uses the rasterization engine to enable shadow reconstruction. In both implementations, a *Nerfstudio* `Cameras` object is instantiated in order to perform rasterization passes from the direction of the light source/sun.

4.3.1 Shadow Mapping

Traditional shadow mapping approximates visibility by rendering the scene from the perspective of the light source and storing the depth of the closest surface along each light ray. For a point $\mathbf{x}$ in the scene, its visibility is determined by projecting it into light space and comparing its depth $z_{\text{light}}(\mathbf{x})$ against the stored depth map D_{light}:

$$V(\mathbf{x}) = \begin{cases} 1, & z_{\text{light}}(\mathbf{x}) \leq D_{\text{light}}(\pi(\mathbf{x})) \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where $\pi(\cdot)$ denotes projection into the light image plane.

In this thesis' implementation, this is accomplished by constructing a sun-aligned orthographic

camera and rasterizing Gaussian centers into a light-space depth map. For any given light ray, the light-space reference depth used for comparison is the expected depth of a pixel, or the weighted average depth of the splats contributing to that light pixel:

$$D_{\text{light}}(\mathbf{u}) = \frac{\sum_{j \in \mathcal{N}(\mathbf{u})} T_j(\mathbf{u}) \alpha_j(\mathbf{u}) z_j}{\sum_{j \in \mathcal{N}(\mathbf{u})} T_j(\mathbf{u}) \alpha_j(\mathbf{u})}, \tag{4.4}$$

where $\alpha_j(\mathbf{u})$ is the projected opacity contribution of Gaussian j at light-space pixel $\mathbf{u}$ and $T_j(\mathbf{u})$ is the accumulated transmittance before Gaussian j along the light ray.

During rendering, each Gaussian is tested against this map to determine whether or not it is shadowed. Rather than performing a strict binary depth comparison, a soft visibility formulation is used to improve stability and reduce aliasing artifacts. For each Gaussian i, its projected 2D location in the light image is used to sample its corresponding reference depth value via bilinear interpolation. Visibility is then computed as a smooth function of the depth difference:

$$v_i = \text{sigmoid}\left(\frac{z_{\text{ref}}(u_i) - (z_i - b_i)}{\epsilon_s}\right) \tag{4.5}$$

where $\text{sigmoid}(\cdot)$ denotes the sigmoid function, z_i is the light-space depth of Gaussian i, $z_{\text{ref}}(u_i)$ is the interpolated reference depth from the light map at its projected location u_i, b_i is a per-Gaussian receiver-depth bias to mitigate self-shadowing (discussed in Section 4.3.4), and ϵ_s is a softness parameter controlling the sharpness of the shadow transition. In this thesis, $\epsilon_s = 10^{-6} \cdot dist_{\max}$, where $dist_{\max}$ denotes the maximum training camera distance. This smoothing provides a continuous approximation of visibility, avoiding incorrect binary classification due to overlapping primitives and

numerical precision. Finally, the occlusion value is obtained as the complement of visibility:

$$\text{occ}_i = 1 - v_i. \tag{4.6}$$

While efficient and simple, this formulation possesses some limitations: it struggles to model partial occlusion, where multiple semi-transparent Gaussians collectively attenuate light, and it has no method for capturing sub-pixel visibility variations due to the discretization of the light-space depth map. These limitations motivated exploration of a splatting-based shadowing methodology.

4.3.2 Shadow Splatting

To better align shadow computation with the 3D Gaussian scene representation, we employ *shadow splatting* [16], which computes visibility through transmittance/opacity accumulation rather than smoothed binary depth comparisons. In shadow splatting, this accumulation of transmittance is performed in light space. Each Gaussian contributes to occlusion based on the amount of light already attenuated along the ray. Due to the anisotropic nature of the Gaussians, this projection results in an elliptical footprint when projected into 2D screen space, characterized by a covariance matrix Σ_i. Thus, the influence of Gaussian i at a pixel location is determined by:

$$\beta_i = \exp(-\sigma_i), \qquad \sigma_i = \frac{1}{2}\boldsymbol{\delta}^\top \Sigma_i^{-1} \boldsymbol{\delta}, \tag{4.7}$$

where $\boldsymbol{\delta}$ is the offset between the pixel center and the projected Gaussian mean. This spatial weighting β_i encodes how strongly a Gaussian overlaps a pixel, and therefore how much influence it is allowed to have on both color and shadow accumulation — enabling continuous, sub-pixel integration of Gaussian

influence.

Gaussians are processed in order of increasing depth from the light source, where the cumulative transmittance up to Gaussian i is:

$$T_i^{\ell} = \prod_{j<i}(1-\alpha_j), \tag{4.8}$$

where the superscript ℓ denotes accumulation in the light-view rasterization path. This quantity represents the fraction of light that reaches Gaussian i. The corresponding raw occlusion signal is then defined as:

$$\mathrm{occ}_i = 1 - T_i^{\ell}, \tag{4.9}$$

capturing the amount of light that has been *absorbed* or blocked along the light ray. $\mathrm{occ}_i = 0$ indicates that the Gaussian is fully visible to the light source, and $\mathrm{occ}_i = 1$ indicates that the Gaussian is fully occluded.

To produce a stable estimate across overlapping Gaussians in light space, shadow contributions are aggregated using the spatial weights:

$$\bar{\mathrm{occ}}_i = \frac{\sum_{k\in\mathcal{N}(i)} \beta_k \,\mathrm{occ}_k}{\sum_{k\in\mathcal{N}(i)} \beta_k}, \tag{4.10}$$

where $\mathcal{N}(i)$ denotes the set of overlapping Gaussian contributions used to estimate the raw occlusion value for Gaussian i. This weighted estimate produces a smoother per-Gaussian shadow signal that is less sensitive to individual rasterization artifacts.

4.3.3 Shadow Refinement

To reduce artifacts in the raw shadow assignment/accumulation processes and represent a broader range of shadowing, a learned shadow refinement stage inspired by GS^3 [16] is adopted for both shadowing methods. This network operates on the initial occlusion estimate and predicts a corrected scalar occlusion value for each Gaussian. The refinement network ingests a shadow-specific feature vector for each Gaussian i, constructed as:

$$\mathbf{h}_i = \left[\mathrm{o\bar{c}c}_i,\ \mathbf{f}_i,\ \psi(\mathbf{s}),\ \tilde{\boldsymbol{\mu}}_i^{\ell},\ \mathbf{b}_i^{\mathrm{info}}\right]. \tag{4.11}$$

Here, $\mathrm{o\bar{c}c}_i$ is the raw occlusion estimate, $\mathbf{f}_i$ is a learned shadow-specific feature vector, $\psi(\mathbf{s})$ is a spherical-harmonic encoding of the sun direction, $\tilde{\boldsymbol{\mu}}_i^{\ell}$ is the model-centered light-space Gaussian mean, and $\mathbf{b}_i^{\mathrm{info}}$ contains geometric bias information derived from the receiver-depth bias formulation introduced in Section 4.3.4. Together, these terms provide the refinement network with the initial shadow estimate, primitive-specific context, illumination direction, object-relative light-space position, and local geometric information related to self-shadowing artifacts. The model-centered light-space Gaussian mean is computed as:

$$\tilde{\boldsymbol{\mu}}_i^{\ell} = \mathbf{R}_{\ell}^{\top}\left(\boldsymbol{\mu}_i^{\mathrm{local}} - \boldsymbol{\mu}_{\mathrm{model}}\right), \tag{4.12}$$

where $\mathbf{R}_{\ell}$ is a canonical light-frame rotation aligned with the current sun direction, $\boldsymbol{\mu}i^{\mathrm{local}}$ is the Gaussian mean expressed in the model's training-scale local frame, and $\boldsymbol{\mu}_{\mathrm{model}}$ is the model center in that same frame. This feature is computed in a canonical sun-aligned frame rather than from the origins

or crop coordinates of the light-view cameras used for shadow rasterization, ensuring that the spatial conditioning remains tied to the object and illumination direction. Since training is performed at unit model scale, transformed models are mapped back to object-scale units before refinement features are evaluated; similarly, length-like terms in $\mathbf{b}_i^{\text{info}}$, such as receiver bias and light-direction thickness, are divided by the model scale before being passed to the refinement network. The shadow rasterization and depth comparisons themselves remain in world scale.

The refinement itself is performed by a small MLP:

$$\hat{\text{occ}}_i = g_\phi(\mathbf{h}_i), \tag{4.13}$$

from which the refined scalar occlusion value $\hat{\text{occ}}_i$ is obtained. The parameters of the shadow refinement pipeline are listed in Table 4.2.

Parameter	Value
Shadow refinement network depth	3 layers
Shadow refinement network width	64 hidden units
Shadow feature dimension	6
Sun direction encoding degree	4

Table 4.2: Shadow refinement configuration used for learned correction of per-Gaussian shadow estimates.

Since color attenuation is applied using visibility, the refined occlusion is converted into a refined visibility value:

$$\hat{v}_i = 1 - \hat{\text{occ}}_i. \tag{4.14}$$

For composed multi-model scenes, the refinement stage is applied only to self-shadowing. Two raw visibility estimates are computed for each model: v_i^{all}, using the full composed scene as shadow casters, and v_i^{self}, using only the receiver model's own Gaussians. The inter-model shadowing term is estimated

as

$$v_i^{\text{inter}} = \text{clamp}\left(\frac{v_i^{\text{all}}}{v_i^{\text{self}} + \epsilon}, 0, 1\right), \tag{4.15}$$

using the approximation $v_i^{\text{all}} \approx v_i^{\text{self}} v_i^{\text{inter}}$. The refinement network is then evaluated only on the self-shadow occlusion, producing $\hat{v}_i^{\text{self}}$, and the final visibility is

$$\hat{v}_i = \hat{v}_i^{\text{self}} v_i^{\text{inter}}. \tag{4.16}$$

This preserves shadows cast by other models while preventing the learned refinement network from incorrectly correcting them as self-shadowing artifacts. The evaluated visibility is then used to modulate the Gaussian color before rasterization accumulation:

$$\tilde{\mathbf{c}}_i = \hat{v}_i \mathbf{c}_i. \tag{4.17}$$

Following the standard 3D Gaussian splatting color accumulation form introduced earlier, the final shadowed pixel color is computed as:

$$\hat{C}_{p_{(x,y)}} = \sum_{i \in N} \tilde{\mathbf{c}}_i \alpha_i T_i. \tag{4.18}$$

In this formulation, shadowing is part of the rasterization path itself: each Gaussian contribution is attenuated before alpha compositing, rather than applying shadow attenuation only as a post-process to the final rendered image. The refined visibility may still be rasterized as an auxiliary shadow image for visualization or diagnostics, but the primary rendered image is accumulated directly from the visibility-modulated Gaussian colors.

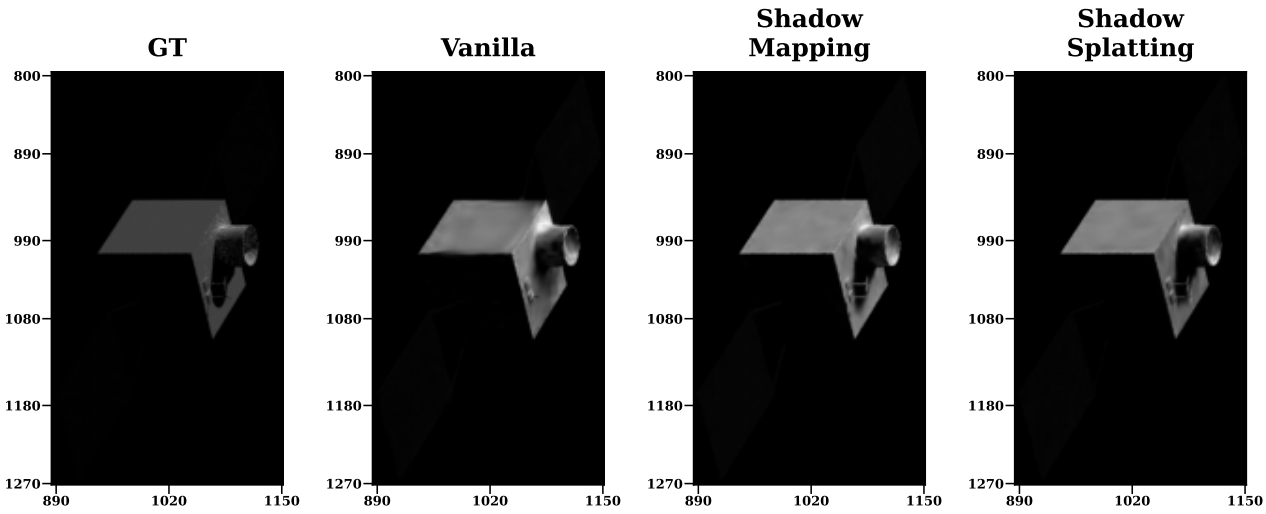


Figure 4.5: Shadowing method progression.

4.3.4 Mitigating Shadow Acne via Receiver-Depth Bias

A challenging aspect of inter- and intra-model shadowing is the appropriate alleviation of surface-level self-shadowing artifacts, dubbed "shadow acne" by the broader computer graphics community. In cases where many closely arranged Gaussians represent a single physical surface perpendicular to the incoming sunlight, minute depth differences and inter-Gaussian overlap can cause Gaussians to incorrectly occlude one another. This produces unnatural superficial darkening across the model, especially on surfaces that should remain directly illuminated.

A simple way to reduce this artifact is to introduce a global light-depth tolerance, so that Gaussians with similar light-space depths are not immediately treated as separate blockers and receivers. However, a global tolerance applies the same correction everywhere in the scene, regardless of local Gaussian size, orientation, or anisotropy. Increasing this tolerance *can* reduce shadow acne, but it can also delay the onset and reduce the sharpness of intended shadows ("peter-panning"). To address this, a per-Gaussian receiver-depth bias, inspired by traditional shadow mapping techniques [53], is

introduced. A light-depth tolerance b_i is computed from the oriented shape of each Gaussian in the scene.

First, each Gaussian's log-scale parameters $\mathbf{l}_i$ are converted to positive scale values as:

$$\boldsymbol{\sigma}_i = \exp(\mathbf{l}_i) = \begin{bmatrix} \sigma_{i,x} & \sigma_{i,y} & \sigma_{i,z} \end{bmatrix}^\top, \tag{4.19}$$

where each scale value $\sigma_{i,a}$ denotes a one-standard-deviation extent of the Gaussian along local axis a. Then, the Gaussian quaternion $\mathbf{q}_i$ is normalized and converted into a rotation matrix:

$$\mathbf{R}_i = \mathrm{Rot}\left(\frac{\mathbf{q}_i}{\|\mathbf{q}_i\|}\right), \tag{4.20}$$

where $\mathbf{R}_i$ maps the local Gaussian axes into world coordinates. Given the current sun direction $\mathbf{s}$, the sun direction in the Gaussian's local coordinate frame is:

$$\mathbf{s}_{\mathrm{local}}^{(i)} = \mathbf{R}_i^\top \mathbf{s}. \tag{4.21}$$

This local light direction is used to estimate how much finite Gaussian thickness may appear along the light ray. The light-direction thickness is computed by projecting the anisotropic Gaussian scale onto the local sun direction:

$$\sigma_i^\ell = \sqrt{\sum_{a\in\{x,y,z\}} \left(\sigma_{i,a} s_{\mathrm{local},a}^{(i)}\right)^2} = \left\| \boldsymbol{\sigma}_i \odot \mathbf{s}_{\mathrm{local}}^{(i)} \right\|_2. \tag{4.22}$$

The σ_i^ℓ term is the primary source of the receiver bias — if a given Gaussian has substantial extent along the light direction, then small light-depth differences between it and nearby Gaussians are more

likely to lead to shadow acne. That Gaussian should therefore receive a larger tolerance before it is considered shadowed by a neighboring primitive.

This initial bias is then modulated by the Gaussian's approximate local normal vector and an anisotropy confidence term. Since a flattened Gaussian often represents a local surface element, its normal can be approximated by the local axis with the smallest scale [54]:

$$a_i^\star = \arg\min_{a\in\{x,y,z\}} \sigma_{i,a}, \qquad \mathbf{n}_i = \mathbf{r}_{i,a_i^\star}, \tag{4.23}$$

where $\mathbf{r}_{i,a}$ is column a of $\mathbf{R}_i$. The normal-light alignment term is then given by

$$\nu_i = \left|\mathbf{n}_i \cdot \mathbf{s}\right|. \tag{4.24}$$

Unlike classic slope-scale shadow mapping, this term applies the finite-thickness tolerance most strongly when the Gaussian has a reliable anisotropic shape and a normal direction that is aligned with the sun. This prioritizes biasing surface-like Gaussians on directly illuminated faces, where small inter-Gaussian depth differences can produce superficial self-shadowing, while avoiding excessive bias near tangent or silhouette-like regions, where valid cast shadows may otherwise be suppressed. The anisotropy confidence is derived from the smallest and largest scales of the Gaussian, and is used to distinguish flattened, truly surface-like Gaussians from more isotropic ones:

$$\gamma_i = \mathrm{clamp}\left(1 - \frac{\sigma_{i,\min}}{\sigma_{i,\max}}, 0, 1\right). \tag{4.25}$$

If a Gaussian is strongly anisotropic, γ_i approaches one, and there is a higher likelihood that the normal-sun alignment can be trusted. Alternatively, if a Gaussian is nearly isotropic, γ_i approaches

zero, reducing reliance on an ambiguous normal estimate. The per-Gaussian receiver bias is therefore:

$$b_i = \lambda_b \sigma_i^{\ell} \nu_i \gamma_i, \tag{4.26}$$

where λ_b controls the strength of the bias. $\lambda_b = 10$ was used for the results presented in this thesis. When trained with a per-Gaussian receiver bias, a model's shadow refinement network is additionally conditioned on $\mathbf{b}_i^{\text{info}}$, comprised of the raw per-Gaussian visibility, normal alignment, light extent, and applied bias.

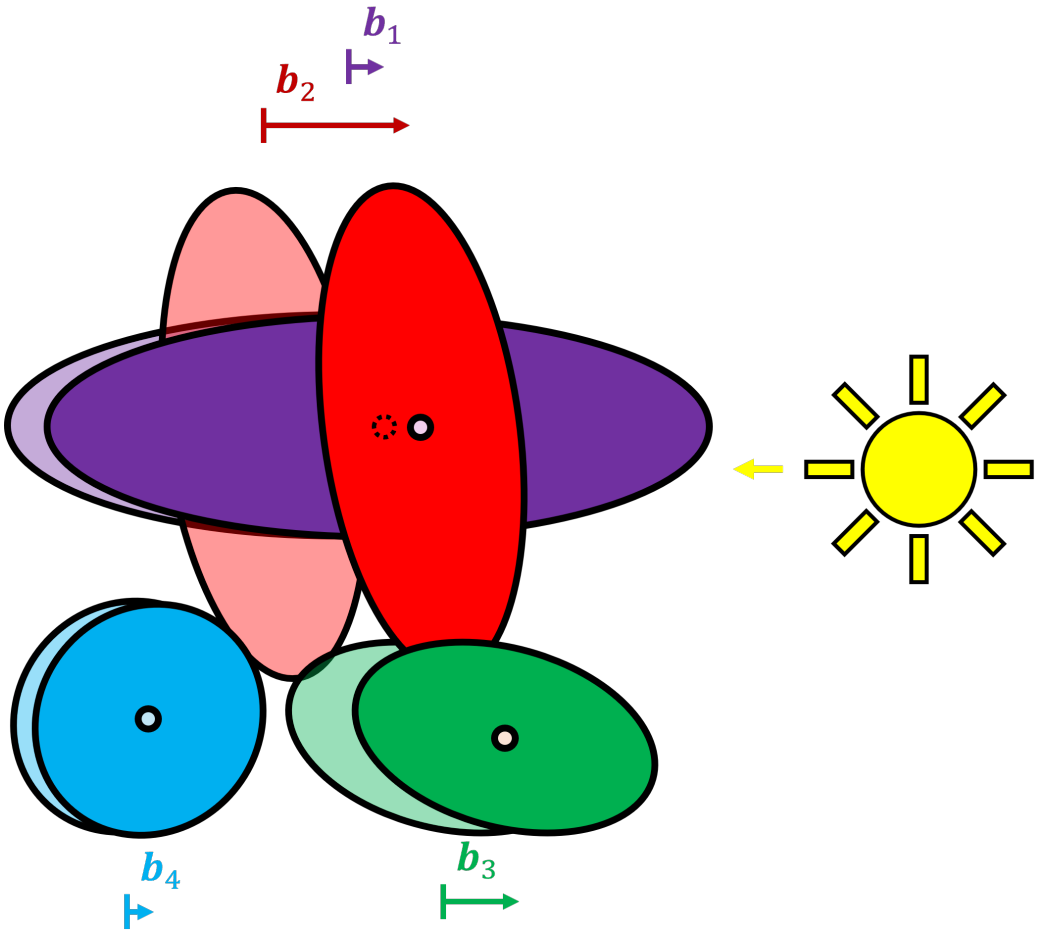


Figure 4.6: Illustration of per-Gaussian receiver-depth biases applied to different Gaussians.

During shadow mapping, this receiver-depth bias is applied directly inside the light-depth

comparison used to determine per-Gaussian visibility in Equation 4.5. For the shadow splatting path, the original receiver/blocker light-depth gap is:

$$\Delta d_{ij} = z_i^\ell - z_j^\ell. \tag{4.27}$$

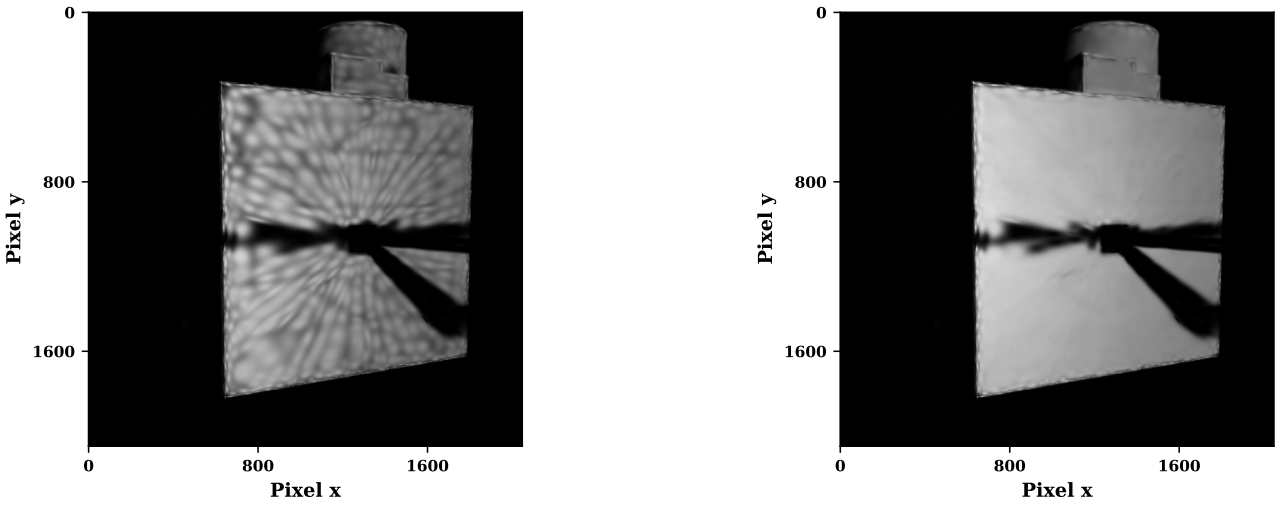
where z_i^ℓ is the receiver depth and z_j^ℓ is the blocker depth in light space. The receiver bias modifies this gap as:

$$\widetilde{\Delta d}_{ij} = \Delta d_{ij} - b_i = z_i^\ell - z_j^\ell - b_i. \tag{4.28}$$

If $\widetilde{\Delta d}_{ij}$ is less than zero, the candidate receiver and blocker are treated as same-depth — effectively shifting all receivers towards the light source and requiring that a candidate blocker must be separated from a receiver by more than that receiver's bias to be treated as a true occluder. In implementation, the receiver bias is computed once per Gaussian in the high-level PyTorch shadow preparation path, using the same combined Gaussian index space passed into the light-view rasterizer. The resulting vector of receiver biases is then passed into the fused shadow splatting rasterization path, where the per-receiver value b_i is subtracted directly from the receiver/blocker depth gap. This keeps the method lightweight, avoiding explicit pairwise surface checks while still adapting the depth tolerance to each Gaussian's local geometry.

In both cases, the receiver-depth bias aims to suppress tiny, same-surface depth gaps while preserving the larger separations associated with real cast shadows on faces perpendicular to sun incidence.

However, a persistent challenge with shadow acne — as is discussed in 4.3.6 — is grazing sun angles where the sun is near-tangent to a face of the object. This is more of an issue for shadow



(a) Shadow acne on the surface of a boxsat model.

(b) After applying per-Gaussian receiver-biases.

Figure 4.7: Fixing shadow acne using receiver-depth biasing.

mapping, where Gaussians are often either shadowed entirely, or not shadowed at all, exacerbating the effect of the splotchy artifacts. Experiments with a more classical slope-scaled depth-bias were unsuccessful in resolving this issue, and caused excessive receding of shadows far more often than it reduced shadow acne.

4.3.5 Adaptive Light Camera Placement for Multi-model Scene Consistency

In multi-model scenes, where independently trained 3DGS models are composed within a shared world coordinate system, shadow computation must remain robust under arbitrary transformations. A fixed light-space camera is insufficient in this regime, as the distributions of Gaussians in a given scene can vary significantly depending on model placement. With a single light camera, edge cases, such as a user moving a model far from the origin or placing multiple models in different regions of the scene, can result in those Gaussian clusters falling outside the view of the light camera and not receiving shadowing support.

Furthermore, an orthographic camera projection is used for the light camera, casting collimated light rays that travel parallel to one another instead of diverging outwards from the screen as they would with a perspective projection. These parallel rays lead to the objects in frame remaining the same size regardless of the camera's distance from them. In *Nerfstudio*'s construction of orthographic cameras, modulating the focal parameter (which will be used interchangeably with "focal length" here) results in a *spreading out* of the light rays/pixels in world-space, or an "enlarging" of the light camera screen. However, smaller focal parameters, and therefore a greater spread of light rays, have adverse effects on shadowing. With fewer light rays intersecting a given region in space, each model is undersampled, and more Gaussians contribute to a single pixel during the light rasterization pass. This produces lower fidelity shadows, as Gaussians further apart in light-space x, y can affect one another.

To support these edge cases, an adaptive light camera system is utilized that recursively splits the scene into smaller clusters with higher individual focal lengths. Given a sun direction $\mathbf{s}$, a light-space coordinate frame is first constructed such that the viewing direction is aligned with $\mathbf{s}$. All Gaussian centers $\boldsymbol{\mu}_i$ are transformed into this light frame:

$$\boldsymbol{\mu}_i^{\ell} = \mathbf{R}_{\ell}^{\top} \left(\boldsymbol{\mu}_i - \mathbf{t}_{\ell} \right). \tag{4.29}$$

This produces a set of light-space locations from which spatial bounds can be estimated. Let $\mathbf{R}_i$ and $\boldsymbol{\sigma}_i$ denote the world-space orientation (converted from normalized quaternions) and scale of Gaussian i, and let $\mathbf{R}_{\ell}$ denote the light-camera-to-world rotation. The covariance of Gaussian i in light space is approximated by:

$$\boldsymbol{\Sigma}_i^{\ell} = \mathbf{R}_{\ell}^{\top} \boldsymbol{\Sigma}_i \mathbf{R}_{\ell}. \tag{4.30}$$

The corresponding estimated projected 3σ support radius in light-camera screen-space is then:

$$\mathbf{r}_i^\ell = 3 \begin{bmatrix} \sqrt{\mathbf{\Sigma}_{i,xx}^\ell} \\ \sqrt{\mathbf{\Sigma}_{i,yy}^\ell} \end{bmatrix}. \tag{4.31}$$

For a cluster $\mathcal{C}_j$, the projected Gaussian supports define light-space bounds:

$$\begin{aligned} x_{\min}^j &= \min_{i \in \mathcal{C}_j} \left(\mu_{i,x}^\ell - r_{i,x}^\ell \right), & x_{\max}^j &= \max_{i \in \mathcal{C}_j} \left(\mu_{i,x}^\ell + r_{i,x}^\ell \right), \\ y_{\min}^j &= \min_{i \in \mathcal{C}_j} \left(\mu_{i,y}^\ell - r_{i,y}^\ell \right), & y_{\max}^j &= \max_{i \in \mathcal{C}_j} \left(\mu_{i,y}^\ell + r_{i,y}^\ell \right). \end{aligned} \tag{4.32}$$

The extent of the cluster in the light-camera image plane is:

$$\Delta x_\ell^j = x_{\max}^j - x_{\min}^j, \qquad \Delta y_\ell^j = y_{\max}^j - y_{\min}^j, \tag{4.33}$$

and a light camera is centered on the midpoint of these fitted bounds:

$$\mathbf{c}_\ell^j = \begin{bmatrix} \frac{1}{2}(x_{\min}^j + x_{\max}^j) \\ \frac{1}{2}(y_{\min}^j + y_{\max}^j) \end{bmatrix}. \tag{4.34}$$

Finally, the orthographic focal parameters required to cover the cluster are computed as

$$f_x^j = \frac{W}{\Delta x_\ell^j}, \qquad f_y^j = \frac{H}{\Delta y_\ell^j}, \tag{4.35}$$

where (W, H) are the light camera image resolution dimensions.

The clustering process operates recursively along the dominant spatial axis in light space: if the spatial extent of a cluster exceeds a minimum focal length threshold, it is split along the axis with the

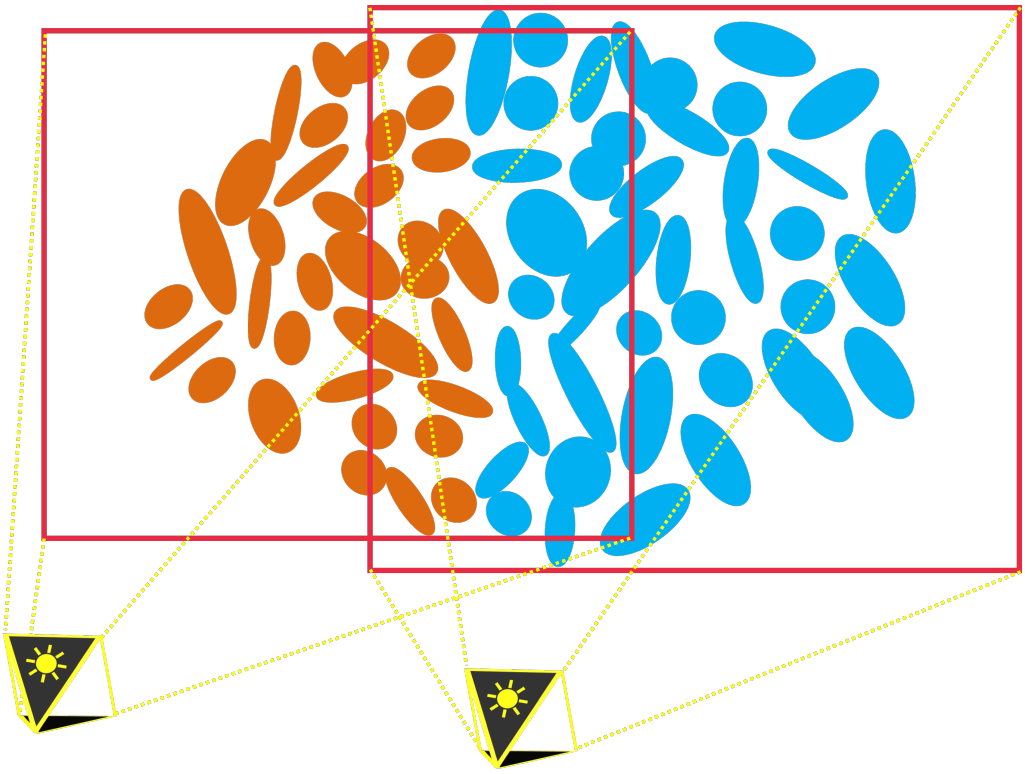


Figure 4.8: Illustration of two Gaussian clusters with their own light cameras.

largest extent using a gap-based heuristic to determine the split location. This process continues until a maximum number of clusters is reached or the focal lengths of all light cameras are sufficient.

The largest of the clusters is designated as the "primary" cluster, and has its rasterization pass performed first. Afterwards, the system re-evaluates which Gaussians were insufficiently covered and performs cluster-specific rasterization passes only for those Gaussians. The resulting shadow values from these fallback rasterization passes are merged with those from the primary cluster. If the cluster budget is exhausted and the focal lengths required to cover the remaining Gaussians are below $f_{\min}$, the algorithm permits rendering with the smaller focal length to ensure no regions of the scene are unaccounted for.

Algorithm 3 Adaptive Light Camera Clustering

Input: Gaussian centers $\{\mu_i\}$, rotations $\{\mathbf{R}_i\}$, scales $\{\boldsymbol{\sigma}_i\}$, sun direction $\mathbf{s}$, max cameras K, cluster-split focal length $f_{\min}$
Output: Raw occlusion values $\mathbf{occ}$
Transform Gaussian centers into light space: $\boldsymbol{\mu}_i^{\ell} = \mathbf{R}_{\ell}^{\top}(\boldsymbol{\mu}_i - \mathbf{t}_{\ell})$
Estimate projected Gaussian support radii $\mathbf{r}_i^{\ell}$ in the light-space x, y plane
Initialize receiver clusters $\mathcal{C} \leftarrow \{\mathcal{C}_0\}$
while $|\mathcal{C}| < K$ **do**
    Select an uncovered cluster $\mathcal{C}_j$
    Fit light-space bounds around $\mathcal{C}_j$ and compute the required focal lengths $\hat{f}_x^j, \hat{f}_y^j$
    if $\min(\hat{f}_x^j, \hat{f}_y^j) \geq f_{\min}$ **then**
        Mark $\mathcal{C}_j$ as sufficiently covered
        break if no remaining cluster requires splitting
    else
        Compute the light-space extent of $\mathcal{C}_j$ in x and y
        Choose the split axis with the largest light extent relative to allowable camera footprint
        Sort receivers along this axis and split at the largest meaningful coordinate gap
            if no meaningful gap exists, split at the median
        Replace $\mathcal{C}_j$ with the two resulting sub-clusters
    end if
end while
for each cluster $\mathcal{C}_j$ **do**
    Fit an orthographic light camera L_j to the cluster bounds
    Rasterize shadow values for receivers assigned to $\mathcal{C}_j$
end for
Merge cluster shadow values into $\mathbf{occ}$
return $\mathbf{occ}$

4.3.6 Shadowing Method Performance

To evaluate the performance of each shadowing method, models were trained on a dataset of 100,000 images of two boxsats, with 1000 Fibonacci sphere-sampled images for each of 100 sun angles (also Fibonacci sampled) at a distance of 50 meters. The boxsat models are trained for a total of 200,000 iterations, and primitive refinement is stopped at iteration 150,000 to allow geometry to settle. All models are trained on an NVIDIA RTX 5000 Ada Generation GPU. For image-space metrics, ray-intersections of a ground-truth boxsat mesh were computed to form a segmentation mask over the

primary boxsat to evaluate overall intra-model performance across all sun angles, and a segmentation mask over the secondary boxsat is computed to evaluate inter-model shadowing fidelity on a select sun angle. The models are evaluated on a dataset of 10,000 Fibonacci sphere-sampled images over 100 sun angles for the all-sun-angle case, and a set of 1,000 Fibonacci sphere-sampled images with sun angle $(45°, 35°)$ for the single-sun case.

Configuration	No Shadow Method	Ours: Shadow Mapping	SoTA: Shadow Splatting [35]	Ours: Shadow Splatting
Shadowing Method	None	Shadow Mapping	Shadow Splatting	Shadow Splatting
Per-Gaussian Feature Dim.	128	128	72	128
Encoding Type	SH	SH	PE	SH
Shadow Refinement Network Size	N/A	3×64	3×32	3×64
Shadow Refine Inputs	N/A	Shadow value; Gaussian features; SH sun direction; light-space means; receiver-bias information	Shadow value; Gaussian features; PE sun direction; PE world-space means	Shadow value; Gaussian features; SH sun direction; light-space means; receiver-bias information
Receiver Bias	No	Yes	No	Yes

Table 4.3: Structural comparison of the four shadowing method configurations.

Table 4.3 gives the notable configuration differences for each type of model evaluated. This thesis adapts the structural implementation from [35] to serve as a literature-inspired shadow-splatting baseline within the proposed training and rendering pipeline. Tables 4.4 and 4.5 compare the multi-sun-angle performance of the four trained models.

Method	MAE (↓)	RMSE (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
No Shadow Method	0.0085 ± 0.0091	0.0249 ± 0.0222	35.1898 ± 8.0153	0.9811 ± 0.0168	0.0502 ± 0.0367
Ours: Shadow Mapping	$\mathbf{0.0071 \pm 0.0078}$	$\mathbf{0.0207 \pm 0.0174}$	$\mathbf{36.3851 \pm 7.4636}$	$\mathbf{0.9833 \pm 0.0140}$	$\mathbf{0.0451 \pm 0.0351}$
SoTA: Shadow Splatting [35]	0.0087 ± 0.0113	0.0236 ± 0.0226	35.8156 ± 8.1478	0.9823 ± 0.0156	0.0472 ± 0.0360
Ours: Shadow Splatting	0.0074 ± 0.0082	0.0212 ± 0.0186	36.2922 ± 7.4749	0.9830 ± 0.0146	0.0463 ± 0.0353

Table 4.4: Mean intra-model photometric performance of shadowing methods over all sun angles.

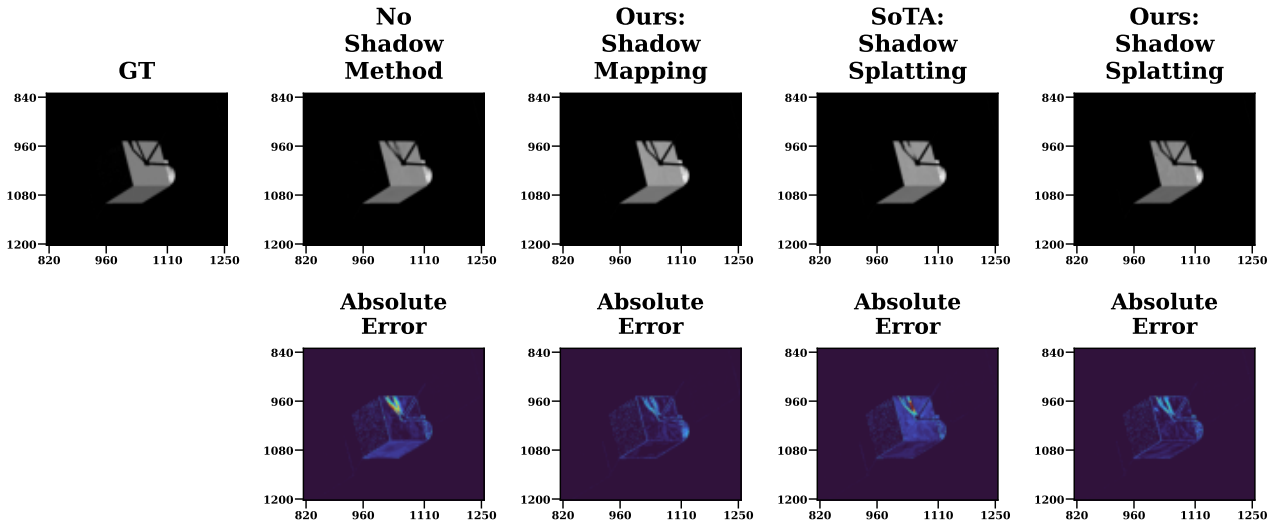
Inspecting the results, it is clear that both shadow mapping and shadow splatting improve global photometric and shadow-specific fidelity. In particular, this thesis' implementation of shadow mapping excels across the board, sweeping both the global photometric and shadow-specific metric categories.

Method	BER (↓)	Shadow Bias (∼ 0)	Boundary MAE (↓)	Sharpness Ratio (∼ 1)
No Shadow Method	0.0160 ± 0.0358	0.0024 ± 0.0071	0.0216 ± 0.0188	0.6808 ± 0.7360
Ours: Shadow Mapping	$\mathbf{0.0118 \pm 0.0310}$	$\mathbf{7.54e-04 \pm 0.0019}$	$\mathbf{0.0184 \pm 0.0142}$	0.7130 ± 0.6437
SoTA: Shadow Splatting [35]	0.0123 ± 0.0318	$7.89e-04 \pm 0.0024$	0.0195 ± 0.0161	0.6436 ± 0.3250
Ours: Shadow Splatting	0.0124 ± 0.0306	0.0015 ± 0.0025	0.0192 ± 0.0157	$\mathbf{0.7347 \pm 0.6774}$

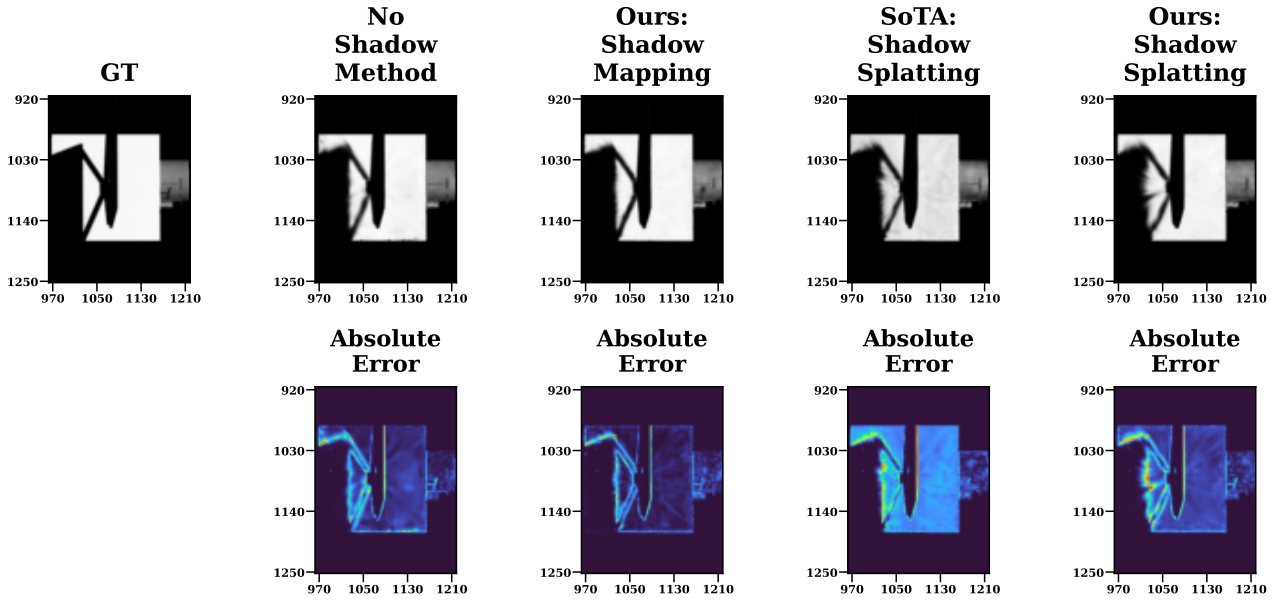
Table 4.5: Mean intra-model shadow-specific performance of shadowing methods over all sun angles.

While shadow splatting's accumulation of occlusion affords it the ability to represent a broader range of shadowing, it tends to result in lighter overall shadows and more gradual shadow onset than shadow mapping. In contrast, shadow mapping's "shadowed-or-not" mechanism is able to more accurately capture the effects of the intense directional light of the sun, producing deeper and sharper shadows. Seen in Figure 4.9, both shadow splatting-based methods tend to present slightly darker illuminated faces, raising error uniformly. Compared to the "SoTA: Shadow Splatting [35]" model, the "Ours: Shadow Mapping" and "Ours: Shadow Splatting" models predict sharper and better-defined shadows, as evidenced by their Boundary MAE and Sharpness Ratios. They are photometrically stronger, indicating their superior image-space reconstructions, and either competitive with or exceeding "SoTA: Shadow Splatting [35]" in shadow specific metrics. Structurally, the larger per-Gaussian feature vectors, wider shadow refinement network, and more illumination-centered geometric conditioning may help the proposed configurations learn more consistent visibility corrections across many sun directions.

Tables 4.6 and 4.7 give the results of each method when evaluated using a sun azimuth and elevation of $(45°, 35°)$. For this evaluation, two individual boxsat models of each kind are composited in a shared scene using the multi-model framework presented in Chapter 3, replicating the relative poses seen in the original dataset (Box 1: $[0m, 0m, 0m]$, Box 2: $[-3m, -3m, -3m]$). Here, the secondary boxsat is near-fully occluded by the primary, testing the depth and accuracy of each



(a) Sun angle $(37°, 12°)$.



(b) Sun angle $(175°, 13°)$.

Figure 4.9: Qualitative performance of the four shadowing methods on two selected evaluation frames from the all-sun-angle case.

method's shadows.

As expected, the "No Shadow Method" model is unable to cast shadows from the primary boxsat onto the secondary boxsat, resulting in substantially worse global performance. In contrast, the shadow mapping and shadow splatting models introduce explicit light-space visibility terms, allowing

Method	MAE (↓)	RMSE (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
No Shadow Method	0.0398 ± 0.0320	0.0694 ± 0.0418	26.3320 ± 10.0901	0.8507 ± 0.1203	0.2018 ± 0.1053
Ours: Shadow Mapping	$\mathbf{0.0015 \pm 0.0015}$	0.0139 ± 0.0136	40.5382 ± 8.5192	0.9809 ± 0.0139	0.1468 ± 0.0834
SoTA: Shadow Splatting [35]	0.0017 ± 0.0017	0.0131 ± 0.0134	41.2144 ± 8.5579	0.9857 ± 0.0106	0.1217 ± 0.0708
Ours: Shadow Splatting	$\mathbf{0.0015 \pm 0.0015}$	$\mathbf{0.0129 \pm 0.0135}$	$\mathbf{41.3919 \pm 8.4955}$	$\mathbf{0.9866 \pm 0.0094}$	$\mathbf{0.1147 \pm 0.0679}$

Table 4.6: Mean inter-model photometric performance of shadowing methods with sun angle $(45°, 35°)$.

Method	BER (↓)	Shadow Bias (~ 0)	Boundary MAE (↓)	Sharpness Ratio (~ 1)
No Shadow Method	0.1505 ± 0.1489	0.0392 ± 0.0319	0.0351 ± 0.0245	1.6572 ± 1.7898
Ours: Shadow Mapping	$\mathbf{0.0030 \pm 0.0028}$	$-7.02e-04 \pm 6.03e-04$	0.0156 ± 0.0150	0.0254 ± 0.1111
SoTA: Shadow Splatting [35]	$\mathbf{0.0030 \pm 0.0028}$	$\mathbf{-9.49e-05 \pm 5.96e-04}$	0.0150 ± 0.0152	0.2027 ± 0.2674
Ours: Shadow Splatting	$\mathbf{0.0030 \pm 0.0028}$	$-2.14e-04 \pm 4.36e-04$	$\mathbf{0.0147 \pm 0.0149}$	$\mathbf{0.2436 \pm 0.3319}$

Table 4.7: Mean inter-model shadow-specific performance of shadowing methods with sun angle $(45°, 35°)$.

the composed models to interact through inter-model shadowing. As shown in Tables 4.6 and 4.7, all explicit-shadow methods perform competitively, with "Ours: Shadow Splatting" achieving the strongest overall photometric scores.

Inspecting the selected frame in Figure 4.10, the "SoTA: Shadow Splatting [35]" and "Ours: Shadow Splatting" models produce visually similar predictions, though the "SoTA: Shadow Splatting [35]" shadows appear marginally less uniform and recede further up the boxsat face. Because the secondary boxsat is positioned nearly directly behind the primary along the sun direction, the shadow mapping model classifies it as fully occluded. This produces a pitch-black shadow, matching that of the ground-truth most completely, but it also removes the faint edge illumination that the shadow splatting-based methods preserve more naturally through accumulated occlusion.

Shadow mapping and shadow splatting have their respective advantages and drawbacks, shown in Figure 4.11. Under more direct illumination, shadow mapping is able to more completely represent details such as the distinct lines of the solar panel struts, whereas shadow splatting produces fuzzier edges. Conversely, in cases of less direct lighting, shadow splatting presents a more uniformly shaded boxsat face, avoiding the shadow acne present in the shadow mapping image. Although the shadow

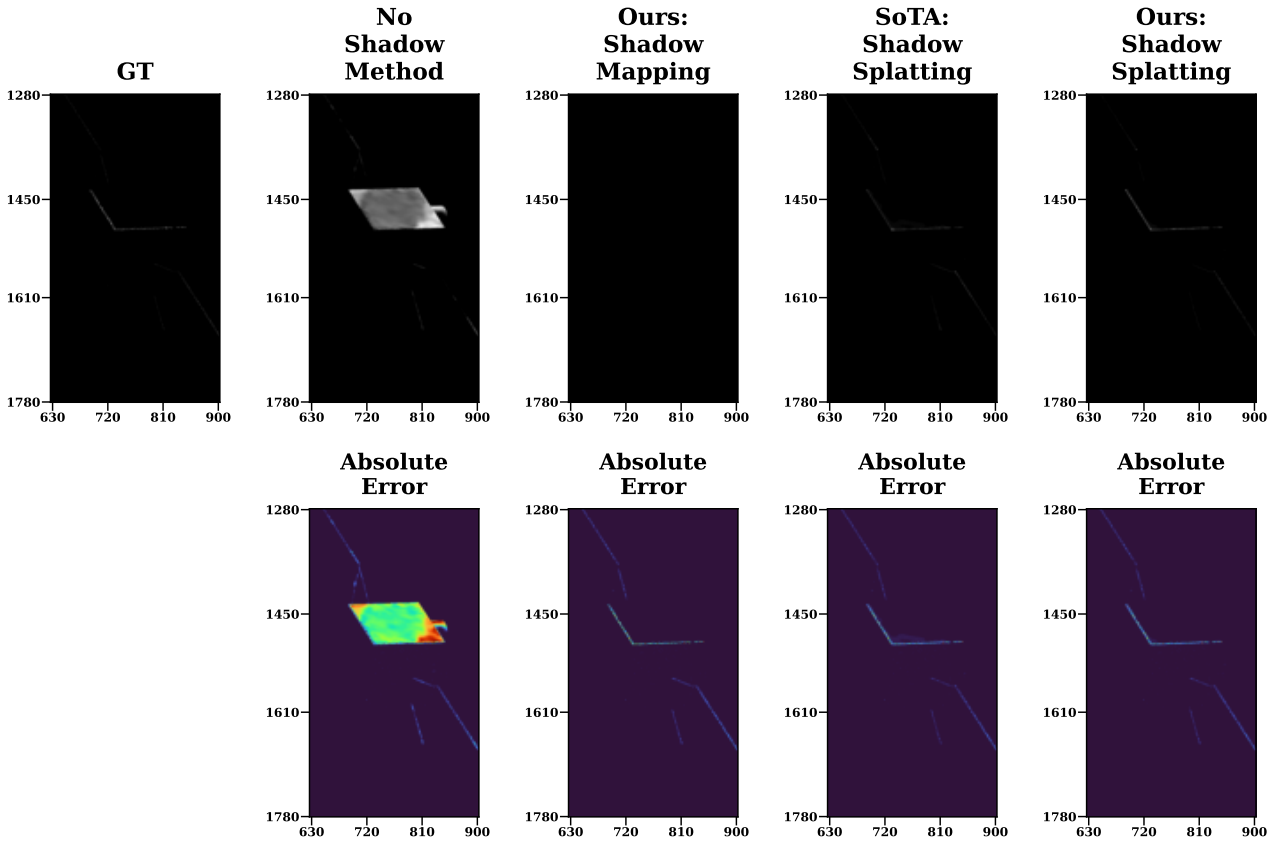


Figure 4.10: Qualitative performance of the four shadowing methods on a selected frame from the sun angle $(45°, 35°)$ case.

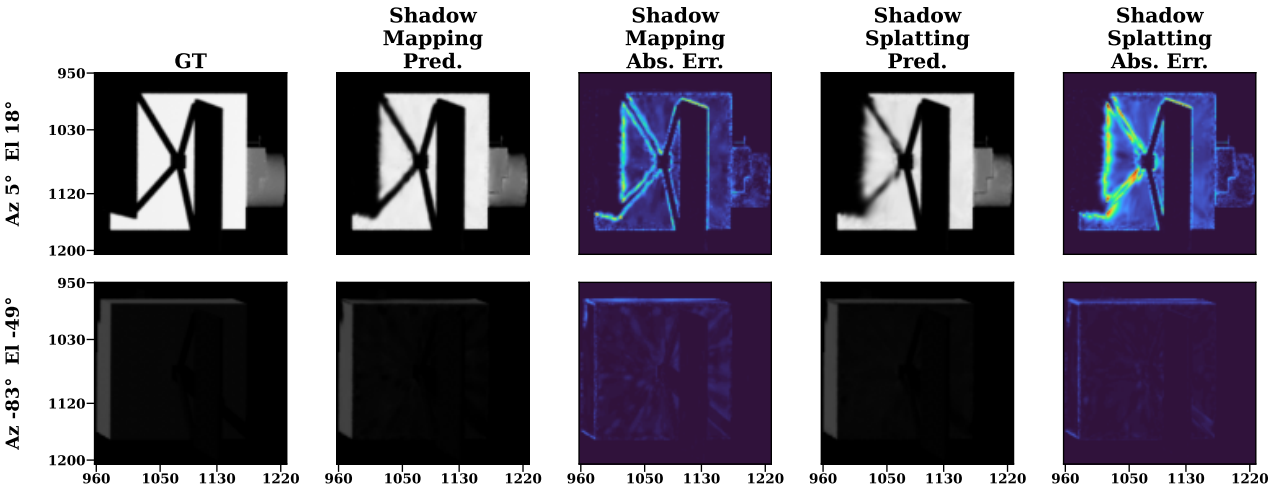


Figure 4.11: Shadow mapping vs. shadow splatting advantage case comparison.

refinement network is capable of correcting some of these artifacts, these grazing angles are difficult to handle using either shadowing method, though the binary nature of shadow mapping makes them especially apparent. Future work focused on exploring 2D Gaussian Splatting [55] and hybrid ray-Gaussian rendering methods will attempt to reduce these effects globally by encouraging surface uniformity and using more physically consistent light-tracing paradigms.

Due to their respective strengths, both shadow mapping and shadow splatting remain available for use in the rendering pipeline. In future work with modeling multiple light sources (Earth albedo in particular), shadow splatting's ability to produce a broader range of shadowing effects may make it more suitable for a wider range of in-orbit modeling. However, for the purposes of this thesis, shadow mapping is used in all subsequent evaluations where applicable.

Chapter 5: Informed Training for Space-based 3D Gaussian Splatting Models

Space-based imagery presents several challenges for 3D Gaussian Splatting training. Strong shadows, sparse viewpoints, dark backgrounds, and limited texture can make photometric supervision insufficient to fully constrain where Gaussians should be created, preserved, or removed. In many cases, the default 3DGS culling and densification strategy is capable of producing strong image-space reconstructions, particularly when camera coverage is good and the object is well observed. However, when the goal is to recover a geometrically plausible shape model, photometric quality alone may not be enough; Gaussians can remain in unsupported regions, and low-opacity but structurally useful Gaussians can be removed.

This chapter therefore investigates whether additional sources of training-time evidence can guide Gaussian allocation in space-based 3DGS models. The mesh-aware strategy (Section 5.2.1) uses an explicit geometric prior when approximate object geometry is available, while the support-aware strategy (Section 5.2.2) uses accumulated multi-view evidence as a weaker image-derived prior when no mesh is provided. Designed as targeted regularization mechanisms for cases where additional spatial or support information can help guide Gaussian allocation, these strategies aim to improve both 2D image and 3D spatial reconstruction of space scenes. The chapter also details a warm-start procedure for improving random initialization by pulling unsupported Gaussians toward regions with foreground evidence.

5.1 Off-the-Shelf Training Tools and Modifications

A particularly useful training tool that ships with *Nerfstudio* is the ability to initialize 3D Gaussian Splatting from a point cloud. Rather than spawning Gaussians randomly within a bounding volume, the initial Gaussian positions can be placed near an approximate scene structure. This

provides a simple but effective spatial prior: before any photometric optimization occurs, the model begins with Gaussians concentrated near regions likely to contain the object.

For spacecraft reconstruction, this can be especially helpful in cases where the object occupies a small portion of a large scene volume. A random initialization may require Gaussians to move long distances across empty space before they contribute meaningfully to the image, while a point cloud initialization begins from a more localized approximation of the target. This is contingent on the point cloud being relatively localized to the true object to maximize its effectiveness, though even a rough estimate of the object location or a sample from a smaller bounding volume is much preferred to a random initialization.

5.2 3DGS Culling and Densification Strategies

The default 3DGS refinement strategy follows the adaptive density-control procedure introduced by Kerbl et al. [2], in which optimization of Gaussian parameters is interleaved with periodic pruning and densification. During training, Gaussians with large image-plane position gradients are treated as evidence of under-reconstructed regions. Small Gaussians that satisfy this condition are duplicated to increase local sampling density, while larger Gaussians are split into smaller components to better distribute their support over the scene. Conversely, Gaussians with sufficiently low opacity, excessive scale, or excessive screen-space footprint are pruned to remove weak or overly massive primitives. Opacities are also periodically reset to encourage continued competition among Gaussians and prevent early opacity saturation. Together, these operations allow the representation to grow where image-space supervision indicates missing detail while removing Gaussians that contribute little or become spatially unstable. Table 5.1 lists the default strategy parameters used throughout the

following experiments, unless stated otherwise.

Parameter	Value
Opacity culling threshold	0.05
Scale culling threshold	0.05
Opacity reset interval	Every 30 refinement steps
Screen-size culling threshold	0.05
Screen-size splitting threshold	0.01
Refinement interval	100 steps
Refinement pause after opacity reset	1000 steps
Maximum Number of Gaussians	250,000

Table 5.1: Default culling and densification strategy parameters.

5.2.1 Mesh-Aware Strategy

Vanilla 3D Gaussian Splatting often produces high-quality renderings from training and nearby novel views, but its learned Gaussian distribution is not guaranteed to correspond to a clean physical surface. This distinction is important for spacecraft reconstruction, where the desired output is often not only an image renderer but also a spatially meaningful shape representation. In sparse-view or high-contrast settings, Gaussians may remain in unsupported regions, or be removed from low-opacity regions that still correspond to valid surface structure.

The mesh-aware strategy introduces an explicit geometric prior into this refinement process. When an approximate spacecraft mesh is available, a signed distance function can be derived from the mesh and queried during training to classify Gaussians relative to the object surface. This information allows culling and densification decisions to account for whether a Gaussian lies near the expected object support, rather than relying only on opacity, projected size, or image-space gradient statistics.

This strategy serves best as a geometry-prior regularization mechanism — if the available mesh is accurate and well-aligned, it can help preserve surface Gaussians and discourage unsupported floating or interior structure. If the mesh is coarse, misaligned, or unavailable, the same prior may

be less useful or even overly restrictive. It is therefore most appropriate in settings where approximate geometry is known and the reconstruction goal emphasizes physically consistent geometry across novel views in addition to global photometric accuracy.

5.2.1.1 Signed Distance Function Shielding and Pruning

Given a Gaussian with center $\mu_i \in \mathbb{R}^3$, its signed distance to the mesh surface is defined as:

$$d_i = \mathrm{SDF}(\boldsymbol{\mu}_i), \tag{5.1}$$

where $d_i < 0$ indicates that the Gaussian lies inside the mesh, $d_i > 0$ indicates that it lies outside, and values near zero correspond to the surface. Using two margin parameters, an exterior margin τ_{out} and interior margin τ_{in}, Gaussians are partitioned into these three regions:

$$\mathcal{O} = \{i \mid d_i > \tau_{\text{out}}\}, \quad \mathcal{I} = \{i \mid d_i < -\tau_{\text{in}}\}, \quad \mathcal{B} = \{i \mid -\tau_{\text{in}} \leq d_i \leq \tau_{\text{out}}\} \tag{5.2}$$

where $\mathcal{O}$, $\mathcal{I}$, and $\mathcal{B}$ denote exterior, interior, and boundary-layer Gaussians, respectively.

The Gaussian pruning strategy can then be augmented with a geometric constraint such that Gaussians in the exterior region are removed at a user-specified cadence (often a multiple of the regular refinement rate) to avoid over-aggressive culling of Gaussians that may be in transit through the scene:

$$m_i^{\text{out}} = \begin{cases} 1, & d_i > \tau_{\text{out}} \\ 0, & \text{otherwise} \end{cases}. \tag{5.3}$$

On steps where exterior region culling is set to occur, this heuristic is combined with the standard

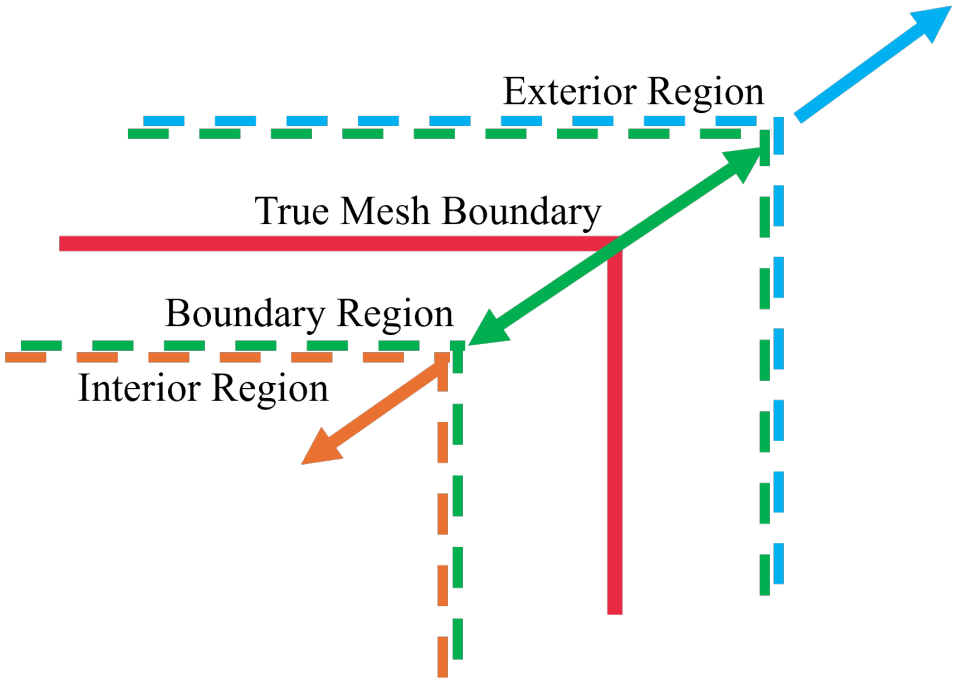


Figure 5.1: Illustration of the Mesh-Aware strategy classification regions.

opacity and scale/screen-size-based pruning thresholds:

$$m_i^{\text{weak}} = \begin{cases} 1, & \alpha_i < \tau_\alpha \\ 0, & \text{otherwise} \end{cases} \qquad m_i^{\text{big}} = \begin{cases} 1, & s_i > \tau_{\text{scale}} \vee \rho_i > \tau_{\text{screen}} \\ 0, & \text{otherwise} \end{cases}. \tag{5.4}$$

Here, s_i denotes a representative world-space Gaussian scale, implemented as the maximum axis scale, and ρ_i denotes the maximum projected screen-space radius (3σ of the largest projected axis) of Gaussian i. With these, the final pruning decision becomes:

$$m_i^{\text{prune}} = m_i^{\text{out}} \vee \left(m_i^{\text{weak}} \wedge \neg m_i^{\text{protect}}\right) \vee \left(m_i^{\text{big}}\right), \tag{5.5}$$

where interior and boundary Gaussians may be protected from weak pruning. This shielding mechanism ensures that Gaussians in the scene remain concentrated within geometrically valid

regions and prevents drift into free space.

5.2.1.2 Mesh Surface Densification

Densification in 3D Gaussian splatting is traditionally driven by image-plane position gradient magnitude, where Gaussians that move larger-than-normal distances through a scene are seen as attempting to reconstruct an under-represented region and are shifted from their previous locations. In the mesh-aware formulation, the threshold for densification can be reduced for Gaussians in the boundary region:

$$\tau_g^{(i)} = \begin{cases} \lambda_b \tau_g, & i \in \mathcal{B} \\ \tau_g, & \text{otherwise} \end{cases} \tag{5.6}$$

where $0 < \lambda_b \leq 1$ is a user-defined scaling factor that encourages densification near the surface. Gaussians that satisfy the densification condition are refined using size-aware operations. Small Gaussians are duplicated to increase local sampling density and larger Gaussians are split into multiple components to better capture local structure. By preferentially densifying Gaussians on the surface of the model, greater geometric fidelity is encouraged in regions where having a solid face is critical.

5.2.1.3 Minimum Gaussian-based Densification

While it is in the nature of the 3DGS optimization process to most accurately reconstruct the scene based on information synthesized from the training images, models may occasionally finish training with see-through or sparsely represented bodies when closely inspected as a result of aggressive culling. To prevent excessive pruning from degrading the representation, a minimum Gaussian count N_{min} is optionally enforced. After a pruning call, the number of Gaussians remaining

is given by:

$$N_{\text{survive}} = N - \sum_i m_i^{\text{prune}} \tag{5.7}$$

If the number of surviving Gaussians falls below the threshold, additional Gaussians are introduced via duplication:

$$N_{\text{add}} = \max(0, N_{\min} - N_{\text{survive}}). \tag{5.8}$$

New Gaussians are generated by duplicating existing Gaussians selected from geometrically valid regions. Selection is biased toward Gaussians that are more reliable, favoring those that are located inside the mesh, near the surface, highly opaque, or have persisted for longer periods of training time. This selection process can be expressed as a scoring function:

$$s_i^{\text{bf}} = \alpha_i + \phi(-d_i) + 0.25\,\tilde{c}_i + 0.5\,\mathbf{1}_{i\in\mathcal{I}} + 0.1\,\mathbf{1}_{i\in\mathcal{B}}. \tag{5.9}$$

where α_i is the Gaussian opacity, $\phi(-d_i) = \max(-d_i/d_{max}, 0)$ is a normalized interior depth that emphasizes Gaussians near the surface of the mesh, $\tilde{c}_i$ is a normalized contribution count reflecting how many training views a Gaussian is seen in, and $\mathbf{1}_{i\in\mathcal{I}}$ and $\mathbf{1}_{i\in\mathcal{B}}$ are indicator functions for interior and boundary regions. By enforcing a minimum representation size and repopulating from geometrically valid regions, this mechanism reduces the detail degradation that comes from over-pruning. Table 5.2 details the mesh-aware strategy parameters used in this thesis' experiments.

Parameter	Value
Mesh-based culling interval	2500 steps
Exterior pruning margin	1×10^{-4}
Interior protection margin	0.05
Boundary-region densification scale factor	0.3
Minimum Gaussian count	20,000

Table 5.2: Mesh-aware refinement parameters.

5.2.2 Support-Aware Strategy

While the mesh-aware training strategy uses an explicit geometric prior to localize Gaussians to the spacecraft object, such a prior is not always available. In training scenarios without mesh information, the model must infer which Gaussians correspond to persistent scene structure and which correspond to transient, weakly supported, or floating artifacts. This is particularly relevant for relightable models, where some regions may appear low-opacity or weakly supervised under one illumination condition but become important under another.

To address this case, a support-aware densification and culling strategy is introduced. Rather than culling Gaussians solely on opacity at a given refinement step, the strategy accumulates persistent evidence that a Gaussian is "useful" across training. This evidence is derived from multi-view visibility, image-plane gradient activity, and screen-space footprint. The motivation is that opacity alone is not always a reliable indicator of geometric or relighting importance: a Gaussian may be weakly visible in a subset of views or sun angles while still contributing to a stable object representation.

5.2.2.1 Persistent Multi-View Support

For each Gaussian i, three persistent exponential moving averages (EMAs) are maintained: a visibility/count value c_i, an image-plane gradient value g_i, and a screen-space radius support value r_i. These quantities are updated at each refinement step using an exponential moving average to retain long-term information on useful Gaussians while still allowing unsupported Gaussians to decay over

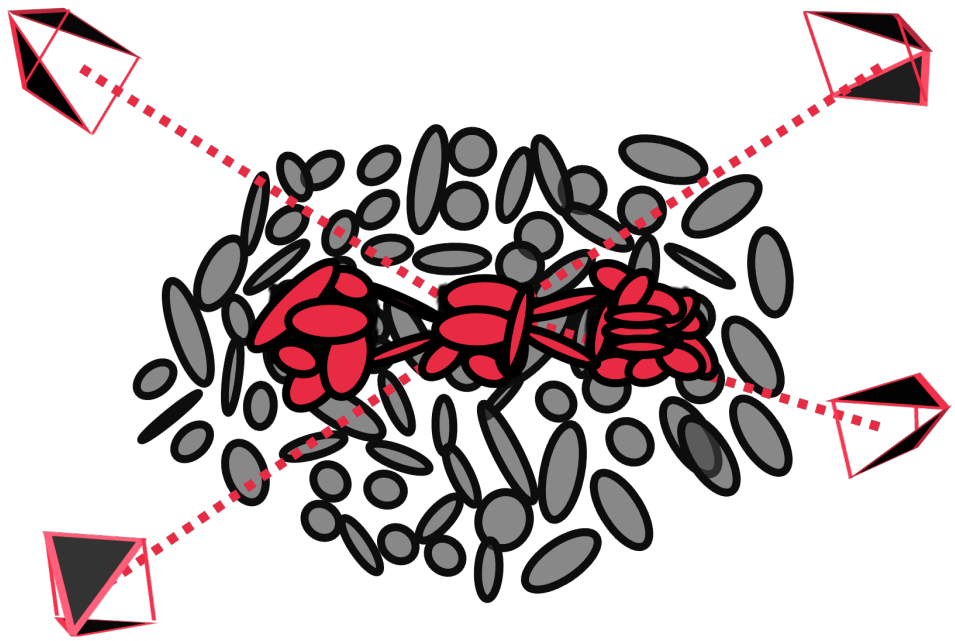


Figure 5.2: Illustration of the Support-Aware strategy's multi-view support.

time. Let $\beta \in [0, 1)$ denote the support decay rate. For a given support metric x_i, the update logic is:

$$x_i^{(t)} = \beta x_i^{(t-1)} + (1 - \beta)\Delta x_i^{(t)}, \tag{5.10}$$

where $x_i^{(t)}$ is the persistent support value for Gaussian i at training step t, and $\Delta x_i^{(t)}$ is the observed support contribution for Gaussian i at training step t. For visibility/count, $\Delta c_i^{(t)}$ records whether the Gaussian was visible. For gradient support, $\Delta g_i^{(t)}$ records the magnitude of the Gaussian's image-plane position gradient. For radius support, $\Delta r_i^{(t)}$ records the Gaussian's normalized screen-space footprint.

Because these support channels have different scales, each channel is robustly normalized before being combined. For a given support statistic x_i, the normalized value is:

$$\widehat{x}_i = \text{clip}\left(\frac{x_i}{Q_{0.95}(\{x_j\}_{j=1}^N) + \epsilon}, 0, 1\right). \tag{5.11}$$

Here, $Q_{0.95}$ is the 95th percentile of that statistic over all N Gaussians, and ϵ is a small numerical stability constant. This prevents massive outliers from dominating the score and skewing the normalization. The final support score S_i is computed as a weighted average of the normalized support channels:

$$S_i = \frac{w_c \widehat{c}_i + w_g \widehat{g}_i + w_r \widehat{r}_i}{w_c + w_g + w_r}, \tag{5.12}$$

where w_c, w_g and w_r are weights controlling each metric's relative importance. A larger S_i indicates that Gaussian i has persistent multi-view evidence of contributing to the reconstruction.

5.2.2.2 Support-Aware Weak Pruning

In the default 3D Gaussian splatting refinement strategy, weak pruning is driven by opacity. A Gaussian is removed if its opacity α_i falls below a pruning threshold τ_i:

$$m_i^{\text{weak}} = \begin{cases} 1, & \alpha_i < \tau_\alpha, \\ 0, & \text{otherwise.} \end{cases} \tag{5.13}$$

This removes low-opacity Gaussians, but it can also remove Gaussians that are temporarily low-opacity despite having persistent multi-view support. The support-aware strategy modifies this rule by protecting Gaussians whose support score is above a threshold τ_s, making the support-aware pruning mask:

$$m_i^{\text{weak-sa}} = m_i^{\text{weak}} \wedge \neg m_i^{\text{support}}. \tag{5.14}$$

In this formulation, a Gaussian may only be weak-pruned if it has both low-opacity and insufficient support. A small warmup period is used before support-aware protection is turned on to allow a

meaningful support distribution to develop. The full pruning mask combines the support-aware condition with the other pruning criteria used by the base strategy:

$$m_i^{\text{prune}} = m_i^{\text{out}} \vee m_i^{\text{weak-sa}} \vee m_i^{\text{big}}. \tag{5.15}$$

5.2.2.3 Support-Eased Densification

As mentioned in subsection 5.2.1.2, densification in 3D Gaussian splatting is typically triggered by large image-plane position gradients, indicating that a Gaussian is being pulled strongly towards an underrepresented region of the scene. The support-aware strategy preserves this behavior, and optionally reduces the densification threshold for Gaussians that exceed a user-specified support score:

$$\tau_g^{(i)} = \begin{cases} \lambda_S \tau_g, & S_i \geq \tau_D, \\ \tau_g, & \text{otherwise.} \end{cases} \tag{5.16}$$

Here, τ_g is the default image-plane gradient threshold for densification, τ_D is the support score threshold for easing densification, and $0 < \lambda_S \leq 1$ controls how much the densification threshold is reduced for highly supported Gaussians. A Gaussian is then selected for growth if its average image-plane gradient $\bar{g}_i$ exceeds this adjusted threshold:

$$m_i^{\text{grow}} = \begin{cases} 1, & \bar{g}_i > \tau_g^{(i)}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.17}$$

As in the base strategy, Gaussians are densified using size-aware operations, duplicating smaller Gaussians to increase local sampling density and splitting large Gaussians to distribute their support across multiple regions.

5.2.2.4 Support-Localized Minimum Gaussian Backfilling

Aggressive pruning can occasionally reduce the scene's Gaussian count enough to create sparse or see-through reconstructions. In the mesh-aware strategy, this issue is addressed by duplicating reliable Gaussians from geometrically valid regions of the mesh. In the support-aware strategy, an analogous selection scheme picks backfill parents from a localized object core inferred from the Gaussian cloud itself.

Let N_{min} be the optional minimum Gaussian count, and let N be the number of Gaussians before pruning. Following the mesh-aware strategy's formulation in Equations 5.7 and 5.8, $N_{survive}$ and N_{add} are derived from the final pruning mask, m_i^{prune}. Then, a candidate pool is formed that takes into account support score, opacity, spatial extent, and proximity to an estimated supported object core. First, the set of sufficiently supported and opaque Gaussians is defined as:

$$\mathcal{P} = \{i \mid S_i \geq \tau_{S,\text{bf}} \wedge \alpha_i \geq \tau_{\alpha,\text{bf}}\}. \tag{5.18}$$

where $\tau_{S,\text{bf}}$ and $\tau_{\alpha,\text{bf}}$ are the support and opacity thresholds for backfill eligibility. The center of the supported cloud is computed as a weighted average of Gaussian centers:

$$\boldsymbol{\mu}_{\mathcal{P}} = \frac{\sum_{i\in\mathcal{P}} S_i \alpha_i \boldsymbol{\mu}_i}{\sum_{i\in\mathcal{P}} S_i \alpha_i}. \tag{5.19}$$

This gives greater influence to Gaussians that are both strong and persistently supported. A robust supported-cloud radius is then estimated using a distance quantile:

$$R_{\mathcal{P}} = Q_q\left(\{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_{\mathcal{P}}\|_2\}_{i\in\mathcal{P}}\right), \tag{5.20}$$

where q is a chosen quantile (0.80 in this thesis' implementation). This gives a mesh-free estimate of the spatial extent of the central-supported object region. The final backfill candidate set is:

$$\mathcal{C} = \{i \mid m_i^{\text{prune}} = 0 \wedge S_i \geq \tau_{S,\text{bf}} \wedge \alpha_i \geq \tau_{\alpha,\text{bf}} \wedge \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_{\mathcal{P}}\|_2 \leq \gamma R_{\mathcal{P}}\}, \tag{5.21}$$

where γ is a scale factor applied to the robust supported-cloud radius. In the reported experiments, $\gamma = 1.25$, allowing the parent pool to extend slightly beyond the estimated supported object core while still excluding Gaussians that lie far from the main supported structure. This produces a mesh-free localization prior for backfilling: parents must be sufficiently supported, sufficiently opaque, survive the current pruning step, and lie near the inferred object-support region.

Parents are then chosen from $\mathcal{C}$ according to a weighted backfill score:

$$s_i^{\text{bf}} = 1.5S_i + 0.75\alpha_i + 0.25\widehat{c}_i + 0.25\widehat{g}_i - 0.5d_i^{\text{core}}. \tag{5.22}$$

Here, d_i^{core} is the normalized distance from Gaussian i to the supported-cloud center $\boldsymbol{\mu}_{\mathcal{P}}$, computed relative to the scaled cloud radius $\gamma R_{\mathcal{P}}$ and clamped to the interval $[0, 1]$. The positive terms encourage high support, opacity, persistent visibility, and persistent gradient activity, while d_i^{core} penalizes Gaussians further from the supported-cloud center. Thus, even within the valid candidate set, Gaussians closer to the inferred object core are preferred for duplication. The highest-scoring

candidates are duplicated first, and additional parents may be sampled with replacement if more Gaussians than unique valid candidates are required. In contrast with the mesh-aware strategy, this backfilling operation is closer to a soft safeguard rather than a hard guarantee. If no sufficiently supported, opaque, and spatially valid parents exist, the strategy does not create arbitrary Gaussians, retaining the intent of the minimum-count mechanism while avoiding maintenance of floating artifacts. Table 5.3 provides the full list of support-aware strategy parameters used in this thesis' implementation.

Parameter	Value
Support moving-average decay	0.99
Visibility count support weight	1.0
Screen-space gradient support weight	0.75
Screen-radius support weight	0.25
Support score protection threshold	0.55
Support warmup duration	500 steps
Densification support threshold	0.95
Support-based densification scale factor	0.3
Minimum Gaussian count	20,000
Minimum Gaussian backfill start step	1000 steps
Minimum backfill support score	0.85
Minimum backfill opacity	0.3
Supported-cloud radius quantile	0.80
Supported-cloud radius scale	1.25
Minimum supported-cloud size	16 Gaussians

Table 5.3: Support-aware refinement strategy parameters.

5.2.3 Mesh-Aware and Support-Aware Performance

Tables 5.4 and 5.5 display the photometric and geometric performance of boxsat models using the default, mesh-aware, and support-aware culling and densification strategies. As with the shadowing tests, all models are trained on datasets of 100,000 images with 100 sun angles (1000 images/sun angle), and each model was trained for 200,000 iterations, with refinement disabled at step 150,000. Evaluation is over the primary boxsat, using a dataset of 10,000 Fibonacci sphere-sampled images

over 100 sun angles. For the 3D metrics, a TSDF mesh is first extracted from each trained 3DGS model. To avoid sampling spurious interior TSDF surfaces, only the exterior-visible mesh surface is retained before surface point sampling. Chamfer and 99th-percentile Hausdorff distances are then computed between equal-sized surface samples from the predicted and ground-truth meshes, while IoU is computed from the corresponding filled-mesh volumetric occupancy grids.

Method	MAE (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
Default	0.0071	36.3851	0.9833	0.0451
Mesh-aware	**0.0068**	36.8452	0.9840	0.0438
Support-aware	0.0068	**36.8906**	**0.9840**	**0.0432**

Table 5.4: Mean photometric performance of training strategies over all sun angles.

Method	Chamfer (↓)	Hausdorff (↓)	IoU (↑)
Default	$\mathbf{1.0391 \times 10^{-3}}$	0.0500	**0.8643**
Mesh-aware	1.0941×10^{-3}	**0.0433**	0.8611
Support-aware	1.1112×10^{-3}	0.0472	0.8627

Table 5.5: Mean geometric performance of training strategies over all sun angles.

Overall, the mesh-aware and support-aware strategies produce modest but measurable improvements over the default model. The support-aware strategy achieves the strongest photometric reconstruction performance, ranking first in PSNR, SSIM, and LPIPS, while matching the mesh-aware model in MAE. This suggests that preserving Gaussians with multi-view support can provide useful additional representational capacity for matching the rendered image distribution. The mesh-aware strategy also improves photometric performance relative to the default model, though it performs slightly below the support-aware strategy in most image-space metrics.

The geometric results are more mixed, though both alternative strategies remain competitive with the default model. The mesh-aware strategy achieves the best 99th-percentile Hausdorff distance,

suggesting that the mesh prior can reduce larger surface deviations or outlier reconstruction errors. The default model achieves the best Chamfer distance and IoU, indicating slightly stronger average surface agreement and volumetric overlap in this particular evaluation. However, the differences among all three models are small, with the mesh- and support-aware models remaining close to the default across Chamfer distance and IoU. Taken together, these results suggest that the alternative strategies do not uniformly improve extracted geometry, but they also do not significantly degrade geometric reconstruction.

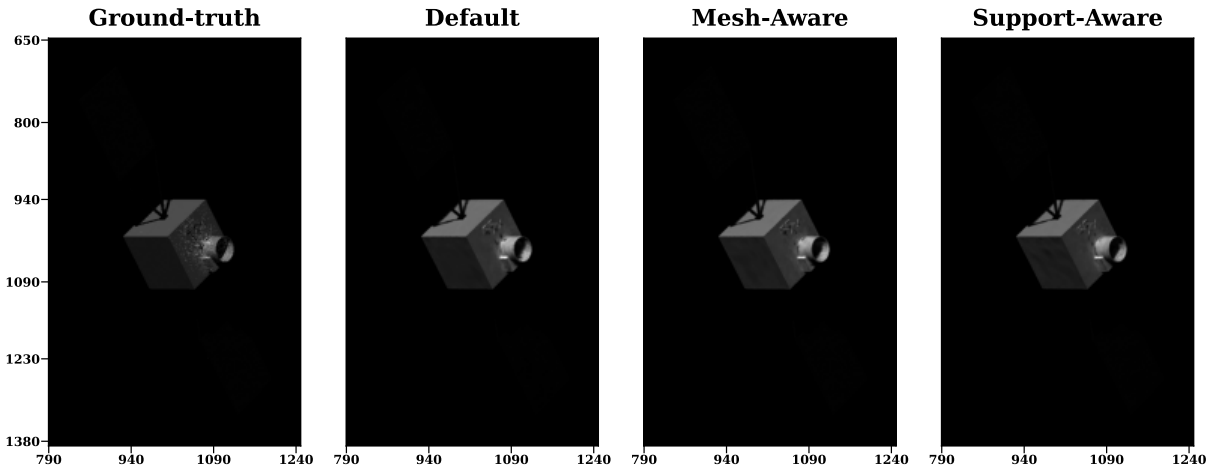


Figure 5.3: Visual comparison of default, mesh-aware, and support-aware strategies.

Looking at a visual comparison of each strategy in Figure 5.3, the results are quite similar, with few visible deviations across methods. Inspecting a depth snapshot for each strategy in Figure 5.4, the differences are minute. As the mesh-aware strategy occasionally prunes Gaussians outside of the mesh extent, it is able to capture concavities like the divot of the boxsat's camera barrel more completely; however, with a well-constrained scene, it does not provide vast improvements in depth consistency over the default strategy.

Both alternative strategies contain several times more Gaussians than the default model, and require more storage space. Table 5.6 lists the Gaussian count, file sizes, and rendering times of the

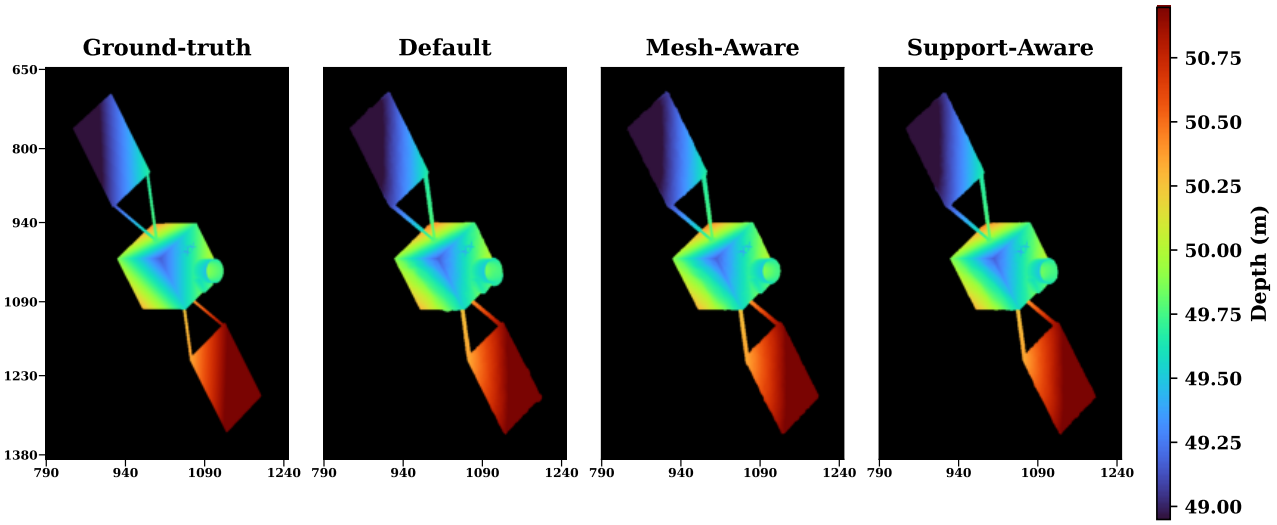


Figure 5.4: Depth comparison of default, mesh-aware, and support-aware strategies.

three models. Since computational efficiency is not a focus of this work, use of additional Gaussians to achieve better photometric performance imposes few restrictions, though it is a potential limiting factor depending on the hardware. Considering that all three models remain above real-time rendering rates, the larger Gaussian count does not substantially limit practical use in this evaluation.

Model	Num. Gaussians	File Size (MB)	Render Wall Clock Time (s)	FPS
Default	31,622	66.26	0.009270	108
Mesh-Aware	101,958	189.69	0.012997	77
Support-Aware	165,495	301.18	0.014243	70

Table 5.6: Default, Mesh-Aware, and Support-Aware model size differences.

In conclusion, the alternative density control strategies provide moderate, task-dependent improvements rather than providing a uniformly superior reconstruction. The default model remains competitive with far fewer Gaussians, indicating that the standard density control strategy is likely sufficient for this simulated regime. The mesh-aware strategy provides the clearest geometric benefit in worst-case deviation, though it may be more difficult to fully constrain the object geometrically with more Gaussians in the mesh- and support-aware cases. On the other hand, the strong photometric

performance of the alternative strategies suggests that their additional Gaussians may provide useful representational capacity for image reconstruction, even if this does not translate into improved extracted surface geometry. Moreover, the increased Gaussian quantity may be providing relighting-specific benefits, as the model is allotted more geometry to distribute across illumination-dependent appearance changes. As discussed in 4.2, relightable models must encode several illumination-dependent appearances within a fixed spatial representation; with more Gaussians available, the model may have greater freedom to allocate geometry toward view- and lighting-specific appearance variation.

In terms of the scale of the benefits provided by these alternative strategies, they likely do not provide as large a performance increase as a well-designed appearance module does. A sun-conditioned appearance field helps disentangle geometry from illumination by allowing shadowed regions in one lighting condition to be informed by their appearance when illuminated under other sun angles. Nevertheless, to the end of learning higher fidelity shape models in these regimes, it is recommended that future work explore surfel-based representations or the fusing of multiple sensing modalities to enforce both visual *and* depth consistency during training.

5.3 Foreground Attraction Warm-Start

When 3D Gaussian splatting models are initialized from a random bounding-box-based spatial distribution, many Gaussians may begin far from the object of interest. This is especially problematic for spacecraft scenes, where the object may occupy only a small portion of the image and the surrounding background may be largely empty or near-black. During early training, randomly initialized Gaussians can receive weak, noisy, or spatially ambiguous gradients before

the representation has begun to organize around the foreground object. If culling is applied too aggressively during this stage, useful Gaussians may be removed before they migrate toward the object, creating a sparse distribution that the model struggles to rebuild from.

To reduce this instability, a foreground attraction warm-start is introduced with the purpose of guiding the initial Gaussian distribution toward likely object regions before normal refinement behavior dominates. This is done by adding an auxiliary opacity-based loss and a projected 2D distance loss over foreground pixels estimated heuristically from the training image.

Let $I(\mathbf{u})$ denote the ground-truth image at pixel $\mathbf{u}$. For RGB images, the luminance $L(\mathbf{u})$ is computed as:

$$L(\mathbf{u}) = 0.2126 I_R(\mathbf{u}) + 0.7152 I_G(\mathbf{u}) + 0.0722 I_B(\mathbf{u}), \tag{5.23}$$

where $I_R(\mathbf{u}), I_G(\mathbf{u})$, and $I_B(\mathbf{u})$ are the red, green and blue channels of the ground-truth image. For grayscale images, $L(\mathbf{u})$ is simply the single-channel pixel intensity. A foreground mask $M_{fg}(\mathbf{u})$ is then formed by thresholding this luminance:

$$M_{\text{fg}}(\mathbf{u}) = \begin{cases} 1, & L(\mathbf{u}) > \tau_{\text{fg}}, \\ 0, & \text{otherwise}, \end{cases} \tag{5.24}$$

where τ_{fg} is the foreground luminance threshold. Pixels brighter than this threshold are treated as likely foreground, while pixels below are treated as background for the purposes of the warm-start. In practice, this foreground mask is dilated by a small number of pixels to make the attraction region more tolerant of object boundaries, shadowed surfaces, or small registration errors.

Let $A(\mathbf{u})$ be the rendered accumulation, or alpha value, of pixel $\mathbf{u}$. The foreground attraction

loss penalizes foreground pixels that remain transparent:

$$\mathcal{L}_{\text{fg-}\alpha} = \frac{\sum_{\mathbf{u}} M_{\text{fg}}(\mathbf{u}) \left(1 - A(\mathbf{u})\right)^2}{\sum_{\mathbf{u}} M_{\text{fg}}(\mathbf{u})}. \tag{5.25}$$

This term encourages the model to place opacity in image regions that are likely to contain the object. Since the accumulation image depends on the projected locations and opacities of the Gaussians, minimizing this loss encourages the representation to explain likely object pixels early in training. However, since the opacity coverage component is defined over image pixels, a relatively small subset of Gaussians can provide sufficient opacity over the foreground region. Therefore, a per-Gaussian projected position term is added to the warm-start loss, encouraging a broader migration of the randomly initialized Gaussian population toward the true object.

From the same foreground mask, a foreground distance image $D_{\text{fg}}(\mathbf{u})$ is computed, where each pixel stores its distance to the nearest foreground pixel. For efficiency, this distance image is computed on a downsampled foreground mask and then sampled in the projected Gaussian loss. In the experiments reported here, the projected foreground term uses a downsampling factor of 4, which preserves the coarse object-support region while reducing the cost of constructing and storing the distance transform. To avoid large inconsistencies in foreground attraction strength across distances, $D_{\text{fg}}(\mathbf{u})$ is normalized by a fraction of the smaller image dimension:

$$\bar{D}_{\text{fg}}(\mathbf{u}) = \frac{D_{\text{fg}}(\mathbf{u})}{\rho \min(H, W)}. \tag{5.26}$$

In this implementation, $\rho = 0.125$. Let μ_i denote the center of Gaussian i, and let $\pi(\mu_i)$ be its projection into the current camera view. The projected foreground component bilinearly samples

the normalized distance image at each valid projected Gaussian center. Let the valid set $\mathcal{V}$ contain Gaussians whose projected centers lie inside the image bounds and in front of the camera. The projected foreground component is then:

$$\mathcal{L}_{\text{fg-proj}} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \sqrt{\bar{D}_{\text{fg}} \left(\pi(\boldsymbol{\mu}_i) \right)^2}, \tag{5.27}$$

where $\pi(\boldsymbol{\mu}_i)$ denotes the projected center of Gaussian i.

The complete foreground attraction objective combines the opacity coverage and projected position components:

$$\mathcal{L}_{\text{fg-ws}} = \mathcal{L}_{\text{fg-}\alpha} + \lambda_{\text{proj}} \mathcal{L}_{\text{fg-proj}}. \tag{5.28}$$

Here, λ_{proj} controls the relative strength of the projected position component.

The foreground attraction loss is applied using a time-dependent weight to avoid a sharp drop in loss signal after the warm-start period. Let T_{ws} be the number of full-strength warm-start steps, T_{decay} be the number of decay steps, and λ_{ws} be the base warm-start weight. The warm-start weight at training step t is then:

$$w_{\text{ws}}(t) = \begin{cases} \lambda_{\text{ws}}, & t < T_{\text{ws}}, \\ \lambda_{\text{ws}} \left(1 - \frac{t - T_{\text{ws}}}{T_{\text{decay}}} \right), & T_{\text{ws}} \leq t < T_{\text{ws}} + T_{\text{decay}}, \\ 0, & t \geq T_{\text{ws}} + T_{\text{decay}}, \end{cases} \tag{5.29}$$

and the full weighted warm-start loss added to the training objective is:

$$\mathcal{L}_{\text{ws}} = w_{\text{ws}}(t) \left(\mathcal{L}_{\text{fg-}\alpha} + \lambda_{\text{proj}} \mathcal{L}_{\text{fg-proj}} \right). \tag{5.30}$$

Table 5.7 provides the full list of warm-start parameters used in the reported experiments.

Parameter	Value
Warm-start duration	2000 steps
Warm-start decay duration	2000 steps
Foreground attraction loss weight	0.5
Foreground luminance threshold	0.03
Foreground mask dilation radius	3 px
Projected foreground attraction weight	0.4
Projected foreground downsampling factor	4
Unseen-Gaussian rescue interpolation factor	0.5
Conservative culling duration	1000 steps
Conservative opacity-culling scale	0.5

Table 5.7: Foreground-attraction warm-start parameters.

During the warm-start, culling and densification are temporarily disabled, preventing the model from duplicating or deleting Gaussians before they have had the opportunity to migrate towards the object. After the initial warm-start period, conservative culling is used for an additional 1000 steps by scaling the low-opacity culling threshold by a factor of 0.5. This transition delays the return to the full support-aware pruning behavior, allowing the Gaussian distribution to stabilize as the warm-start attraction decays. For random initialization volumes that extend beyond the camera-observed region, the warm-start may also periodically rescue recently unseen Gaussians by interpolating them toward the visible Gaussian cloud or, when insufficient visible support exists, toward the scene center. Let $\boldsymbol{\mu}_{\text{vis}}$ denote the center of the recently visible Gaussian cloud and let α_{rescue} denote the rescue interpolation factor. For an unseen Gaussian center $\boldsymbol{\mu}_i$, the rescue update is:

$$\boldsymbol{\mu}_i \leftarrow (1 - \alpha_{\text{rescue}})\boldsymbol{\mu}_i + \alpha_{\text{rescue}}\boldsymbol{\mu}_{\text{vis}}. \tag{5.31}$$

In the reported experiments, $\alpha_{\text{rescue}} = 0.5$, moving unseen Gaussians halfway toward the visible cloud at each rescue step. This rescue is applied only during the warm-start and only to Gaussians with no

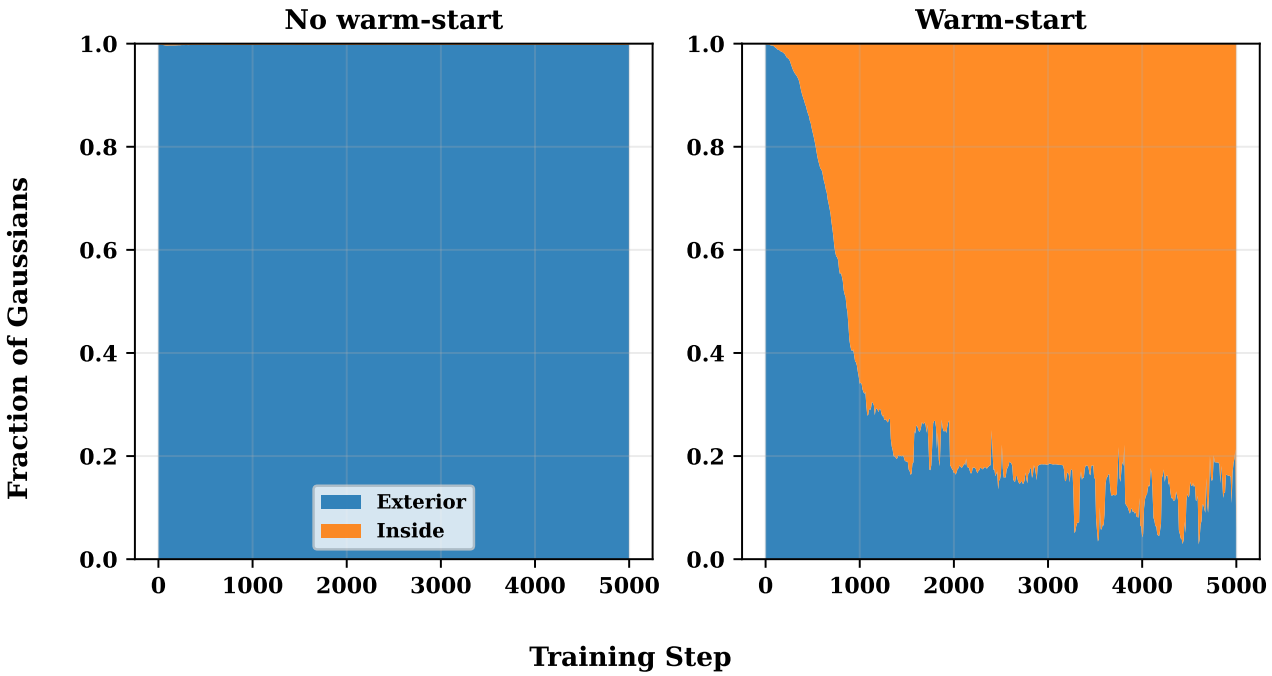


Figure 5.5: Comparison of the fraction of total Gaussians located inside a ground-truth mesh region between a nominal and warm-started random initialization with 100,000 Gaussians.

recent visibility, preventing permanently stranded Gaussians from remaining outside the influence of the image-space losses.

Figure 5.5 compares the number of Gaussians located inside the true object mesh during the first 5000 training steps of two randomly initialized runs — one regular and one that uses the warm-start. During these steps, culling and densification are turned off to avoid fraction-of-total artifacts. These results show that the warm-start draws more Gaussians to useful regions of the scene, and as seen in Figure 5.6, also results in better photometric performance and a higher quality shape model. Where the default strategy prunes nearly all Gaussians in the early stages of training, the warm-start allows Gaussians to migrate towards the model before support-aware training kicks in, protecting geometry from aggressive culling as the warm-start effects decay.

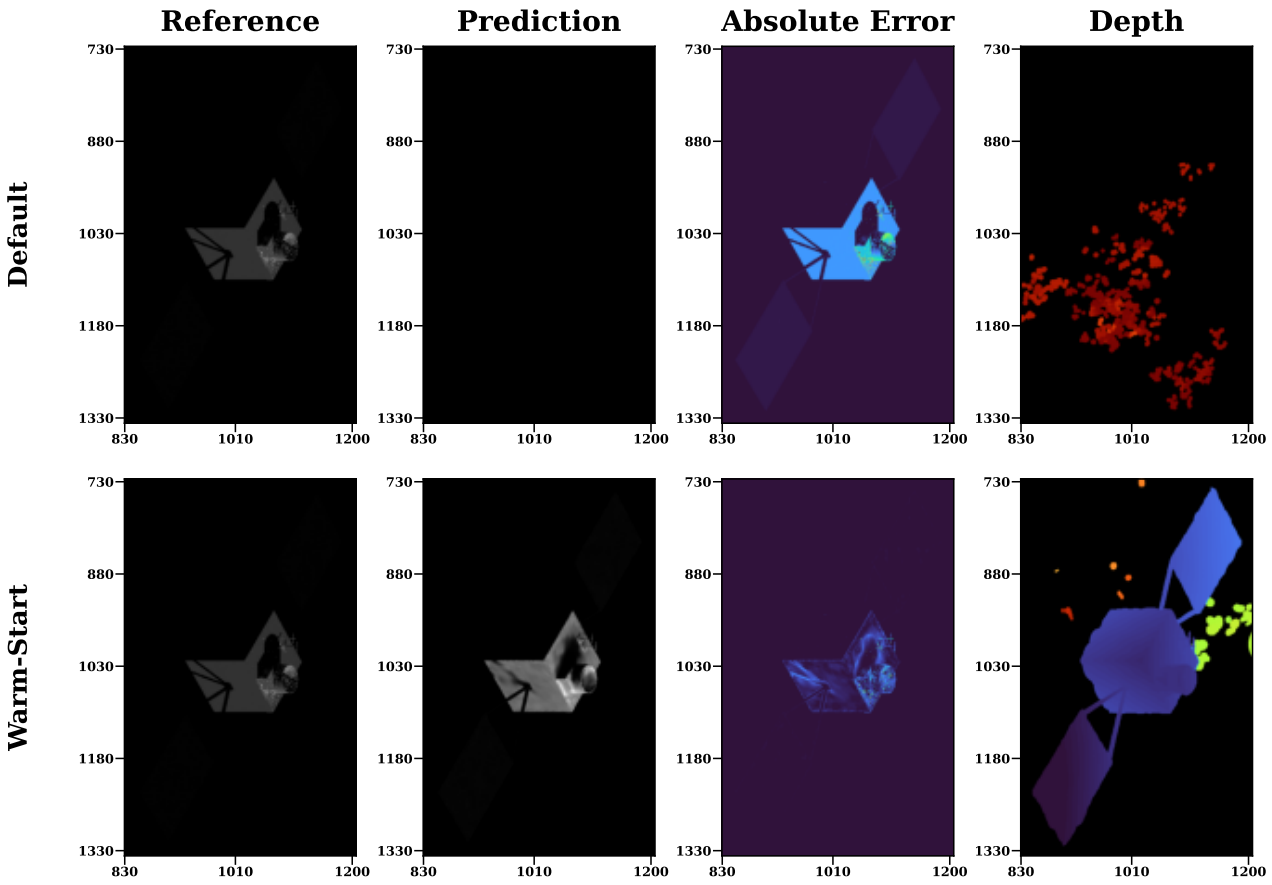


Figure 5.6: Post-training comparison of a nominal and warm-started random initialization.

Chapter 6: Case Studies

6.1 Description of Case Study Format

The methods developed in the previous chapters are evaluated across three space-imagery regimes of increasing realism. This progression is important because reconstruction and relighting performance depends strongly on the amount and quality of information available during training. A method that performs well with known poses, known illumination, and clean simulation data may behave differently when lighting is unobserved, camera poses must be estimated, or the imagery comes from an operational mission.

Accordingly, this chapter uses the case studies as a staged stress test of the proposed pipeline. The simulation dataset provides a controlled setting for validating the intended behavior of the relighting and shadowing methods. The synthetic SPEED+ dataset removes explicit sun information while retaining known camera geometry, testing how the model behaves when illumination must be learned implicitly. The ADRAS-J dataset then evaluates the practical workflow on real orbital imagery, where pose recovery, sparse coverage, and real material effects introduce additional uncertainty.

Together, these datasets help distinguish which aspects of the pipeline are robust across data regimes and which depend on favorable metadata or controlled imaging conditions.

6.2 Operating on Simulation Data

The simulation data used for the majority of this thesis' work comes from The Johns Hopkins Applied Physics Laboratory, generated using their high-fidelity simulation software. Unlike the synthetic and real-world cases, this simulation data provides known camera poses, camera intrinsics, sun directions, clean object imagery, and a ground-truth mesh.

6.2.1 Data Preprocessing

In collaboration with APL, the simulation data is provided in a format that makes conversion to *Nerfstudio*'s convention simple, requiring a single transformation of their native defined coordinate frame and reading of relevant pose and camera information into a `transforms.json` file. This file is pointed to during training and contains the camera-to-world extrinsic matrices for each camera, along with the camera intrinsic parameters used to capture them. For the simulation datasets, a ground-truth mesh is also supplied, providing a geometrically meaningful initialization when sampled into a point cloud for training.

6.2.2 Training

The availability of accurate poses, known illumination directions, and ground-truth geometry makes the simulation case study the most favorable setting for the proposed pipeline. In this regime, the appearance field can be conditioned directly on the known sun direction and shadowing methods can be evaluated against consistent lighting geometry. In addition to the boxsat model used in earlier experiments, a Hubble Space Telescope model is used to provide cross-object validation of the methods developed in previous chapters. As with the boxsat dataset, the Hubble dataset contains images across 24 sun angles with 614 images per sun angle. Unlike the Fibonacci-sampled boxsat images/sun directions, the sun angles form a polar arc over the Hubble model and the cameras are placed as shown in Figure 6.1. The presented model was trained for 50,000 iterations, with refinement stopping at 40,000.

A qualitative relighting showcase for the Hubble model is given in Figure 6.2. The predicted renders reproduce the primary illumination changes across the selected sun directions, including the

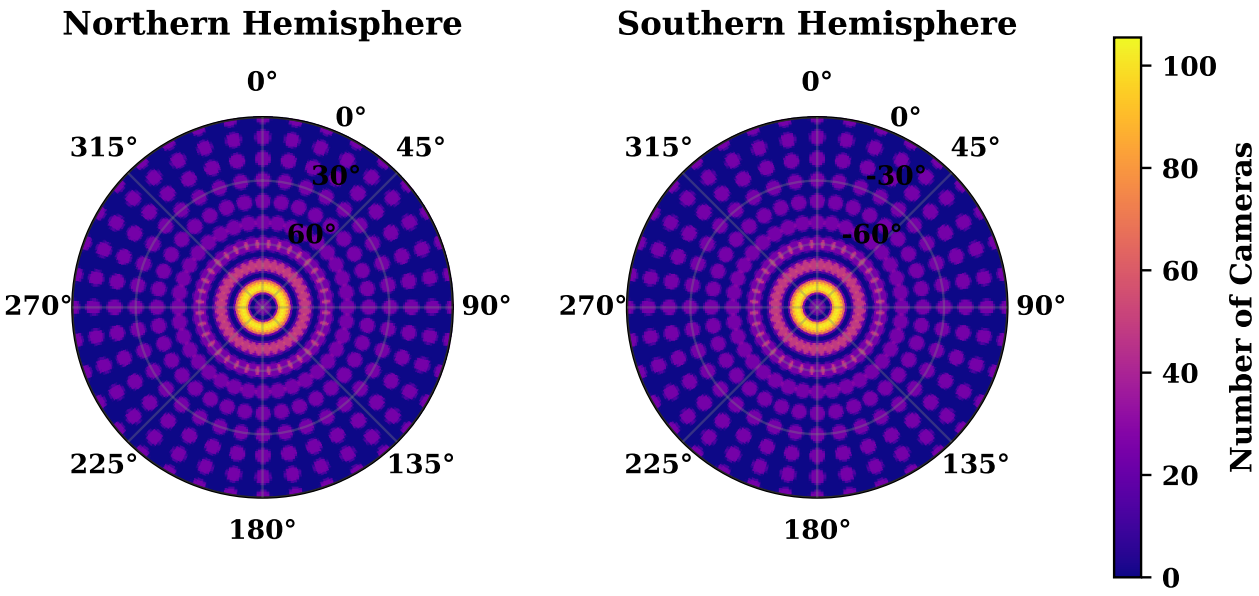


Figure 6.1: Heatmap of Hubble training camera density.

migration of bright regions and shadowed areas across the barrel and aft shroud. This demonstrates the main advantage of the simulation setting: because sun direction metadata is available during training, the appearance field can be queried at different lighting conditions during inference, resulting in a controlled, relightable representation. Concurrently, this study also highlights the reliance of the methods on metadata availability; in the later synthetic and real-world case studies, this assumption is not always satisfied.

Figure 6.3 shows a summary of results on the simulated Hubble telescope dataset, with highlight images given by best, mean, and worst SSIM performance. From the best case, as well as several other high-scoring evaluation images, the reconstructed model nearly perfectly represents the shadowed faces of the spacecraft. Even in the mean and worst cases, the reconstruction continues to perform well, with slight shadow artifacting below the small component to the left of the aperture door in the mean case, and thin bands of high error on the end of the aft shroud.

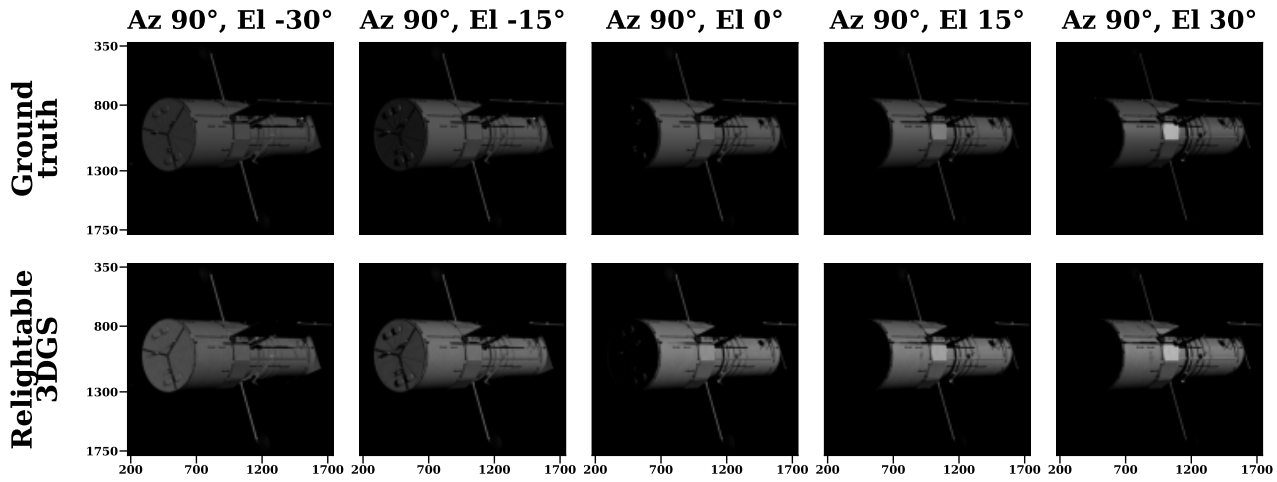


Figure 6.2: Hubble model relighting showcase.

6.3 Operating on Synthetic Data

This thesis categorizes "synthetic data" as an intermediate regime between fully simulated imagery and real-world orbital imagery. In the fully simulated setting, camera poses, object geometry, and illumination metadata are all known and controlled. In real-world imagery, these quantities may be unknown and must often be estimated from the images themselves. The synthetic benchmark occupies a useful middle ground, in which accurate camera pose labels are provided, but not sun information — testing the reconstruction and relighting pipeline when illumination metadata is missing. The synthetic dataset chosen for evaluation comes from the Next-Generation Spacecraft Pose Estimation Dataset (SPEED+) [56]. SPEED+ is an advanced dataset for vision-based spacecraft pose estimation with specific emphasis on evaluating the robustness of machine learning models, containing synthetic, as well as lightbox and sunlamp-illuminated images of the Tango spacecraft from the European Space Agency's PRISMA mission. For the purposes of this thesis, the focus will be on the synthetic component of this dataset.

Mean Photometric Performance

PSNR ↑	SSIM ↑	LPIPS ↓	MAE ↓
31.8046 ± 7.4455	0.9812 ± 0.0175	0.0554 ± 0.0312	0.0113 ± 0.0130

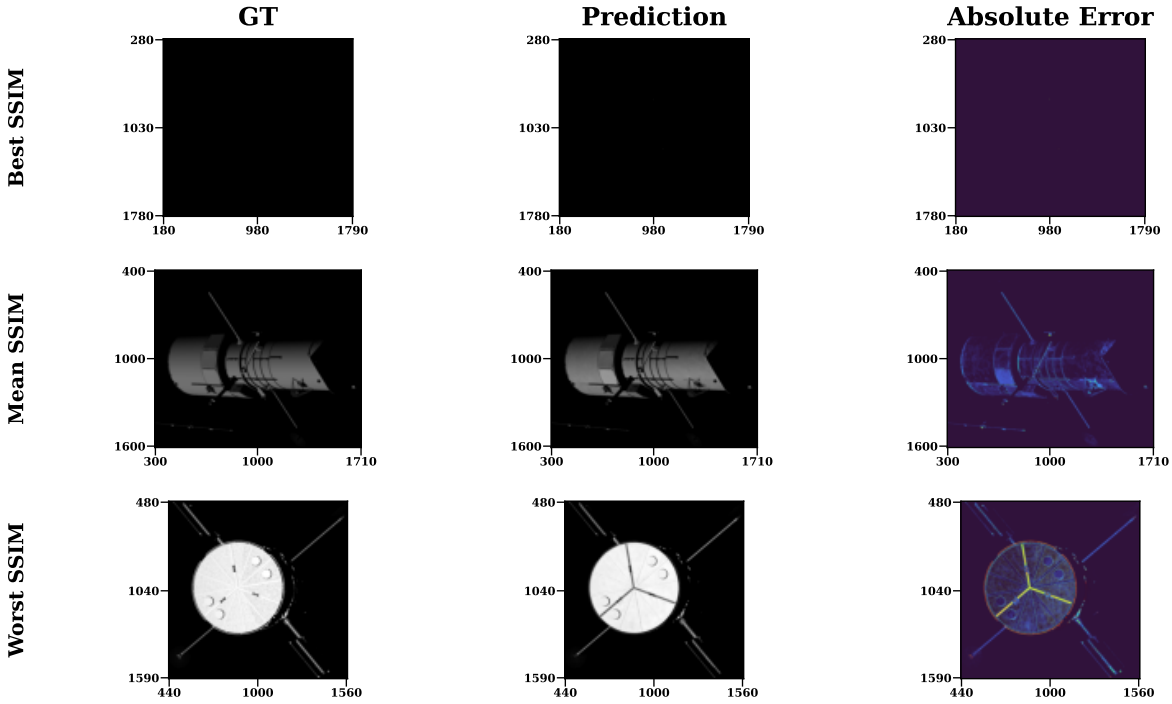


Figure 6.3: Summary of results on Hubble dataset.

6.3.1 Data Preprocessing

In its native form, the provided SPEED+ pose information and metadata is not directly formatted for training in *Nerfstudio*'s framework. In order to enable training on this third-party dataset, the SPEED+ poses and provided camera intrinsics were represented in the *Nerfstudio* format by converting the world-to-camera quaternion orientations and positions into camera-to-world rotation matrices ($\mathbb{R}^{3x3}$) and position vectors ($\mathbb{R}^{3x1}$). These new rotation matrices and position vectors form the camera extrinsic matrices, while the provided camera parameters form the intrinsic matrices for each frame, which are then written to a *Nerfstudio* `transforms.json` file to be ingested for training. Although SPEED+ provides camera poses, a sparse point cloud was generated using COLMAP [57]

only as an initialization source for the model, shown in Figure 6.4.

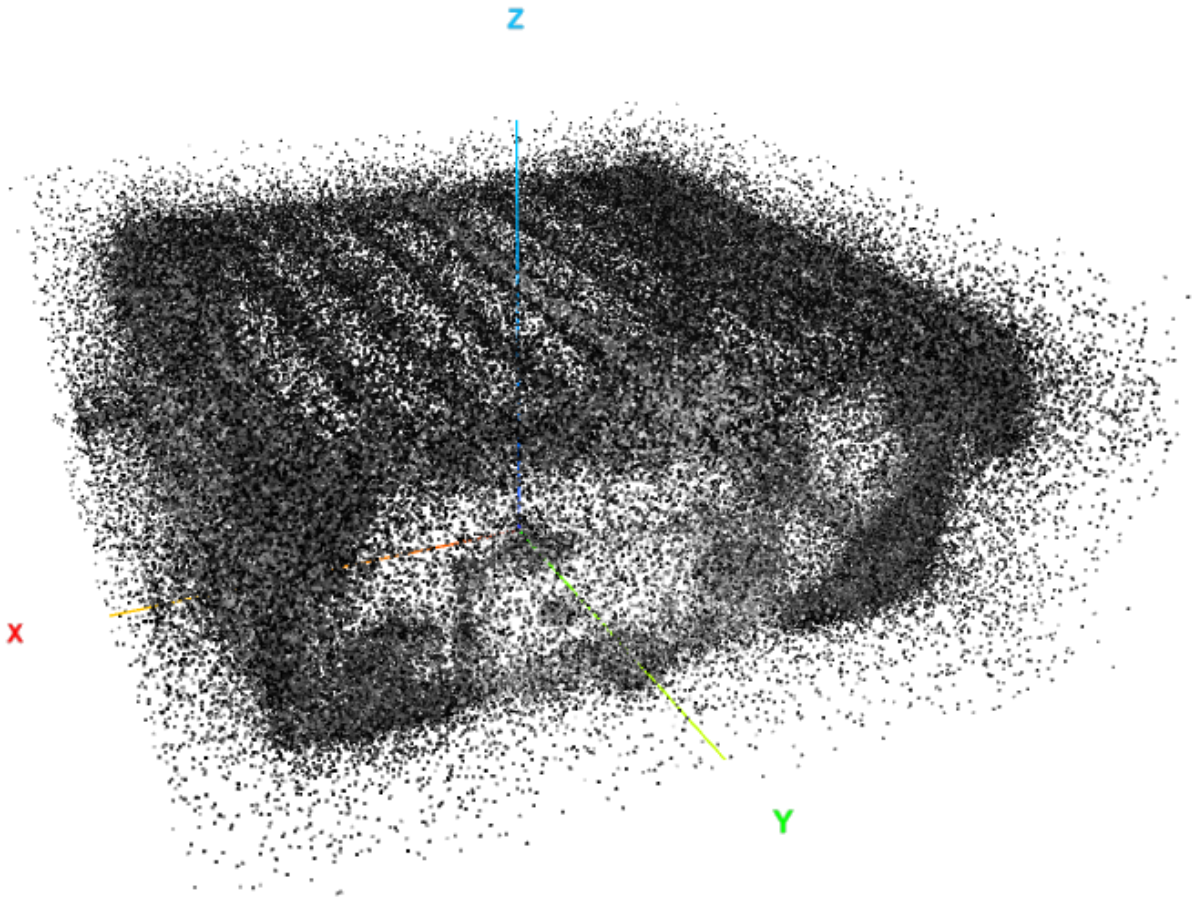


Figure 6.4: SPEED+ point cloud retrieved by COLMAP.

6.3.2 Training

The SPEED+ dataset contains just under 48,000 synthetic training images and 12,000 validation images, with no repeat poses. The sampling distribution of the training camera poses is near-uniform and containing images from all angles around the Tango spacecraft at a maximum distance of 10 meters, as seen in Figure 6.5. This generally leads to better reconstruction quality, as the model takes up more of the frame, leading to more informative training signals from each image. During the creation of the synthetic dataset, random Earth backgrounds were inserted into half of the training

images, but because their placement is inconsistent with respect to Tango's pose, there are no issues with floating Gaussian artifacts. The presented model was trained for 120,000 iterations, with refinement stopping at 100,000.

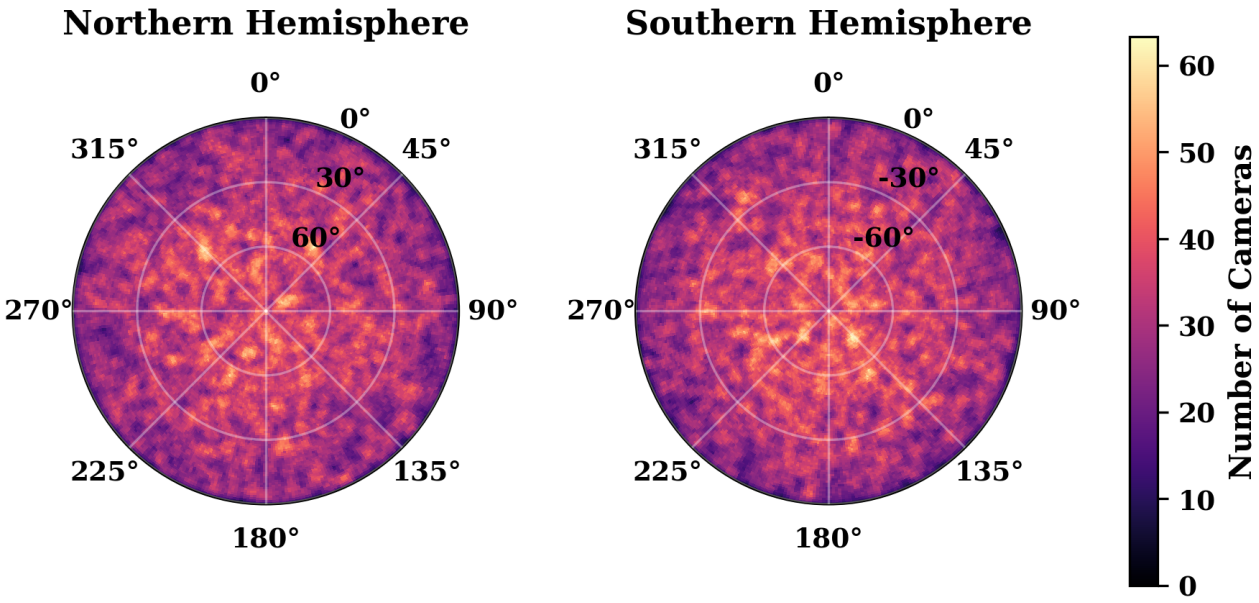


Figure 6.5: Heatmap of SPEED+ training camera density.

Figure 6.6 summarizes the trained model's behavior using the best, median, and worst photometric reconstruction cases. Image-space metrics are evaluated over ground-truth segmentation masks. While the close proximity of the training views lends itself to geometric recovery of the Tango spacecraft, reconstruction quality is *not* uniform across lighting configurations, as evidenced by the worst-case reconstruction in particular. When the observed lighting is compatible with the dominant appearance learned by the model, the reconstruction remains visually accurate. In particular, the top side of the Tango spacecraft, which is covered by dark solar panels, is approximated well because the solar panel appearance exhibits less extreme brightness variation under different sun angles.

The more difficult cases occur when the test image contains strong illumination conditions that are not well represented by a single averaged appearance. This is especially visible on the lower

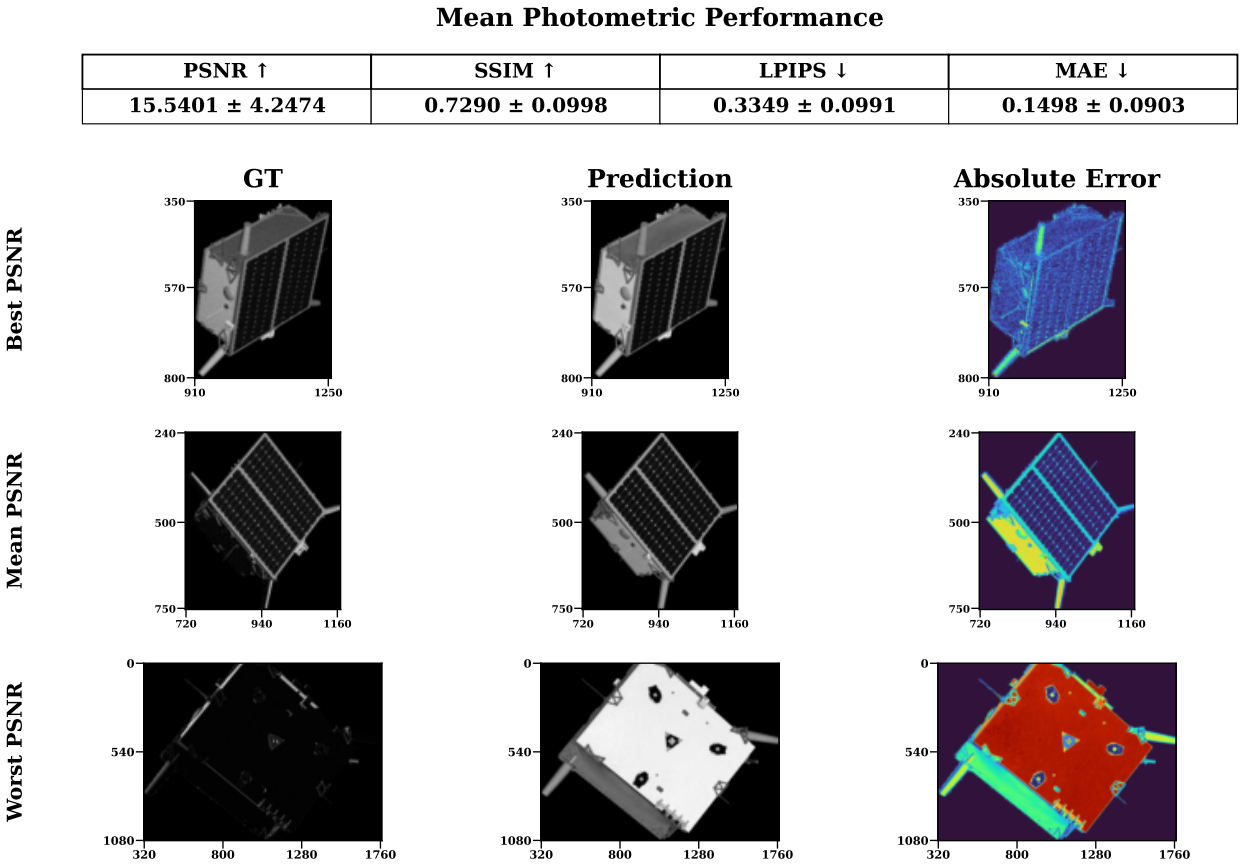


Figure 6.6: Summary of SPEED+ training results; trained using training split, evaluated on validation split.

portions of the spacecraft, where direct illumination can create far darker regions and sharper contrast than the model can consistently reproduce without sun conditioning. Images with near-perpendicular sun incidence on the top or bottom of the spacecraft therefore produce larger errors. In these cases, the model is not necessarily failing to reconstruct the spacecraft geometry; rather, it is being asked to represent multiple incompatible lighting states without being told which lighting state applies to a given image.

While per-image embeddings may help absorb some of these image-specific appearance variations, they do so implicitly rather than by encoding the underlying sun direction, limiting their usefulness for physically controlled relighting. In addition, because SPEED+ contains a large number

of images, a per-image embedding approach would require learning a separate appearance code for each training image. Under standard image-sampling training, this means each code receives only a small number of updates unless the model is trained for substantially more iterations, making these embeddings an inefficient substitute for explicit illumination conditioning.

6.4 Operating on Real-World Data

Deploying 3D Gaussian splatting on real-world spacecraft data introduces a different set of challenges than the simulated and synthetic datasets. In the simulation case, camera pose, scene scale, object geometry, and sun direction are known. With SPEED+, camera poses are provided, but illumination labels are unavailable. In the real-world case study, neither camera poses nor sun information is provided directly. As a result, the problem shifts from controlled reconstruction and relighting toward a broader real-world preprocessing pipeline to faithfully build a model from the training views.

The real-world data used in this thesis comes from Astroscale's ADRAS-J mission [6], which captured images of a JAXA space debris object. Astroscale released a stitched video of these captures online, from which individual image frames were retrieved and processed to be used in training. Although explicit sun direction metadata is not available, the ADRAS-J images were captured over a short fly-around period, and are thus treated as an approximately single-illumination sequence.

6.4.1 Data Preprocessing

For this case, where the only data provided is the images themselves, the most extensive preprocessing steps of the three cases are required. First, individual image frames are extracted

from the video (103 total), and the Earth is removed from the background of images where it is present to reduce the risk that the model allocates Gaussians to background structure rather than the target object. Finally, COLMAP's Structure-from-Motion (SfM) pipeline is used to perform feature extraction, sequential image matching, and pose estimation on the images to retrieve approximate pose information.

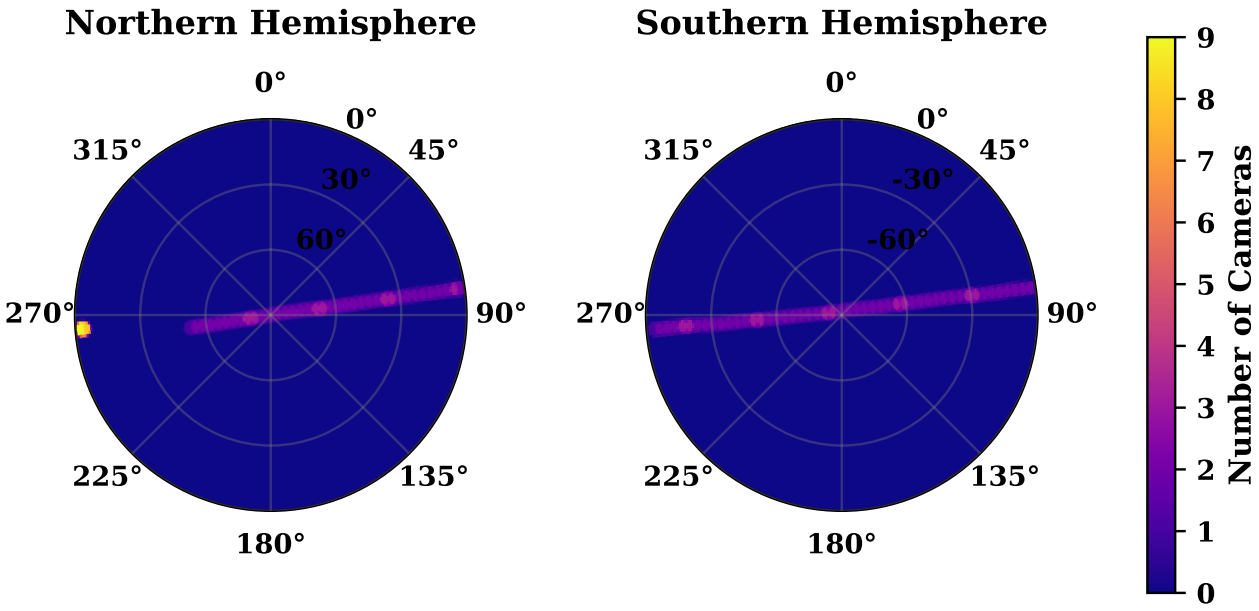


Figure 6.7: Heatmap of ADRAS-J training camera density.

Figure 6.7 shows the camera poses retrieved by COLMAP, forming the near-polar arc seen in Astroscale's video. This trajectory provides useful coverage of the object from the observed side, but it also reveals an important limitation of the dataset: large portions of the viewing sphere are weakly constrained or entirely unobserved. As a result, the model can be expected to perform best near the recovered training trajectory and to degrade when rendered from viewpoints far from that arc.

In addition to pose estimates for each training view, COLMAP also produces a sparse point cloud, which is recentered to the origin and trimmed to remove floating outliers. To maintain consistency between the point cloud and the *Nerfstudio* scene representation, the scene origin is

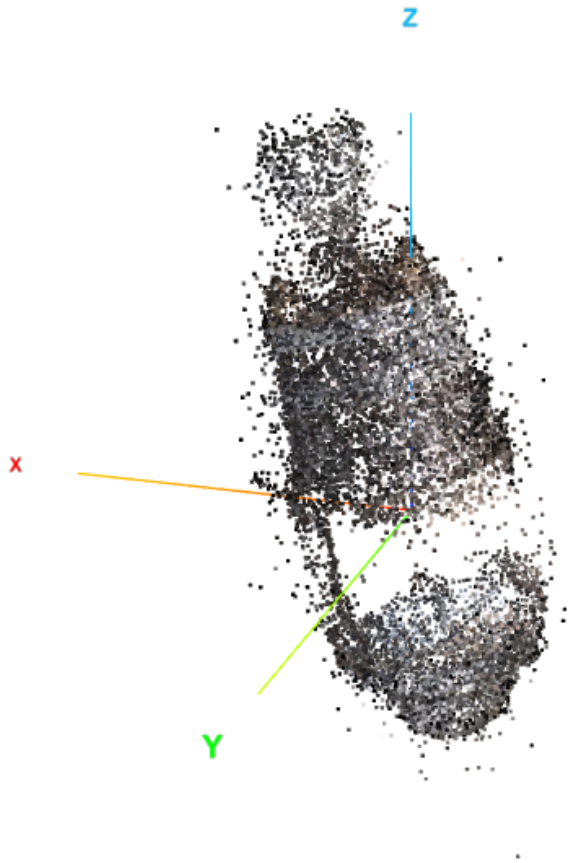


Figure 6.8: COLMAP-retrieved point cloud used to initialize ADRAS-J training.

shifted by the same amount. Figure 6.8 shows the point cloud used for training the ADRAS-J model.

6.4.2 Training

The ADRAS-J model was trained using the estimated COLMAP camera poses and the sparse point cloud initialization. Unlike the simulation data, the sequence does not possess ground-truth camera poses, object scale, detailed object geometry, or sun direction metadata. Furthermore, the illumination setting itself differs from the prior cases in that it contains only a single illumination condition, making this case study a test of whether a coherent reconstruction can be recovered from real spacecraft imagery when camera poses and scene scale must be estimated.

Figure 6.9 provides a photometric performance summary of a model trained on a background-removed version of the ADRAS-J dataset. The pictured model was trained for 40,000 iterations, with refinement stopping at 30,000 to allow geometry to settle. The LPIPS metric is used to highlight frames that emphasize the strengths and weaknesses of the model on this dataset. While the primary structure of the debris object is generally well-represented, it struggles to recreate the specular reflections glinting from metallic surfaces of the model. Since 3DGS renders images via rasterization, it cannot represent rays of reflected light unless it places geometry in those regions. As these methods mature to operate on more realistic space data, it is recommended that mechanisms for representing more dynamic ranges of color and aspects of physical light transport should be explored to capture these kinds of effects.

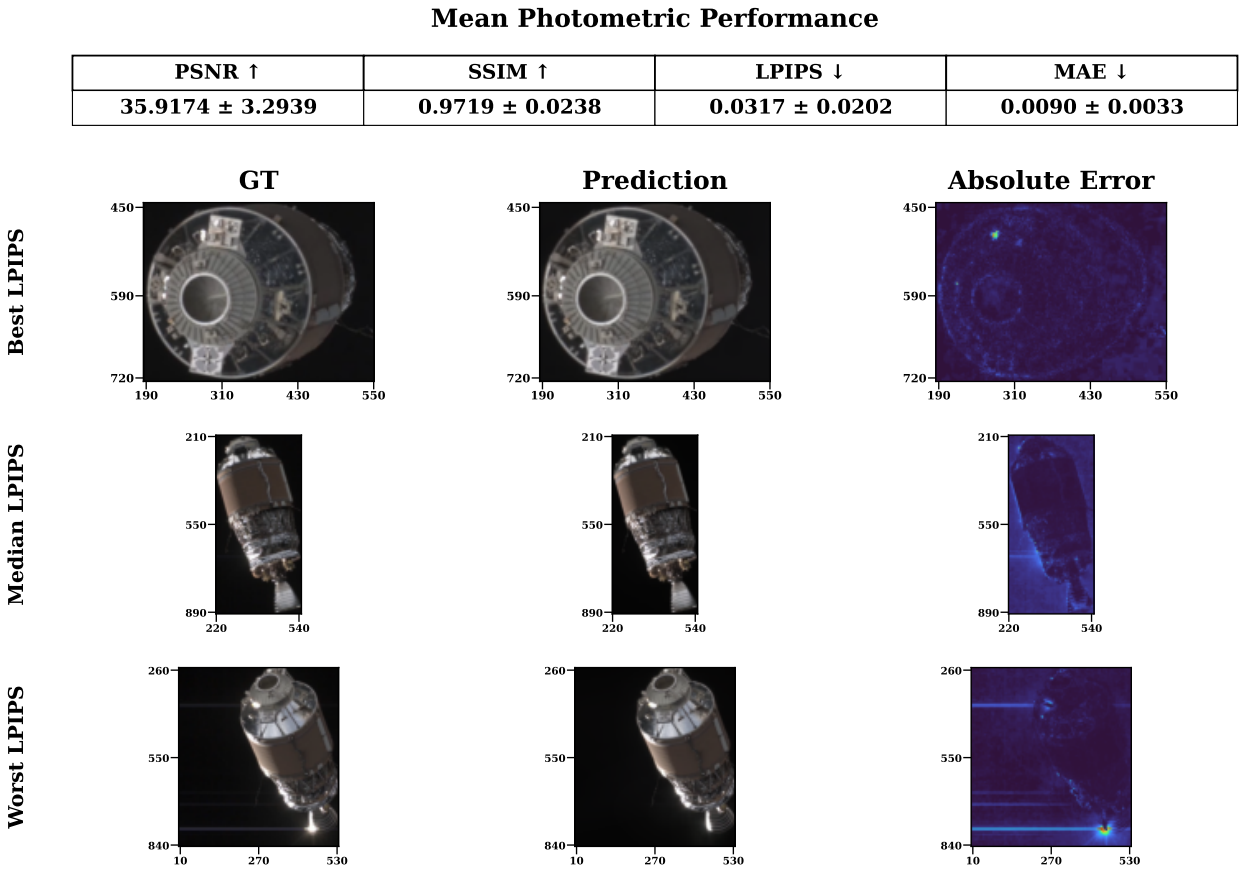


Figure 6.9: Summary of results on ADRAS-J dataset.

The limitations of the dataset sparsity become more apparent when the model is rendered from viewpoints far from the training camera. Figure 6.10 shows a side view of the debris object, where the coarse shape remains partially recognizable, but the appearance quality degrades substantially. This behavior is expected: because the available training cameras occupy a single polar arc, the model receives little or no direct image evidence for many side regions. The learned appearance in these regions is therefore weakly constrained, causing color, texture, and fine detail to break down outside the observed view range.

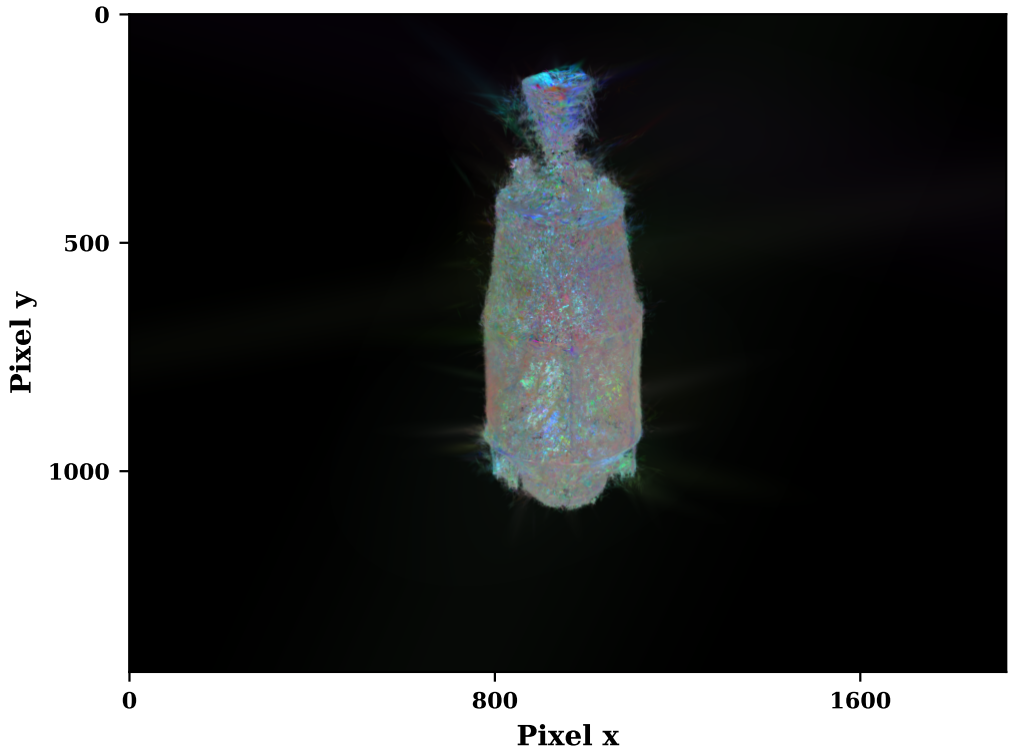


Figure 6.10: Side view of ADRAS-J 3DGS model.

The ADRAS-J case study demonstrates both the promise and the current limitations of applying 3D Gaussian Splatting to real orbital imagery. Coherent visual representations can be formed after pose recovery and preprocessing, but the absence of proper coverage prevents the same level of controlled reconstruction possible in simulation.

Chapter 7: Conclusion

7.1 Summary of Contributions

This thesis investigated the use of radiance fields — in particular, 3D Gaussian Splatting — for reconstruction of objects imaged in space environments. Two algorithms were developed to render multiple independently-trained neural radiance field and 3D Gaussian splatting models together in a shared world scene. Combined with UI additions to *Nerfstudio*'s interactive viewer, these algorithms enable visualization and arbitrary transformation of radiance field models for rendering and analysis. A sun-conditioned appearance and shadowing framework was introduced, allowing for dynamic relighting and inference-time inter- and intra-model shadowing of one or more 3D Gaussian splatting models. A per-Gaussian receiver-depth bias is used to reduce the effects of shadow acne on directly-illuminated faces of 3DGS models, and an adaptive light camera placement algorithm is created to ensure appropriate shadow support across multi-model scenes. Two new culling and densification strategies for training 3D Gaussian splatting models were developed, designed to take advantage of mesh information when available, and protect low-opacity Gaussians from aggressive culling if supported across multiple training views. Additionally, a warm-start procedure for random-box 3DGS initializations was provided. Lastly, these models were trained across a spectrum of datasets with increasing realism: fully simulated data with known poses, and multiple known sun angles; synthetic data with known poses and multiple unknown sun angles; and real-world data, with neither known poses nor sun-angle information.

From qualitative and quantitative comparisons against vanilla 3DGS models, it was found that the appearance module modifications enabled sun-conditioned dynamic relighting and greater flexibility in multi-illumination scenes. Shadow mapping and shadow splatting both improved shadow

fidelity over the no-shadow baseline and enabled inter-model shadowing. Shadow mapping performed strongest in the multi-sun evaluation, while shadow splatting remained highly competitive and achieved the best overall photometric performance in the composed two-object case. The receiver-depth bias succeeded at mitigating shadow acne on model faces perpendicular to incident sunlight, though sun-grazing faces remain an issue. The mesh-aware and support-aware strategies introduced moderately improve photometric reconstruction, though their file size and memory tradeoffs make their use cases rather specialized. Deployment of these models on simulated, synthetic, and real-world data emphasized their high expressivity when appropriate pose and sun angle labels exist, and showed their limitations when working without that information.

7.2 Future Work

The methods developed in this thesis move radiance field representations toward more flexible modeling under varied illumination in space environments, and several natural extensions remain:

- **More Physically Decomposed Appearance Models:** A potentially useful enhancement of the appearance module would consider further decomposition of the learned appearance representation into physically meaningful factors. Prior work has shown that associating extra lighting parameters, such as material properties, normal vectors, and BRDFs/incident lighting can improve novel view synthesis and relighting [11]. Introducing either learned or explicit estimates of these lower-level appearance components may allow the model to better generalize to new lighting conditions and produce more physically consistent reflective effects for spacecraft materials like solar panels or metallic components.

- **2D Gaussian Splatting for Surface-Consistent Relighting and Shadowing:** Another

promising direction would be to investigate 2D Gaussian Splatting [55] as a more surface-consistent representation for spacecraft radiance fields. Unlike volumetric 3D Gaussians, 2DGS encourages Gaussians to localize around coherent surfaces, thereby reducing the ambiguous volumetric thickness that can make true depth and light-space visibility difficult to define. By producing firmer surface localization, a 2DGS formulation could provide more stable receiver depths, defined normals, and cleaner light-ray intersections for shadow estimation. Moreover, access to well-defined surface intersection and normal vectors may open the door for use of more physically-grounded lighting configurations, as well as ray-Gaussian hybrid rendering pipelines. Combined with the decomposition or learning of lower-level illumination components, this framework may assist with higher-fidelity inter-model effects, and potentially make inference-time reflections between models more feasible.

- **Modeling of Multiple Illumination Sources:** A further extension of the relighting framework would be to model multiple illumination sources rather than assuming that the Sun is the only dominant light source. In orbital imagery, the Earth's albedo can provide a non-negligible source of secondary illumination, particularly for spacecraft in low Earth orbit or for viewing geometries in which the Sun does not fully explain the observed brightness distribution. Incorporating both direct solar illumination and diffuse albedo could allow the model to better represent shadowed regions, soft fill lighting, and appearance changes that arise from the spacecraft's orbital environment. A multi-source formulation could condition the appearance or shadowing modules on separate lighting terms for the Sun and Earth, enabling more physically faithful reconstruction and relighting in realistic space scenes.

- **Tone Mapping for High-Dynamic-Range Space Imagery:** Another useful direction would

be to incorporate tone mapping or high-dynamic-range intensity modeling into the radiance field training pipeline. Space imagery can contain extreme intensity variation, particularly when spacecraft surfaces are directly illuminated by the Sun, when specular reflections occur on metallic or solar-panel materials, or when saturated highlights coexist with dark shadowed regions. Standard image representations and losses can make it difficult for the model to preserve radiance values that exceed the nominal image range or are compressed during preprocessing. A differentiable tone-mapping strategy could allow training on a compressed but informative representation while preserving gradients across both bright and dark regions. This may improve reconstruction of directly illuminated surfaces, reduce saturation-related artifacts, and provide a more physically meaningful basis for relighting under strong solar illumination.

In summary, the findings of this study indicate that radiance fields are a promising representation for spacecraft modeling, but one that benefits substantially from domain-specific modifications when applied to the unique constraints of space imagery. Future work exploring surface element-based scene formulations, alternative approaches for decoupling geometry and appearance, and explicit modeling of Earth albedo would help move these methods toward more reliable use on real-world datasets.

Bibliography

[1] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, August 2020.

[2] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering, August 2023.

[3] Guikun Chen and Wenguan Wang. A Survey on 3D Gaussian Splatting, October 2025.

[4] Basilio Caruso, Trupti Mahendrakar, Van Minh Nguyen, Ryan T. White, and Todd Steffen. 3D Reconstruction of Non-cooperative Resident Space Objects using Instant NGP-accelerated NeRF and D-NeRF, June 2023.

[5] Sajjad Kazemi, Nasser L. Azad, K. Andrea Scott, Haroon B. Oqab, and George B. Dietrich. Orbit determination for space situational awareness: A survey. *Acta Astronautica*, 222:272–295, 2024.

[6] Astroscale. ADRAS-j mission, 2026.

[7] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections, January 2021.

[8] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural Reflectance Fields for Appearance Acquisition, August 2020.

[9] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields, December 2021.

[10] Yao Yao, Jingyang Zhang, Jingbo Liu, Yihang Qu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. NeILF: Neural Incident Light Field for Physically-based Material Estimation, March 2022.

[11] Jian Gao, Chun Gu, Youtian Lin, Zhihao Li, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3D Gaussians: Realistic Point Cloud Relighting with BRDF Decomposition and Ray Tracing, August 2024.

[12] Yahao Shi, Yanmin Wu, Chenming Wu, Xing Liu, Chen Zhao, Haocheng Feng, Jian Zhang, Bin Zhou, Errui Ding, and Jingdong Wang. GIR: 3D Gaussian Inverse Rendering for Relightable Scene Factorization, August 2024.

[13] Zhenyuan Liu, Yu Guo, Xinyuan Li, Bernd Bickel, and Ran Zhang. BiGS: Bidirectional Gaussian Primitives for Relightable 3D Gaussian Splatting, August 2024.

[14] Luca Savant Aira, Gabriele Facciolo, and Thibaud Ehret. Gaussian splatting for efficient satellite image photogrammetry, 2025.

[15] Aymen Mir, Riza Alp Guler, Jian Wang, Gerard Pons-Moll, and Bing Zhou. Animated 3DGS avatars in diverse scenes with consistent lighting and shadows, 2026.

[16] Zoubin Bi, Yixin Zeng, Chong Zeng, Fan Pei, Xiang Feng, Kun Zhou, and Hongzhi Wu. GSˆ3: Efficient Relighting with Triple Gaussian Splatting. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–12, December 2024.

[17] Hiba Dahmani, Moussab Bennehar, Nathan Piasco, Luis Roldao, and Dzmitry Tsishkou. SWAG: Splatting in the Wild images with Appearance-conditioned Gaussians, April 2024.

[18] Jonas Kulhanek, Songyou Peng, Zuzana Kukelova, Marc Pollefeys, and Torsten Sattler. WildGaussians: 3D Gaussian Splatting in the Wild, October 2024.

[19] Congrong Xu, Justin Kerr, and Angjoo Kanazawa. Splatfacto-W: A Nerfstudio Implementation of Gaussian Splatting for Unconstrained Photo Collections, September 2024.

[20] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-Centric Neural Scene Rendering, December 2020.

[21] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering, September 2021.

[22] Xinyu Gao, Ziyi Yang, Yunlu Zhao, Yuxiang Sun, Xiaogang Jin, and Changqing Zou. A General Implicit Framework for Fast NeRF Composition and Rendering, January 2024.

[23] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly-Throughs, March 2022.

[24] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable Large Scene Neural View Synthesis, February 2022.

[25] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs, August 2021.

[26] Ruilong Li, Sanja Fidler, Angjoo Kanazawa, and Francis Williams. NeRF-XL: Scaling NeRFs with Multiple GPUs, April 2024.

[27] Jian Gao, Mengqi Yuan, Yifei Zeng, Chang Zeng, Zhihao Li, Zhenyu Chen, Weichao Qiu, Xiao-Xiao Long, Hao Zhu, Xun Cao, and Yao Yao. ComGS: Efficient 3D Object-Scene Composition via Surface Octahedral Probes, October 2025.

[28] Anne Mergy, Gurvan Lecuyer, Dawa Derksen, and Dario Izzo. Vision-based neural scene representations for spacecraft. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2002–2011, June 2021.

[29] Yang Liu, Zhihao Sun, Lele Zhang, Lele Xi, Wei Dong, and Fang Deng. Spacecraft-NeRF: High-fidelity reconstruction of spacecraft by neural radiance field based implicit representation. *IEEE Transactions on Aerospace and Electronic Systems*, 61(6):15182–15194, 2025.

[30] Aneesh M. Heintz and Mason Peck. Spacecraft State Estimation Using Neural Radiance Fields. *Journal of Guidance, Control, and Dynamics*, 46(8):1596–1609, August 2023.

[31] Antoine Legrand, Renaud Detry, and Christophe De Vleeschouwer. Leveraging neural radiance fields for pose estimation of an unknown space object during proximity operations, 2024.

[32] Van Minh Nguyen, Emma Sandidge, Trupti Mahendrakar, and Ryan T. White. Characterizing satellite geometry via accelerated 3D gaussian splatting. *Aerospace*, 11(183), 2024.

[33] Yibin Zhao, Jianjun Yi, Yihan Pan, and Liwei Chen. 3D reconstruction of non-cooperative space targets of poor lighting based on 3D gaussian splatting. *Signal, Image and Video Processing*, 19(6):509, May 2025.

[34] Zhang, Xiaojie, Cui, Linyan, Wei, Xiaodong, and Chen, Yicong. Asteroid-GS: 3D Gaussian splatting for fast surface reconstruction of asteroids. *antike und abendland*, 704:A247, 2025.

[35] Tae Ha Park and Simone D'Amico. Improved 3D gaussian splatting of unknown spacecraft structure using space environment illumination knowledge, 2025.

[36] Xiaodong Wei, Linyan Cui, Chuankai Liu, and Jihao Yin. Integrating neural radiance fields with deep reinforcement learning for autonomous mapping of small bodies. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–17, 2024.

[37] Matt Pharr, Wenzel Jakob, and Greg Humphreys. PBRT v4: Physically based rendering toolkit, 2023.

[38] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. MIT Press, 4 edition, 2023.

[39] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. Mitsuba 3 renderer, 2022.

[40] Blender Foundation. Blender (cycles path-tracing renderer), 2024.

[41] Blender Foundation. Blender (EEVEE real-time renderer), 2024.

[42] LuxCoreRender Team. LuxCoreRender, 2024.

[43] NASA Goddard Space Flight Center. Vira: Scientific ray-tracing and visualization framework, 2024.

[44] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, July 2022.

[45] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields, March 2022.

[46] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul Srinivasan, and Jonathan T. Barron. NeRF in the Dark: High Dynamic Range View Synthesis from Noisy Raw Images, November 2021.

[47] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: An efficient JAX implementation of NeRF, 2020.

[48] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, pages 1–12, July 2023.

[49] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. NerfAcc: A General NeRF Acceleration Toolbox, May 2023.

[50] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. Gsplat: An Open-Source Library for Gaussian Splatting, September 2024.

[51] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian Grouping: Segment and Edit Anything in 3D Scenes, July 2024.

[52] Brent Yi, Chung Min Kim, Justin Kerr, Gina Wu, Rebecca Feng, Anthony Zhang, Jonas Kulhanek, Hongsuk Choi, Yi Ma, Matthew Tancik, and Angjoo Kanazawa. Viser: Imperative, web-based 3d visualization in python. *arXiv preprint arXiv:2507.22885*, 2025.

[53] Daniel Scherzer, Michael Wimmer, and Werner Purgathofer. A survey of real-time hard shadow mapping methods. *Computer Graphics Forum*, 30(1):169–186, 2011.

[54] Qianyi Wu, Jianmin Zheng, and Jianfei Cai. Surface reconstruction from 3D gaussian splatting via local structural hints. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 441–458, Cham, 2025. Springer Nature Switzerland.

[55] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D gaussian splatting for geometrically accurate radiance fields. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Papers*, Siggraph '24, pages 1–11. ACM, July 2024.

[56] Tae Ha Park, Marcus Märtens, Gurvan Lecuyer, Dario Izzo, and Simone D'Amico. SPEED+: Next-Generation Dataset for Spacecraft Pose Estimation across Domain Gap. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–15, March 2022.

[57] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.