Erasmus+

COSI Computational Colour and Spectral Imaging

Master in Computational Colour and Spectral Imaging (COSI)

UNIVERSITY OF EASTERN FINLAND · UNIVERSIDAD DE GRANADA · UNIVERSITÉ DE LYON · UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE · NTNU Norwegian University of Science and Technology

Mesh Reconstruction via Differentiable Rendering Pipelines
GS–2M: Gaussian Splatting for Joint Mesh Reconstruction and Material Decomposition

Master Thesis Report

Presented by

Dinh Minh Nguyen

and defended at the

Norwegian University of Science and Technology

September 2025

Academic Supervisor(s): Associate Professor, Dr. Philippe COLANTONI
Host Supervisor: Dr. Malte Avenhaus and Dr. Thomas Lindemeier

Submission of the thesis: 15th July 2025
Day of the oral defense: 2st September 2025

Abstract

Advances in neural rendering have given rise to many learnable scene representations that surpass traditional mesh-based primitives. While polygonal meshes remain the de facto standard representation for graphics editing and GPU-powered rasterization, their discrete yet highly structured nature makes them ineffective in image-based rendering tasks. This has led to a new array of optimization-friendly methods to encode scene information, among which radiance fields enjoy significant attention due to the excellent work of Neural Radiance Fields, widely known as NeRF, that achieved impressive results in novel view synthesis. Despite carrying multiple advantages with numerous derivative works aiming at quality and efficiency improvement, NeRF and neural-based methods suffer from high computing resource consumption, often requiring hours of training on high-end GPUs for a single scene.

Recently, Gaussian splatting has emerged as a resource-friendly technique for radiance fields, achieving comparable rendering quality in novel view synthesis while enabling real-time rasterization. The method leverages discrete 3D anisotropic Gaussians enriched with learnable appearance features, resulting in a scene representation substantially faster to train while allowing primitives to be adaptively refined during optimization. This has sparked a new line of research to adopt Gaussian splatting in various domains such as surface reconstruction, material decomposition, scene relighting, deformation, composition, and inverse rendering.

To pick up on this thriving development of Gaussian splatting, we propose a unified solution for *mesh* reconstruction and *material* decomposition from multi-view images, referred to as GS–2M. Previous works address these tasks separately and either struggle to reconstruct highly reflective surfaces or require priors from external models to enhance the decomposition results. While there are contemporary works that faithfully address these tasks together, they often employ sophisticated multi-layer perceptrons to learn scene components. By contrast, our method tackles these two problems by jointly optimizing attributes relevant to the quality of rendered depth and normal under a unified framework. To eliminate sophisticated modeling of neural components, we propose a roughness supervision strategy based on multi-view photometric clues. When combined with a carefully designed loss and optimization process, GS–2M produces comparable results to state-of-the-art methods, delivering triangle meshes and their associated material components for downstream tasks. We validate the effectiveness of our approach with datasets extensively used in previous works and qualitative results showcasing its performance.

To further make GS–2M approachable, we investigate practical situations where mesh reconstruction based on Gaussian splatting is applied. In this context, alternatives to COLMAP may be desirable for camera pose and intrinsic estimations—two essential ingredients for Gaussian splatting pipelines. Insight into this replacement process is provided, using examples from industrial calibration pipelines to highlight the unexpected catches and their accompanying solutions when adopting pre-defined camera parameters. Additionally, the topic of foreground object reconstruction is addressed, with examples from both real life and synthetic scenes.

Acknowledgment

Firstly, I would like to express my gratitude toward all supervisors, Dr. Malte Avenhaus, Dr. Thomas Lindemeier, and Dr. Philippe Colantoni, for their warm support throughout the completion of this thesis work.

I am also grateful to Carl ZEISS AG for providing the necessary resources and funding to keep this journey on track and enjoyable.

Additionally, the entire project is based on previous amazing Gaussian splatting works, and we acknowledge all the great publications and publicly available code that made this research possible.

Finally, this thesis work concludes my Master's with COSI — a joyful Erasmus program that has always been supportive of their students. I am profoundly thankful to the COSI board for their approval of my decision to work with Carl ZEISS AG.

Acronyms

NeRF	Neural Radiance Fields
NVS	Novel View Synthesis
3DGS	3D Gaussian Splatting
MVS	Multi-view Stereo
SfM	Structure-from-Motion
SoTA	State-of-The-Art
MLP	Multi-Layer Perceptron
SDF	Signed-Distance Function
PBR	Physically-based Rendering
BRDF	Bidirectional Reflectance Distribution Function

Notations

$\mathcal{G}$	a Gaussian primitive, packed with learnable parameters
$\mu,\ \mathbf{q},\ \mathbf{s}$	a Gaussian center in $\mathbb{R}^3$, quaterion in $\mathbb{R}^4$, and scaling vector in $\mathbb{R}^3$
$\mathbf{n},\ \hat{\mathbf{n}}$	a normal and Sobel normal vector, in $\mathbb{R}^3$
$\mathbf{a},\ \mathbf{r},\ \mathbf{m}$	BRDF parameters: albedo $\in \mathbb{R}^3$, roughness $\in \mathbb{R}$, metallic $\in \mathbb{R}$
$R,\ S$	a 3×3 rotation built from $\mathbf{q}$ and 3×3 scaling built from $\mathbf{s}$
$\Sigma,\ \Sigma'$	a 3D covariance matrix and its 2D projection in image space
$V,\ K,\ H$	a 4×4 view transform, 3×3 intrinsic, and 3×3 homography matrix
$\mathbf{p},\ \tilde{\mathbf{p}}$	pixels in image space and their homogeneous coordinates in camera space
$\mathcal{N},\ \mathcal{D},\ \mathcal{T}$	a normal, depth, and alpha map, defined for all $\mathbf{p}$
$\mathcal{A},\ \mathcal{R},\ \mathcal{M}$	an albedo, roughness, and metallic map, defined for all $\mathbf{p}$
α_i	the blending factor for Gaussian $\mathcal{G}_i$
T_i	the accumulated α-blending factor up to Gaussian $\mathcal{G}_i$ (exclusive)
$\mathcal{L}_{\text{planar}},\ \lambda_{\text{p}}$	the planar loss and its weight
$\mathcal{L}_{\text{sparse}},\ \lambda_{\text{s}}$	the sparse loss and its weight
$\mathcal{L}_{\text{mv}},\ \lambda_{\text{mv}}$	the multi-view loss and its weight
$\mathcal{L}_{\text{dn}},\ \lambda_{\text{dn}}$	the depth-normal consistency loss and its weight
$\mathcal{L}_{\text{ro}},\ \lambda_{\text{ro}}$	the roughness loss and its weight
$\mathcal{L}_{\text{sm}},\ \lambda_{\text{sm}}$	the smoothness loss and its weight
$\mathcal{L}_{\text{geo}},\ \mathcal{L}_{\text{mat}}$	the geometry and material loss
$\mathcal{L}_{\text{rgb}},\ \mathcal{L}_{\text{pbr}}$	the RGB and PBR photometric loss

Contents

Figure 1.1: *Neural rendering helps modify scene appearance without manual laborious work. The left-most column shows the original images captured under different lighting conditions. These appearances are latent coded and fed to the neural renderer to synthesize novel views with the same look, as shown in the remaining columns. By contrast, achieving the same result in traditional pipelines would involve remodeling the scene geometry and careful design of synthetic light sources. Image is taken and adapted from Meshry et al. (2019).*

1 Introduction

Neural rendering has been receiving increasing attention over the past few years. Unlike traditional forward rendering approaches that require the availability of all scene parameters before the image formation process, neural rendering optimizes its compact model parameters to generate photo-realistic imagery in a controllable way. It learns to perform the task by analyzing existing visual data from the scene, including images or videos. As described by Tewari et al. (2020), this capability of neural rendering frees content creators from the vast amount of tedious and expensive manual work of skilled artists for the creation of the underlying scene representations in terms of surface geometry, appearance/material, light sources, and even animation. Figure 1.1 illustrates one such application of neural rendering, where the appearance of the original scene is comfortably altered just by feeding the neural renderer with the latent vector of the target appearance.

1.1 Neural radiance fields

The development of neural rendering "explodes" with the advent of Neural Radiance Fields (NeRF) (Mildenhall et al., 2020). While early works in the field learn to map a semantic label or a rasterized proxy geometry directly to the output image, NeRF only optimizes its *scene representation* and delegates the rendering task to a well-developed differential graphics engine. Concretely, NeRF does not learn how to render per se. It disentangles the modeling and rendering processes to estimate the radiance and density field of a 3D scene as continuous functions. As a result, NeRF learns physically meaningful color and density values in 3D space, which physics-inspired ray casting and volume integration can then render consistently into novel views. This paradigm gives NeRF an upper hand over previous methods because optimizing only the scene representation is less subject to the hard trade-off between under- and over-fitting—classic neural network optimization challenges.

NeRF's excellence in novel view synthesis (NVS) has led to rapid developments to improve its applicability, controllability, adaptability to scenes' dynamics, and training/inference times (Tewari et al., 2021). One of these active research themes is to optimize the efficiency of NeRF and general neural rendering approaches in terms of computing resources and rendering speed. Liu et al. (2020), for example, subdivides the commonly used volumetric representation into a sparse voxel octree structure to accelerate rendering at inference time. Lindell et al. (2021), on the other hand, bypasses the bottleneck of volume integration by efficiently learning closed-form integral solutions via coordinate-based neural networks. Neff et al. (2021) and Piala and Clark (2021) further leverage depth queried from a learned network to greatly reduce the number of samples needed for computations along rays. PlenOctress (Yu et al., 2021) is another great example where a pretrained NeRF is extracted into an optimizable sparse voxel grid, resulting in three orders of magnitude rendering speedup. Plenoxel (Fridovich-Keil et al., 2022) later generalizes this idea to support voxel grids of arbitrary resolution and further accelerates training. Most recently, InstantNGP (Müller et al., 2022) achieves state-of-the-art performance speedup via efficient implementation of multi-resolution hash encoding augmented into a small neural network. Their approach seamlessly applies to NeRF and other neural-based graphics tasks, enabling near-instant training of neural graphics primitives.

Despite all these continuing advances in performance, their focus is largely within the NVS domain. In reality, NeRF family and neural-based approaches designed for accurate surface reconstruction (Yariv et al., 2021a), (Wang et al., 2021a), (Darmon et al., 2022), (Li et al., 2023) or scene decomposition (Jin et al., 2023), (Liu et al., 2023), (Li et al., 2024b) still often require hours of training on high-end GPUs to reach sufficient output quality. While there are attempts aiming for reducing runtime (Wang et al., 2023), they are limited and may come at the expense of reconstruction performance. To this end, a more efficient scene representation is desirable for neural rendering in domains that require extended training steps.

1.2 Gaussian splatting

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) emerges as an explicit representation of radiance fields. Unlike NeRF and its variants, which model empty or occupied space as continuous fields, 3DGS represents the scene using discrete, unstructured point-based primitives—3D Gaussians. To accompany the training and rendering of these primitives, 3DGS designs a tile-based rasterizer capable of back-propagating gradients to splat parameters for optimization. Thanks to their explicit nature, 3DGS eliminates the need for sampling along camera rays, a computational bottleneck in volumetric implicit-surface methods. Figure 1.2 compares the performance of 3DGS with InstantNGP and Plenoxels to highlight its excellence in NVS under a much shorter training time. Additionally, their fast Gaussian rasterizer renders a trained model in real-time and outperforms InstantNGP by a large margin.

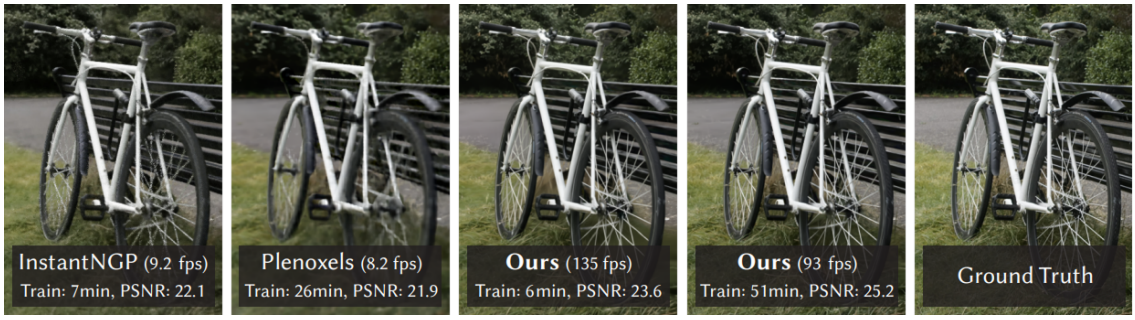

Figure 1.2: *The training time (minutes), rendering speed at inference (fps), and NVS quality (PSNR) between 3DGS and two NeRF-based state-of-the-art models: InstantNGP and Plenoxels. Image taken and adapted from Kerbl et al. (2023).*

Similar to NeRF, 3DGS has been extensively praised by the community. Since its publication, there have been numerous follow-up improvements and adoptions of 3DGS to different visual learning domains (Chen and Wang, 2025). To address aliasing artifacts, for example, Yu et al. (2024a) leverages 3D smoothing and 2D mip filters when the camera is close to or far from the Gaussians. On the other hand, Radl et al. (2024) eliminates the popping artifacts and view inconsistencies by hierarchically resorting and culling projected splats. For better densification strategies, Ye et al. (2024d) addresses 3DGS's gradient collision issue, Lyu et al. (2025) designs an adaptive densification pyramid, while Li et al. (2024c) splits thin Gaussians in the tangential space. Another line of research focuses on reconstruction quality under sparse input conditions. These usually involve the use of generative models to compensate for the lack of training data, such as diffusion (Yang et al., 2024), transformers (Chen et al., 2024b), (Zhang et al., 2025a), or monocular depth priors from pre-trained models (Zhu et al., 2023). 3DGS is also adopted for large-scale reconstructions (Kerbl et al., 2024), (Lu et al., 2024), (Lin et al., 2024), scene editing (Guédon and Lepetit, 2024a), and ray-tracing (Moenne-Loccoz et al., 2024).

1.3 Project scope and contributions

While there are many areas to explore the application of 3DGS, we focus on two domains: mesh reconstruction and material decomposition from multi-view images. These tasks have identical training input and setup, yet their end products serve different goals. In mesh reconstruction, the process takes as input captures of a target 3D object at various *viewpoints*. These captures usually manifest as 2D images or frames extracted from a video. We then optimize a set of 3D Gaussians supervised by these inputs that, after the training, closely approximate the underlying scene. Since these Gaussians tend to settle into positions where the actual object surface is constituted, they can meaningfully estimate the structure of the target object. From there, the *depth maps* from each viewpoint are rendered, using the trained Gaussians to define depth values at each pixel. These depth maps are then fused (Newcombe et al., 2011) to obtain the final 3D triangle mesh as seen in the input captures. The desire for triangle meshes is motivated by their compatibility with standard graphics pipelines and their efficiency in downstream tasks such as simulation, editing, and rendering. Yet to reconstruct them from 2D images without manual modeling work, Gaussian splatting provides a powerful intermediate representation that bridges raw image observations and geometric surfaces.

While mesh reconstruction focuses on the positions of Gaussians, material decomposition tasks rely on their orientations. These help define the *surface normals* of the target object and directly influence the reconstruction quality of material parameters, some of which include *albedo*, *roughness*, and *metallic*. Disentangling these intrinsic attributes is a challenging task because there is a certain ambiguity when the surface is highly reflective (specular). As a result, shading models are usually employed to handle the problem, with lighting information inferred and encoded in a special *cubemap*. Together, they collectively define the appearance attributes of the object in question, and like triangle meshes, these material properties are well-defined and widely consumed in conventional graphics pipelines. Extracting these parameters directly from 2D observations saves countless hours of manual work that would otherwise be required for asset creation.

There have been many works addressing either of these tasks independently. For example, the need for precise depth estimation in mesh reconstruction gears the literature toward flattening Gaussian points (Huang et al., 2024a), (Dai et al., 2024), and introducing loss terms that encourage depth-normal consistency (Yu et al., 2024b), (Chen et al., 2024a), (Wang et al., 2024). Recent works in the domain leverage multi-view clues to supervise the training (Chen et al., 2024a), (Wang et al., 2024), (Huang et al., 2024b), significantly improving the reconstruction result. Despite being excellent at recovering high-quality and water-tight meshes, these methods suffer from reflective surfaces since they cannot distinguish surface geometry from reflection images. This leads to bumpy artifacts at strongly view-dependent regions, degrading the reconstruction quality.

On the other hand, 3DGS-based works focusing on material decomposition faithfully address the issue created by reflective surfaces. They introduce shading parameters (Jiang et al., 2024), (Keyang et al., 2024) to capture the intrinsic properties of the underlying material, resulting in smoother surfaces at highly specular regions. These are quickly developed into sophisticated scene decomposition frameworks such as relighting (Bi et al., 2024), (Shi et al., 2025), or inverse rendering (Ye et al., 2024c), (Liang et al., 2024). Despite being better at handling reflective surfaces, these methods often struggle to supervise Gaussians for accurate surface geometry, with recent works borrowing neural-based backbones (Zhu et al., 2025) or building on existing mesh reconstruction approaches (Zhang et al., 2025b) to aid the geometric consistency. Yet, these attempts still require sophisticated uses of encoder/decoder and multi-layer perceptron (MLP) networks to decompose material properties, hindering their performance at scale.

To address these limitations, we propose a unified framework for jointly optimizing Gaussians for mesh reconstruction and material decomposition, referred to as GS–2M. The method eliminates the plaguing issue of reflective surfaces common to mesh reconstruction approaches by incorporating material parameters into the training pipeline to identify appearance properties of the underlying surface. At the same time, we leverage the multi-view consistency that helps supervise high-quality geometry to identify rough and smooth regions in the scene, eliminating the need for external neural-based dependencies or sophisticated MLPs. Furthermore, we construct our model design so that training can be tweaked from two intuitive hyperparameters to help adapt the training for different reconstruction goals. Experiments on widely adopted datasets reveal that our method robustly reconstructs high-fidelity meshes while being resilient to strong view-dependent effects caused by reflection images. The contributions are summarized as follows:

- A unified framework jointly optimizing Gaussians for both mesh reconstruction and material decomposition from multi-view images, delivering comparable mesh reconstruction quality to SoTA, while being resilient to reflective surfaces.
- An integration of occlusion-aware check and multi-view normal consistency for SoTA mesh reconstruction methods, improving NVS performance while maintaining the reconstruction quality.
- A roughness supervision strategy based on multi-view photometric consistency, eliminating the dependence on MLP networks for appearance modeling.
- In-practice adoption of 3DGS-based methods for mesh reconstruction, including predefined camera parameters and foreground object extractions.

The code for our implementation is published on GitHub and can be accessed via the project page at: https://ndming.github.io/publications/gs2m/. We modified the CUDA kernels and Python scripts for the new design and visualization.

2 Related Work

This chapter reviews recent Gaussian splatting methods for mesh reconstruction and material decomposition from multi-view images. For completeness, it starts from classical and neural-based reconstruction approaches in Section 2.1 and 2.2. Section 2.3 then extensively reviews the recent state-of-the-art (SoTA) 3DGS-based works that inspired GS–2M's design. Finally, Section 2.4 examines relevant datasets widely adopted for validating reconstruction performance in both domains.

2.1 Traditional methods

In their tutorial, Furukawa and Hernández (2015) highlighted reconstruction of 3D geometry from photographs as an ill-posed problem that must rely on certain assumptions of the underlying scene. Among these cues, stereo correspondence has exhibited reasonable robustness and propelled a class of reconstruction algorithms known as multi-view stereo (MVS) (Seitz et al., 2006). MVS shares largely the same principle as GS–2M and other 3DGS-based methods: reconstruct the 3D geometry of the scene from multi-view images and corresponding *camera parameters*. Structure-from-Motion (SfM) (Snavely et al., 2006) is widely adopted to recover these camera parameters for each image, with an initial sparse point cloud as a byproduct. Note, however, that for online reconstructions, as in real-time processing of video frames, Visual Simultaneous Localization and Mapping (Davison et al., 2007) is more suitable in this context. To simplify and accelerate the reconstruction process, the estimated camera parameters are generally assumed to come from a *pinhole* camera. While this assumption does not necessarily hold for cameras in practice, MVS software like COLMAP (Schönberger and Frahm, 2016) can help undistort input images to align with the pinhole camera model. Given the undistorted assets and their associated camera parameters, the 3D geometry can be reconstructed via voxel-based optimization (Seitz and Dyer, 1997), (Sinha et al., 2007), feature point growing (Furukawa and Ponce, 2010), (Wu et al., 2010), or depth fusion (Schönberger et al., 2016), (Huang et al., 2024b). The last two methods have similar characteristics to our design, in which they jointly optimize depth and surface normal for better reconstruction quality. Yet, like other traditional methods, their performance is affected by the inconsistency in appearance across input images,

making it challenging to accurately capture complete geometric representations due to the ambiguities in the correspondence. By contrast, we model the appearance properties of the target object with a handful of material parameters as part of its pipeline, making it resilient to abrupt photometric changes across views caused by reflective surfaces. Nevertheless, Huang et al. (2024b)'s visibility-aware method plays an important role in our multi-view geometric consistency implementation for handling occlusion.

2.2 Neural-based methods

As NeRF is not mainly designed for surface reconstruction, radiance fields are replaced with implicit surfaces (Niemeyer et al., 2020), (Oechsle et al., 2021), or SDFs (Yariv et al., 2020) to better define isosurfaces from a volume density and avoid high-frequency noise. Wang et al. (2021a) and Yariv et al. (2021a) later reparameterized the underlying representations of neural implicit surface to be compatible with neural volume rendering as employed by NeRF. To further increase the reconstruction quality, auxiliary information is baked into the training pipeline as priors, such as patch warping with co-visibility masks (Darmon et al., 2022), sparse SfM points (Fu et al., 2022), (Zhang et al., 2022), semantic segmentation (Guo et al., 2022), monocular depth (Sun et al., 2022), or geometric monocular (Yu et al., 2022). On the other hand, Neuralangelo (Li et al., 2023) leverages hash encodings as introduced by InstantNGP to eliminate the need for auxiliary guidance, while HF-NeuS (Wang et al., 2022) adopts coarse-to-fine optimization for improved surface details. As previously stated, these neural-based methods are SoTA in mesh reconstruction but demand expensive training resources. It is noteworthy that attempts have been made to speed up the training time of these methods (Wang et al., 2023), (Yariv et al., 2023), (Li et al., 2024a), to the point they can be comparable to the training time of 3DGS-based approaches. Yet these enhancements often come at the expense of reconstruction quality, while methods based on Gaussian splatting retain sufficient performance without requiring high computing resources.

Neural-based approaches for material decomposition often stem from inverse rendering frameworks. Early methods only consider direct lighting (Boss et al., 2021a), (Boss et al., 2021b), (Zhang et al., 2021) to trade quality for computation speed. Subsequent works (Srinivasan et al., 2021), (Yao et al., 2022), (Jin et al., 2023) introduce indirect lighting but produce inferior results on highly reflective objects. To handle objects with strong reflections, Ref-NeRF (Verbin et al., 2022) decomposes colors into diffuse and specular terms, while NeRO (Liu et al., 2023) designs a novel light representation based on Karis and Games (2013)'s split-sum approximation. Most recently, TensoSDF (Li et al., 2024b) achieves SoTA performance by expressing geometry as the implicit SDF and incorporating roughness-aware radiance fields.

2.3 3DGS-based methods

To our knowledge, limited research has explored the joint reconstruction of meshes and material parameters within Gaussian splatting frameworks. GS–ROR^2 (Zhu et al., 2025) stands out as a comparable candidate, but it largely focuses on the material decomposition task and provides relatively few experiments on its performance for mesh reconstruction. In addition, GS–ROR^2 employs TensoSDF as its backbone for SDF priors to supervise scene geometry. By doing so, they inherit TensoSDF's expensive computation cost and create a tight dependency on deep neural-based approaches. At the other extreme, GeoGaussian (Li et al., 2024c) pays more attention to the geometric properties of the Gaussian points with little regard for their appearance parameters. As a result, they improved NVS quality, but little performance was shown for mesh reconstruction or material decomposition. GS–2M, on the other hand, is the novel approach to combine these two tasks and deliver comparable SoTA reconstruction quality while not relying on any external models. The rest of this section thus reviews previous works that excel in either domain and highlights the design choices inspiring GS–2M's implementation.

Mesh reconstruction. SuGaR (Guédon and Lepetit, 2024b) was the first to adapt 3DGS for mesh reconstruction. By minimizing the SDF derived from the Gaussians and the scene density function, they encourage points to better align with the underlying surface. However, SuGaR only approximates the planarity of Gaussians with their minimum scaling axes, resulting in insufficient constraints on their overall shape during training. To address this loose approximation, 2D Gaussian Splatting (2DGS) (Huang et al., 2024a) and GaussianSurfels (Dai et al., 2024) replace 3D Gaussian points with planar primitives. In particular, they share the idea of collapsing 3D Gaussians to planar ellipses to maximize the alignment between points and the object surface. As a key takeaway, the two methods highlight the significance of the regularization term enforcing depth-normal consistency, in which the Gaussian normals and normals estimated by the gradient of depth maps are minimized. Despite achieving view-consistent geometry, 2DGS and GaussianSurfels rely on mean/median or blended z-depth in camera space, which may cause ambiguity or biased depth rendering for some scenes. Gaussian Opacity Fields (GOF) (Yu et al., 2024b), on the other hand, draws inspiration from ray-traced volume rendering of 3D Gaussians and constructs an opacity field from the trained Gaussians to extract the underlying surface by identifying its level-set. While outperforming 2DGS and GaussianSurfels to some extent, GOF suffers from the expensive resources required for its marching tetrahedra computation. Above all, PGSR (Chen et al., 2024a) and GausSurf (Wang et al., 2024) are SoTA works in mesh reconstruction. PGSR introduces unbiased depth rendering by leveraging plane depths defined by the plane normal and its distance to the camera, with multi-view constraints applied on geometry and appearance. GausSurf, meanwhile, borrows the patch-matching technique from

MVS to refine the rendered depth across views and employs normal priors from StableNormal (Ye et al., 2024b) to further supervise depth-normal consistency. Despite achieving SoTA performance, both PGSR (its exposure compensation) and GausSurf (its priors from StableNormal) depend on external MLP networks to mitigate sudden appearance variations in target objects during training. By contrast, our method solely relies on captured appearance via the integrated material decomposition pipeline and eliminates the need for external models, making its performance consistent and predictable.

Once Gaussian points are trained, the target's triangle mesh can be extracted in several ways. SuGaR and GaussianSurfels, for instance, employ Poisson reconstruction (Kazhdan et al., 2006), while GOF designs their unique marching tetrahedra method to derive meshes from the constructed opacity fields. Despite their capability to recover faithful surfaces, extracting only foreground objects with Poisson reconstruction is challenging, and GOF's method is computationally expensive. Consequently, most methods leverage TSDF fusion (Curless and Levoy, 1996) to efficiently fuse RGB-D images and extract triangle meshes from the voxel grid. This allows masking depth maps before the fusion process, enabling only the clean reconstruction of foreground objects without additional processing.

Material decomposition. As with neural-based approaches, reflective objects require special treatment among 3DGS-based methods tackling material decomposition. To address view-dependent issues stemming from high specular surfaces, Jiang et al. (2024) introduces simplified shading functions to capture light-surface interactions, while Keyang et al. (2024) employs deferred rendering with a trainable reflection strength parameter. These ideas are quickly adopted by the community and further enhanced for scene relighting (Bi et al., 2024), (Shi et al., 2025), and inverse rendering frameworks (Ye et al., 2024c), (Liang et al., 2024). At the high level, these approaches augment Gaussians with BRDF parameters and perform deferred shading where G-buffer contents are α-blended and composited into the final render. The choice of the compositing strategy varies depending on the complexity of the problem and the BRDF formulation. It can be as simple as a weighted sum between diffuse and specular components, or as sophisticated as the split-sum approximation (Karis and Games, 2013) of Kajiya (1986)'s rendering equation. To further factorize lighting, most works employ differential environment cubemaps (Laine et al., 2020) and sample them during training at various mip levels. This helps separate environment lighting from the object's intrinsic appearance and allows for more accurate material decomposition. GS-ROR^2 (Zhu et al., 2025), GlossyGS (Lai et al., 2025), and Ref-GS (Zhang et al., 2025b) are SoTA works in this context by realizing the significance of the underlying geometry to the decomposition performance. Ref-GS builds upon 2DGS to inherit its strength in surface reconstruction, while GlossyGS incorporates micro-facet features to generate neural materials. As with GS-ROR^2, they all use MLP networks to encode/decode trainable parameters.

2.4 Validation datasets

To compare different approaches to mesh and material reconstructions, several well-known datasets are employed in the literature. This section reviews four datasets making extensive appearances in previous works that we employ for experiments.

DTU. Jensen et al. (2014) published a collection of 124 thoroughly captured scenes using an industrial robot arm equipped with a structured light scanner. Each scene (scan) was imaged from either 49 or 64 calibrated viewpoints at a 1600×1200 resolution, with precise camera poses and intrinsics determined via MATLAB's Calibration Toolbox. High-quality ground truth point clouds are acquired from structured light scans, supporting accurate performance assessment for multi-view mesh reconstruction methods. While the collection encompasses a broad spectrum of objects, 15 scenes are selected for benchmarking mesh reconstruction performance in this report, as shown in Figure 2.1. Note that the choice of these scenes conforms to previous approaches that also employed the exact same set of scenes to evaluate the reconstructed mesh quality. To evaluate reconstruction performance for the DTU dataset, we use the Chamfer L_1 distance to measure the geometric similarity between the reconstructed mesh and the ground truth reference. It calculates the bidirectional distances between corresponding point sets sampled from both surfaces, and the final result is derived from the mean of the two numbers. Following previous works, the provided masks for each scene are adopted to remove non-visual hull parts before TSDF fusion, and the reconstructed mesh undergoes post-processing to keep a single largest connected component.

Figure 2.1: *15 scans of the DTU dataset for evaluating mesh reconstruction performance. The quantitative results are reported using the Chamfer distance between the extracted mesh and its corresponding ground truth point cloud.*

Figure 2.2: *6 scenes of the TnT dataset focusing on mesh reconstruction performance for uncontrolled/outdoor environments. The quantitative results are reported using the F1 score between the reconstructed points and their ground truth.*

TnT. To address the limitations of previous evaluation frameworks that were primarily conducted in controlled laboratory environments, Knapitsch et al. (2017) introduced the Tanks and Temples (TnT) dataset. Unlike earlier benchmarks, the TnT dataset features sequences acquired outside the lab under realistic conditions, including both challenging outdoor scenes and complex indoor environments, with ground-truth data captured using high-precision industrial laser scanners. The dataset is organized into training, intermediate, and advanced testing sets, where the intermediate group contains sculptures, large vehicles, and house-scale buildings with outside-looking-in camera trajectories, while the advanced group features large indoor scenes imaged from within and complex outdoor scenes with intricate geometric layouts. Following previous works (Wang et al., 2021b), (Li et al., 2023), (Huang et al., 2024a), (Yu et al., 2024b), (Chen et al., 2024a), (Wang et al., 2024), 6 scenes are selected from the TnT dataset to evaluate the performance of our method under uncontrolled and outdoor environments, as shown in Figure 2.2. As with previous works benchmarking on the TnT dataset, we use the F1 score to report the quantitative reconstruction performance for the TnT dataset. The score balances both precision and recall to provide a comprehensive assessment of reconstruction quality. Specifically, precision measures the fraction of reconstructed points that fall within a specified distance threshold from the ground truth geometry, while recall quantifies the fraction of ground truth points that have corresponding reconstructed points within the same distance threshold. The F1 score is then calculated as the harmonic mean of precision and recall, as shown in Equation 2.1. This ensures that both metrics must be high to achieve a good overall score, forming a comprehensive assessment of reconstruction quality.

$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.1}$$

Shiny Blender Synthetic. In their 2022 work, Verbin et al. (2022) introduced the Shiny Blender Synthetic dataset as a key benchmark for evaluating neural rendering methods on scenes with complex, view-dependent effects, particularly glossy and specular materials. Figure 2.3 lists all 6 objects in the dataset, each exhibits a certain appearance of highly reflective materials. The dataset helps quantify the ability of GS–2M to disentangle reflection images from the object's true color, providing qualitative comparisons with other 3DGS-based methods.

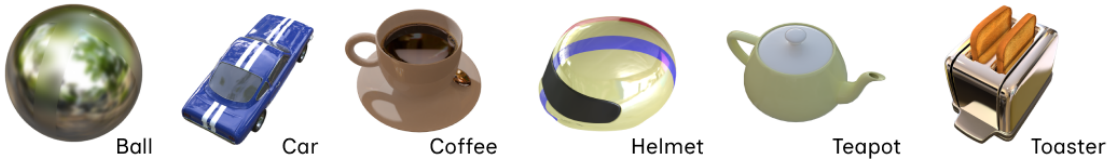

Figure 2.3: *6 highly specular objects from the Shiny Blender dataset. The strong view-dependent nature of reflective surfaces produces significant changes in appearance depending on the viewing angle, resulting in overly fitted Gaussians if material properties are not taken into account.*

Glossy Blender Synthetic. Liu et al. (2023) introduced yet another challenging benchmark full of highly intricate reflective objects—the Glossy Blender Synthetic dataset, shown in Figure 2.4. Just like with the shiny dataset, the glossy blender one is employed to quantify the limit of GS–2M when exposed to highly specular surfaces, providing qualitative comparisons with other 3DGS-based methods.

Figure 2.4: *8 challenging reflective objects from the Glossy Blender dataset designed for the reconstruction and material estimation of highly specular objects.*

3 Method

This chapter describes the theoretical design of GS–2M for joint reconstructions of meshes and materials. To set up the necessary terminology and notation, Section 3.1 gives a prelude to Gaussian splatting at a high level. Section 3.2 then provides the detailed construct of GS–2M, including the choice of point primitive, definitions of depth and normals, and augmented material parameters.

3.1 Preliminary

To represent a scene, Kerbl et al. (2023) initialized a set $\mathcal{G} = \{\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_{n-1}\}$ containing n anisotropic 3D Gaussian points. $\mathcal{G}$ can be randomly instantiated by uniformly sampling $n = 100,000$ points within the scene volume or, more commonly, constructed from an estimated sparse SfM point cloud (Snavely et al., 2006).

Each Gaussian $\mathcal{G}_i \in \mathcal{G}$ is parameterized by a center $\mu_i \in \mathbb{R}^3$, a scaling vector $\mathbf{s}_i \in \mathbb{R}^3$, and a quaternion $\mathbf{q}_i \in \mathbb{R}^4$. These are learnable parameters, with $\mathbf{s}_i$ and $\mathbf{q}_i$ further defining a scaling matrix $S_i \in \mathbb{R}^{3\times3}$ and a rotation matrix $R_i \in \mathbb{R}^{3\times3}$. Evaluating $\mathcal{G}_i$ at a position $\mathbf{x} \in \mathbb{R}^3$ thus follows Equation 3.1, where $\Sigma_i = R_i S_i S_i^\top R_i^\top$ is the 3D covariance matrix of $\mathcal{G}_i$.

$$\mathcal{G}_i(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i)} \tag{3.1}$$

Suppose there's a camera $\mathcal{C}$ in the scene whose viewing transform is $V \in \mathbb{R}^{4\times4}$, we can apply Zwicker et al. (2001)'s volume splatting method to find the 2D projection Σ_i' of Σ_i as seen by $\mathcal{C}$ using Equation 3.2. Note that V follows the convention of defining the world-to-camera (w2c) transformation for homogeneous coordinates.

$$\Sigma_i' = J_i W \Sigma_i W^\top J_i^\top \tag{3.2}$$

Here, $W \in \mathbb{R}^{3\times3}$ is the leading 3×3 rotation component of V, while J_i defines the Jacobian of the affine approximation when projecting $\mathcal{G}_i$ with $\mathcal{C}$. By discarding the third row and column of Σ_i', we obtain the 2×2 variance matrix Σ_i^{2D} describing the 2D Gaussian $\mathcal{G}_i^{\text{2D}}$ of $\mathcal{G}_i$. Note that it is preferable to define $\mathcal{G}_i^{\text{2D}}$ in image space, for which the focal lengths of $\mathcal{C}$ are incorporated in the construct of J_i.

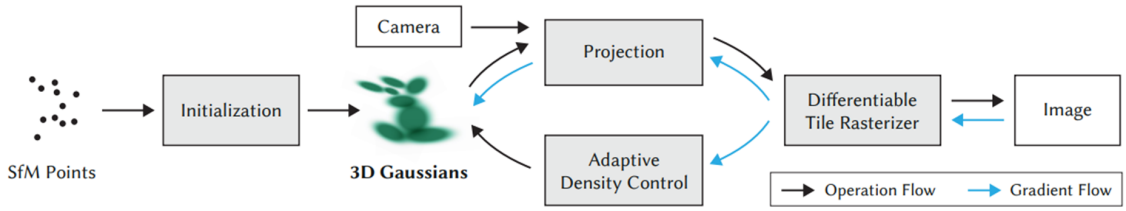

Figure 3.1: *The Gaussian splatting (Kerbl et al., 2023) pipeline. Gaussian points are first constructed from the sparse SfM point cloud, and then a series of rasterization/optimization steps is performed to update their learnable parameters.*

Given projected 2D Gaussians in image space, the alpha value α_i of a $\mathcal{G}_i^{\text{2D}}$ can be evaluated for any pixel $\mathbf{p} \in \mathbb{R}^2$ by using its variance matrix Σ_i^{2D} and projected center $\mu_i^{\text{2D}} \in \mathbb{R}^2$ of μ_i, multiplied with a learnable opacity value corresponding to $\mathcal{G}_i$. Computing μ_i^{2D} follows standard computer graphics point transforms, i.e. multiplying $[\mu_i,\ 1]^\top$ with V and subsequently the intrinsic matrix $K \in \mathbb{R}^{3\times3}$ after performing perspective division. Equation 3.3 formulates the α-blending process to calculate the linear color $C \in \mathbb{R}^3$ for any pixel $\mathbf{p}$, where the view-dependent color $\mathbf{c}_i \in \mathbb{R}^3$ is evaluated from $\mathcal{G}_i$'s learnable spherical harmonics (SH) coefficients.

$$C_{\mathbf{p}} = \sum_{G_{\mathbf{p}}} T_i \alpha_i \mathbf{c}_i, \quad T_i = \prod_{j=0}^{i-1} (1 - \alpha_j) \tag{3.3}$$

Here, $G_{\mathbf{p}}$ denotes the subset of all Gaussians $\mathcal{G}_i$ potentially contributing to the color of pixel $\mathbf{p}$. Whether a Gaussian $\mathcal{G}_i$ belongs to $G_{\mathbf{p}}$ is estimated by inspecting μ_i^{2D} and the maximum eigenvalue of Σ_i^{2D} against a threshold value. Note that the order of Gaussians in $G_{\mathbf{p}}$ matters since Gaussians must be blended front to back. As a result, a sorting procedure (Adinets and Merrill, 2022) precedes the blending process to sort Gaussians for each $G_{\mathbf{p}}$ using their depths z_i in camera space.

Once pixel colors are found for all $\mathbf{p}$, they effectively define the rendered image $\hat{\mathcal{I}}$ of all Gaussians in $\mathcal{G}$ given a viewpoint $\mathcal{C}$. Since most learnable parameters are randomly initialized, $\hat{\mathcal{I}}$ is not yet similar to the ground truth image $\mathcal{I}$ captured under the same viewpoint. A loss function $\mathcal{L}$ can therefore be defined to measure how far $\hat{\mathcal{I}}$ is to $\mathcal{I}$, and a training process is then established to minimize $\mathcal{L}$ by optimizing all learnable parameters contributing to the rasterization of $\hat{\mathcal{I}}$. Once $\mathcal{L}$ is measured, gradient descent is adopted to update parameters, as illustrated in Figure 3.1. The newly updated Gaussians are then rasterized again following the same procedure but with another viewpoint, resulting in better $\hat{\mathcal{I}}$ over time across views. This process repeats for a finite number of iterations, using the training set $\mathcal{S}$ composed of ground truth images $\mathcal{I}_k$ and their associated viewpoints $\mathcal{C}_k$. Once the training finishes, a test set containing non-overlapping viewpoints in $\mathcal{S}$ is employed to validate how well the trained Gaussians generalize to views they've never seen.

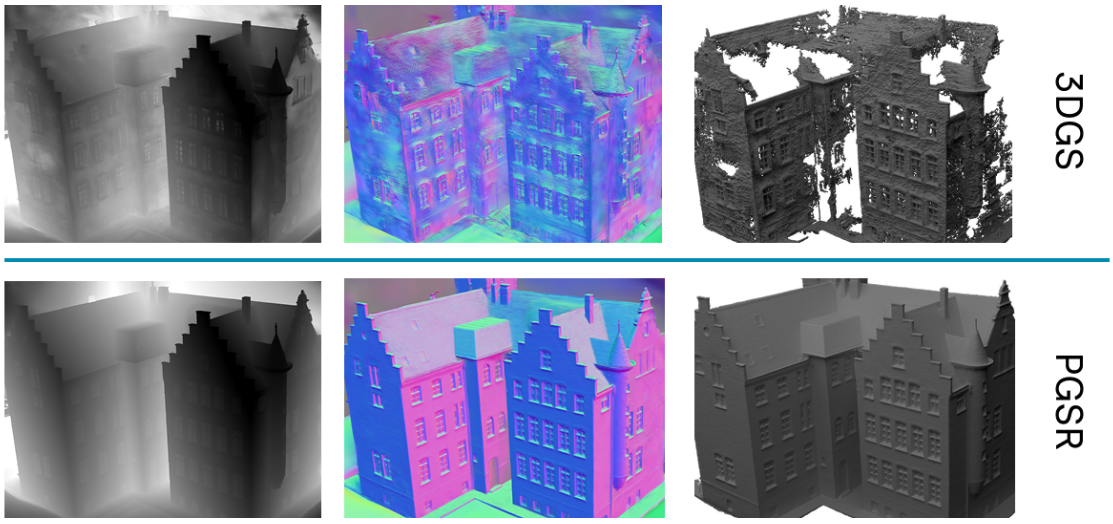

Figure 3.2: *3DGS vs. PGSR (SoTA) methods on mesh reconstruction. From left to right: the blended depth map, normal map, and reconstructed triangle mesh.*

Mesh reconstruction. The process described so far is the original design of 3DGS. After training, the Gaussians are likely to settle into positions close to the underlying surface of objects in the scene. Leveraging this insight, we can extract depth maps for all viewpoints $\mathcal{C}_k$ by modifying the blending process to replace color $\mathbf{c}_i$ with depth z_i of Gaussian $\mathcal{G}_i$ in camera space, as shown in Equation 3.4.

$$\mathcal{D} = \sum T_i \alpha_i z_i, \quad T_i = \prod_{j=0}^{i-1} (1 - \alpha_j) \tag{3.4}$$

Despite high-quality NVS results, these depth maps extracted post-training are still noisy since vanilla 3DGS is not designed for mesh reconstruction. Specifically, 3DGS's $\mathcal{L}$ only focuses on the photometric loss without regard for the consistency of Gaussian positions near the surface. Consequently, meshes reconstructed with vanilla 3DGS are not water-tight, as shown in Figure 3.2.

Material decomposition. Contrary to mesh reconstruction, material decomposition pipelines focus on the fidelity of the recovered appearance attributes of the target object. One key aspect of the trained Gaussians that play an important part in this process is the distribution of their orientations, defined by normal vectors of the underlying 3D surface. In particular, the more consistent these normals, the higher quality the recovered material parameters and lighting. Given a normal vector $\mathbf{n}_i \in \mathbb{R}^3$ defined for Gaussian $\mathcal{G}_i$, the normal map follows Equation 3.5, similar to $\mathcal{D}$.

$$\mathcal{N} = \sum T_i \alpha_i \mathbf{n}_i, \quad T_i = \prod_{j=0}^{i-1} (1 - \alpha_j) \tag{3.5}$$

3.2 GS–2M

We propose a Gaussian model built on previous outstanding works in mesh reconstruction and material decomposition domains—GS–2M. Its construct comprises different components that have been proven responsible for high-quality mesh and material extraction. This section provides an in-depth explanation of how these components are integrated into the original 3DGS work and how they are modified to tackle both tasks jointly.

Normal as shortest axis. Following the tactic employed by previous works (Keyang et al., 2024), (Jiang et al., 2024), (Liang et al., 2024), (Chen et al., 2024a), (Zhu et al., 2025), we define the normal $\mathbf{n}_i$ of each Gaussian $\mathcal{G}_i$ as the direction of its shortest axis. In particular, given a rotation matrix $R_i = \begin{bmatrix}\mathbf{r}_0 & \mathbf{r}_1 & \mathbf{r}_2\end{bmatrix}$ derived from quaternion $\mathbf{q}_i$ where each $\mathbf{r}_j \in \mathbb{R}^3$ is a column vector, the normal vector $\mathbf{n}_i$ of Gaussian $\mathcal{G}_i$ is defined following Equation 3.6, where each s_j is a scalar component from scaling vector $\mathbf{s}_i$.

$$\mathbf{n}_i = \mathbf{r}_t, \quad t = \underset{j}{\operatorname{argmin}}(s_0, s_1, s_2) \tag{3.6}$$

To put it differently, the normal direction $\mathbf{n}_i$ is the normalized column vector of R_i sitting at the position corresponding to the position of the minimum scaling factor in $\mathbf{s}_i$. This effectively provides a normal direction in a local camera space of $\mathcal{C}$, and an additional rotation step shall be performed using the leading 3×3 component of V to convert the normal to world space. Note that blending of normal maps $\mathcal{N}$ in our pipeline is performed in camera space, which is a conventional choice and should not be confused with other works (Keyang et al., 2024), (Jiang et al., 2024), (Liang et al., 2024) where normals are blended in world space.

To ensure that each normal vector $\mathbf{n}_i$ faces toward the camera, we apply a flipping step based on the view direction. Specifically, let $\mathbf{c}$ denote the position of $\mathcal{C}$ in world space, the view direction $\mathbf{v}_i$ from Gaussian $\mathcal{G}_i$ to the camera is defined as $\mathbf{v}_i = \mathbf{c} - \mu_i$. If the dot product between $\mathbf{n}_i$ and $\mathbf{v}_i$ is negative, the normal is flipped to face $\mathcal{C}$, as shown in Equation 3.7. If $\mathbf{n}_i$ is perpendicular to $\mathbf{v}_i$, its direction remains intact.

$$\mathbf{n}_i \leftarrow \begin{cases} \mathbf{n}_i, & \text{if } \mathbf{n}_i^\top \mathbf{v}_i \geq 0 \\ -\mathbf{n}_i, & \text{if } \mathbf{n}_i^\top \mathbf{v}_i < 0 \end{cases} \tag{3.7}$$

This definition of normal directions relies on the assumption that Gaussians are planar, i.e., scaling vectors with a minimum value. However, the scaling factors of Gaussians at initialization do not necessarily obey such an assumption. While there are elaborate approaches to initialize Gaussians conforming to disc-like shapes (Li et al., 2024c), we instead adopt a simple solution where a loss term is used to encourage the planar assumption, detailed in Chapter 4.

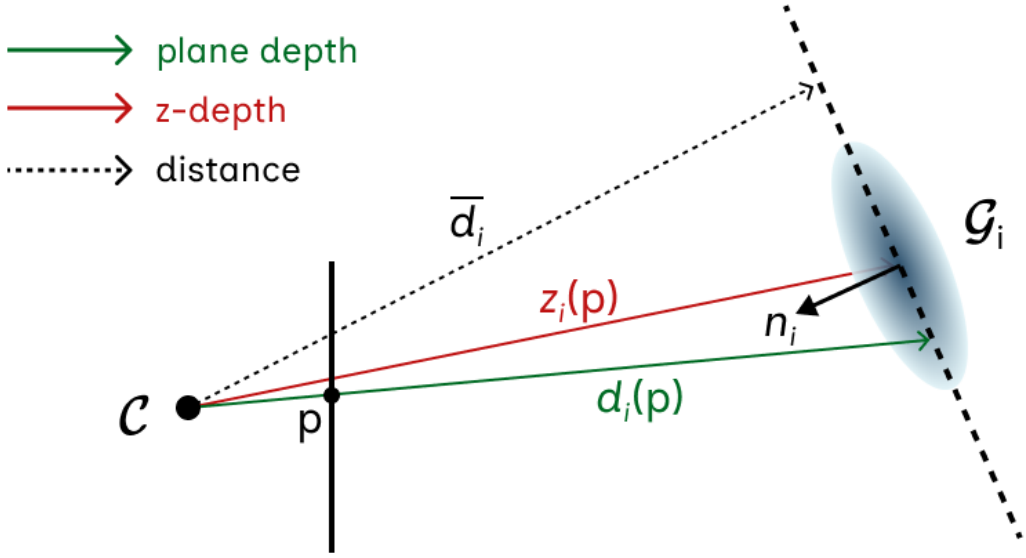

Figure 3.3: *Plane depth vs. $z-$depth. Instead of taking the $z-$coordinate of the Gaussian's center in camera space, depth values are defined by the lengths of camera rays passing through image pixels to the plane perpendicular to the Gaussian's normal. This plane depth is derived for each pixel based on the rendered distance and normal maps. The distance map is created by $\alpha-$blending the distance values $\bar{d}_i$ of all Gaussians from their planes to the camera center.*

Unbiased depth rendering. Following Chen et al. (2024a) and Huang et al. (2024b), $z-$depth in camera space is replaced with *plane depth* in our method. Specifically, depth d_i of Gaussian $\mathcal{G}_i$ for an image pixel $\mathbf{p}$ is calculated as the length of the camera ray passing through $\mathbf{p}$ to the Gaussian's normal plane. This plane is realized by the plane passing through the Gaussian's center μ_i' in camera space and accepting $\mathbf{n}_i$ as one of its normal vectors, as shown in Figure 3.3. The computation of d_i is derived from the camera-to-plane distance $\bar{d}_i$ and the normal $\mathbf{n}_i$, given a viewpoint $\mathcal{C}$. Equation 3.8 describes the calculation of $\bar{d}_i$, where $\mu_i' \in \mathbb{R}^3$ is the Gaussian center of $\mathcal{G}_i$ in camera space, and $R \in \mathbb{R}^{3\times3}$ is the rotation component (the leading 3×3 sub-matrix) of V.

$$\bar{d}_i = \mu_i' \cdot (R\mathbf{n}_i)^\top \tag{3.8}$$

These distances are $\alpha-$blended to create the distance map $\bar{\mathcal{D}}$ defined for all pixels $\mathbf{p}$. Finally, the depth map is then derived following Equation 3.9, where $\mathcal{N}$ is the blended normal map from Equation 3.5 using normal vectors defined in Equation 3.6. Note that $\tilde{\mathbf{p}}$ is the homogeneous version of $\mathbf{p}$, i.e $\tilde{\mathbf{p}} = [\mathbf{p}, 1]^\top$

$$\mathcal{D}(\mathbf{p}) = \frac{\bar{\mathcal{D}}(\mathbf{p})}{\mathcal{N}(\mathbf{p})K^{-1}\tilde{\mathbf{p}}} \tag{3.9}$$

BRDF parameters for PBR. To enable the recovery of material parameters, a physically based rendering (PBR) pipeline is integrated into GS–2M. As reviewed in Chapter 2, the shading model choice varies, depending on the ultimate goal of the application. For studies focusing on handling reflective surfaces alone, a simplified shading function (Jiang et al., 2024) or an additional reflection parameter (Keyang et al., 2024) could well do the job. For complicated scene decomposition tasks such as relighting or inverse rendering, however, more sophisticated shading pipelines are adopted (Bi et al., 2024), (Shi et al., 2025), (Ye et al., 2024c), (Liang et al., 2024). At the extreme, neural-based backbones or priors from MLP networks are incorporated to aid the optimization process (Zhu et al., 2025), (Lai et al., 2025), (Zhang et al., 2025b). As we consider material decomposition as one of the important tasks to handle, the split-sum approximation (Karis and Games, 2013) of the rendering equation is employed as part of our training pipeline. To recap, Kajiya (1986) introduced the rendering equation calculating the outgoing radiance $L_o(\mathbf{x}, \omega_o)$ from a surface point $\mathbf{x}$ in direction ω_o as in Equation 3.10.

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{S^2} f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) |\cos \theta_i| d\omega_i \tag{3.10}$$

The outgoing radiance is described as the sum of the emitted radiance $L_e(\mathbf{x}, \omega_o)$ at $\mathbf{x}$ along ω_o, and the incident radiance from all directions on the sphere S^2 centered at $\mathbf{x}$ scaled by the bidirectional reflectance distribution function (BRDF) $f_r(\mathbf{x}, \omega_i, \omega_o)$ and a cosine term. As with previous works (Liang et al., 2024), (Zhu et al., 2025), we employ the Cook-Torrance microfacet model (Cook and Torrance, 1982), (Walter et al., 2007) to formulate the BRDF, as shown in Equation 3.11.

$$f_r(\omega_i, \omega_o) = (1 - m)\frac{\mathbf{a}}{\pi} + \frac{D \times F \times G}{4(\mathbf{n} \cdot \omega_o)(\mathbf{n} \cdot \omega_i)} \tag{3.11}$$

Here, $\mathbf{a} \in [0, 1]^3$ is learnable albedo, and $m \in [0, 1]$ is estimated from the learnable roughness $\rho \in [0, 1]$ as $m = 1 - \rho$. The Fresnel term F quantifies how reflectance changes with viewing angle and is computed using Schlick's approximation (Schlick, 1994). The geometry term G accounts for self-shadowing and masking, where microfacets occlude light or block reflected rays. The normal distribution function D describes the statistical orientation of microfacets relative to the surface normal. It determines the concentration of microfacets aligned with the halfway vector $\mathbf{h}$ between light and view directions, i.e, $\mathbf{h} = (\omega_o + \omega_i)/|\omega_o + \omega_i|$. Equation 3.12 details the D term, where $\theta_\mathbf{h}$ is the angle between $\mathbf{h}$ and the surface normal $\mathbf{n}$.

$$D(\rho, \theta_\mathbf{h}) = \frac{\rho^2}{\pi(\cos^2 \theta_\mathbf{h}(\rho^2 - 1) + 1)^2} \tag{3.12}$$

The final PBR image $\bar{\mathcal{I}}$ is composed via a deferred rendering pipeline (Keyang et al., 2024), using the albedo $\mathcal{A}$, metallic $\mathcal{M}$, and roughness $\mathcal{R}$ maps blended from their respective BRDF parameters, similar to Equation 3.3.

Differential environment lighting. Since the integral term in Equation 3.10 is intractable in practice and requires specialized ray-tracing hardware for efficient sampling, Karis and Games (2013) proposed an approximation to the rendering equation. Specifically, when substituting the BRDF formulation of Equation 3.11 into Equation 3.10, the rendering equation can be decomposed into a diffuse and specular component. As light-emitting surfaces are not within the scope of GS–2M, the L_e term is dropped, and Equation 3.10 can be rewritten as:

$$L_o(\mathbf{x}, \omega_o) = \underbrace{(1-m)\frac{\mathbf{a}}{\pi}\int_\Omega L_i(\mathbf{x}, \omega_i)(\mathbf{n}\cdot\omega_i)d\omega_i}_{\text{diffuse } L_\text{d}} + \underbrace{\int_\Omega \frac{D\times F\times G}{4(\mathbf{n}\cdot\omega_o)(\mathbf{n}\cdot\omega_i)} L_i(\mathbf{x}, \omega_i)(\mathbf{n}\cdot\omega_i)d\omega_i}_{\text{specular } L_\text{s}}$$

Note that the integral domain S^2 is replaced with the hemisphere Ω about the surface point $\mathbf{x}$ and normal $\mathbf{n}$ as we do not model transmission. Additionally, the absolute cosine term becomes a simple dot product as shading now confines ω_i and ω_o to the same hemisphere. The integral term of L_d can be interpreted as the incoming irradiance to $\mathbf{x}$ given a direction; thus, it can be *prefiltered* by convolving a cubemap storing the environment radiance. At runtime, the cubemap is sampled given a viewing direction ω_o to retrieve its corresponding integral value—the diffuse light. The integral in the specular term, however, depends on ω_o and cannot be pre-computed like L_d. Therefore, the split-sum approximation proposed by Karis and Games (2013) treats L_s as two components following Equation 3.13.

$$L_\text{s} \approx \int_\Omega L_i(\mathbf{x}, \omega_i)d\omega_i \int_\Omega \frac{D\times F\times G}{4(\mathbf{n}\cdot\omega_o)(\mathbf{n}\cdot\omega_i)}(\mathbf{n}\cdot\omega_i)d\omega_i \tag{3.13}$$

The first component of this approximation assumes constant incoming lighting and is fairly accurate for common environments. This component can be prefiltered in the same manner as L_d, but it must take roughness into account, with increasing roughness values gradually blurring the cubemap. As a result, the environment lighting is precomputed at multiple *mip resolutions*, each corresponding to a roughness level. At runtime, the supplied roughness ρ will select the closest environment mipmap to sample irradiance from, avoiding the need for expensive computation. Thanks to this splitting, the BRDF's response values in the remaining term can now be pre-computed and stored in a 2D lookup texture (LUT). Similar to the first term, the 2D LUT encodes information for each roughness level and can be efficiently sampled during runtime to retrieve the integral value. The prefiltered cubemap and its mip scales make up the environment light map $\mathcal{E}$ of the scene. This environment lighting is learnable (Laine et al., 2020) to help the optimization process decompose scene and material properties.

4 Implementation

This chapter describes in detail the implementation of the total loss $\mathcal{L}$ employed by GS–2M for training and its optimization strategy to handle two reconstruction tasks in parallel. In particular, Section 4.1 provides the formulations of all loss components playing the key role in achieving detailed meshes and smooth normal distributions. Section 4.2 then explains how GS–2M is trained to reach SoTA reconstruction performance for both tasks.

4.1 Loss

As an overview, we employ up to eight loss terms for the training of GS–2M, all contributing to the total loss $\mathcal{L}$ at certain points during the optimization process. As an overview, these terms include the RGB photometric loss $\mathcal{L}_{\text{rgb}}$, the planar loss $\mathcal{L}_{\text{planar}}$, the sparse loss $\mathcal{L}_{\text{sparse}}$, the depth-normal consistency loss $\mathcal{L}_{\text{dn}}$, the multi-view loss $\mathcal{L}_{\text{mv}}$, the PBR photometric loss $\mathcal{L}_{\text{pbr}}$, the multi-view roughness loss $\mathcal{L}_{\text{ro}}$, and the smoothness loss $\mathcal{L}_{\text{sm}}$. Table 4.1 provides a high-level outline of these losses and their accompanying weights derived from previous works and empirical experiments.

Table 4.1: *Loss terms and their weights in GS–2M training. The pipeline utilizes a variety of loss terms working together to jointly deliver high-quality mesh extraction and material decomposition. Note that the provided weights are either based on empirical experiments or taken from previous works.*

Loss Term	**Symbol** $\mathcal{L}_{[\cdot]}$	**Weight** $\lambda_{[\cdot]}$
RGB photometric loss	$\mathcal{L}_{\text{rgb}}$	1.0
Planar (scaling) loss	$\mathcal{L}_{\text{planar}}$	100.0
Sparse loss	$\mathcal{L}_{\text{sparse}}$	0.001
Depth-normal consistency loss	$\mathcal{L}_{\text{dn}}$	0.015
Multi-view loss	$\mathcal{L}_{\text{mv}}$	1.0
PBR photometric loss	$\mathcal{L}_{\text{pbr}}$	1.0
Roughness loss	$\mathcal{L}_{\text{ro}}$	0.3
Smoothness loss	$\mathcal{L}_{\text{sm}}$	0.01

Equation 4.1, 4.2, and 4.3 illustrate how all loss terms are combined into the total loss, where they are further divided into groups, i.e., $\mathcal{L}_{\text{geo}}$ and $\mathcal{L}_{\text{mat}}$. In particular, loss terms not part of any group are active over the full training course. By contrast, terms belonging to a group are only activated when Gaussians have been trained for a predefined number of iterations. Section 4.2 covers this process in more detail.

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{p}}\mathcal{L}_{\text{planar}} + \lambda_{\text{s}}\mathcal{L}_{\text{sparse}} + \mathcal{L}_{\text{geo}} + \mathcal{L}_{\text{mat}} \tag{4.1}$$

$$\mathcal{L}_{\text{geo}} = \lambda_{\text{dn}}\mathcal{L}_{\text{dn}} + \lambda_{\text{mv}}\mathcal{L}_{\text{mv}} \tag{4.2}$$

$$\mathcal{L}_{\text{mat}} = \mathcal{L}_{\text{pbr}} + \lambda_{\text{sm}}\mathcal{L}_{\text{sm}} + \lambda_{\text{ro}}\mathcal{L}_{\text{ro}} \tag{4.3}$$

RGB photometric loss. $\mathcal{L}_{\text{rgb}}$ is the original loss proposed by 3DGS to supervise Gaussians with ground truth images. Concretely, $\mathcal{L}_{\text{rgb}}$ helps the initialized points settle into states ready for further optimization at later stages, with its construct provided in Equation 4.4.

$$\mathcal{L}_{\text{rgb}} = (1 - \lambda_{\text{SSIM}})L_1 + \lambda_{\text{SSIM}}L_{\text{D-SSIM}} \tag{4.4}$$

Here, L_1 is the pixel-wise mean absolute error between the rendered image $\hat{\mathcal{I}}$ (not to be confused with the PBR image $\bar{\mathcal{I}}$) and its ground truth $\mathcal{I}$, while $L_{\text{D-SSIM}}$ measures their structural similarity index following Baker et al. (2024). λ_{SSIM} in the equation is set to 0.2 identical to 3DGS and previous works.

Planar and sparse loss. These loss terms regulate the shape of Gaussians and their opacity values during training, as illustrated in Figure 4.1. Akin to $\mathcal{L}_{\text{rgb}}$, they are active over the full training course to encourage flat Gaussians and thin surfaces. Equation 4.5 defines these losses mathematically, where min selects the smallest component of an input vector, $\mathcal{V}$ denotes the set of Gaussians that are visible during the rasterization process, and $\mathcal{T}$ is the blended alpha map defined for all pixels $\mathbf{p}$.

$$\mathcal{L}_{\text{planar}} = \frac{1}{|\mathcal{V}|}\sum_{\mathcal{V}} ||\min(\mathbf{s}_i)||_1, \quad \mathcal{L}_{\text{sparse}} = \frac{1}{|\mathcal{T}|}\sum_{\mathcal{T}} [\log(\alpha_{\mathbf{p}}) + \log(1 - \alpha_{\mathbf{p}})] \tag{4.5}$$

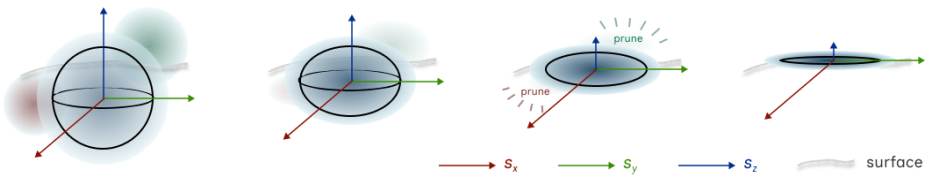

Figure 4.1: $\mathcal{L}_{\text{planar}}$ *and* $\mathcal{L}_{\text{sparse}}$ *encourage planar Gaussians and thin surfaces.*

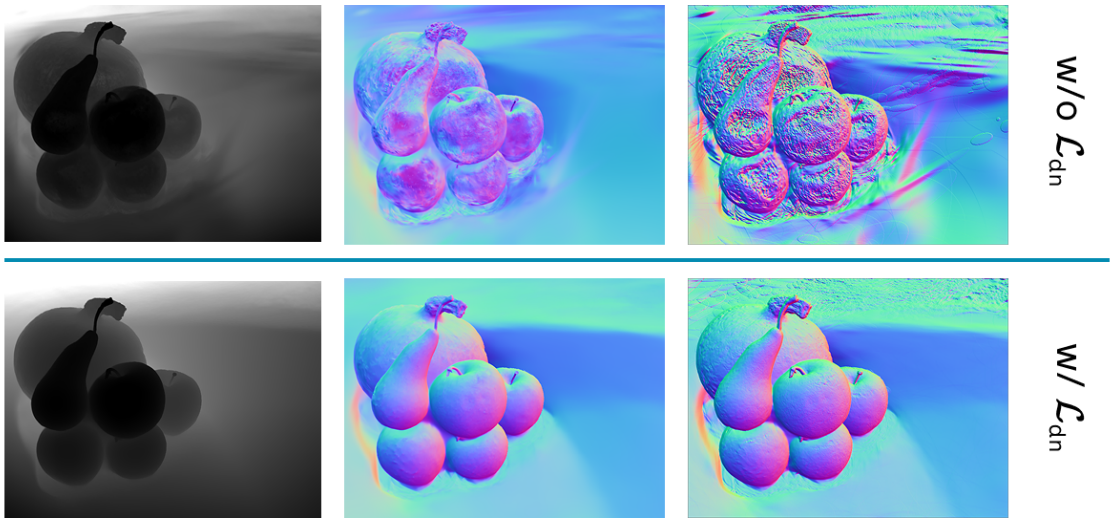

Figure 4.2: *Depth-normal consistency loss remarkably smooths out normals and refines depth precision after it is activated (bottom row), following the bootstrap stage (top row). From left to right: rendered depth maps, rendered normal maps, and normal maps derived from depth gradients.*

Depth-normal consistency loss. The depth-normal consistency loss is arguably one of the most essential loss terms responsible for high-quality depth and normal supervision (Huang et al., 2024a), (Huang et al., 2024b), (Dai et al., 2024), (Yu et al., 2024b), (Chen et al., 2024a), (Wang et al., 2024), (Jiang et al., 2024), (Liang et al., 2024), (Zhu et al., 2025). Figure 4.2 illustrates the effect of this loss term just a few optimization steps after it is activated. The term helps refine depth precision to discard floaters and smooth out normal directions to stay consistent with the underlying surface. While this mutual supervision substantially increases reconstruction quality, our experiments show that it negatively affects NVS performance—a small price to pay given that GS–2M's design is for mesh and material extraction.

The insight that makes $\mathcal{L}_{\text{dn}}$ powerful is that a normal map $\hat{\mathcal{N}}$ can be estimated for a viewpoint by applying a Sobel-like operator to the rendered depth map $\mathcal{D}$. The process first uses depth values from $\mathcal{D}$ and camera intrinsics K to find 3D points $\mathbf{p}_c$ for all pixel coordinates $\mathbf{p}$ in camera space. It then computes for each $\mathbf{p}_c$ two directional vectors, $\boldsymbol{g_u}$ and $\boldsymbol{g_v}$, estimating the local gradients by using 4 neighbors of $\mathbf{p}_c$. The normal direction at $\mathbf{p}_c$ can now be derived by taking the cross product of these gradients, as described in Equation 4.6. Equation 4.7 derives $\mathcal{L}_{\text{dn}}$, using norm-1 to compute the error between the rendered normal map $\mathcal{N}$ and $\hat{\mathcal{N}}$.

$$\mathbf{p}_c = \mathcal{D}(\mathbf{p})K^{-1}\tilde{\mathbf{p}}, \quad \hat{\mathcal{N}} = \frac{\boldsymbol{g_u} \times \boldsymbol{g_v}}{|\boldsymbol{g_u} \times \boldsymbol{g_v}|} \tag{4.6}$$

$$\mathcal{L}_{\text{dn}} = \frac{1}{|\mathcal{N}|} \sum_{\mathcal{N}} ||\mathbf{n_p} - \hat{\mathbf{n}}_\mathbf{p}||_1 \tag{4.7}$$

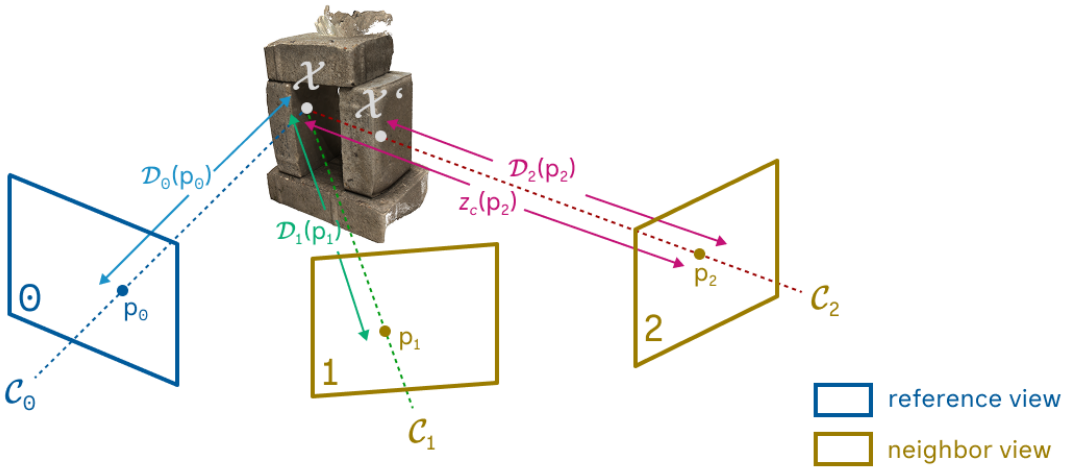

Figure 4.3: *Robust occlusion check employed by GS–2M for multi-view geometric consistency. Only pixels satisfying $z_c < \mathcal{D}(\mathbf{p})$ in neighbor views are counted.*

Multi-view loss. Until recently, multi-view consistency received credit for boosting mesh reconstruction performance (Chen et al., 2024a), (Wang et al., 2024), (Huang et al., 2024b). Consequently, GS–2M gracefully adopts and builds upon Chen et al. (2024a)'s implementation to incorporate and enhance the multi-view loss term.

$\mathcal{L}_{\text{mv}}$ comprises of multi-view geometric and photometric errors, denoted L_{g} and L_{p}. The loss is computed as their weighted sum, as shown in Equation 4.8, where λ_{g} and λ_{p} are set to 0.03 and 0.15, respectively.

$$\mathcal{L}_{\text{mv}} = \lambda_{\text{g}} L_{\text{g}} + \lambda_{\text{p}} L_{\text{p}} \tag{4.8}$$

To stimulate geometric consistency across views, Chen et al. (2024a) performs forward and backward projections of pixels in a reference view $\mathcal{C}_{\text{r}}$ with a neighbor view $\mathcal{C}_{\text{n}}$ and minimizes the reprojection error. However, they approximate occlusion regions by thresholding large projection noises, which is unreliable and ambiguous. Taking inspiration from Huang et al. (2024b), GS–2M instead explicitly detects and rejects these invalid pixels by comparing depth values in the neighbor view's depth map with z-depths of back-projected points in the same camera space. Figure 4.3 depicts this robust occlusion check, where corresponding pixels in neighbor views with depth values less than their camera z-depth are rejected from multi-view consideration. Additionally, a decay rate r_{g} is incorporated into the exponential weight involved in L_{g} computation, as shown in Equation 4.9. Here, $\hat{\mathbf{p}}_{\text{n}}$ and $\hat{\mathbf{n}}_{\text{n}}$ are pixels and their normals in a neighbor view corresponding to pixels $\mathbf{p}_{\text{r}}$ and normals $\mathbf{n}_{\text{r}}$ in the reference view, respectively. These correspondences are identified through forward and backward projections with occlusion checks, as discussed above.

$$L_{\text{g}} = \exp(-r_{\text{g}} ||\hat{\mathbf{p}}_{\text{n}} - \mathbf{p}_{\text{r}}||_1)(||\hat{\mathbf{p}}_{\text{n}} - \mathbf{p}_{\text{r}}||_1 + 2\arccos(\hat{\mathbf{n}}_{\text{n}} \cdot \mathbf{n}_{\text{r}})) \tag{4.9}$$

For multi-view photometric consistency loss, we compute the L_{p} term identically to Chen et al. (2024a)'s implementation, i.e., the normalized cross-correlation (NCC) (Yoo and Han, 2009) between warped patches $\mathcal{P}_{\mathrm{r}}$ and $\hat{\mathcal{P}}_{\mathrm{n}}$ sampled from reference and neighbor views, respectively. As before, corresponding patches in neighbor views are found via homography matrices H_{rn} that transform pixels from $\mathcal{C}_{\mathrm{r}}$ to $\mathcal{C}_{\mathrm{n}}$. In particular, Equation 4.10 defines L_{p}, where exponential weighting is applied as with the geometric term.

$$L_{\mathrm{p}} = \frac{1}{|\mathcal{P}|} \sum_{\mathcal{P}} \exp(-r_{\mathrm{g}} ||\hat{\mathbf{p}}_{\mathrm{n}} - \mathbf{p}_{\mathrm{r}}||_1)(1 - \mathrm{NCC}(\hat{\mathcal{P}}_{\mathrm{n}}(\hat{\mathbf{p}}_{\mathrm{n}}), \mathcal{P}_{\mathrm{r}}(\mathbf{p}_{\mathrm{r}}))) \tag{4.10}$$

Multi-view supervision for roughness. As stated earlier, we recognize the multi-view photometric NCC error as an indicator for highly reflective regions. The intuition is that if the surface nature is strongly view-dependent, switching to nearby viewpoints will cause the warped patches to change significantly. To take advantage of this observation, roughness values at pixels whose NCC error exceeds a certain threshold are added to $\mathcal{L}_{\mathrm{ro}}$, while those less than the same threshold are subtracted from $\mathcal{L}_{\mathrm{ro}}$. This threshold can be specified before training based on different reflective appearances, allowing the model to correctly identify surface roughness. To aid the selection of this threshold, the NCC error is rescaled using the θ function to mimic the behavior of the D term in Equation 3.12, as illustrated in Figure 4.4.

$$\theta(x) = (1 - \mathrm{sigmoid}(x)) \times \sqrt{x} + \mathrm{sigmoid}(x) \times x^{2.5} \tag{4.11}$$

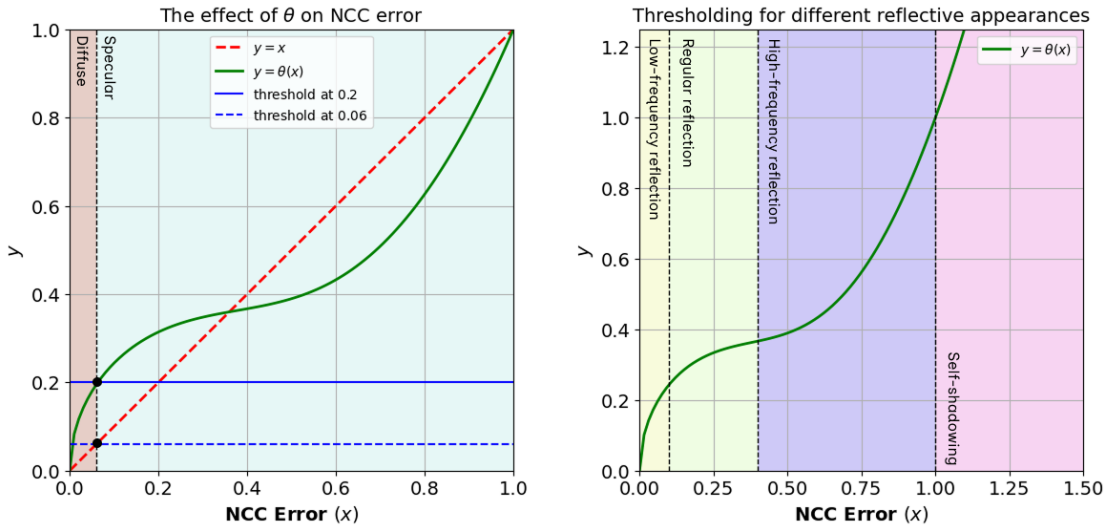

Figure 4.4: *NCC errors are rescaled with the θ function (left) to aid the selection of the reflection threshold. Different reflective appearances (right) correspond to an approximate range of thresholding values.*

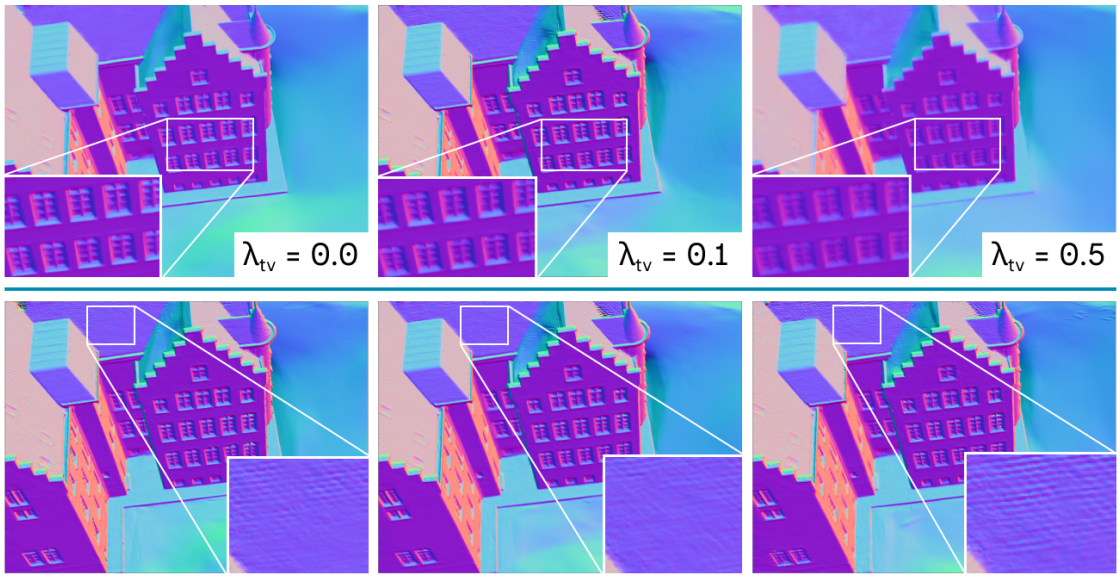

Figure 4.5: *Total variation loss applied on the distribution of normal directions doesn't negatively affect geometric details. The top row shows the rendered normal maps at different weight values of $\mathcal{L}_{\text{tv}}$, while the bottom row shows their corresponding depth-derived normal maps. As depth maps are directly responsible for surface accuracy, the geometric details are preserved regardless of how uniform the normals are due to TV loss. This not only boosts mesh reconstruction performance on some scenes, but also smooths out normals for material decomposition.*

Smoothness loss. The $\mathcal{L}_{\text{sm}}$ term is responsible for the smoothness of normals and BRDF parameters. It combines the weighted total variance (TV) of the normal map and roughness maps, as shown in Equation 4.12.

$$\mathcal{L}_{\text{sm}} = \frac{\lambda_{\text{tv}}}{|\mathcal{N}|} \sum_{u,v} \Delta_{uv}^{\mathcal{N}} + \frac{\lambda_{\text{sm}}}{|\mathcal{R}|} \sum_{u,v} \Delta_{uv}^{\mathcal{R}} \tag{4.12}$$

The $\Delta_{uv}^{\mathcal{N}}$ term further smooths out normal directions (Jiang et al., 2024), (Chen et al., 2025) without negatively affecting the geometric details. Figure 4.5 illustrates this idea, where λ_{tv} is gradually increased to amplify the effect of $\Delta_{uv}^{\mathcal{N}}$. The resulting normal map is more uniform as λ_{tv} grows, yet the depth map still preserves its details regardless of how large λ_{tv} is. Equation 4.13 formulates $\Delta_{uv}^{\mathcal{N}}$ given the blended albedo map $\mathcal{A}$, and normal map $\mathcal{N}$, defined for all image pixels $\mathbf{p} = (u, v)$. Unlike previous works, we employ the rendered albedo map instead of ground truth images for edge-aware weighting to prevent reflection artifacts from contaminating the smoothness of normal vectors.

$$\begin{aligned} \Delta_{uv}^{\mathcal{N}} = &\exp(-||\mathcal{A}_{u,v} - \mathcal{A}_{u-1,v}||_1)||\mathcal{N}_{u,v} - \mathcal{N}_{u-1,v}||_1 \\ &+ \exp(-||\mathcal{A}_{u,v} - \mathcal{A}_{u,v-1}||_1)||\mathcal{N}_{u,v} - \mathcal{N}_{u,v-1}||_1 \end{aligned} \tag{4.13}$$

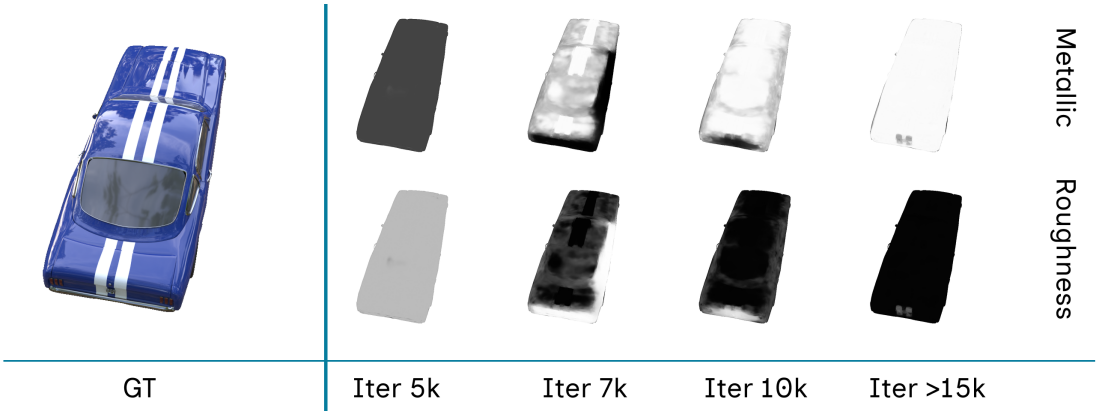

Figure 4.6: *Provoking smoothness for rendered roughness and metallic maps via the $\Delta_{uv}^{\mathcal{X}}$ term of $\mathcal{L}_{\text{sm}}$. For the first few thousand iterations, only regions with photometric variations due to reflection are penalized (via $\mathcal{L}_{\text{ro}}$) and become smoother (darker for roughness maps) than regions lacking reflection images. $\Delta_{uv}^{\mathcal{X}}$ helps propagate these correctly identified smooth regions to the surrounding, resulting in uniform roughness maps. Since $\mathcal{L}_{\text{ro}}$ relies entirely on multi-view photometric inconsistency, smooth regions with very low-frequency reflective clues are hard to detect. They must therefore be supervised via $\mathcal{L}_{\text{sm}}$, a loss term controlled by the optimization hyperparameter λ_{sm}.*

While $\Delta_{uv}^{\mathcal{N}}$ helps smooth out normal variations, the $\Delta_{uv}^{\mathcal{R}}$ term is responsible for propagating correctly identified roughness regions. As illustrated in Figure 4.6, $\mathcal{L}_{\text{ro}}$ helps identify reflective regions at the start of training, yet it cannot detect the same appearance at the remaining parts due to the lack of reflection clues. With the help of $\Delta_{uv}^{\mathcal{R}}$, these identified smooth regions are propagated as the training proceeds, resulting in more uniform surfaces over time. This behavior can be controlled via λ_{sm} to help the optimization process converge toward the desired surface appearance. Equation 4.14 defines $\Delta_{uv}^{\mathcal{R}}$, where the blended albedo map $\mathcal{A}$ is again employed for edge-aware weighting.

$$\begin{aligned}\Delta_{uv}^{\mathcal{R}} = &\exp(-||\mathcal{A}_{u,v} - \mathcal{A}_{u-1,v}||_1)||\mathcal{R}_{u,v} - \mathcal{R}_{u-1,v}||_1 \\ &+ \exp(-||\mathcal{A}_{u,v} - \mathcal{A}_{u,v-1}||_1)||\mathcal{R}_{u,v} - \mathcal{R}_{u,v-1}||_1\end{aligned} \tag{4.14}$$

PBR photometric loss. The PBR photometric loss is defined similarly to $\mathcal{L}_{\text{rgb}}$, i.e., combining the pixel-wise mean absolute error and the D-SSIM error between the PBR composite $\bar{\mathcal{I}}$ and the ground truth image $\mathcal{I}$. Equation 4.15 defines $\mathcal{L}_{\text{pbr}}$, where SSIM measures the D-SSIM index between $\bar{\mathcal{I}}$ and $\mathcal{I}$.

$$\mathcal{L}_{\text{pbr}} = (1 - \lambda_{\text{SSIM}})||\bar{\mathcal{I}} - \mathcal{I}||_1 + \lambda_{\text{SSIM}}(1 - \text{SSIM}(\bar{\mathcal{I}},\ \mathcal{I})) \tag{4.15}$$

4.2 Optimization

As with other works building on 3DGS, GS–2M employs largely the same set of optimization parameters as the original work. These include the learning rates of trainable parameters and densification thresholding hyperparameters. The only exceptions are the learning rate of opacity, which is raised to 0.05, and the densification percentage, which is lowered to 0.001.

Two-stage training. GS–2M is trained for 30,000 iterations for each scene and split into two stages: bootstrap and joint optimization. During the bootstrap stage, the total loss only accounts for the RGB photometric, planar, and sparse terms. This helps set up the general point shapes and positions before the joint optimization stage, during which the remaining losses are activated. To help provoke smooth normals at highly reflective surfaces, a weight map $\mathcal{W}$ is derived from the roughness map $\mathcal{R}$ following Equation 4.16 to weight $\Delta_{uv}^{\mathcal{N}}$. The effect of $\mathcal{W}$ helps amplify uniform normals at highly specular regions and suppress them at diffuse parts, resulting in smooth normals while preserving geometric details. The 2.5 power further suppresses the effect of $\Delta_{uv}^{\mathcal{N}}$ at rough regions and makes the weighting aligned with the scaling θ function discussed in Equation 4.11.

$$\mathcal{W} = (1.0 - \mathcal{R})^{2.5} \tag{4.16}$$

To further improve $\mathcal{L}_{\text{dn}}$, gradients from ground truth RGB images are also incorporated as edge-aware weighting. However, these gradients are contaminated by the reflection images and cannot be employed as reliable weighting sources. As a result, we replace them with rendered albedo maps $\mathcal{A}$, and suppress the $L1$ contribution (between $\hat{\mathcal{I}}$ and $\mathcal{I}$) to $\mathcal{L}_{\text{rgb}}$ in the joint optimization stage. The weighting is computed as shown in Equation 4.17, where the power of two further suppresses high-frequency details from the image gradients.

$$\mathcal{W}_{\text{grad}} = (1 - ||\nabla \mathcal{A}||_1)^2 \tag{4.17}$$

Densification strategy. As Ye et al. (2024d) points out that the original densification strategy of 3DGS suffers from gradient collision, their absolute gradient method (AbsGS) is adopted for GS–2M. This helps eliminate blurry artifacts at high-frequency regions and allows for a more fine-grained splitting strategy. In addition, their simple pruning method based on COLMAP visualization is integrated into the pipeline to prune initial outliers before training. It's worth mentioning that Kheradmand et al. (2024)'s work, where the densification strategy is formulated as a Monte-Carlo Markov Chain sampling process, is outstanding among densification methods. However, adopting their approach to GS–2M requires careful fine-tuning of parameters, which is left as a future perspective.

Mesh extraction. As discussed in Chapter 2, we employ TSDF fusion (Newcombe et al., 2011) to extract triangle meshes post-training. The procedure starts by rendering color and depth maps for all viewpoints to obtain RGB-D images. The rendered depth maps are further filtered using ground-truth alpha masks if they are provided, allowing for foreground mesh extraction. The RGB-D images are then integrated via Open3D (Zhou et al., 2018) to obtain the TSDF volume. From there, the resulting triangle mesh is extracted and further post-processed to keep only one cluster. This process produces a polygonal mesh with an RGB color baked to each vertex. However, fusing depth maps with extended attributes is not currently supported by Open3D, and extending its capability to bake auxiliary vertex attributes is an interesting derivative work.

5 Experiments

We conduct experiments to validate the effectiveness of GS–2M. Specifically, Section 5.1 provides results when benchmarking GS–2M with datasets designed for mesh reconstruction assessment. Section 5.2, on the other hand, provides qualitative results when applying GS–2M to datasets containing highly reflective surfaces. All experiments are carried out using a single RTX 4090 GPU with 24GB VRAM, with the model split into two variants:

- **Ours** w/o BRDF: training is performed on GS–2M without joint optimization. This helps validate the effectiveness of the new occlusion-aware and multi-view normal consistency integrated into the multi-view geometric loss.
- **Ours**: the full GS–2M training, as described in Chapter 3 and 4.

5.1 Mesh reconstruction

Table 5.1 compares mesh reconstruction performance of GS–2M with SoTA neural-based and 3DGS-based methods using the DTU benchmark. The Chamfer Distances (CD) are reported for 15 scans as introduced in Chapter 2, together with the average runtime and GPU employed for each method. The results of previous works (including runtime and GPU) are taken directly as reported in their original papers. Note that runtimes are not directly comparable, as these methods are trained on different hardware. Nevertheless, it is clear that all 3DGS-based methods, including GS–2M, consume significantly less training time and computing resources.

The quantitative results show that the integration of the new occlusion-aware check still keeps GS–2M's performance on par with SoTA reconstruction methods. However, the new integration helps GS–2M reduce the total number of Gaussian points post-training, as detailed in Table 5.2. Moreover, the new occlusion-aware check boosts GS–2M NVS performance, outperforming all SoTA reconstruction methods, as shown in Table 5.3. On the other hand, when GS–2M is augmented with BRDF parameters, there's a slight drop in its reconstruction performance, but the full model still outperforms all SoTA neural-based approaches and most 3DGS-based methods. Despite this, GS–2M is able to cope with highly reflective surfaces, as will be shown in the next section.

Table 5.1: *Quantitative results of mesh reconstruction performance for the DTU dataset. The Chamfer Distance (CD) ↓ for each scan of the 15 scenes in the DTU dataset is reported. We compare our approach with SoTA neural-based methods: VolSDF (Yariv et al., 2021b), NeuS (Wang et al., 2021b), RegSDF (Zhang et al., 2022), NeuS2 (Wang et al., 2023), NeuralWarp (Darmon et al., 2022), Neuralangelo (Li et al., 2023); and 3DGS-based methods: SuGaR (Guédon and Lepetit, 2024b), GaussianSurfels (Dai et al., 2024), 2DGS (Huang et al., 2024a), GOF (Yu et al., 2024b), PGSR (Chen et al., 2024a), GausSurf (Wang et al., 2024). The top-three best performing results are highlighted in color for ease of visualization, with red, orange, and yellow indicating 1st, 2nd, and 3rd, respectively. Note that the reconstruction runtimes listed here are not directly comparable since these methods are trained on different GPUs. The hardware and runtime info was taken from the original papers and GitHub discussions, where unpublished GPU info is assigned as H-E for High-End.*

CD ↓	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean	Time	GPU
VolSDF	1.14	1.26	0.81	0.49	1.25	0.70	0.72	1.29	1.18	0.70	0.66	1.08	0.42	0.61	0.55	**0.86**	∼ 12h	V100
NeuS	0.83	0.98	0.56	0.37	1.13	0.59	0.60	1.45	0.95	0.78	0.52	1.43	0.36	0.45	0.45	**0.77**	∼ 8h	RTX 3090
RegSDF	0.60	1.41	0.64	0.43	1.34	0.62	0.60	0.90	0.92	1.02	0.60	0.59	0.30	0.41	0.39	**0.72**	∼ 3.5h	H-E
NeuS2	0.56	0.76	0.49	0.37	0.92	0.71	0.76	1.22	1.08	0.63	0.59	0.89	0.40	0.48	0.55	**0.70**	∼ 5m	RTX 3090
NeuralWarp	0.49	0.71	0.38	0.38	0.79	0.81	0.82	1.20	1.06	0.68	0.66	0.74	0.41	0.63	0.51	**0.68**	> 12h	H-E
Neuralangelo	0.37	0.72	0.35	0.35	0.87	0.54	0.53	1.29	0.97	0.73	0.47	0.74	0.32	0.41	0.43	**0.61**	∼ 16h	A100
SuGaR	1.47	1.33	1.13	0.61	2.25	1.71	1.15	1.63	1.62	1.07	0.79	2.45	0.98	0.88	0.79	**1.33**	15–45m	V100
GaussianSurfels	0.66	0.93	0.54	0.41	1.06	1.14	0.85	1.29	1.53	0.79	0.82	1.58	0.45	0.66	0.53	**0.88**	6.67m	RTX 4090
2DGS	0.48	0.91	0.39	0.39	1.01	0.83	0.81	1.36	1.27	0.76	0.70	1.40	0.40	0.76	0.52	**0.80**	10.9m	RTX 3090
GOF	0.50	0.82	0.37	0.37	1.12	0.74	0.73	1.18	1.29	0.68	0.77	0.90	0.42	0.66	0.49	**0.74**	18.4m	A100
PGSR	0.36	0.57	0.38	0.33	0.78	0.58	0.50	1.08	0.63	0.59	0.46	0.54	0.30	0.38	0.34	**0.52**	30m	RTX 4090
GausSurf	0.35	0.55	0.34	0.34	0.77	0.58	0.51	1.10	0.69	0.60	0.43	0.49	0.32	0.40	0.37	**0.52**	7.2m	RTX 3090
Ours w/o BRDF	0.34	0.55	0.41	0.34	0.75	0.49	0.48	1.06	0.62	0.57	0.49	0.54	0.32	0.38	0.35	**0.51**	22.4m	RTX 4090
Ours	0.36	0.60	0.38	0.33	0.73	0.53	0.53	1.09	0.78	0.59	0.54	0.64	0.77	0.40	0.38	**0.58**	33.1m	RTX 4090

Table 5.2: *Comparing GS–2M's total number of Gaussian points post-training with leading 3DGS-based methods in mesh reconstruction: 2DGS (Huang et al., 2024a), GOF (Yu et al., 2024b), and PGSR (SoTA) (Chen et al., 2024a). The top-three best optimal numbers are highlighted, with red, orange, and yellow indicating 1st, 2nd, and 3rd, respectively. Thanks to the enhanced occlusion-aware multi-view loss, GS–2M significantly reduces the total number of points post-training while maintaining the reconstruction quality on par with SoTA methods. For scenes where GS–2M produces more points than the others, it tends to give better reconstruction quality (scans 83, for example).*

# Points ↓	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
3DGS	681k	790k	987k	787k	257k	328k	298k	213k	471k	282k	302k	135k	343k	170k	168k
GOF	584k	549k	739k	686k	277k	298k	307k	220k	450k	353k	280k	133k	620k	169k	156k
PGSR (SoTA)	337k	392k	437k	530k	284k	233k	221k	116k	394k	159k	301k	126k	142k	158k	158k
2DGS	300k	372k	379k	311k	177k	178k	187k	177k	366k	154k	138k	97k	161k	107k	147k
Ours w/o BRDF	185k	256k	381k	290k	130k	134k	106k	527k	226k	191k	81k	156k	185k	136k	92k
Ours	290k	376k	444k	396k	133k	161k	170k	164k	273k	292k	128k	193k	246k	141k	172k

Table 5.3: *Quantitative results of NVS performance for the DTU dataset. The Peak Signal-to-Noise Ratio (PSNR) ↑ for each scan of the 15 scenes in the DTU dataset is reported. We compare our approach with SoTA neural-based methods: VolSDF (Yariv et al., 2021b), NeuS (Wang et al., 2021b), RegSDF (Zhang et al., 2022), NeuS2 (Wang et al., 2023), Neuralangelo (Li et al., 2023); and PGSR (Chen et al., 2024a). Except for neural-based approaches, all NVS metrics are reproduced using the published code and evaluated on unmasked images. The top-three best performing results are highlighted, with red, orange, and yellow indicating 1st, 2nd, and 3rd, respectively. The incorporation of multi-view normal consistency and occlusion-aware check improves NVS performance without negatively affecting the extracted mesh quality. The new integration helps GS–2M outperform all SoTA reconstruction methods. Despite this improvement, our full model with BRDF parameters suffers from a drop in NVS due to the immature fine-tuning. We suspect the root cause lies in the blending of normal vectors in local space, which may not be transformed well to world space when performing PBR. Note that the NVS performance for the DTU dataset is evaluated on the whole training data, aligning with previous works.*

PSNR ↑	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean
RegSDF	24.78	23.06	23.47	22.21	28.57	25.53	21.81	28.89	26.81	27.91	24.71	25.13	26.84	21.67	28.25	**25.31**
NeuS	26.49	26.17	27.66	27.78	30.63	27.42	25.38	30.00	26.40	29.63	25.87	28.82	28.80	27.36	31.19	**28.00**
NeuS2	28.44	27.14	29.70	29.67	31.75	27.83	24.84	31.24	26.86	30.57	26.05	28.93	28.98	27.82	32.48	**28.82**
VolSDF	26.28	25.61	26.55	26.76	31.57	31.50	29.38	33.23	28.03	32.13	33.16	31.49	30.33	34.90	34.75	**30.38**
Neuralangelo	30.64	27.78	32.70	34.18	35.15	35.89	31.47	36.82	30.13	35.92	36.61	32.60	31.20	38.41	38.05	**33.84**
PGSR	32.20	28.07	31.10	33.41	33.99	33.22	32.16	32.83	31.31	33.82	36.03	34.64	32.71	37.33	37.09	**33.33**
Ours w/o BRDF	33.41	30.25	32.54	33.76	35.26	33.32	31.91	32.82	31.14	34.16	36.53	35.15	32.83	37.79	37.37	**33.88**
Ours	27.84	24.33	26.83	27.27	28.86	25.22	26.79	24.91	25.00	24.72	28.01	26.15	28.47	28.14	28.47	**26.73**

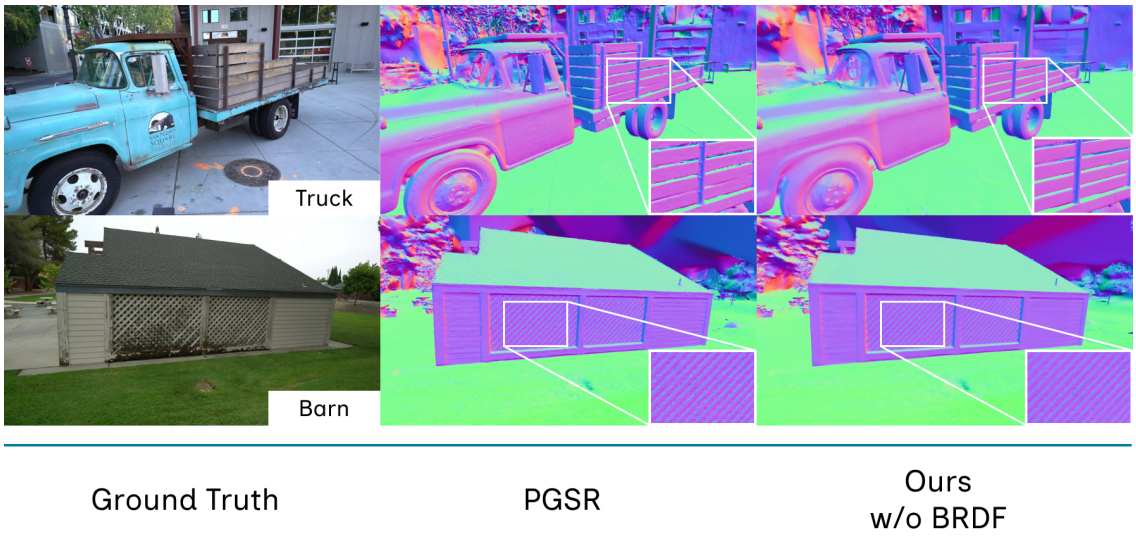

Figure 5.1: *Qualitative comparisons between PGSR (Chen et al., 2024a) and our method w/o BRDF for the Truck and Barn scenes from the TnT dataset. This highlights the effectiveness of the incorporation of multi-view normal consistency into $\mathcal{L}_{\text{mv}}$, resulting in smoother and more consistent normals.*

Table 5.4: *Quantitative results of mesh reconstruction performance for the Truck and Barn scenes from the TnT dataset, where the F1-score ↑ is reported. We compare our reconstruction with NeuS (Wang et al., 2021b), Neuralangelo (Li et al., 2023), SuGaR (Guédon and Lepetit, 2024b), 2DGS (Huang et al., 2024a), GOF (Yu et al., 2024b), and PGSR (Chen et al., 2024a).*

F1-Score ↑	NeuS	Geo-NeuS	Neuralangelo	SuGaR	2DGS	GOF	PGSR	**Ours** w/o BRDF
Barn	0.29	0.33	0.70	0.14	0.36	0.51	0.66	0.57
Truck	0.45	0.45	0.48	0.26	0.26	0.58	0.66	0.67
Mean	0.37	0.39	0.59	0.15	0.31	0.55	0.66	0.62

For experiments performed on the TnT dataset, we only select the Barn and Truck scenes as reconstructions on the remaining scenes result in out-of-memory errors at runtime. This is one of the drawbacks of our method since there is not yet a strategy to cap the maximum number of Gaussians produced during training. The promising approach to address this issue is to reformulate the densification as Monte-Carlo Markov Chains, proposed by Kheradmand et al. (2024). Nevertheless, Figure 5.1 reveals that our incorporation of multi-view normal consistency into the multi-view loss produces smoother and more consistent normals at regions with high-frequency textures. We do not test our full model on the TnT dataset as it is designed for surface reconstruction only, and the training of BRDF parameters would be overwhelmed by the vast background details. Table 5.4 provides quantitative comparisons using the F1-score between our method and other SoTA approaches.

5.2 Material decomposition

Figure 5.2 showcases the qualitative performance of GS–2M for decomposing material properties on the challenging Glossy Blender Synthetic dataset. We include the reconstructed normals, BRDF parameters, shaded diffuse, specular, and final composite images. The experiment reveals that our model performs faithfully for the majority of the dataset, with only two failed reconstructions on the Potion and T-bell scenes. These scenes possess a mixture of low- and high-frequency reflective clues, causing challenges for the training process to select an appropriate reflection threshold without distorting the geometry. This is yet another drawback of our approach as we aim for non-MLP training, resulting in insufficient learnable capabilities for intricate reflection images. Nevertheless, our method is still able to robustly decompose surface appearance for the remaining scenes of the Glossy Blender Synthetic dataset, hinting at the potential to improve the multi-view roughness supervision for more intricate reflective clues. Despite this, we notice our PBR pipeline struggles to prevent the environment lighting from bleeding into the albedo component, as can be observed for the Cat and Bell scenes. This artifact stems from the insufficient constraints on the environment lighting, forcing the albedo component to compensate for its noise. Lastly, we observe the incoherent decomposition result for the Luyu scene, where half of the object is incorrectly identified as rough regions. This reveals another shortcoming of not relying on MLPs for material decomposition, as there's not enough variation across views for supervising roughness. While we can increase the smoothness term to enforce roughness propagation as discussed in Chapter 4, doing so affects the highly intricate geometric detail of the Luyu scene, resulting in overly smooth reconstruction. Despite these drawbacks, the effectiveness of our method serves as the starting point for more refined designs in the future.

Figure 5.3 provides qualitative comparisons between GS–2M and other 3DGS-based methods using the Shiny Blender Synthetic dataset. It is clear that while SoTA mesh reconstruction methods excel in recovering non-reflective surfaces, they suffer from the ambiguity caused by reflection images and attribute them as part of the object's geometry. This results in highly distorted reconstructions and poor quality meshes. Our method, on the other hand, gracefully copes with reflective surfaces and produces more accurate geometries. Thanks to the smoothness constraints applied on normals and BRDF parameters, regions with correctly identified smoothness are propagated, amplifying the normal consistency. Together with the depth-normal loss, depth values are encouraged to fill in the holes, resulting in better surfaces. On top of that, as discussed in the Section 5.1, our full model still excels in regular surface reconstruction and outperforms most 3DGS-based approaches in this regard. As a result, we consider GS–2M as a starting point to bridge the gap between high-quality surface reconstruction and material decomposition. While there's still a lot of room for improvement, our framework proves its effectiveness and opens the door for model designs that do not rely on external dependencies.

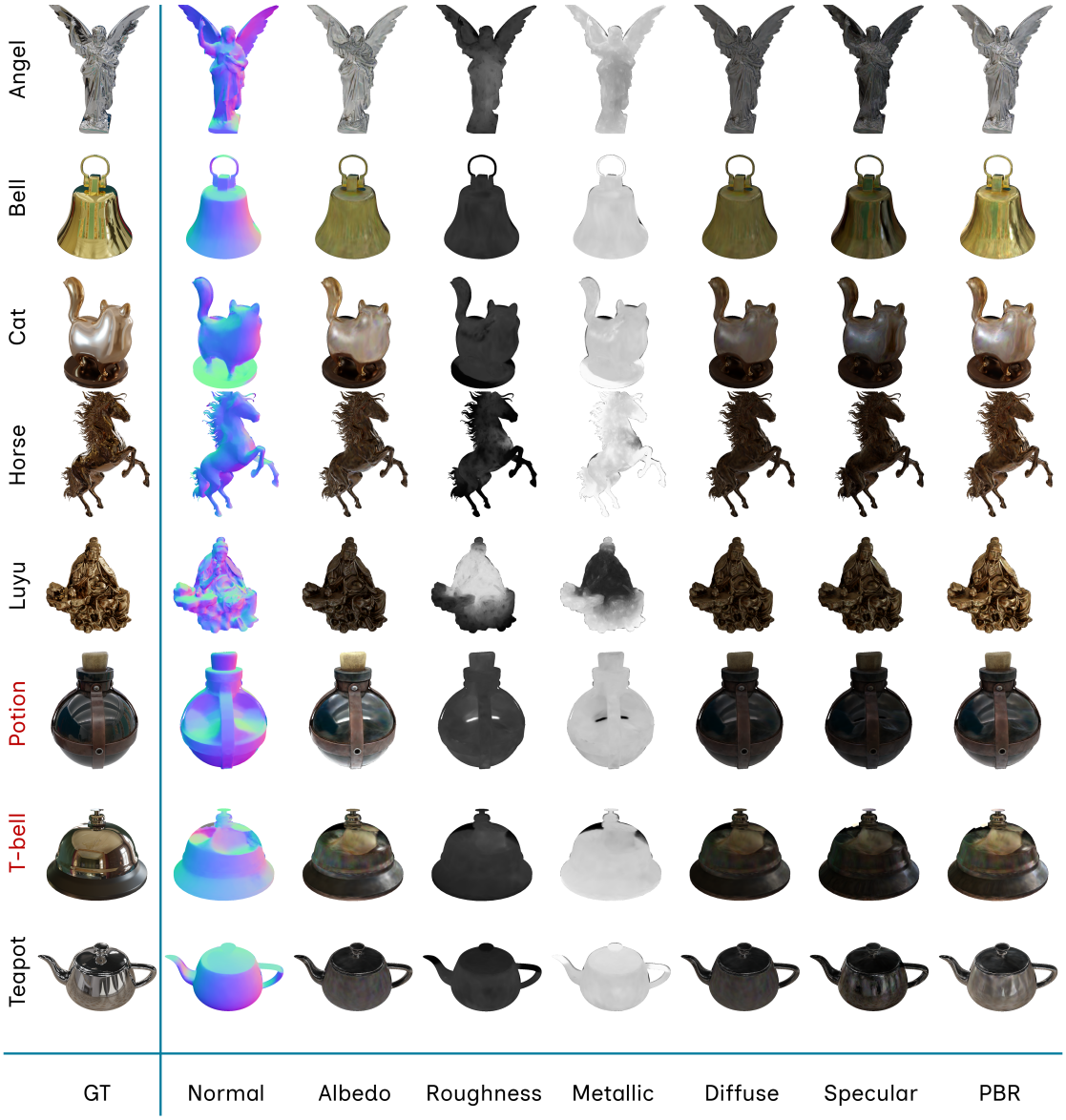

Figure 5.2: *We provide qualitative performance of GS–2M on the challenging Glossy Blender Synthetic dataset. Our method is able to faithfully decompose material properties for 6 out of 8 scenes, with the two failed scenes colored in red. These failures are caused by the intricate reflection images that make it challenging for our multi-view roughness strategy to detect. In particular, these surfaces contain both low- and high-frequency reflective clues, making it hard to find a good thresholding value for roughness supervision. However, the remaining reconstruction demonstrates that our method robustly recovers material attributes of the target object, highlighting the potential of multi-view clues for roughness-aware training without relying on external priors or sophisticated MLPs. More experiments for the Glossy Blender Synthetic dataset can be found in Appendix A.*

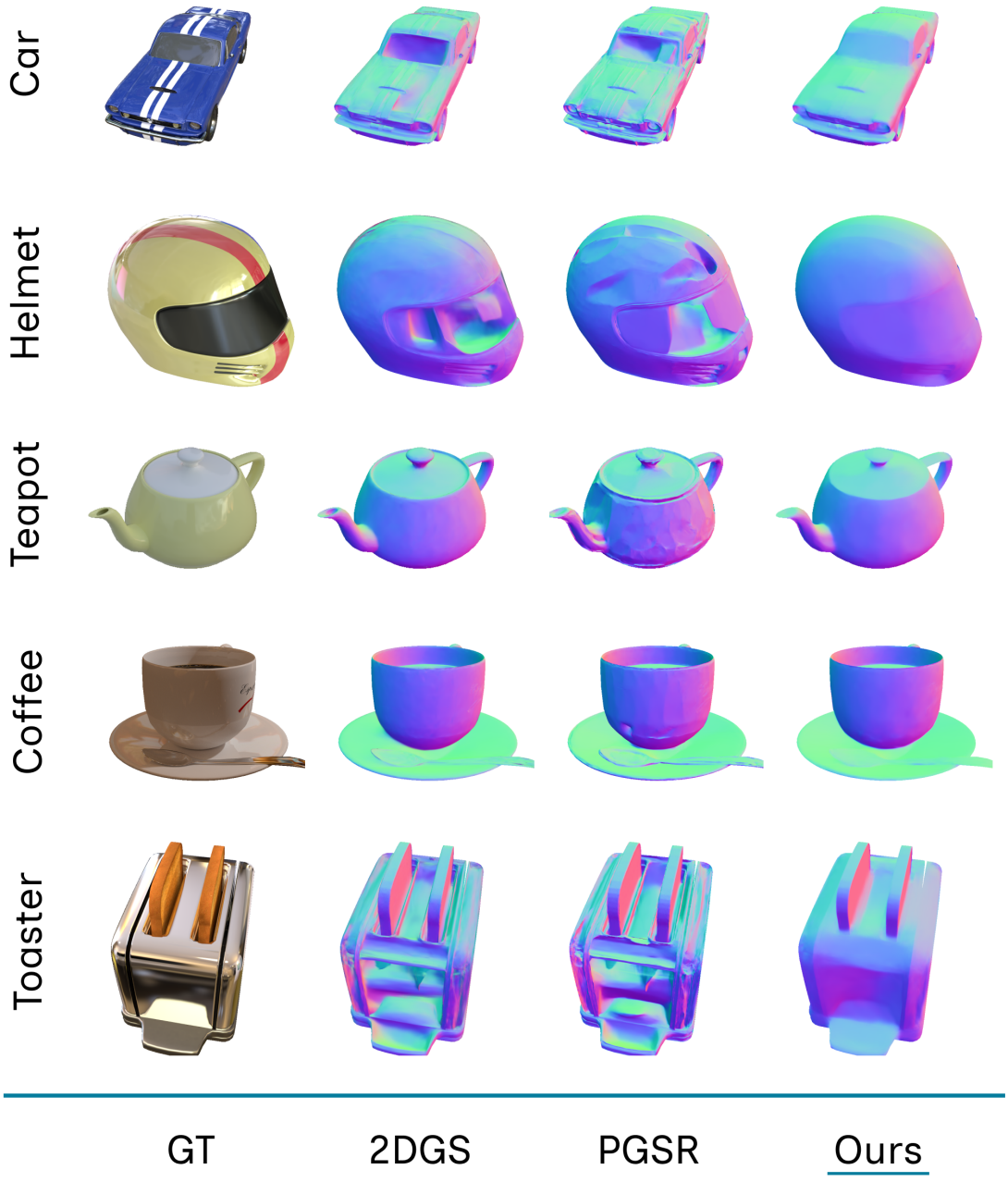

Figure 5.3: *Qualitative comparisons for the Shiny Blender Synthetic dataset. We compare the reconstructed normals of our full model with two SoTA methods: 2DGS (Huang et al., 2024a) and PGSR (Chen et al., 2024a). It is clear that SoTA 3DGS-based methods suffer from the ambiguity caused by reflection images, resulting in meshes with distorted geometries. Our method, on the other hand, produces more accurate reconstructed surfaces thanks to the smoothness constraint of normals and BRDF parameters.*

6 In Practice

This chapter examines how mesh reconstruction could be adopted in practice. While GS–2M and other Gaussian splatting or even neural-based methods excel in this domain, applying them to real-life scenarios may encounter unexpected shortcomings. For instance, performing mesh reconstruction from in-the-wild images or videos likely requires an additional camera registration process for each captured image/frame. In particular, the camera intrinsic parameters, including focal lengths and distortion coefficients, and camera poses at each frame, all play a major role in the reconstruction process. In addition, it is usually preferable to only extract the mesh of the foreground object in the scene. Given that most in-the-wild captures always contain background artifacts of some sort, reconstructing a clean foreground mesh is not as straightforward as it may seem. To this end, this chapter reports two adaptation attempts of Gaussian splatting for mesh reconstruction in practice, one focusing on working with predefined camera poses (Section 6.1), while the other addresses foreground object extraction (Section 6.2).

6.1 Predefined camera parameters

Given that Augmented Reality (AR) and its application are constantly rising, recent camera generations often support acquiring camera poses during shooting (Alatise and Hancke, 2017), (Yang et al., 2020), (Shen et al., 2021). At the same time, more sophisticated methods are under active development for deducing camera poses from plain captures as alternatives to COLMAP (Blanton et al., 2022), (Cheng et al., 2023), (Smith et al., 2024), (Zhang et al., 2024). As a result, the in-practice training process of Gaussian splatting methods is more likely to consume camera parameters that do not originate from COLMAP. When this happens, certain assumptions and modifications must be made aware of. These will be illustrated by a small example where predefined camera poses from an industrial assembly pipeline are replaced with COLMAP for mesh reconstruction. It is worth noting, nonetheless, that there is an active line of research specifically focusing on eliminating COLMAP from Gaussian splatting pipelines (Ye et al., 2024a), (Meuleman et al., 2025). Despite being interesting and inspiring under NVS frameworks, they are not within this report's scope since we aim at high-quality mesh and material reconstruction.

Figure 6.1: *The industrial part (TestCube) with predefined camera poses derived from the assembly pipeline. The scene comprises 73 views capturing highly overlapped images with rich background textures. The study attempts to bridge these predefined camera poses with the training process interface.*

The object under study is an intricate industrial part exhibiting a shiny surface, as shown in Figure 6.1. The experiment encompasses multiple predefined camera poses originating from an involved acquisition pipeline, consisting of 73 views illustrated in the same figure. As stated before, the goal of this practical study is to streamline these predefined camera parameters for the reconstruction of the TestCube without relying on COLMAP. Listing 6.1 gives an example of the input format of these predefined parameters. As with standard camera calibration processes, these include the intrinsic, extrinsic, and distortion parameters. The intrinsic in this scenario is a 3×3 matrix composed of focal lengths and image centers, while the extrinsic is a 4×4 world-to-camera matrix.

Listing 6.1: *Direct predefined camera parameters dumped from worker RegistrationFine. The JSON provides the camera's 3-by-3 intrinsic matrix and its distortion coefficients. In turn, the camera pose is defined by a 4-by-4 w2c matrix.*

```
{
  "cam_matrix": [
    [1525.2380952380952, 0.0, 954.3809523809523],
    [0.0, 1525.2380952380952, 718.1428571428571],
    [0.0, 0.0, 1.0]
  ],
  "cam_distortion": [0, 0, 0, 0, 0],
  "pose_refined": [
    [0.83811283, -0.53609866, 0.092942454, -48.261074],
    [-0.238565654, -0.51651978, -0.82261997, -72.595039],
    [0.4881791, 0.66793710, -0.56154108, 998.99860],
    [0.0, 0.0, 0.0, 1.0]
  ],
}
```

Projection model. Although the camera distortion coefficients are given as zeros, it is worth stressing that most Gaussian splatting models expect *pinhole* or *simple pinhole* camera projection models. While zero distortion coefficients imply that the input captures have been correctly undistorted, it is easy to forget that not all input captures in practice come from pinhole cameras. When this is the case, the provided distortion coefficients must be used to undistort input images before using them for training (Weng et al., 1992). On the other hand, there are relevant works in the literature addressing this issue specifically, either by refining the EWA transform (Liao et al., 2024), or employing a different projection model (Wu et al., 2025). Despite being interesting on their own, working with undistorted camera models is not the scope of this report.

Predefined intrinsics. The predefined intrinsic parameters are often given by a 3×3 matrix containing focal lengths in $x-$ and $y-$directions and the image principal point in pixels. While it is uncommon to have the image principal point deviate from the image center in literature, it may not be located at the exact center in practice due to the camera's optical system. This is again a sign of distorted images, and the undistortion routine should reproject input captures to the correct form. Once done, the given principal point can be omitted entirely and replaced with half the dimension of the undistorted images. It's important to keep in mind that if the principal point is not at the center of the training images, the optimization process may diverge, as discussed in previous works (Huang et al., 2024a).

Predefined poses. The challenge of adopting existing camera parameters largely stems from the predefined poses. On the one hand, cameras in Gaussian splatting models follow the convention employed by the OpenCV library (Bradski, 2000), i.e., right-handed coordinate systems with $z-$forward and $y-$down. On the other hand, the optimization expects the input poses to define 4×4 transforms from world to camera space. While the former causes little hazard since most calibration pipelines take OpenCV's standard as the norm, the latter is more prone to error since the ambiguity between world-to-camera (w2c) and camera-to-world (c2w) is extremely common. As a recap, the image formation model projects a 3D world point $\mathbf{P} = [x, y, z, 1]^\top$ to 2D image coordinates $\mathbf{p} = [u, v, 1]^\top$ via Equation 6.1:

$$s\mathbf{p} = \underbrace{\begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}}_{\text{w2c}} \mathbf{P} \tag{6.1}$$

where s is the depth scaling factor, K is the intrinsic matrix, and the 4×4 w2c matrix transforms world coordinates to the camera's right-handed OpenCV coordinate system ($+z$ forward, $+y$ down). It is a good practice to plot the predefined camera poses to validate their interpretation, as shown in Figure 6.2. In particular, if the

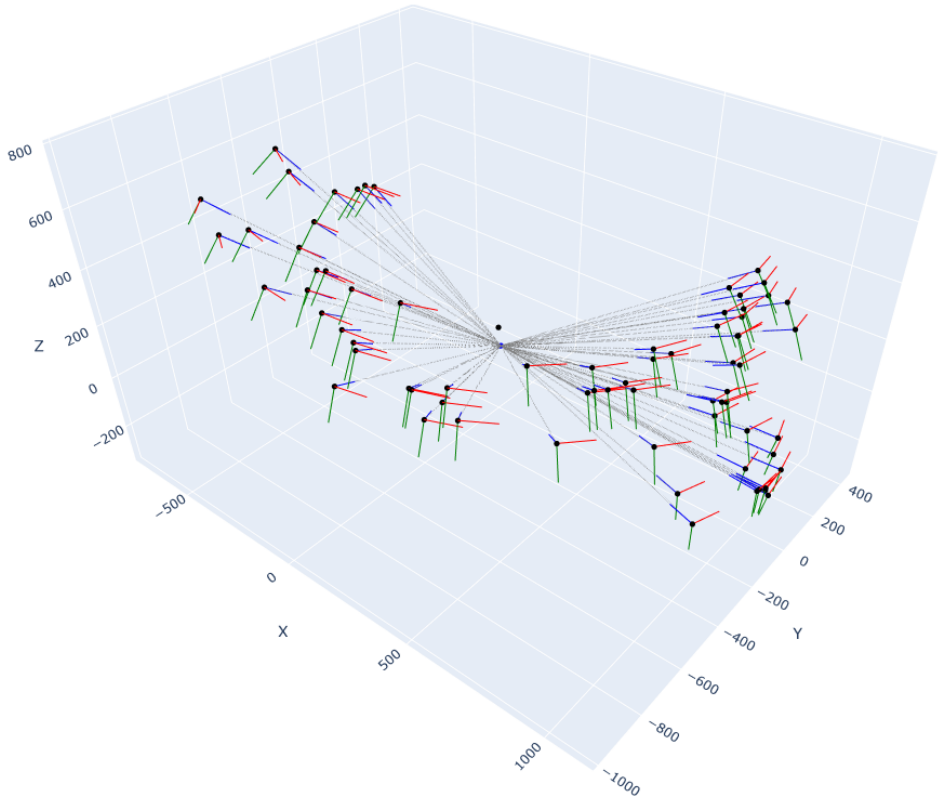

Figure 6.2: *Validate the predefined poses by plotting their orientations. The transforms that make the visualization correct must be inverted before using them for training. The visualization also reveals that the target point of all cameras (blue dot) is not necessarily at the world space's origin (black dot).*

predefined rotation and translation components can be consumed directly to plot the camera orientations, they effectively define the c2w transform and must be inverted before plugging into the training. Yet, the visualization also reveals another issue: the scale of these predefined poses is three orders of magnitude bigger than the typical scale of COLMAP-prepared scenes. This can be roughly estimated by looking at the translation components, where the dominant values always span up to 1000. If these poses were to be consumed directly, the rasterization would clip a large portion of the scene because the viewing frustum employed in Gaussian splatting scales to at most 100. While this frustum can be adjusted, large distances between the near and far plane in rendering have been known to cause numerical instability (Reed, 2019). As a result, it is preferable to instead scale the poses to have smaller overall translation components. At first glance, performing vector scaling on these translations seems to be a sufficient solution. However, this is only correct if the poses point at the scene origin, which is not the case for the parameters at hand based on the visualization. The correct approach is to instead scale the translation components relative to an approximate intersection point, as detailed in Listing 6.2.

Listing 6.2: *Python procedure to find the intersection point of multiple lines. Each line is defined by point P_i and direction vector d_i. The solution minimizes the sum of squared distances to all lines. All translation vectors should be scaled relative to this approximate intersection point.*

```python
import numpy as np

def find_intersection(P, d):
    n = P.shape[0]
    A = np.zeros((3, 3))
    b = np.zeros(3)

    for i in range(n):
        di = d[i]
        Pi = P[i]
        A += np.eye(3) - np.outer(di, di)
        b += (np.eye(3) - np.outer(di, di)) @ Pi

    p = np.linalg.solve(A, b)
    return p
```

However, the above scaling procedure assumes that the predefined rotation components are orthonormal to a sufficient threshold level. Specifically, the scaling procedure is only reliable if all predefined rotations R satisfy the orthonormality check of Listing 6.3 with at least 1^{-6} tolerance level. This tolerance level is loosely based on empirical experiments, and as a reference, camera poses derived by COLMAP have rotations passing the orthonormality check with 1^{-15} tolerance. Once the interpretation of all predefined poses is validated and their extents are rescaled to a small enough viewing volume, they can be injected into the optimization process as with other scene formats. Notice that this would require defining a dedicated dataset reader to consume these poses and their accompanying ground truth images.

Listing 6.3: *Python procedure to validate the orthornomality of a rotation matrix. The rotation R is a 3×3 matrix, and the tolerance level should be at the minimum 1^{-6}. The scaling procedure is only reliable if all predefined rotations pass the check.*

```python
import numpy as np

def is_orthonormal(R, atol=1e-6):
    if R.shape != (3, 3):
        return False

    orthogonal = np.allclose(np.dot(R.T, R), np.eye(3), atol=atol)

    norms = np.linalg.norm(R, axis=0)
    normal = np.allclose(norms, np.ones(3), atol=atol)

    return orthogonal and normal
```

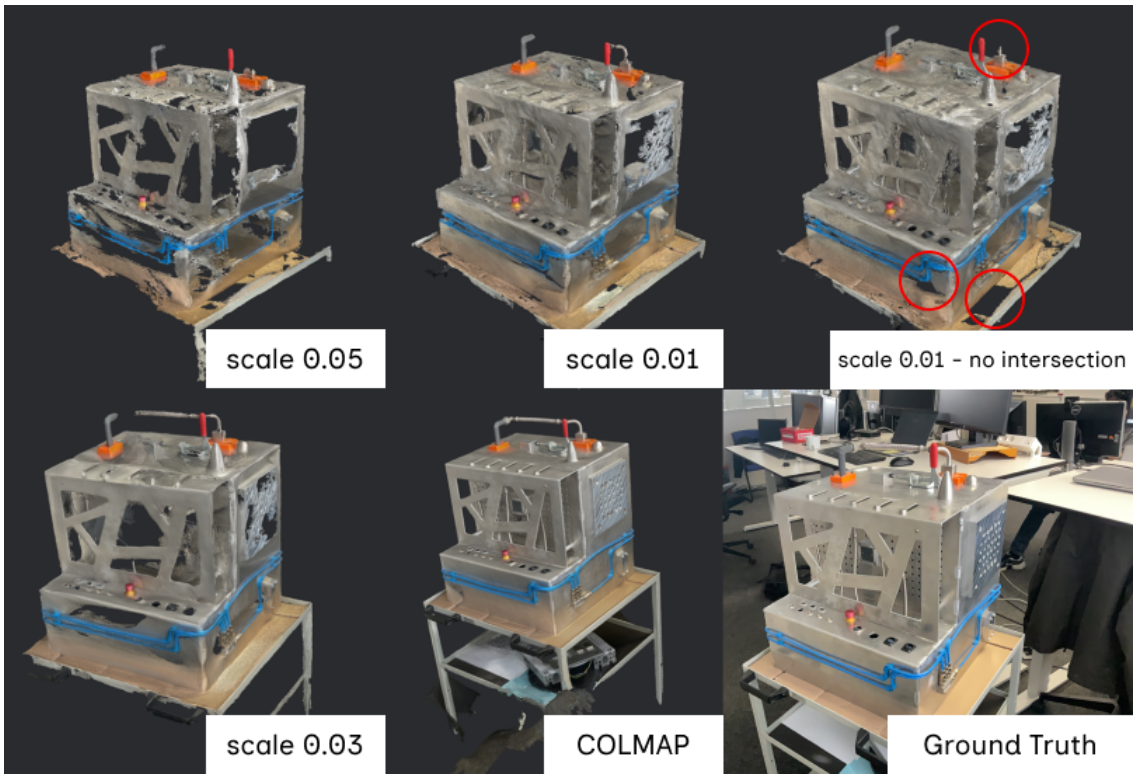

Figure 6.3: *The reconstructed TestCube with predefined camera parameters. This demonstrates the importance of rescaling the scene to smaller extents, with the scale factor of 0.03 being a sweet value with an overall good-quality mesh. However, non-orthonormal rotations cause a drop in the reconstruction performance.*

Random initialization. The last issue for non-COLMAP training paths is the initialization of random points at the beginning of training. Since there are no SfM points to start with, they are replaced with random 3D points sampled within the scene extent. However, the existing randomization procedure of 3DGS uniformly samples points within a cube whose center is at the scene's origin. While this tactic serves its purpose well for synthetic datasets, practical scenes in real life may have the target object located significantly off the center. As a result, it is advisable to shift the whole structure so that the estimated intersection point matches the scene's origin. Alternatively, a much simpler approach is to shift the sampling cube so that its center is at the estimated intersection.

Key points (see Figure 6.3 for more insight)

- Ensure intrinsics follow pinhole or simple pinhole camera models
- Rescaling may not perform well if rotations are not orthonormal

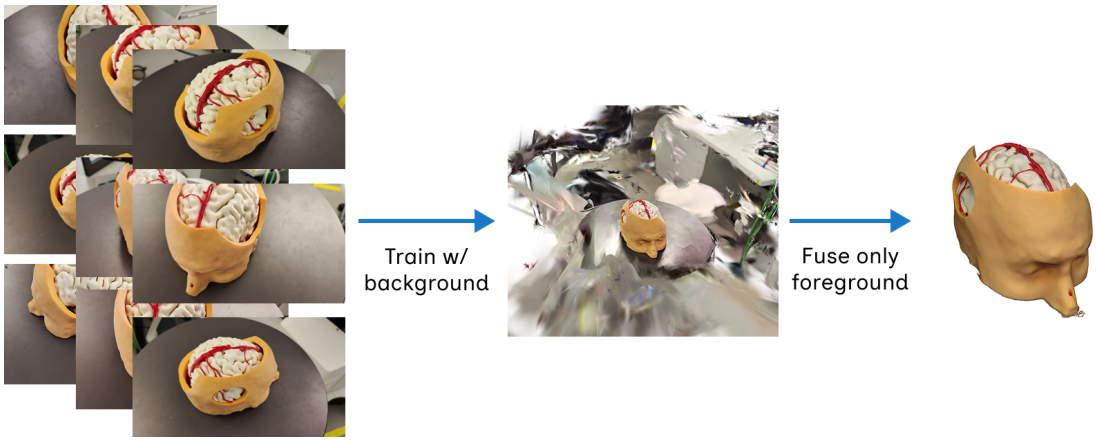

Figure 6.4: *Practical mesh reconstruction scenarios often perform training on images with full background and only apply masks during TSDF fusion.*

6.2 Foreground object reconstruction

It is desirable in practice to extract only the foreground mesh from a scene containing background noise. This is especially common for in-the-wild captures where the cameras may not be equipped with sophisticated visual models to eliminate these background artifacts. On the other hand, rich background textures are preferable for COLMAP training paths since COLMAP relies on these clues to perform feature matching. Scenes with no background thus hinder COLMAP's ability to robustly detect and match features across multiple views, often resulting in incomplete or inaccurate camera pose estimations and sparse reconstructions. Consequently, the training process is likely to consume input images with more scene contents than just the target foreground object, as illustrated in Figure 6.4. While validation datasets in the literature provide masks to extract only the foreground mesh for evaluation, practical scenarios, however, are not designed for experiments, and manual procedures must be adopted to produce masks for clean mesh reconstruction.

Masking with BiRefNet. Zheng et al. (2024) introduces Bilateral Reference for High-Resolution Dichotomous Image Segmentation (BiRefNet) to accurately segment foreground objects from high-resolution images, particularly excelling in scenarios with complex and fine structural details. The method integrates two core components: a Localization Module (LM) that leverages global semantic information to precisely locate objects, and a Reconstruction Module (RM) that employs a novel bilateral reference mechanism combining hierarchical image patches as source references and gradient maps as target references to reconstruct detailed segmentation masks. This bilateral referencing strategy enables BiRefNet to effectively capture both local fine-grained features and global contextual cues, resulting in high-quality binary masks that distinguish foreground objects from backgrounds.

Listing 6.4: *Run a pre-trained BiRefNet to obtain a mask given an RGB image. The pre-trained model can be obtained from the author's official website.*

```python
import torch
from torchvision import transforms
from PIL import Image

from models.birefnet import BiRefNet
from utils import check_state_dict

birefnet = BiRefNet(bb_pretrained=False)
state_dict = torch.load("path/to/weight", map_location='cpu')
state_dict = check_state_dict(state_dict)
birefnet.load_state_dict(state_dict)

torch.set_float32_matmul_precision(['high', 'highest'][0])
birefnet.to('cuda')
birefnet.eval()
birefnet.half()

def extract_mask(birefnet, imagepath, tf_size=(1024, 1024)):
    # Data settings
    transform_image = transforms.Compose([
        transforms.Resize(tf_size),
        transforms.ToTensor(),
        transforms.Normalize(
            [0.485, 0.456, 0.406],
            [0.229, 0.224, 0.225])])

    image = Image.open(imagepath)
    input_images = transform_image(image).unsqueeze(0).to('cuda').
       half()

    # Prediction
    with torch.no_grad():
        preds = birefnet(input_images)[-1].sigmoid().cpu()
    pred = preds[0].squeeze()
    pred_pil = transforms.ToPILImage()(pred)
    mask = pred_pil.resize(image.size)
    return mask
```

Listing 6.4 provides an example of how a pre-trained BiRefNet model is employed to produce foreground masks from RGB input images. These masks are then consumed during the TSDF fusion process to eliminate invalid depth values, resulting in a clean reconstruction of the foreground triangle mesh. Note that the mask extraction procedure must be performed **after** any undistortion process, i.e, masks must be produced from undistorted images, rather than the original ones. The reason is that such undistorting operations would potentially warp training images. If the masks are generated using the original data, they will lead to incorrect segmentation and contaminated depth fusion.

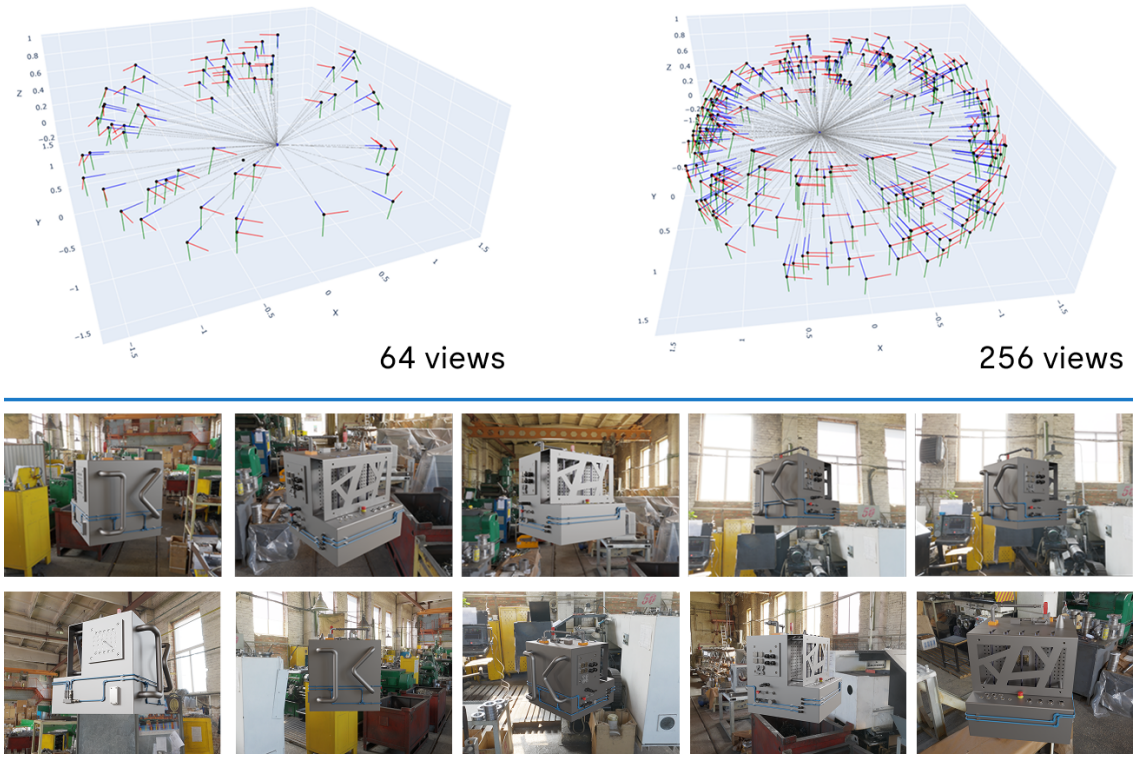

Figure 6.5: *A synthetically rendered scene of the TestCube, extremely rich in background. The goal is to reconstruct the cube from a sparse (64) and dense (256) set of viewpoints. These viewpoints are uniformly sampled within a 4-unit-wide sphere and fed directly to the training.*

Ground truth RGB masking. The next study concerns situations where the target object is extracted from an extremely rich-textured background. The setup is a synthetic scene of the same TestCube demonstrated earlier, rendered with a detailed environment map featuring a wide collection of background objects, as shown in Figure 6.5. To make the study robust, the reconstruction will be performed under two distinct conditions: sparse and dense captures of the scene. In particular, a randomized procedure of the camera trajectory is set up to uniformly sample viewpoints at positions that likely resemble how a person would capture an object. This trajectory is 360 degrees around the object, with elevation angles ranging from -10 to 40 degrees. These numbers are derived based on the assumption that a typical height of a person is around 170cm, and values are drawn from a normal distribution centered at this height. Once the trajectory is set up, 64 and 256 views are sampled for sparse and dense reconstruction, respectively. These poses are fed directly to the training via a separate dataset reader implementation. Following the previous discussion, Gaussians are trained on input images with background, with masking only applied during TSDF fusion.

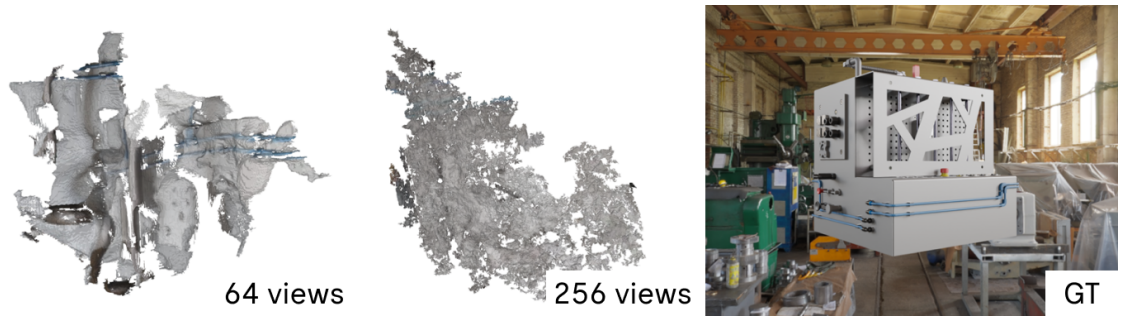

Figure 6.6: *Training on ground truth images with rich background textures results in poor reconstruction results. Gaussians are busy fitting to the vast background details, leaving little room for the target object.*

Figure 6.6 shows the reconstruction result post-training. The extracted meshes for both sparse and dense reconstructions do not match the overall shape of the ground truth object. The reason for such poor reconstructions is that Gaussian points are busy fitting to the vast background details, leaving no room for the target object to gain their attention. As a result, computing resources ramp up during training, with the optimization process producing more than 4 million points for the unnecessary background. The solution to combat this problem is to mask the ground truth RGB images prior to the training process. As simple as it may sound, the masking may cause intricate artifacts if not handled properly, as shown in Figure 6.7. At the high level, masking of ground truth should be performed manually before image resizing to prevent imperfect binarization. The issue has been raised and can be found on the PULL REQUEST PAGE of the original 3DGS code base.

Figure 6.7: *Masking is done automatically on RGBA images when using PIL for image processing. Floaters are produced in NVS views due to the imperfect binarization of the alpha channel. This only happens when ground truth images are resized via the resolution parameter of the train script, upon which the automatic masking of PIL takes effect. Despite being irrelevant to mesh reconstruction, this floater artifact causes significant damage when assessing NVS performance.*

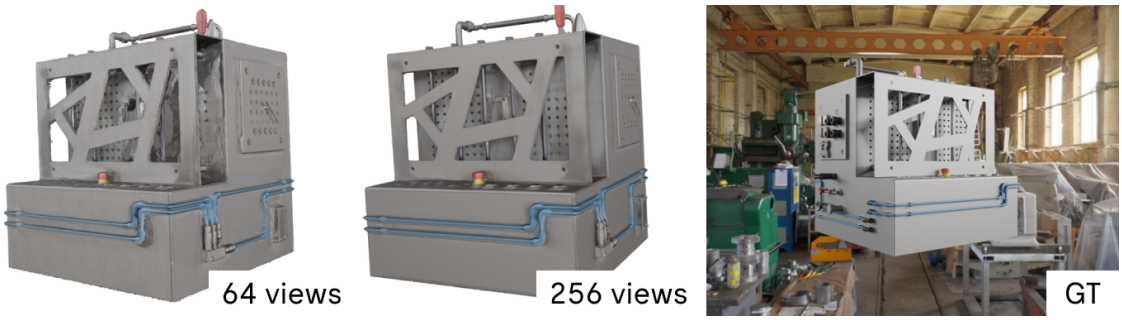

Figure 6.8: *Training with masked ground truth produces significantly better reconstructed meshes. The intricate geometry of the TestCube is recovered faithfully even with only 64 views. As for the dense reconstruction, occlusion regions are handled better, resulting in a high-fidelity mesh. This highlights the strength of masking GT images if done properly.*

Once ground truth images are masked properly, the optimization process can now focus only on the target object without having to worry about the background details. The final reconstruction result is illustrated in Figure 6.8, where the quality of the dense reconstruction is slightly better than its sparse counterpart. This signifies the significance of masking ground truth images if they are captured with an enormous amount of background pollution. If the optimization process only focuses on the foreground as shown here, not only is the resulting mesh much better, but computing resources are also reduced considerably.

Key points

- For simple scenes, training can be done on the original GT images
- For scenes with overwhelming background, masking it is advisable

7 Conclusion

We have presented and discussed our unified framework for jointly optimizing Gaussian splatting models for mesh reconstruction and material decomposition. We started introducing the emergence of neural rendering as a faithful paradigm to replace manual, laborious work in visual computing domains. While neural-based methods promise their effectiveness to address a large number of vision tasks, we chose to pursue Gaussian splatting approaches due to their explicit nature and resource-friendly training. On assessing the capabilities of SoTA 3DGS-based methods for mesh reconstruction, we discovered that they struggle to reconstruct highly reflective surfaces due to the lack of appearance modeling. Although there have been many publications proposing different methods to handle reflective surfaces for Gaussian splatting models, they often incorporate external priors from neural-based approaches, or model appearance parameters with MLP networks. We therefore propose an optimization strategy independent of external dependencies and rely purely on multi-view photometric clues to identify reflective regions. Our approach helps cope with reconstruction artifacts caused by reflection images, and not using external priors or MLPs allows us to scale up the runtime performance without worrying about these external components. To validate the effectiveness of our method, we perform reconstructions on four widely consumed datasets relevant to mesh and material reconstruction tasks. The experiments on the DTU dataset reveal that our occlusion check for multi-view geometric loss produces considerably fewer Gaussian points while maintaining the reconstruction quality to be on par with current SoTA methods. On the other hand, experiments on the TnT dataset show that the incorporation of multi-view normal consistency helps recover smoother normals at regions with high-frequency textures. Although our full model's performance cannot match SoTA approaches under the DTU benchmark, it still outperforms all neural-based and most 3DGS-based approaches in this domain. Most importantly, our full model has the capability to recover highly reflective surfaces, as demonstrated via qualitative comparisons using the Shiny Blender Synthetic dataset. This is largely attributed to the proposed multi-view roughness supervision based on inconsistent photometric clues and roughness propagation via smoothness constraints. Furthermore, our method produces reasonable decomposition results when tested with the challenging Glossy Blender Synthetic dataset, highlighting its potential for further improvements without relying on external dependencies.

Despite the advantages, the experiments also reveal that our method still bears a certain number of limitations, and there is a lot of room for future refinements. We summarize these limitations as follows:

- NVS performance of the full model: as shown in Table 5.3, our full model struggles to match the NVS performance with current SoTA mesh reconstruction approaches. This stems from the unrefined PBR pipeline in our design. For example, our normal maps are blended in camera space for the convenience of plane depth computation, yet shading requires normals to be in world space. We speculate that the transformation of blended local normals to world space causes certain issues with the composite image, as some visualizations of these normals are not aligned with the GT normal maps in Blender datasets.

- Maximum number of Gaussians: as discussed in Section 5.1, we currently have no strategy to cap the maximum number of Gaussians due to densification, resulting in limited experiments with the TnT dataset. While the masking of background discussed in Chapter 6 would indeed solve the issue, more decent and prior-agnostic approaches, such as reformulating densification as Monte-Carlo Markov Chains (Kheradmand et al., 2024), are more preferable for future improvements.

- More robust roughness supervision: our proposed multi-view roughness supervision works well for a decent number of scenes from the Glossy Blender Synthetic, yet our design still has several limitations. The experiment on the Luyu scene reveals that multi-view photometric inconsistency may not be enough to detect reflective regions. While smoothness constraints can help propagate correctly identified smooth regions, doing so may negatively affect the geometric details of the object. On top of that, the approximation of metallic from roughness does not hold for objects containing both reflective and non-reflective parts. As a result, we cannot robustly distinguish different roughness levels with the current strategy, and more sophisticated enhancements for supervising roughness from multi-view clues are worth investigating.

- Self-reflection: while the incorporation of BRDF training helps cope with reflective surfaces, the method cannot faithfully decompose the appearances of objects containing self-reflection images. This limitation can be observed with the reconstruction of the Toaster scene in Figure 5.3. While our reconstruction result gives the most favorable surface, the reflective shiny part is due to self-reflection, causing artifacts in the reconstructed surface. With the current PBR pipeline of GS–2M, the training has no options to correctly handle self-reflection images, and it thus sacrifices geometry to compensate for the photometric loss. Resolving self-reflection would require redesigning the entire shading model for ray-tracing (Moenne-Loccoz et al., 2024), and we consider this a promising direction to enhance our model's capability.

A | More experiments

This supplementary section provides additional reconstruction results we conducted with GS–2M and other methods for comparison, in particular:

- Figure A.1, A.2, and A.3 provide qualitative mesh reconstructions for the DTU dataset. The figures compare 2DGS (Huang et al., 2024a), PGSR (Chen et al., 2024a), and two variants of our model.
- Figure A.4 and A.5 showcase the reconstructed meshes of the Barn and Truck scenes from the TnT dataset, respectively.
- Figure A.6 provides the reconstructed lighting of our model for all 8 scenes from the Glossy Blender Synthetic dataset.
- Figure A.7 compares the quality of the extracted foreground meshes using BiRefNet's masks and ground-truth masks, while Figure A.8 demonstrates the benefit of masking ground-truth images during training for scenes with overwhelming background.

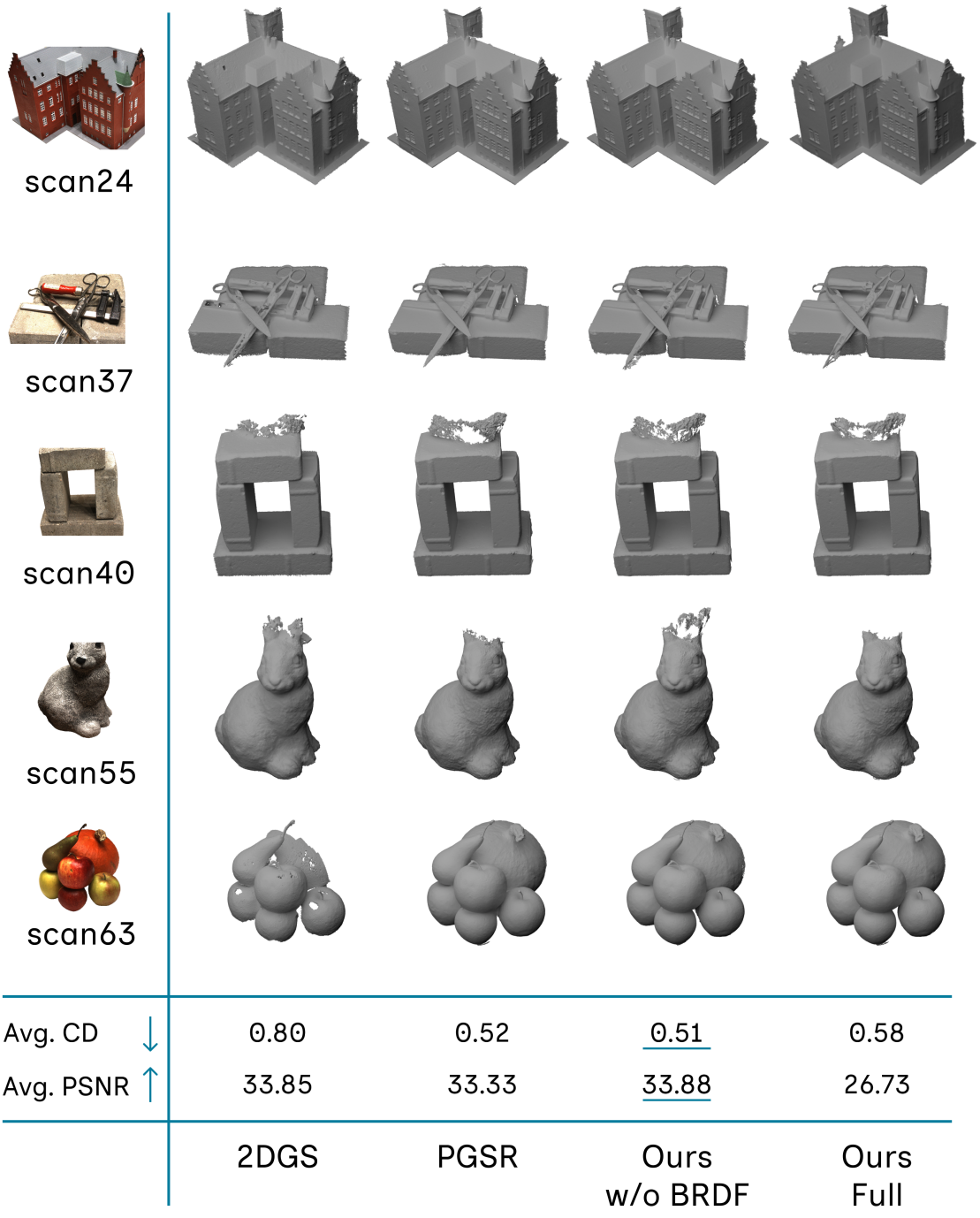

Figure A.1: *Visualizing reconstructed meshes from the DTU dataset - Part 1. The figure provides qualitative comparisons between 2DGS (Huang et al., 2024a), PGSR (Chen et al., 2024a), GS–2M w/o BRDF, and GS–2M full model. The average chamfer distance and PSNR of each method are also provided, with the best performing results underlined.*

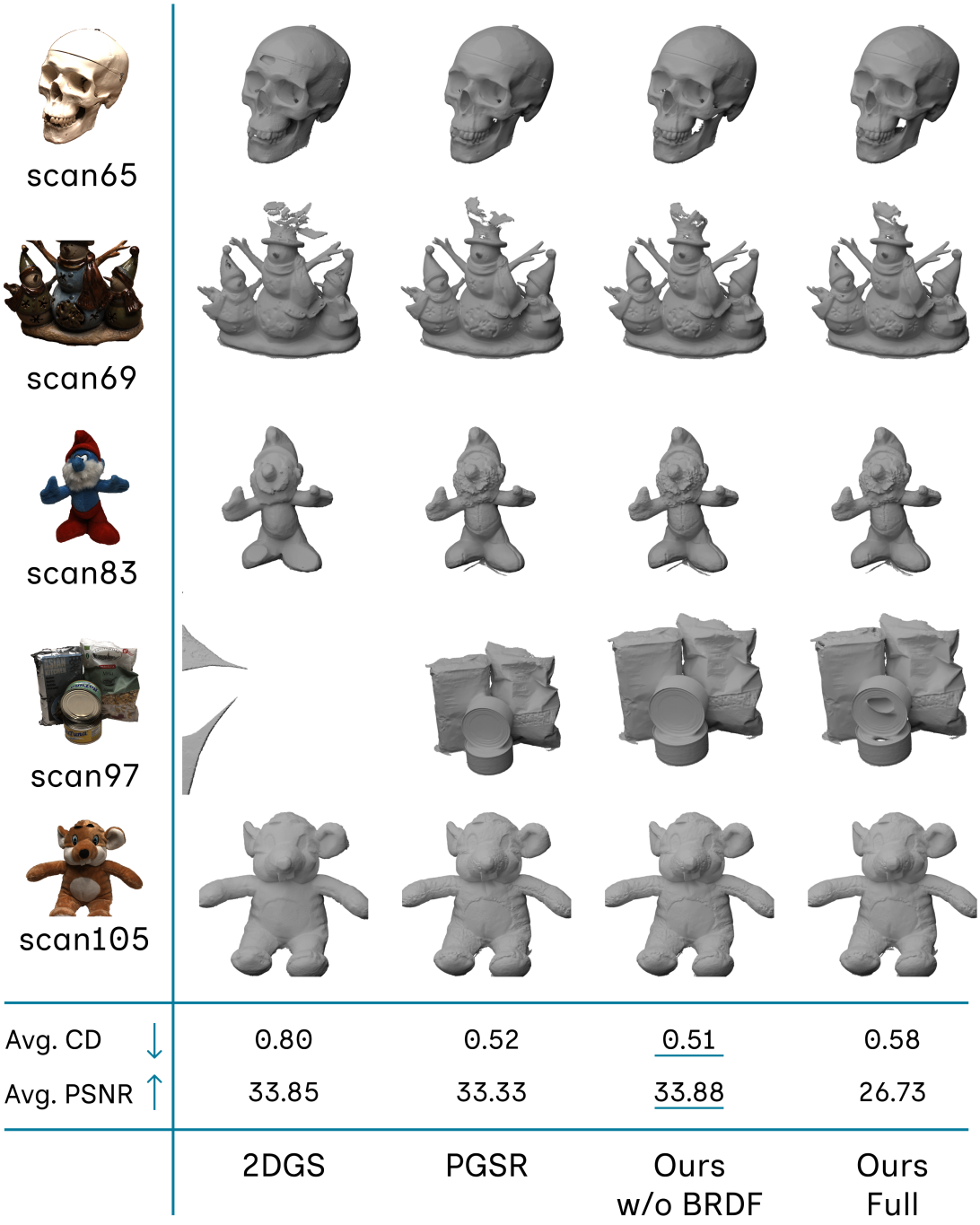

Figure A.2: *Visualizing reconstructed meshes from the DTU dataset - Part 1. The figure provides qualitative comparisons between 2DGS (Huang et al., 2024a), PGSR (Chen et al., 2024a), GS–2M w/o BRDF, and GS–2M full model. The average chamfer distance and PSNR of each method are also provided, with the best performing results underlined.*

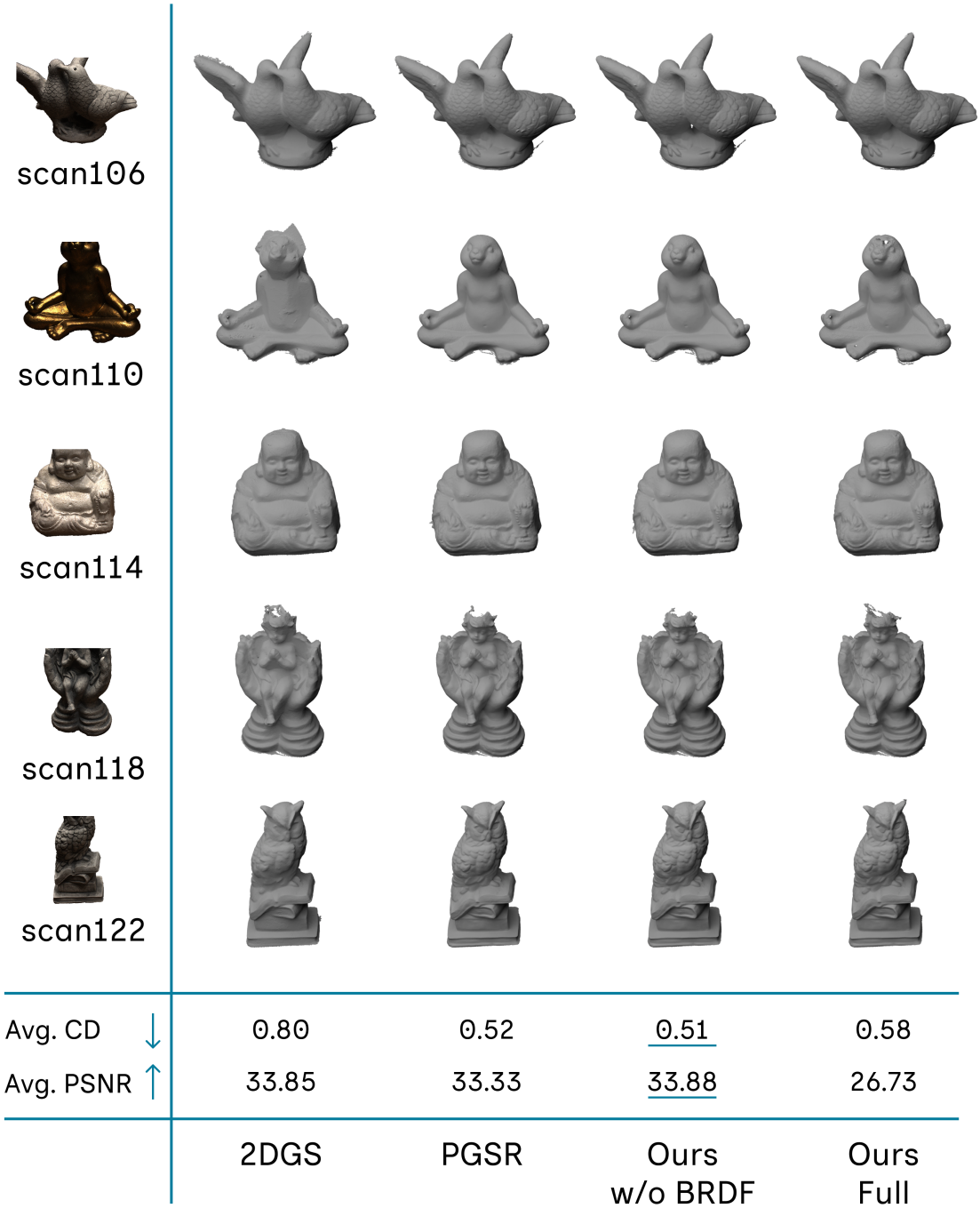

Figure A.3: *Visualizing reconstructed meshes from the DTU dataset - Part 3. The figure provides qualitative comparisons between 2DGS (Huang et al., 2024a), PGSR (Chen et al., 2024a), GS–2M w/o BRDF, and GS–2M full model. The average chamfer distance and PSNR of each method are also provided, with the best performing results underlined.*

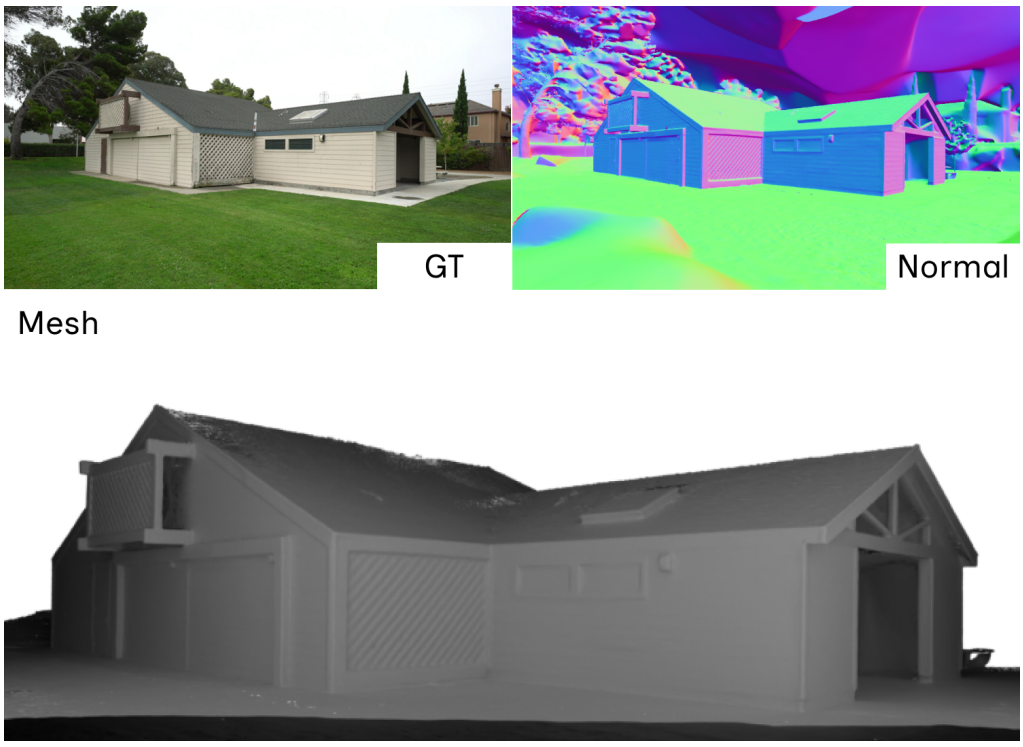

Figure A.4: *Reconstruction of the Barn scene from the TnT dataset*

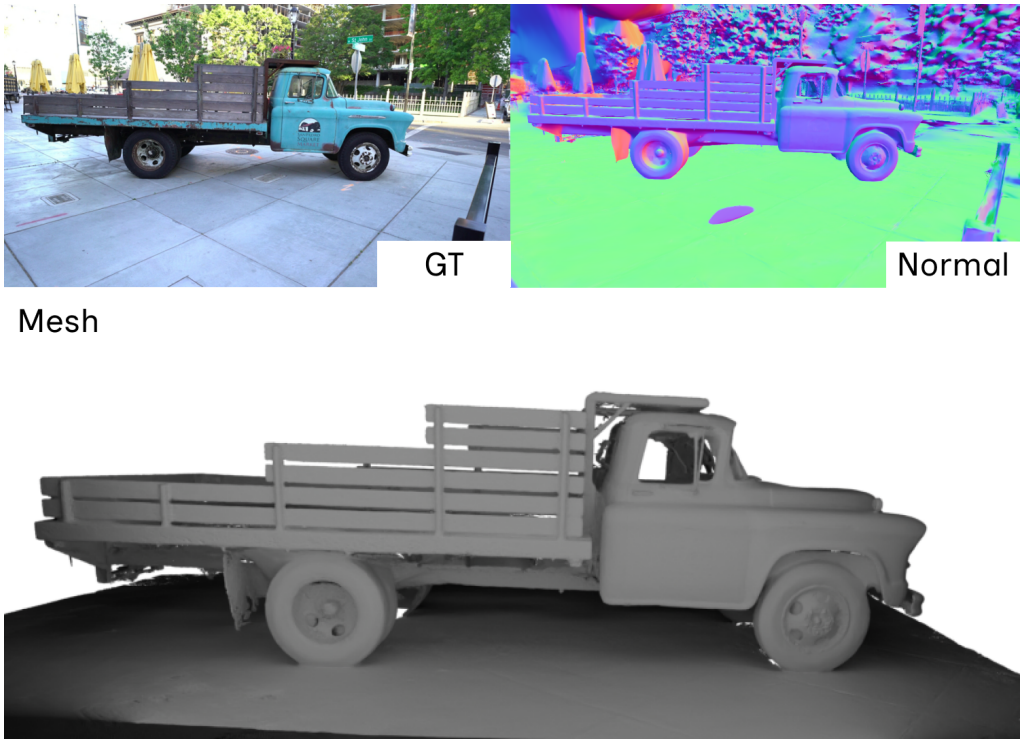

Figure A.5: *Reconstruction of the Truck scene from the TnT dataset*

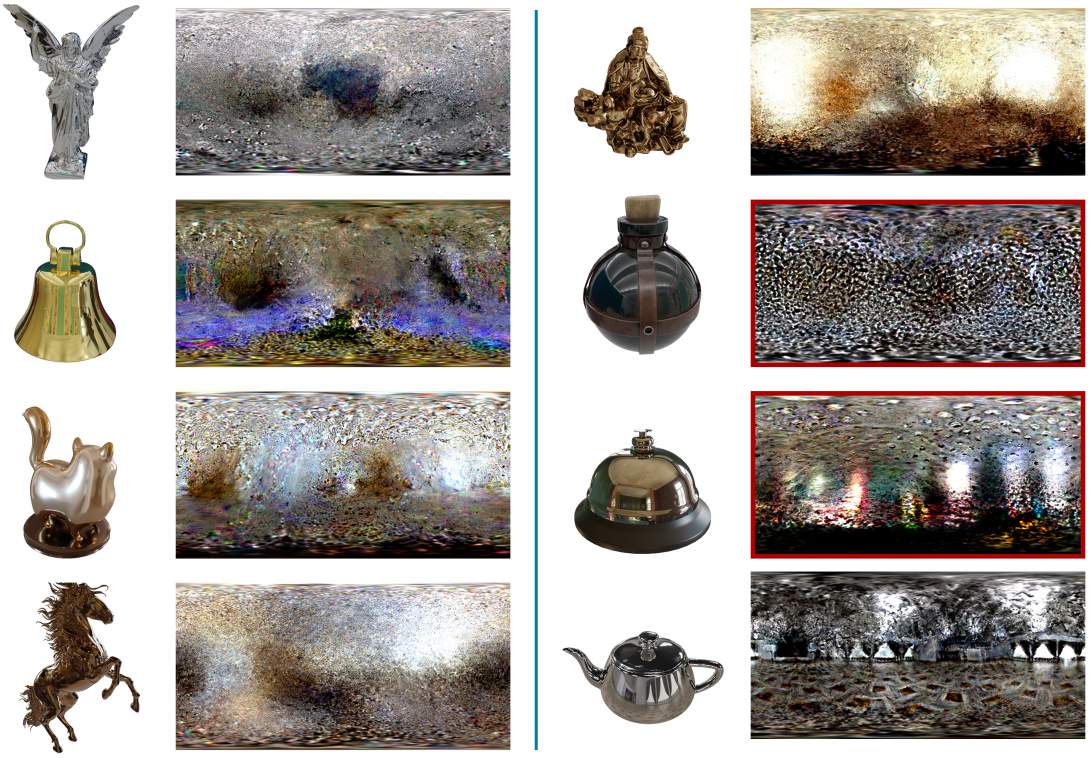

Figure A.6: *The reconstructed environment lighting for the Glossy Blender Synthetic dataset. Note that the Potion and T-bell scenes (colored in red) are failed reconstruction attempts. Our model is able to recover lighting, but more refined constraints and shading designs are advisable to discard noise.*

Figure A.7: *Extracting the foreground triangle mesh with ground-truth masks vs. masks inferred by BiRefNet (Zheng et al., 2024). The qualitative comparisons show that BiRefNet produces consistent masking compared to ground-truth masks and only gives artifacts along edges.*

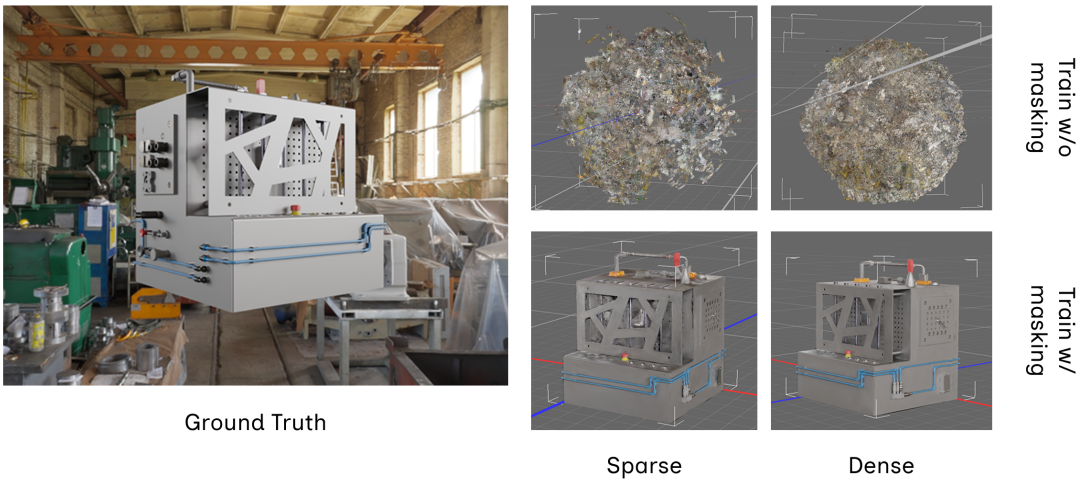

Figure A.8: *The Gaussians post-training for the synthetic scene of the TestCube (visualized via PlayCanvas's SuperSplat). The top row shows the resulting Gaussians when trained without masking, while the bottom row shows the trained Gaussians when ground truth images are masked during training.*

Bibliography

Adinets, A. and Merrill, D. (2022). Onesweep: A faster least significant digit radix sort for gpus. (cited on page 16)

Alatise, M. and Hancke, G. P. (2017). Pose estimation of a mobile robot using monocular vision and inertial sensors data. In *2017 IEEE AFRICON*, pages 1552–1557. (cited on page 41)

Baker, A. H., Pinard, A., and Hammerling, D. M. (2024). On a structural similarity index approach for floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, 30(9):6261–6274. (cited on page 24)

Bi, Z., Zeng, Y., Zeng, C., Pei, F., Feng, X., Zhou, K., and Wu, H. (2024). Gs^3: Efficient relighting with triple gaussian splatting. In *SIGGRAPH Asia 2024 Conference Papers*. (cited on pages 5, 10, and 20)

Blanton, H., Workman, S., and Jacobs, N. (2022). A structure-aware method for direct pose estimation. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 205–214. (cited on page 41)

Boss, M., Braun, R., Jampani, V., Barron, J. T., Liu, C., and Lensch, H. P. (2021a). Nerd: Neural reflectance decomposition from image collections. In *IEEE International Conference on Computer Vision (ICCV)*. (cited on page 8)

Boss, M., Jampani, V., Braun, R., Liu, C., Barron, J. T., and Lensch, H. P. (2021b). Neural-pil: Neural pre-integrated lighting for reflectance decomposition. In *Advances in Neural Information Processing Systems (NeurIPS)*. (cited on page 8)

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. (cited on page 43)

Chen, D., Li, H., Ye, W., Wang, Y., Xie, W., Zhai, S., Wang, N., Liu, H., Bao, H., and Zhang, G. (2024a). Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, page 1–12. (cited on pages 4, 9, 12, 18, 19, 25, 26, 27, 34, 35, 36, 39, 55, 56, 57, 58, 77, 78, 81, and 82)

Chen, G. and Wang, W. (2025). A survey on 3d gaussian splatting. (cited on page 3)

Chen, H., Lin, Z., and Zhang, J. (2025). Gi-gs: Global illumination decomposition on gaussian splatting for inverse rendering. (cited on page 28)

Chen, Y., Xu, H., Zheng, C., Zhuang, B., Pollefeys, M., Geiger, A., Cham, T.-J., and Cai, J. (2024b). Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. *arXiv preprint arXiv:2403.14627*. (cited on page 3)

Cheng, W., Cao, Y.-P., and Shan, Y. (2023). Id-pose: Sparse-view camera pose estimation by inverting diffusion models. *arXiv preprint arXiv:2306.17140*. (cited on page 41)

Cook, R. L. and Torrance, K. E. (1982). A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24. (cited on page 20)

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 303–312, New York, NY, USA. Association for Computing Machinery. (cited on page 10)

Dai, P., Xu, J., Xie, W., Liu, X., Wang, H., and Xu, W. (2024). High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery. (cited on pages 4, 9, 25, 34, and 81)

Darmon, F., Bascle, B., Devaux, J.-C., Monasse, P., and Aubry, M. (2022). Improving neural implicit surfaces geometry with patch warping . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6250–6259, Los Alamitos, CA, USA. IEEE Computer Society. (cited on pages 2, 8, 34, and 81)

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067. (cited on page 7)

Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. (2022). Plenoxels: Radiance fields without neural networks. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5491–5500. (cited on page 2)

Fu, Q., Xu, Q., Ong, Y.-S., and Tao, W. (2022). Geo-neus: geometry-consistent neural implicit surfaces learning for multi-view reconstruction. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc. (cited on page 8)

Furukawa, Y. and Hernández, C. (2015). *Multi-View Stereo: A Tutorial*. Now Foundations and Trends. (cited on page 7)

Furukawa, Y. and Ponce, J. (2010). Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376. (cited on page 7)

Guédon, A. and Lepetit, V. (2024a). Gaussian frosting: Editable complex radiance fields with real-time rendering. *ECCV*. (cited on page 3)

Guédon, A. and Lepetit, V. (2024b). Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR*. (cited on pages 9, 34, 36, 81, and 82)

Guo, H., Peng, S., Lin, H., Wang, Q., Zhang, G., Bao, H., and Zhou, X. (2022). Neural 3d scene reconstruction with the manhattan-world assumption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5511–5520. (cited on page 8)

Huang, B., Yu, Z., Chen, A., Geiger, A., and Gao, S. (2024a). 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery. (cited on pages 4, 9, 12, 25, 34, 35, 36, 39, 43, 55, 56, 57, 58, 77, 78, 81, and 82)

Huang, Z., Shi, Y., and Gong, M. (2024b). Visibility-aware pixelwise view selection for multi-view stereo matching. *International Conference on Pattern Recognition (ICPR)*. (cited on pages 4, 7, 8, 19, 25, and 26)

Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., and Aanæs, H. (2014). Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE. (cited on page 11)

Jiang, Y., Tu, J., Liu, Y., Gao, X., Long, X., Wang, W., and Ma, Y. (2024). Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5322–5332. (cited on pages 5, 10, 18, 20, 25, and 28)

Jin, H., Liu, I., Xu, P., Zhang, X., Han, S., Bi, S., Zhou, X., Xu, Z., and Su, H. (2023). TensoIR: Tensorial Inverse Rendering . In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, Los Alamitos, CA, USA. IEEE Computer Society. (cited on pages 2 and 8)

Kajiya, J. T. (1986). The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150. (cited on pages 10 and 20)

Karis, B. and Games, E. (2013). Real shading in unreal engine 4. In *Proceedings of Physically Based Shading Theory Practice*, volume 4, page 1. (cited on pages 8, 10, 20, and 21)

Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, page 61–70, Goslar, DEU. Eurographics Association. (cited on page 10)

Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4). (cited on pages 3, 15, 16, and 75)

Kerbl, B., Meuleman, A., Kopanas, G., Wimmer, M., Lanvin, A., and Drettakis, G. (2024). A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics*, 43(4). (cited on page 3)

Keyang, Y., Qiming, H., and Kun, Z. (2024). 3d gaussian splatting with deferred reflection. *ACM SIGGRAPH Conference Proceedings, Denver, CO, United States.* (cited on pages 5, 10, 18, and 20)

Kheradmand, S., Rebain, D., Sharma, G., Sun, W., Tseng, Y.-C., Isack, H., Kar, A., Tagliasacchi, A., and Yi, K. M. (2024). 3d gaussian splatting as markov chain monte carlo. In *Advances in Neural Information Processing Systems (NeurIPS)*. Spotlight Presentation. (cited on pages 30, 36, and 54)

Knapitsch, A., Park, J., Zhou, Q.-Y., and Koltun, V. (2017). Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4). (cited on page 12)

Lai, S., Huang, L., Guo, J., Cheng, K., Pan, B., Long, X., Lyu, J., Lv, C., and Guo, Y. (2025). Glossygs: Inverse rendering of glossy objects with 3d gaussian splatting. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–14. (cited on pages 10 and 20)

Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., and Aila, T. (2020). Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6). (cited on pages 10 and 21)

Li, H., Yang, X., Zhai, H., Liu, Y., Bao, H., and Zhang, G. (2024a). Vox-surf: Voxel-based implicit surface representation. *IEEE Transactions on Visualization and Computer Graphics*, 30(3):1743–1755. (cited on page 8)

Li, J., Wang, L., Zhang, L., and Wang, B. (2024b). Tensosdf: Roughness-aware tensorial representation for robust geometry and material reconstruction. *ACM Trans. Graph.*, 43(4). (cited on pages 2 and 8)

Li, Y., Lyu, C., Di, Y., Zhai, G., Lee, G. H., and Tombari, F. (2024c). Geogaussian: Geometry-aware gaussian splatting for scene rendering. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29 – October 4, 2024, Proceedings, Part XXXV*, page 441–457, Berlin, Heidelberg. Springer-Verlag. (cited on pages 3, 9, and 18)

Li, Z., Müller, T., Evans, A., Taylor, R. H., Unberath, M., Liu, M.-Y., and Lin, C.-H. (2023). Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 2, 8, 12, 34, 35, 36, 81, and 82)

Liang, Z., Zhang, Q., Feng, Y., Shan, Y., and Jia, K. (2024). Gs-ir: 3d gaussian splatting for inverse rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21644–21653. (cited on pages 5, 10, 18, 20, and 25)

Liao, Z., Chen, S., Fu, R., Wang, Y., Su, Z., Luo, H., Ma, L., Xu, L., Dai, B., Li, H., Pei, Z., and Zhang, X. (2024). Fisheye-gs: Lightweight and extensible gaussian splatting module for fisheye cameras. (cited on page 43)

Lin, J., Li, Z., Tang, X., Liu, J., Liu, S., Liu, J., Lu, Y., Wu, X., Xu, S., Yan, Y., and Yang, W. (2024). Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *CVPR*. (cited on page 3)

Lindell, D. B., Martel, J. N. P., and Wetzstein, G. (2021). Autoint: Automatic integration for fast neural volume rendering. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14551–14560. (cited on page 2)

Liu, L., Gu, J., Lin, K. Z., Chua, T.-S., and Theobalt, C. (2020). Neural sparse voxel fields. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc. (cited on page 2)

Liu, Y., Wang, P., Lin, C., Long, X., Wang, J., Liu, L., Komura, T., and Wang, W. (2023). Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images. *ACM Trans. Graph.*, 42(4). (cited on pages 2, 8, and 13)

Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., and Dai, B. (2024). Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664. (cited on page 3)

Lyu, Y., Cheng, K., Kang, X., and Chen, X. (2025). Resgs: Residual densification of 3d gaussian for efficient detail recovery. (cited on page 3)

Meshry, M., Goldman, D. B., Khamis, S., Hoppe, H., Pandey, R., Snavely, N., and Martin-Brualla, R. (2019). Neural rerendering in the wild. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6871–6880. (cited on pages 1 and 75)

Meuleman, A., Shah, I., Lanvin, A., Kerbl, B., and Drettakis, G. (2025). On-the-fly reconstruction for large-scale novel view synthesis from unposed images. *ACM Transactions on Graphics*, 44(4). (cited on page 41)

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*. (cited on page 2)

Moenne-Loccoz, N., Mirzaei, A., Perel, O., de Lutio, R., Esturo, J. M., State, G., Fidler, S., Sharp, N., and Gojcic, Z. (2024). 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics and SIGGRAPH Asia*. (cited on pages 3 and 54)

Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15. (cited on page 2)

Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J. H., Chaitanya, C. R. A., Kaplanyan, A. S., and Steinberger, M. (2021). DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4). (cited on page 2)

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, page 127–136, USA. IEEE Computer Society. (cited on pages 4 and 31)

Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. (2020). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, pages 3501 – 3512, Piscataway, NJ. IEEE. (cited on page 8)

Oechsle, M., Peng, S., and Geiger, A. (2021). Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*. (cited on page 8)

Piala, M. and Clark, R. (2021). Terminerf: Ray termination prediction for efficient neural rendering. In *International Conference on 3D Vision (3DV)*. (cited on page 2)

Radl, L., Steiner, M., Parger, M., Weinrauch, A., Kerbl, B., and Steinberger, M. (2024). Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering. *ACM Trans. Graph.*, 43(4). (cited on page 3)

Reed, N. (2019). Visualizing depth precision. NVIDIA Technical Blog. (cited on page 44)

Schlick, C. (1994). An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum*. (cited on page 20)

Schönberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 7)

Schönberger, J. L., Zheng, E., Pollefeys, M., and Frahm, J.-M. (2016). Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*. (cited on page 7)

Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 519–528. (cited on page 7)

Seitz, S. and Dyer, C. (1997). Photorealistic scene reconstruction by voxel coloring. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1067–1073. (cited on page 7)

Shen, H.-M., Lian, C., Wu, X.-W., Bian, F., Yu, P., and Yang, G. (2021). Full-pose estimation using inertial and magnetic sensor fusion in structurized magnetic field for hand motion tracking. *Measurement*, 170:108697. (cited on page 41)

Shi, Y., Wu, Y., Wu, C., Liu, X., Zhao, C., Feng, H., Zhang, J., Zhou, B., Ding, E., and Wang, J. (2025). Gir: 3d gaussian inverse rendering for relightable scene factorization. *IEEE Transactions on Transactions on Pattern Analysis and Machine Intelligence*. (cited on pages 5, 10, and 20)

Sinha, S. N., Mordohai, P., and Pollefeys, M. (2007). Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. (cited on page 7)

Smith, C., Charatan, D., Tewari, A., and Sitzmann, V. (2024). Flowmap: High-quality camera poses, intrinsics, and depth via gradient descent. (cited on page 41)

Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846. (cited on pages 7 and 15)

Srinivasan, P. P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., and Barron, J. T. (2021). Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*. (cited on page 8)

Sun, J., Chen, X., Wang, Q., Li, Z., Averbuch-Elor, H., Zhou, X., and Snavely, N. (2022). Neural 3d reconstruction in the wild. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, New York, NY, USA. Association for Computing Machinery. (cited on page 8)

Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J., Nießner, M., Pandey, R., Fanello, S., Wetzstein, G., Zhu, J.-Y., Theobalt, C., Agrawala, M., Shechtman, E., Goldman, D. B., and Zollhöfer, M. (2020). State of the art on neural rendering. (cited on page 1)

Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Xu, Z., Simon, T., Nießner, M., Tretschk, E., Liu, L., Mildenhall, B., Srinivasan, P., Pandey, R., Orts-Escolano, S., Fanello, S., Guo, M., Wetzstein, G., Zhu, J.-Y., Theobalt, C., Agrawala, M., Goldman, D. B., and Zollhöfer, M. (2021). Advances in neural rendering. In *ACM SIGGRAPH 2021 Courses*, SIGGRAPH '21, New York, NY, USA. Association for Computing Machinery. (cited on page 2)

Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J. T., and Srinivasan, P. P. (2022). Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*. (cited on pages 8 and 13)

Walter, B., Marschner, S. R., Li, H., and Torrance, K. E. (2007). Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, page 195–206, Goslar, DEU. Eurographics Association. (cited on page 20)

Wang, J., Liu, Y., Wang, P., Lin, C., Hou, J., Li, X., Komura, T., and Wang, W. (2024). Gaussurf: Geometry-guided 3d gaussian splatting for surface reconstruction. *arXiv preprint arXiv:2411.19454*. (cited on pages 4, 9, 12, 25, 26, 34, and 81)

Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. (2021a). Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*. (cited on pages 2 and 8)

Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. (2021b). Neus: learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA. Curran Associates Inc. (cited on pages 12, 34, 35, 36, 81, and 82)

Wang, Y., Han, Q., Habermann, M., Daniilidis, K., Theobalt, C., and Liu, L. (2023). Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. (cited on pages 2, 8, 34, 35, 81, and 82)

Wang, Y., Skorokhodov, I., and Wonka, P. (2022). Hf-neus: improved surface reconstruction using high-frequency details. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc. (cited on page 8)

Weng, J., Cohen, P., and Herniou, M. (1992). Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980. (cited on page 43)

Wu, Q., Martinez Esturo, J., Mirzaei, A., Moenne-Loccoz, N., and Gojcic, Z. (2025). 3dgut: Enabling distorted cameras and secondary rays in gaussian splatting. *Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 43)

Wu, T.-P., Yeung, S.-K., Jia, J., and Tang, C.-K. (2010). Quasi-dense 3d reconstruction using tensor-based multiview stereo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1482–1489. (cited on page 7)

Yang, C., Li, S., Fang, J., Liang, R., Xie, L., Zhang, X., Shen, W., and Tian, Q. (2024). Gaussianobject: High-quality 3d object reconstruction from four views with gaussian splatting. *ACM Transactions on Graphics*, 43(6). (cited on page 3)

Yang, C., Simon, G., See, J., Berger, M.-O., and Wang, W. (2020). Watchpose: A view-aware approach for camera pose data collection in industrial environments. *Sensors*, 20(11):3045. (cited on page 41)

Yao, Y., Zhang, J., Liu, J., Qu, Y., Fang, T., McKinnon, D., Tsin, Y., and Quan, L. (2022). Neilf: Neural incident light field for physically-based material estimation. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXI*, page 700–716, Berlin, Heidelberg. Springer-Verlag. (cited on page 8)

Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. (2021a). Volume rendering of neural implicit surfaces. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA. Curran Associates Inc. (cited on pages 2 and 8)

Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. (2021b). Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*. (cited on pages 34, 35, 81, and 82)

Yariv, L., Hedman, P., Reiser, C., Verbin, D., Srinivasan, P. P., Szeliski, R., Barron, J. T., and Mildenhall, B. (2023). Bakedsdf: Meshing neural sdfs for real-time view synthesis. *arXiv*. (cited on page 8)

Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., and Lipman, Y. (2020). Multiview neural surface reconstruction by disentangling geometry and appearance. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2492–2502. Curran Associates, Inc. (cited on page 8)

Ye, B., Liu, S., Xu, H., Xueting, L., Pollefeys, M., Yang, M.-H., and Songyou, P. (2024a). No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *arXiv preprint arXiv:2410.24207*. (cited on page 41)

Ye, C., Qiu, L., Gu, X., Zuo, Q., Wu, Y., Dong, Z., Bo, L., Xiu, Y., and Han, X. (2024b). Stablenormal: Reducing diffusion variance for stable and sharp normal. *ACM Transactions on Graphics*. (cited on page 10)

Ye, K., Gao, C., Li, G., Chen, W., and Chen, B. (2024c). Geosplatting: Towards geometry guided gaussian splatting for physically-based inverse rendering. (cited on pages 5, 10, and 20)

Ye, Z., Li, W., Liu, S., Qiao, P., and Dou, Y. (2024d). Absgs: Recovering fine details in 3d gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 1053–1061, New York, NY, USA. Association for Computing Machinery. (cited on pages 3 and 30)

Yoo, J.-C. and Han, T. H. (2009). Fast normalized cross-correlation. *Circuits Syst. Signal Process.*, 28(6):819–843. (cited on page 27)

Yu, A., Li, R., Tancik, M., Li, H., Ng, R., and Kanazawa, A. (2021). Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. (cited on page 2)

Yu, Z., Chen, A., Huang, B., Sattler, T., and Geiger, A. (2024a). Mip-splatting: Alias-free 3d gaussian splatting. *Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 3)

Yu, Z., Peng, S., Niemeyer, M., Sattler, T., and Geiger, A. (2022). Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*. (cited on page 8)

Yu, Z., Sattler, T., and Geiger, A. (2024b). Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*. (cited on pages 4, 9, 12, 25, 34, 35, 36, 81, and 82)

Zhang, C., Zou, Y., Li, Z., Yi, M., and Wang, H. (2025a). Transplat: Generalizable 3d gaussian splatting from sparse multi-view images with transformers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(9):9869–9877. (cited on page 3)

Zhang, J., Yao, Y., Li, S., Fang, T., McKinnon, D., Tsin, Y., and Quan, L. (2022). Critical regularizations for neural surface reconstruction in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6270–6279. (cited on pages 8, 34, 35, 81, and 82)

Zhang, J. Y., Lin, A., Kumar, M., Yang, T.-H., Ramanan, D., and Tulsiani, S. (2024). Cameras as rays: Pose estimation via ray diffusion. In *International Conference on Learning Representations (ICLR)*. (cited on page 41)

Zhang, K., Luan, F., Wang, Q., Bala, K., and Snavely, N. (2021). PhySG: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 8)

Zhang, Y., Chen, A., Wan, Y., Song, Z., Yu, J., Luo, Y., and Yang, W. (2025b). Ref-gs: Directional factorization for 2d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. (cited on pages 5, 10, and 20)

Zheng, P., Gao, D., Fan, D.-P., Liu, L., Laaksonen, J., Ouyang, W., and Sebe, N. (2024). Bilateral reference for high-resolution dichotomous image segmentation. *CAAI Artificial Intelligence Research*, 3:9150038. (cited on pages 47, 61, and 79)

Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*. (cited on page 31)

Zhu, Z., Fan, Z., Jiang, Y., and Wang, Z. (2023). Fsgs: Real-time few-shot view synthesis using gaussian splatting. (cited on page 3)

Zhu, Z.-L., Wang, B., and Yang, J. (2025). Gs-ror^2: Bidirectional-guided 3dgs and sdf for reflective object relighting and reconstruction. (cited on pages 5, 9, 10, 18, 20, and 25)

Zwicker, M., Pfister, H., van Baar, J., and Gross, M. (2001). Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538. (cited on page 15)

BIBLIOGRAPHY

List of Figures

LIST OF FIGURES

List of Tables