# RENDERING-OPTIMAL IS NOT MESH-OPTIMAL: COMPOSING MULTI-BUDGET GAUSSIANS FOR COMPACT SURFACE EXTRACTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

3D Gaussian Splatting (3DGS) achieves high-quality rendering, but its Gaussians do not form an explicit mesh. Gaussian-based surface reconstruction methods fit a surface to the learned Gaussians and mesh it on a spatial structure built around them, yielding an explicit mesh. In these methods, mesh complexity is largely determined by the spatial distribution of Gaussians, with denser Gaussians tending to yield denser triangulation. Obtaining a compact yet detailed mesh therefore depends on allocating Gaussians densely where the geometry is complex and sparsely elsewhere. However, the placement and density of these Gaussians are driven by photometric supervision and may not align with the geometry required for compact mesh reconstruction. We address this mismatch by composing Gaussians trained at different budgets, *i.e.*, Gaussian counts, into a single set of Gaussians, drawing each region from the budget its geometry needs. Starting from the lowest budget, we keep its Gaussians where they already capture the geometry and replace them with higher-budget Gaussians only where rendered depth indicates a sufficient geometric gain. The resulting composite is then meshed with the underlying reconstruction method's own extraction procedure, yielding more compact meshes while preserving geometric quality. Across multiple Gaussian-based surface reconstruction methods and datasets, our composites improve the geometry–mesh-size trade-off, reducing mesh size by 26–59% relative to extraction from the finest budget while maintaining comparable or better geometric accuracy and rendering quality.

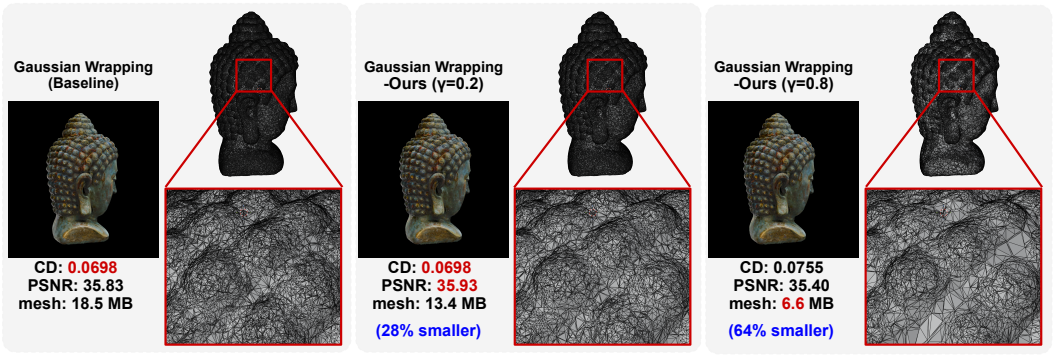


Figure 1: **Compact surface extraction through geometry-aware Gaussian composition.** We compare a mesh extracted directly from Gaussian Wrapping (Gomez et al., 2026), a Gaussian-based surface reconstruction method (left), with meshes extracted from our composites of its lower-budget Gaussians (middle, right). Each panel shows the mesh rendering, its wireframe, and an enlarged wireframe of the red box on the curls of the Buddha's head. Our meshes are 28% and 64% smaller in file size (18.5 MB vs. 13.4 and 6.6 MB) with comparable rendering PSNR. The threshold $\gamma$ on the relative depth gain controls how many higher-budget Gaussians are introduced: a larger $\gamma$ introduces fewer Gaussians and yields a smaller mesh. In the close-ups, the flat patches between the curls are covered by fewer and larger triangles, whereas the curls keep dense triangulation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

# 1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) enables high-quality, real-time novel-view rendering, but its Gaussians do not form an explicit mesh, which limits its use in mesh-based graphics workflows such as editing and simulation. Gaussian-based surface reconstruction methods address this limitation by recovering explicit meshes (Guédon & Lepetit, 2024; Yu et al., 2024; Guédon et al., 2025; Gomez et al., 2026). These meshes should not only capture accurate geometry but also preserve geometric detail while remaining compact for downstream graphics applications. Meeting these goals requires allocating mesh complexity according to scene geometry, which is difficult to achieve with standard fixed-resolution extraction such as TSDF fusion (Curless & Levoy, 1996): a fine grid wastes triangles on flat regions and a coarse grid loses thin structures.

Unlike fixed-resolution extraction, Gaussian-based extraction varies mesh resolution across regions, since the mesh follows the non-uniform distribution of Gaussians; their positions, scales, and density largely determine the complexity of the extracted mesh. However, this distribution is mainly driven by photometric supervision. Geometric regularizers such as depth–normal consistency (Huang et al., 2024) constrain the shape and orientation of individual Gaussians, but do not directly control their placement or density. As a result, the spatial allocation of Gaussians need not align with the geometric complexity required by the extracted mesh: a rendering-optimal allocation of Gaussians therefore need not be mesh-optimal. Textured-Gaussian methods (Rong et al., 2025; Zhang et al., 2025; Papantonakis et al., 2025) decouple appearance from the Gaussians through textures, so fine appearance detail no longer requires many small Gaussians. This reduces the number of Gaussians needed for rendering, but does not explicitly allocate them according to geometric complexity.

To better align the spatial allocation of Gaussians with geometric complexity before mesh extraction, we compose Gaussians trained at different budgets,*i.e.*, Gaussian counts, into a single set of Gaussians. Our approach builds on the intuition that higher-budget Gaussians provide varying geometric benefit across regions and add little where lower-budget Gaussians already capture the geometry. We therefore compose Gaussians progressively, starting from the lowest budget. At each stage, we measure the benefit of higher-budget Gaussians as the relative reduction in error to a reference depth rendered from the finest budget, and replace Gaussians only where this relative depth gain reaches a threshold $\gamma$, which controls the geometry–mesh-size trade-off (Fig. 1). We then apply the underlying method's own mesh extraction procedure to the final composite to obtain compact meshes.

Across three Gaussian-based surface reconstruction methods and multiple datasets, our composites improve the geometry–mesh-size trade-off, producing meshes that are 26–59% smaller than those extracted from the finest budget, with comparable or better geometric accuracy and rendering quality. Notably, with comparable Gaussian rendering PSNR, our composites yield smaller meshes with lower CD while preserving mesh rendering PSNR, supporting that rendering-optimal Gaussians need not be mesh-optimal (see Sec. 4.4).

Our contributions are summarized as follows:

- We show that the spatial arrangement of Gaussians optimized for rendering need not align with the allocation of mesh complexity required to represent scene geometry.
- Motivated by this observation, we propose a geometry-aware composition method that combines Gaussians across budgets based on depth-guided geometric improvement.
- We demonstrate that our method generalizes across three Gaussian-based surface reconstruction methods and multiple datasets, consistently improving the geometry–mesh-size trade-off.

# 2 RELATED WORK

**Neural and Gaussian-based surface reconstruction.** Neural implicit methods represent surfaces as continuous occupancy or signed distance fields (Mescheder et al., 2019; Park et al., 2019; Yariv et al., 2020; Oechsle et al., 2021; Wang et al., 2021; Yariv et al., 2021; Li et al., 2023), but their uniform sampling for mesh extraction (Lorensen & Cline, 1987; Curless & Levoy, 1996) allocates excessive resolution to smooth regions. Analytic extraction recovers the exact zero-level set of piecewise-linear networks (Lei & Jia, 2020; Kim, 2024; Oh et al., 2026), but applies only to such architectures, and its mesh complexity follows the network's linear regions rather than the scene

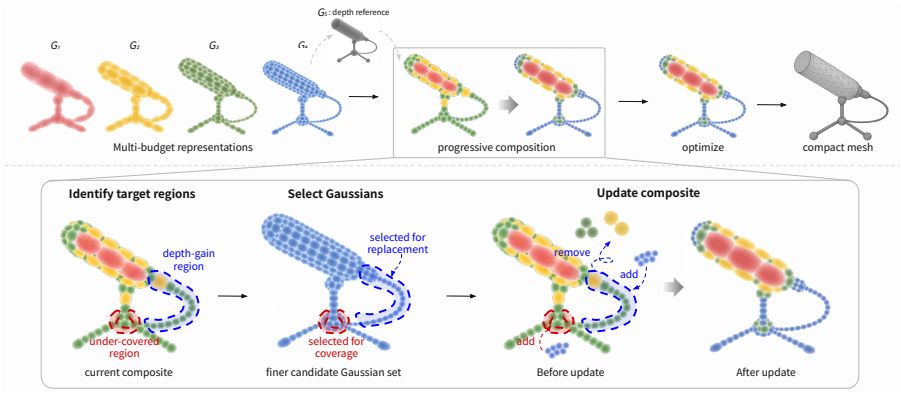


Figure 2: **Method overview.** We progressively compose $\mathcal{G}_1$–$\mathcal{G}_4$, using $\mathcal{G}_5$ only as a fixed depth reference. At each level, we identify under-covered regions and regions where finer Gaussians provide sufficient geometric improvement, then select Gaussians for coverage and replacement. The composite is updated by removing and adding the selected Gaussians, then briefly optimized and passed to the original mesh extraction procedure to obtain a compact mesh.

geometry. Differentiable isosurface methods (Shen et al., 2021; 2023) adaptively optimize surface geometry but increase memory and computational costs, limiting scalability at high resolutions and for large scenes. Gaussian-based approaches extract meshes through Poisson reconstruction (Kazhdan et al., 2006) of sampled surface points (Guédon & Lepetit, 2024) or exploit the spatially non-uniform distribution of primitives for adaptive surface extraction (Yu et al., 2024; Gomez et al., 2026; Guédon et al., 2025; Zhang et al., 2026). While these methods focus on accurate surface reconstruction, we aim to preserve geometric quality under a limited budget by selectively composing Gaussian primitives before applying existing mesh extraction procedures.

**Mesh compactness.** Vertex contraction and edge-collapse methods reduce mesh complexity while controlling geometric deviation, using quadric error metrics or other local approximation criteria (Hoppe, 1996; Garland & Heckbert, 1997; Lindstrom & Turk, 1999). Appearance-aware simplification additionally accounts for texture and normal variation (Cohen et al., 1998). Mesh compression methods reduce storage while maintaining triangle count (Deering, 1995; Rossignac, 1999). In contrast, our method addresses mesh compactness before extraction by composing Gaussian representations at different budgets.

**Efficient Gaussian representations.** Efficient Gaussian methods aim to preserve rendering quality under a limited Gaussian budget (Kheradmand et al., 2024; Mallick et al., 2024). Other methods reduce the number of primitives through redundant primitive removal or sampling-based simplification (Fan et al., 2024; Lee et al., 2024; Deng et al., 2025; Fang & Wang, 2024). Multiscale and hierarchical representations select Gaussian detail according to image resolution or viewing conditions (Yan et al., 2024; Ren et al., 2025; Seo et al., 2025). These methods, including textured Gaussians that decouple appearance through textures (Rong et al., 2025; Zhang et al., 2025; Papantonakis et al., 2025), primarily reduce Gaussian count or adapt detail for rendering, whereas our method explicitly reallocates Gaussians according to geometric improvement.

## 3 Method

We formulate multi-budget composition (Sec. 3.1), define a relative-depth-gain criterion (Sec. 3.2), and use it for progressive composition followed by optimization and mesh extraction (Sec. 3.3). Fig. 2 provides an overview of our pipeline.

### 3.1 Problem Formulation

Our inputs are $K$ candidate Gaussian sets $\mathcal{G}_1, \ldots, \mathcal{G}_K$ of the same scene, ordered by increasing primitive count, where $N_l = |\mathcal{G}_l|$ and $N_1 < \cdots < N_K$. We refer to Gaussian sets with larger

primitive counts as finer Gaussian sets. Since Gaussian count and spatial arrangement influence mesh complexity in the extraction procedures of the underlying reconstruction methods we use, our goal is to allocate primitives according to their geometric benefit to obtain compact meshes.

We progressively compose the candidate Gaussian sets and denote the composite after processing level $l$ by $\mathcal{C}_l$. We additionally use the finest Gaussian set $\mathcal{G}_R$, where $R > K$ and $N_R > N_K$, only to render reference depth maps. Its Gaussians are not included in the composite. These depth maps serve as a common geometric reference for both the selection of Gaussians from finer candidate sets and the subsequent optimization of the composed Gaussians.

### 3.2 Relative-Depth-Gain Refinement Criterion

We aim to introduce finer Gaussians where they provide geometric improvement over the previous composite. To assess this improvement, we measure how much the finer candidate set reduces the absolute error to the reference depth relative to the previous composite. We refer to this absolute difference as the reference-depth error and use its reduction as a proxy for geometric improvement. Since the magnitude of the reference-depth error varies across scenes and candidate levels, we normalize the error reduction by the previous composite's reference-depth error.

**Relative-depth-gain criterion.** Let $D_v(p; X)$ denote the alpha-blended expected depth at pixel $p$ rendered from Gaussian set $X$ in training view $v$. At each level $l \geq 2$, we compare the previous composite $\mathcal{C}_{l-1}$ and the full candidate set $\mathcal{G}_l$ using the reference depth rendered from $\mathcal{G}_R$. We define the reference-depth error of $X$ at pixel $p$ as

$$e_v(p; X) = |D_v(p; X) - D_v(p; \mathcal{G}_R)| \,. \tag{1}$$

Using this reference-depth error, we quantify the pixel-wise relative depth gain of the finer candidate set $\mathcal{G}_l$ over the previous composite $\mathcal{C}_{l-1}$ by the relative reduction in this error:

$$r_v^{(l)}(p) = \frac{e_v(p; \mathcal{C}_{l-1}) - e_v(p; \mathcal{G}_l)}{e_v(p; \mathcal{C}_{l-1}) + \epsilon_d}. \tag{2}$$

Here $\epsilon_d > 0$ stabilizes the denominator, and $r_v^{(l)}(p)$ measures error reduction relative to the previous composite's reference-depth error.

**Relative-depth-gain threshold.** We identify pixels with sufficient relative depth gain by thresholding $r_v^{(l)}(p)$ with $\gamma$. For pixels where at least one of $D_v(p; \mathcal{C}_{l-1})$, $D_v(p; \mathcal{G}_l)$, or $D_v(p; \mathcal{G}_R)$ is positive, we define the pixel-selection mask as

$$H_v^{(l)}(p) = \mathbb{I}[r_v^{(l)}(p) \geq \gamma], \qquad 0 \leq \gamma < 1. \tag{3}$$

Pixels for which all three depths are zero are assigned $H_v^{(l)}(p) = 0$. $\mathbb{I}[\cdot]$ is the indicator function.

**Gaussian-wise relative-depth-gain score.** We extend the relative depth gains at these selected pixels to a Gaussian-wise score by weighting them according to each Gaussian's rendering contribution. Let $w_{v,g}(p; X)$ denote the standard front-to-back alpha-compositing weight $\mathcal{T}_{v,g}(p; X)\alpha_{v,g}(p)$ of Gaussian $g$ at pixel $p$ when rendering $X$, where $\mathcal{T}_{v,g}(p; X)$ denotes the accumulated transmittance before Gaussian $g$ and $\alpha_{v,g}(p)$ its opacity at $p$. For $l \geq 2$, we define the Gaussian-wise score as

$$q_l(g; X) = \frac{\sum_{v,p} w_{v,g}(p; X) H_v^{(l)}(p) r_v^{(l)}(p)}{\sum_{v,p} w_{v,g}(p; X) + \epsilon_w}, \tag{4}$$

where the sums run over all training views and pixels. $X$ is the Gaussian set used for rendering, and $\epsilon_w > 0$ is a numerical stabilizer. We normalize by the Gaussian's total contribution so that a Gaussian scores high only when a sufficient fraction of its contribution falls within selected regions.

### 3.3 Progressive Gaussian Composition

We progressively construct the Gaussian composite $\mathcal{C}_l$ using the candidate Gaussian sets in order of increasing budget. At each level $l$, we first select Gaussians from $\mathcal{G}_l$ for coverage in under-covered foreground regions and then use the Gaussian-wise relative depth gain score to select Gaussians for replacement. Both steps are computed with $\mathcal{C}_{l-1}$ fixed before updating the composite.

**Coverage-guided Gaussian addition.** When foreground masks are available, we identify regions requiring additional coverage using the residual between the foreground mask and the composite's rendered alpha. Let $M_v$ denote the binary foreground mask in view $v$ and $A_v(X)$ the rendered alpha of Gaussian set $X$. The coverage target is

$$T_v^{(1)} = M_v, \qquad T_v^{(l)} = [M_v - A_v(\mathcal{C}_{l-1})]_+ \quad (l > 1), \tag{5}$$

where $[x]_+ = \max(x, 0)$ restricts the target to pixels with insufficient coverage.

Using the rendering contribution weights $w_{v,g}(p; \mathcal{G}_l)$, we score each Gaussian in $\mathcal{G}_l$ according to its contribution to the coverage target $T_v^{(l)}$ and define the coverage score $s_l(g)$ and selected set $\mathcal{S}_l$ as

$$s_l(g) = \frac{\sum_{v,p} w_{v,g}(p; \mathcal{G}_l) T_v^{(l)}(p)}{\sum_{v,p} w_{v,g}(p; \mathcal{G}_l) + \epsilon_w}, \qquad \mathcal{S}_l = \{g \in \mathcal{G}_l : s_l(g) \geq \tau_l\}, \tag{6}$$

where $\tau_l$ is the selection threshold at level $l$ (Appendix A.2). The Gaussians selected from $\mathcal{G}_1$ form the initial composite, $\mathcal{C}_1 = \mathcal{S}_1$. When masks are unavailable, we omit coverage-guided Gaussian addition. In this case, we initialize $\mathcal{C}_1 = \mathcal{G}_1$ and set $\mathcal{S}_l = \varnothing$ for all subsequent levels $l \geq 2$. Further details for this setting are provided in Appendix A.2.

**Relative-depth-gain Gaussian replacement.** We use the Gaussian-wise relative-depth-gain score $q_l(g; X)$ defined in Sec. 3.2 to select Gaussians for replacement. For a Gaussian $g \in \mathcal{C}_{l-1}$, a high $q_l(g; \mathcal{C}_{l-1})$ indicates that it contributes strongly to regions where the finer candidate set $\mathcal{G}_l$ provides sufficient relative depth gain. We therefore remove existing Gaussians whose scores are at least $\gamma$:

$$\mathcal{R}_l = \{g \in \mathcal{C}_{l-1} : q_l(g; \mathcal{C}_{l-1}) \geq \gamma\}. \tag{7}$$

We then select Gaussians from the finer candidate set $\mathcal{G}_l$ to replace the removed Gaussians. To avoid reselecting Gaussians already chosen for coverage-guided addition, we exclude $\mathcal{S}_l$ and define the remaining set as $\mathcal{B}_l = \mathcal{G}_l \setminus \mathcal{S}_l$. We evaluate $q_l(g; \mathcal{B}_l)$ over this set and select Gaussians whose scores are at least $\gamma$ as the replacement set $\mathcal{A}_l$:

$$\mathcal{A}_l = \{g \in \mathcal{B}_l : q_l(g; \mathcal{B}_l) \geq \gamma\}. \tag{8}$$

After determining the removal set $\mathcal{R}_l$ and replacement set $\mathcal{A}_l$, we update the composite by removing $\mathcal{R}_l$ from $\mathcal{C}_{l-1}$ and adding the Gaussians selected for coverage ($\mathcal{S}_l$) and replacement ($\mathcal{A}_l$):

$$\mathcal{C}_l = (\mathcal{C}_{l-1} \setminus \mathcal{R}_l) \cup \mathcal{S}_l \cup \mathcal{A}_l. \tag{9}$$

We use the same threshold $\gamma$ for pixel selection and Gaussian selection. This shared threshold controls the extent of relative-depth-gain replacement and the resulting geometry–mesh-size trade-off. Further details on the shared threshold and its effect are provided in Appendices A.2 and B.

**Gaussian optimization and mesh extraction.** After composing all candidate Gaussian sets, we optimize the resulting Gaussian composite $\mathcal{C}_K$. Combining Gaussians from different Gaussian sets changes their occlusion relationships and contributions to rendered pixels. We therefore perform a brief optimization of their parameters while keeping the Gaussian count fixed. The loss combines RGB reconstruction, reference-depth alignment, and depth-normal consistency:

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{depth}} \mathcal{L}_{\text{depth}} + \lambda_{\text{normal}} \mathcal{L}_{\text{normal}}. \tag{10}$$

The losses and optimization settings are provided in Appendix A.3. We then apply each method's own extraction procedure to the optimized composite to obtain the final mesh.

# 4 Experiments

## 4.1 Experimental Setup

**Datasets and metrics.** We evaluate geometric reconstruction quality on the DeMapGS dataset (Zhou et al., 2025) and mesh rendering quality on the NeRF Synthetic dataset (Mildenhall

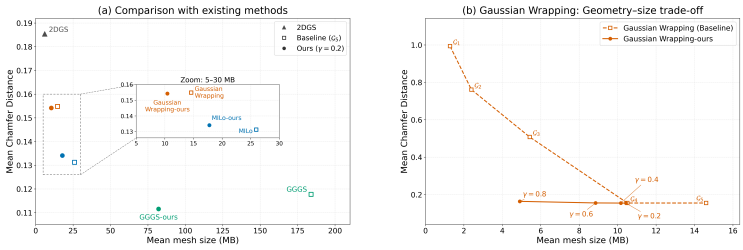


Figure 3: **Geometry–mesh-size trade-offs on the DeMapGS dataset.** (a) Comparison with $\mathcal{G}_5$ baselines across reconstruction methods, with Ours evaluated at $\gamma = 0.2$ and 2DGS (Huang et al., 2024) included as an additional baseline. (b) Gaussian Wrapping at different budgets and Ours varying $\gamma$.

et al., 2020). We further evaluate real-world scenes from the Mip-NeRF 360 dataset (Barron et al., 2022) to assess applicability beyond object-centric reconstruction. Geometric quality is measured using Chamfer distance (CD). Mesh rendering is evaluated using PSNR, SSIM, and LPIPS after appearance fitting with fixed mesh geometry. We measure mesh size in megabytes (MB) to compare storage requirements. For the DeMapGS dataset, we average results over five scenes and report the Box scene separately because its substantially larger reconstruction errors dominate the six-scene mean. Both five- and six-scene averages are provided in Appendix C. Dataset descriptions, scene lists, evaluation details, and additional DTU (Jensen et al., 2014) and Tanks and Temples (Knapitsch et al., 2017) results are provided in Appendices C to E.

**Reconstruction methods.** We apply our framework to Gaussian representations obtained from MILo (Guédon et al., 2025), GGGS (Zhang et al., 2026), and Gaussian Wrapping (Gomez et al., 2026). For each method, we compare our composites with meshes extracted directly from its representations at individual budgets. For MILo, both the comparison baseline and the inputs to our framework exclude mesh–Gaussian joint optimization. We use the Gaussian representations obtained before this stage, as they yielded lower CD in our setting while requiring substantially less training than the joint-optimization stage. All meshes are extracted using the original extraction procedure of the underlying reconstruction method.

**Multi-budget representations.** We construct five Gaussian representations $\mathcal{G}_1, \ldots, \mathcal{G}_5$ per scene through budget-controlled training or sampling, ordered by increasing Gaussian count. In our main setting, we compose $\mathcal{G}_1, \ldots, \mathcal{G}_4$ and use $\mathcal{G}_5$ only as a depth reference for selection and optimization. Method-specific construction settings and Gaussian budgets are provided in Appendix A.1.

### 4.2 Geometry–Mesh-Size Trade-offs on the DeMapGS Dataset

We evaluate the trade-off between geometric accuracy and mesh size on the DeMapGS dataset using Gaussian Wrapping, MILo, and GGGS. We measure geometric accuracy using CD and additionally evaluate novel-view mesh rendering. For each method, meshes extracted from $\mathcal{G}_1, \ldots, \mathcal{G}_4$ define the baseline operating points. We generate our curve by varying the relative-depth-gain threshold $\gamma$ while keeping these four candidate representations and the reference depth from $\mathcal{G}_5$ fixed.

Fig. 3(a) compares Ours across the three reconstruction methods, while Fig. 3(b) compares Gaussian Wrapping baselines at different budgets with Ours under varying $\gamma$. Our method achieves comparable CD with smaller meshes, showing improved mesh efficiency over direct extraction at individual budgets. Table 1 reports mesh sizes and CD for meshes extracted from $\mathcal{G}_5$ and $\mathcal{G}_4$ and those produced by our method at $\gamma \in \{0.2, 0.6\}$. At $\gamma = 0.2$, Ours reduces mesh size by approximately 29%, 32%, and 55% relative to $\mathcal{G}_5$ for Gaussian Wrapping, MILo, and GGGS, respectively. Gaussian Wrapping and GGGS achieve these reductions with lower CD. For MILo, mesh size decreases from 25.97 to 17.78 MB, while CD increases slightly from 0.131 to 0.134. The comparison with $G_4$ further shows that composition can improve the size–quality trade-off beyond simply using a lower-budget representation. At $\gamma = 0.6$, Ours further reduces mesh size by approximately 15–48% relative to $\gamma = 0.2$, at the cost of some decrease in geometric accuracy. This reduction illustrates how $\gamma$ controls the balance between geometric accuracy and mesh size.

Table 1: **Geometric accuracy and novel-view mesh rendering results on the DeMapGS dataset.** Ours composes $\mathcal{G}_1$–$\mathcal{G}_4$ with $\gamma \in \{0.2, 0.6\}$. Bold indicates the best per baseline.

| Baseline | Representation | Mesh (MB) ↓ | CD ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
|---|---|---|---|---|---|---|
| Gaussian Wrapping | $\mathcal{G}_5$ | 14.61 | 0.1548 | 37.873 | 0.9812 | **0.0325** |
| | $\mathcal{G}_4$ | 10.51 | 0.1548 | 37.798 | 0.9811 | 0.0329 |
| | Ours ($\gamma = 0.2$) | 10.43 | **0.1542** | **37.961** | **0.9813** | 0.0328 |
| | Ours ($\gamma = 0.6$) | **8.85** | 0.1552 | 37.874 | 0.9809 | 0.0334 |
| MILo | $\mathcal{G}_5$ | 25.97 | **0.1312** | **38.010** | **0.9803** | **0.0335** |
| | $\mathcal{G}_4$ | 18.29 | 0.1381 | 37.714 | 0.9793 | 0.0346 |
| | Ours ($\gamma = 0.2$) | 17.78 | 0.1341 | 37.832 | 0.9797 | 0.0343 |
| | Ours ($\gamma = 0.6$) | **9.16** | 0.1579 | 37.271 | 0.9770 | 0.0368 |
| GGGS | $\mathcal{G}_5$ | 184.00 | 0.1177 | 37.332 | 0.9809 | 0.0313 |
| | $\mathcal{G}_4$ | 77.15 | 0.1180 | 37.066 | 0.9803 | 0.0321 |
| | Ours ($\gamma = 0.2$) | 82.10 | **0.1116** | **37.562** | **0.9812** | **0.0307** |
| | Ours ($\gamma = 0.6$) | **59.07** | 0.1137 | 37.308 | 0.9803 | 0.0316 |

Table 2: **Novel-view mesh rendering on the NeRF Synthetic dataset.** Ours composes $\mathcal{G}_1$–$\mathcal{G}_4$ with $\gamma \in \{0.2, 0.6\}$. Bold indicates the best per baseline.

| Baseline | Representation | Mesh (MB) ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
|---|---|---|---|---|---|
| Gaussian Wrapping | $\mathcal{G}_5$ | 29.13 | 27.302 | 0.9231 | 0.0848 |
| | $\mathcal{G}_4$ | 21.30 | 27.161 | 0.9219 | 0.0859 |
| | Ours ($\gamma = 0.2$) | 21.12 | **27.327** | **0.9235** | **0.0846** |
| | Ours ($\gamma = 0.6$) | **17.90** | 27.247 | 0.9226 | 0.0855 |
| MILo | $\mathcal{G}_5$ | 24.14 | 27.666 | 0.9225 | 0.0902 |
| | $\mathcal{G}_4$ | 17.91 | 27.361 | 0.9192 | 0.0929 |
| | Ours ($\gamma = 0.2$) | 17.86 | **27.775** | **0.9242** | **0.0873** |
| | Ours ($\gamma = 0.6$) | **10.34** | 27.466 | 0.9208 | 0.0904 |
| GGGS | $\mathcal{G}_5$ | 245.75 | 25.685 | 0.9034 | 0.1091 |
| | $\mathcal{G}_4$ | 97.38 | 25.338 | 0.9028 | 0.1053 |
| | Ours ($\gamma = 0.2$) | 100.47 | **25.986** | **0.9094** | **0.0981** |
| | Ours ($\gamma = 0.6$) | **60.16** | 25.695 | 0.9056 | 0.1017 |

Table 1 also reports metrics using MILo's color-field rendering with fixed mesh geometry. At $\gamma = 0.2$, the smaller meshes maintain comparable rendering quality across the three methods, with GGGS improving all three metrics. At $\gamma = 0.6$, further mesh-size reductions are obtained with only minor drops in rendering quality. Additional renderings and wireframe close-ups are in Appendix F.

### 4.3 Novel-View Rendering on the NeRF Synthetic Dataset

As in Sec. 4.2, we evaluate novel-view mesh rendering on the NeRF Synthetic dataset using MILo's color-field approach, with mesh geometry fixed during appearance optimization. Table 2 compares meshes extracted from $\mathcal{G}_4$ and $\mathcal{G}_5$ with those produced by our method at $\gamma \in \{0.2, 0.6\}$. At $\gamma = 0.2$, Ours reduces mesh size by approximately 28%, 26%, and 59% relative to $\mathcal{G}_5$ for Gaussian Wrapping, MILo, and GGGS, respectively, while improving PSNR, SSIM, and LPIPS for all three methods. Compared with $\mathcal{G}_4$, Ours also achieves a better rendering-quality–mesh-size trade-off, indicating that the improvement is not explained by simply using a lower-budget representation. At $\gamma = 0.6$, Ours further reduces mesh size by approximately 15–42% relative to $\gamma = 0.2$, while rendering quality changes only slightly.

Fig. 4 provides qualitative comparisons at $\gamma = 0.2$ and $0.6$, illustrating the trade-off at different mesh sizes. The rendered images exhibit comparable quality, and the wireframe close-ups show the spatial distribution of triangle density. Ours reduces density in smooth regions while maintaining dense

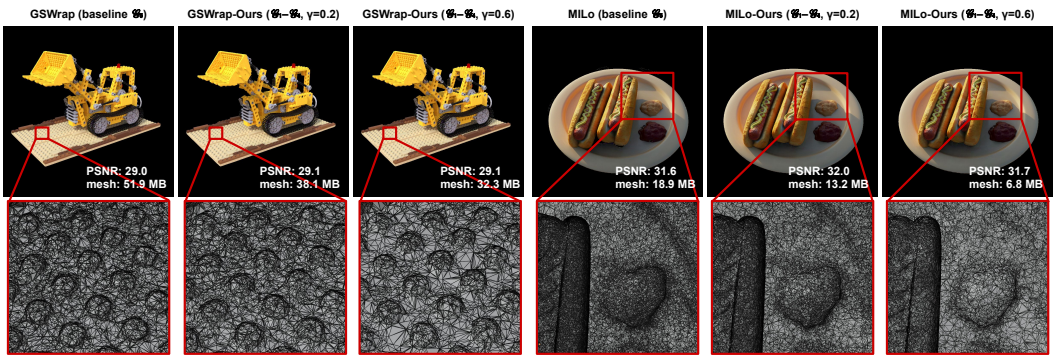


Figure 4: **Qualitative comparisons on the NeRF Synthetic dataset.** Rendered images and wireframe close-ups compare the $\mathcal{G}_5$ baselines with our results at $\gamma = 0.2$ and $0.6$.

Table 3: **Gaussian rendering quality and extracted-mesh quality on the DeMapGS dataset.** Ours uses $\gamma = 0.2$. Bold indicates the better result between Ours and $\mathcal{G}_5$ for Gaussian Wrapping, and between Ours and $\mathcal{G}_4$ for MILo. MILo's $\mathcal{G}_5$ results are included for reference.

| | | Gaussian | Mesh | | |
|---|---|---|---|---|---|
| Baseline | Representation | PSNR↑ | MB↓ | CD↓ | PSNR↑ |
| Gaussian Wrapping | $\mathcal{G}_5$ | **43.267** | 14.61 | 0.1548 | 37.873 |
| | Ours ($\mathcal{G}_1$–$\mathcal{G}_4$) | 42.911 | **10.43** | **0.1542** | **37.961** |
| MILo | $\mathcal{G}_5$ (reference) | 42.114 | 25.97 | 0.1312 | 38.010 |
| | $\mathcal{G}_4$ | **41.238** | 18.29 | 0.1381 | 37.714 |
| | Ours ($\mathcal{G}_1$–$\mathcal{G}_4$) | 40.953 | **17.78** | **0.1341** | **37.832** |

triangulation around geometric details. These observations further support the improved trade-off between rendering quality and mesh size achieved by Ours. Additional trade-off curves comparing Ours with meshes extracted directly at different Gaussian budgets are provided in Appendix G.

### 4.4 Analysis and Ablation Studies

**Gaussian rendering quality and mesh efficiency.** Table 3 shows that, compared with $\mathcal{G}_5$ for Gaussian Wrapping and $\mathcal{G}_4$ for MILo, our composites have comparable Gaussian rendering PSNR yet produce smaller meshes with lower CD, while mesh rendering PSNR is preserved or slightly higher. These results support the distinction between Gaussian rendering quality and mesh efficiency, indicating that Gaussian rendering quality does not determine the accuracy, compactness, or rendering quality of the extracted meshes: rendering-optimal Gaussians need not be mesh-optimal.

**Depth-guided Gaussian selection.** We examine whether relative-depth-gain selection improves mesh efficiency compared with RGB-based selection. For RGB-based selection, we use the same MILo candidate representations and composition procedure, replacing reference-depth error with the RGB error between rendered and ground-truth images. Fig. 5(b) shows that relative-depth-gain selection achieves a better CD–mesh-size trade-off than RGB-based selection.

**Effects of Gaussian replacement, coverage addition, and optimization.** With both coverage addition and optimization disabled, replacement alone already improves on direct extraction from $\mathcal{G}_4$ at $\gamma = 0.2$, reducing mesh size from 18.29 to 17.96 MB and CD from 0.138 to 0.137 (see MILo $\mathcal{G}_4$ in Table 3; first row in Fig. 5(a)). This provides evidence that replacement itself contributes to the improvement, even before Gaussian optimization. Coverage addition improves CD at $\gamma = 0.6$, where replacement is more restrictive, but matters little at $\gamma = 0.2$; optimization improves CD at both thresholds, and combining the two yields the lowest CD. Further analysis of $\gamma$ and its effect on Gaussian selection and mesh complexity is provided in Appendix B.

(a) Component ablation

| Cover. | GS optim. | $\gamma = 0.6$ | | $\gamma = 0.2$ | |
|---|---|---|---|---|---|
| | | MB ↓ | CD ↓ | MB ↓ | CD ↓ |
| × | × | 9.01 | 0.285 | 17.96 | 0.137 |
| ✓ | × | 9.43 | 0.203 | 17.97 | 0.137 |
| × | ✓ | 8.78 | 0.160 | 17.78 | **0.134** |
| ✓ | ✓ | 9.16 | 0.158 | 17.78 | **0.134** |

(b) Selection criterion

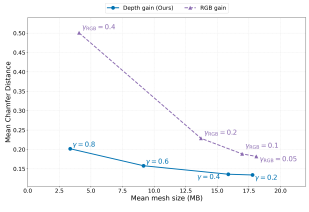


Figure 5: **Ablation studies on the DeMapGS dataset using MILo.** (a) Effects of coverage-guided Gaussian addition (Cover.) and Gaussian optimization (GS optim.) at two thresholds; all configurations use relative-depth-gain Gaussian replacement. Mesh size is reported in MB. (b) CD and mesh-size trade-off of relative-depth-gain selection (Ours) and RGB-gain selection, obtained by sweeping the respective thresholds $\gamma$ and $\gamma_{\text{RGB}}$; each point is labeled with its threshold.

Table 4: **Novel-view mesh rendering results on the Mip-NeRF 360 dataset.** Ours composes $\mathcal{G}_1$–$\mathcal{G}_4$ at $\gamma = 0.2$. Bold marks the better. $\mathcal{G}_5$ is listed for reference only.

| Baseline | Representation | Mesh (MB) ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
|---|---|---|---|---|---|
| Gaussian Wrapping | $\mathcal{G}_5$ | 228.54 | 23.671 | 0.7061 | 0.2982 |
| | $\mathcal{G}_4$ | 156.48 | 23.413 | **0.6999** | **0.3045** |
| | Ours ($\mathcal{G}_1$–$\mathcal{G}_4$) | **144.38** | **23.455** | 0.6994 | 0.3062 |
| MILo | $\mathcal{G}_5$ | 261.53 | 23.357 | 0.6765 | 0.3235 |
| | $\mathcal{G}_4$ | 208.37 | 22.867 | 0.6638 | 0.3354 |
| | Ours ($\mathcal{G}_1$–$\mathcal{G}_4$) | **180.99** | **23.510** | **0.6693** | **0.3286** |
| GGGS | $\mathcal{G}_5$ | 383.73 | 23.282 | 0.6806 | 0.3164 |
| | $\mathcal{G}_4$ | 293.94 | 22.903 | 0.6725 | 0.3242 |
| | Ours ($\mathcal{G}_1$–$\mathcal{G}_4$) | **278.77** | **23.106** | **0.6849** | **0.3150** |

### 4.5 Extension to Unbounded Scenes

We evaluate the trade-off between mesh rendering quality and mesh size on the Mip-NeRF 360 dataset. Without ground-truth foreground masks, we omit coverage-guided Gaussian addition while retaining relative-depth-gain replacement and Gaussian optimization. For Gaussian Wrapping and GGGS, mesh extraction from the full $\mathcal{G}_5$ representations causes out-of-memory errors. We therefore subsample $\mathcal{G}_5$ and use the resulting representations for both baseline mesh extraction and reference-depth supervision in our method.

As shown in Table 4, compared with direct extraction from $\mathcal{G}_4$, our method produces smaller meshes across all three reconstruction methods, while improving PSNR, SSIM, and LPIPS for MILo and GGGS and maintaining comparable rendering quality for Gaussian Wrapping. Compared with the $\mathcal{G}_5$ results, our method also achieves comparable PSNR with smaller meshes. Additional geometric evaluation on the Tanks and Temples dataset is provided in Appendix E.

## 5 Conclusion

We presented a geometry-aware Gaussian composition method for compact surface extraction. Our method spatially combines Gaussian representations across budgets, retaining lower-budget Gaussians where finer representations offer limited relative improvement and selectively incorporating finer Gaussians based on relative reference-depth gain. Across multiple reconstruction methods, the resulting composites improve geometry–mesh-size trade-offs while preserving mesh rendering quality. These results show that Gaussian representations optimized for rendering do not necessarily allocate spatial complexity efficiently for mesh extraction, motivating geometry-aware Gaussian representation design for surface reconstruction.

## Reproducibility Statement

Appendix A.1 lists the budget settings for each reconstruction method, Appendix A.2 gives all composition hyperparameters ($\epsilon_d$, $\epsilon_w$, $\gamma$, $\tau_l$), and Appendix A.3 describes the optimization losses and schedule. Scene lists and evaluation protocols are given in Appendix C. We will release our code upon acceptance.

## AI Use Disclosure

Generative AI tools were used to assist with literature search, provide feedback on manuscript organization and clarity, draft and polish portions of the manuscript, and create elements of the method overview figure. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final manuscript.

## References

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5460–5469, 2022.

Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *ACM SIGGRAPH*, pp. 115–122, 1998.

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *ACM SIGGRAPH*, pp. 303–312, 1996.

Michael Deering. Geometry compression. In *ACM SIGGRAPH*, pp. 13–20, 1995.

Xiaobin Deng, Qiuli Yu, Changyu Diao, Min Li, and Duanqing Xu. Gradient-driven natural selection for compact 3D gaussian splatting. *arXiv preprint arXiv:2511.16980*, 2025.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D gaussian compression with 15x reduction and 200+ FPS. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 140138–140158, 2024.

Guangchi Fang and Bing Wang. Mini-Splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision (ECCV)*, pp. 165–181, 2024.

Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *ACM SIGGRAPH*, pp. 209–216, 1997.

Diego Gomez, Antoine Guédon, Nissim Maruani, Bingchen Gong, and Maks Ovsjanikov. From blobs to spokes: High-fidelity surface reconstruction via oriented gaussians. In *European Conference on Computer Vision (ECCV)*, pp. 54–72, 2026.

Antoine Guédon, Diego Gomez, Nissim Maruani, Bingchen Gong, George Drettakis, and Maks Ovsjanikov. MILo: Mesh-in-the-loop gaussian splatting for detailed and efficient surface reconstruction. *ACM Transactions on Graphics (TOG)*, 44(6), 2025.

Antoine Guédon and Vincent Lepetit. SuGaR: Surface-aligned gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5354–5363, 2024.

Hugues Hoppe. Progressive meshes. In *ACM SIGGRAPH*, pp. 99–108, 1996.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH*, 2024.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 406–413, 2014.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing (SGP)*, pp. 61–70, 2006.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4), 2023.

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D gaussian splatting as Markov Chain Monte Carlo. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 80965–80986, 2024.

Jin-Hwa Kim. Polyhedral complex derivation from piecewise trilinear networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 37138–37175, 2024.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (TOG)*, 36(4), 2017.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D gaussian representation for radiance field. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21719–21728, 2024.

Jiabao Lei and Kui Jia. Analytic marching: An analytic meshing solution from deep implicit surface networks. In *International Conference on Machine Learning (ICML)*, volume 119, pp. 5789–5798, 2020.

Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H. Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8456–8465, 2023.

Peter Lindstrom and Greg Turk. Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 5(2):98–115, 1999.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM SIGGRAPH*, pp. 163–169, 1987.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3DGS: High-quality radiance fields with limited resources. In *ACM SIGGRAPH Asia*, 2024.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4455–4465, 2019.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pp. 405–421, 2020.

Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 5569–5579, 2021.

Seonghun Oh, Youngjung Uh, and Jin-Hwa Kim. TetraSDF: Analytic isosurface extraction with multi-resolution tetrahedral grid. In *European Conference on Computer Vision (ECCV)*, pp. 538–556, 2026.

Panagiotis Papantonakis, Georgios Kopanas, Frédo Durand, and George Drettakis. Content-aware texturing for gaussian splatting. In *Eurographics Symposium on Rendering (EGSR)*, 2025.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 165–174, 2019.

Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-GS: Towards consistent real-time rendering with LOD-structured 3D gaussians. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pp. 1–15, 2025.

Victor Rong, Jingxiang Chen, Sherwin Bahmani, Kiriakos N. Kutulakos, and David B. Lindell. GStex: Per-primitive texturing of 2D gaussian splatting for decoupled appearance and geometry modeling. In *Winter Conference on Applications of Computer Vision (WACV)*, pp. 3508–3518, 2025.

Jarek Rossignac. Edgebreaker: connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 5(1):47–61, 1999.

Yunji Seo, Young Sun Choi, HyunSeung Son, and Youngjung Uh. FLoD: Integrating flexible level of detail into 3D gaussian splatting for customizable rendering. *ACM Transactions on Graphics (TOG)*, 44(4), 2025.

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: A hybrid representation for high-resolution 3D shape synthesis. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 6087–6101, 2021.

Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Transactions on Graphics (TOG)*, 42(4), 2023.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 27171–27183, 2021.

Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3D gaussian splatting for anti-aliased rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20923–20931, 2024.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 2492–2502, 2020.

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 4805–4815, 2021.

Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 43(6), 2024.

Baowen Zhang, Chenxing Jiang, Heng Li, Shaojie Shen, and Ping Tan. Geometry-grounded gaussian splatting. *arXiv preprint arXiv:2601.17835*, 2026.

Xin Zhang, Anpei Chen, Jincheng Xiong, Pinxuan Dai, Yujun Shen, and Weiwei Xu. Neural shell texture splatting: More details and fewer primitives. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 25229–25238, 2025.

Shuyi Zhou, Shengze Zhong, Kenshi Takayama, Takafumi Taketomi, and Takeshi Oishi. DeMapGS: Simultaneous mesh deformation and surface attribute mapping via gaussian splatting. In *ACM SIGGRAPH Asia*, 2025.

# A Implementation and Evaluation Details

## A.1 Multi-Budget Representation Settings

We construct multi-budget representations using different procedures depending on the reconstruction method. MILo (Guédon et al., 2025) supports budget control through its surface-importance-based Gaussian sampling stage. The sampling parameter $p_s$ controls the sampling budget after importance and visibility filtering, rather than directly specifying the fraction of the original Gaussians retained. We perform five separate training runs with $p_s \in \{0.05, 0.1, 0.2, 0.4, 0.6\}$ and use the checkpoints saved at 18,000 iterations. The finest level uses MILo's default $p_s = 0.6$ setting. For all MILo comparisons, we use the representations before mesh–Gaussian joint optimization. In our DeMapGS experiments, the pre-joint representations achieved better geometric accuracy, with a mean CD of 0.1312 compared with 0.9455 after joint optimization, while reducing training time from approximately 3 hours to 30 minutes per scene.

GGGS (Zhang et al., 2026) and Gaussian Wrapping (Gomez et al., 2026) do not provide a budget-control stage in their original training pipelines. For each method, we train a single representation and apply MILo's sampling procedure at different budgets. Each sampled representation is then optimized for 10,000 iterations with the Gaussian count fixed. On the DeMapGS and NeRF Synthetic datasets, both methods use $p_s \in \{0.08, 0.15, 0.35, 0.7\}$ to construct the four candidate representations, while retaining the original trained representation as the finest reference.

On the Mip-NeRF 360 dataset, mesh extraction from the full GGGS and Gaussian Wrapping representations causes out-of-memory errors. We therefore construct all five representations using $p_s \in \{0.02, 0.04, 0.07, 0.15, 0.2\}$ for GGGS and $p_s \in \{0.05, 0.1, 0.2, 0.4, 0.6\}$ for Gaussian Wrapping. For each method, the representation with the largest $p_s$ is used for both baseline mesh extraction and reference-depth supervision.

## A.2 Gaussian Composition

We compute each Gaussian's score by summing the numerator and denominator terms over all training views and pixels before normalization. We use separate numerical stabilizers $\epsilon_d$ and $\epsilon_w$ for the relative-depth-gain score and Gaussian contribution score denominators, respectively, and set both to $10^{-8}$ in all experiments. For the Tanks and Temples evaluation (Appendix E), we use $\gamma = 0.1$. When coverage-guided Gaussian addition is enabled, we use the level-dependent threshold $\tau_l = 0.5(0.5)^{l-1}$.

When coverage-guided Gaussian addition is disabled, we initialize the composite with the full lowest-budget representation, $\mathcal{C}_1 = \mathcal{G}_1$. For each subsequent level $l \geq 2$, we set $\mathcal{S}_l = \varnothing$ and $\mathcal{B}_l = \mathcal{G}_l$, and update the composite using only relative-depth-gain Gaussian replacement.

The Gaussian score $q_l(g; X)$ aggregates pixel-wise relative gains using rendering contributions as weights and remains dimensionless. Pixels outside the selection mask contribute zero to the numerator, while their rendering contributions remain in the denominator. Thus, the score reflects both the relative gains within selected regions and the fraction of the Gaussian's total rendering contribution to those regions. We use the same threshold $\gamma$ at the pixel and Gaussian levels to control both selection stages with a single parameter, without introducing a separate Gaussian-level threshold. This is a design choice rather than a requirement of the score definition.

## A.3 Gaussian Optimization

We optimize the composite for 2,000 iterations, updating Gaussian positions, scales, rotations, opacities, and spherical-harmonic coefficients. We perform no densification or pruning and keep the reference representation fixed. The optimization objective is

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{normal}}\mathcal{L}_{\text{normal}}, \tag{11}$$

where $\lambda_{\text{depth}} = 1$ and $\lambda_{\text{normal}} = 0.005$.

**RGB reconstruction.** Let $\hat{I}_v$ and $I_v$ denote the rendered and ground-truth images in training view $v$. We combine the mean absolute reconstruction error with an SSIM term:

$$\mathcal{L}_{\text{rgb}} = \mathbb{E}_v \left[ 0.8 \ \text{mean} \left| \hat{I}_v - I_v \right| + 0.2 \big( 1 - \text{SSIM}(\hat{I}_v, I_v) \big) \right]. \tag{12}$$

**Depth alignment.** Let $\hat{d}_v$ and $d_v^{\text{ref}}$ denote the depth maps rendered from the composite and the fixed reference representation, respectively. We align these depth maps using an $L_1$ loss:

$$\mathcal{L}_{\text{depth}} = \mathbb{E}_v \left[ \text{mean} \left| \hat{d}_v - d_v^{\text{ref}} \right| \right]. \tag{13}$$

**Depth-normal consistency.** We encourage consistency between normals rendered from the composite and normals derived from its rendered depth:

$$\mathcal{L}_{\text{normal}} = \mathbb{E}_v \left[ \ell_{\text{normal}} \left( \hat{N}_v, \mathcal{N}_v(\hat{d}_v) \right) \right], \tag{14}$$

where $\hat{N}_v$ denotes rendered normals and $\mathcal{N}_v$ computes normals from rendered depth. The loss $\ell_{\text{normal}}$ follows the normal-consistency formulation of the underlying reconstruction method.

### A.4 Mesh Extraction and Size Measurement

For each Gaussian-based surface reconstruction method, we apply its original mesh extraction procedure to both representations at individual budgets and our composites.

We measure mesh complexity by the extracted mesh file size in megabytes (MB). Unlike vertex or face counts alone, this measure accounts for the storage of both vertex coordinates and mesh connectivity. For consistent measurement, we save all extracted meshes in a common binary PLY format with double-precision vertex coordinates and 32-bit triangle indices. We report file sizes in decimal megabytes, where $1\,\text{MB} = 10^6$ bytes.

### A.5 Geometric Evaluation

For the DeMapGS dataset, we follow the surface sampling, normalization, and distance computation in the official evaluation code released with the DeMapGS paper (Zhou et al., 2025). We uniformly sample 100,000 points over the surface of each extracted mesh and its corresponding ground-truth mesh. Both point sets are scaled by the same factor so that the longest side of the ground-truth bounding box has length 10. We compute the mean squared nearest-neighbor distance in each direction, average the two values, and multiply the result by 100. We report the mean across five sampling seeds.

For the DTU dataset, we evaluate meshes extracted from GGGS representations and our composites using the evaluation pipeline provided in the GGGS implementation. We align each reconstructed mesh to the DTU coordinate system using a similarity transformation estimated from corresponding camera centers, without foreground-mask culling. We then apply the DTU evaluation protocol, including observation-mask and ground-plane filtering. We report the average of the reconstruction-to-reference and reference-to-reconstruction mean nearest-neighbor distances in millimeters.

### A.6 Mesh Rendering Evaluation

For the DeMapGS, NeRF Synthetic, and Mip-NeRF 360 datasets, we use MILo's novel-view mesh rendering evaluation pipeline. We fit a color field to each mesh using the training images for 5,000 iterations, keeping the mesh geometry fixed. We evaluate the resulting mesh renderings on test views using PSNR, SSIM, and LPIPS.

### A.7 Runtime

Given five trained Gaussian representations at different budgets, our method takes 40.5–91.2 seconds per scene on average on the NeRF Synthetic dataset (Table 5). Relative-depth-gain replacement takes only 1.9–11.1 seconds, with most of the runtime spent on Gaussian optimization.

Table 5: **Runtime breakdown on the NeRF Synthetic dataset.** Times are reported in seconds and averaged over eight scenes, excluding pretraining of the input Gaussian representations and mesh extraction.

| Baseline | Relative-depth-gain replacement | Gaussian optimization | Total |
|---|---|---|---|
| Gaussian Wrapping | 7.7 | 83.5 | 91.2 |
| MILo | 1.9 | 38.6 | 40.5 |
| GGGS | 11.1 | 74.0 | 85.2 |

Table 6: **Level-wise Gaussian selection on the NeRF Synthetic dataset across relative-depth-gain thresholds.** For each reconstruction method, Ours composes $\mathcal{G}_1$–$\mathcal{G}_4$. The $\mathcal{G}_1$–$\mathcal{G}_4$ columns report the number of Gaussians from each level retained in the final composite. Percentages in parentheses give the retained count divided by the original count at the same level. Total denotes the Gaussian count used for mesh extraction. Gaussian counts, mesh sizes (MB), and novel-view mesh rendering PSNR (dB) are averaged over the eight scenes.

| $\gamma$ | $\mathcal{G}_1$ | $\mathcal{G}_2$ | $\mathcal{G}_3$ | $\mathcal{G}_4$ | Total | Mesh (MB) ↓ | PSNR ↑ |
|---|---|---|---|---|---|---|---|
| **Gaussian Wrapping** | | | | | | | |
| 0.8 | 11,363 (83.47%) | 156 (0.61%) | 3,012 (5.06%) | 34,152 (28.67%) | 48,683 | 9.65 | 26.95 |
| 0.6 | 2,422 (17.79%) | 115 (0.45%) | 5,682 (9.54%) | 89,344 (75.01%) | 97,562 | 17.90 | 27.25 |
| 0.4 | 56.1 (0.41%) | 44.6 (0.17%) | 2,847 (4.78%) | 113,186 (95.02%) | 116,133 | 20.68 | 27.32 |
| 0.2 | 0.4 (0.00%) | 6.5 (0.03%) | 124 (0.21%) | 118,911 (99.83%) | 119,041 | 21.12 | 27.33 |
| **MILo** | | | | | | | |
| 0.8 | 3,712 (96.72%) | 57.0 (0.76%) | 893 (6.18%) | 3,591 (13.39%) | 8,252 | 5.69 | 26.99 |
| 0.6 | 2,245 (58.49%) | 108 (1.43%) | 2,498 (17.30%) | 10,423 (38.85%) | 15,274 | 10.34 | 27.47 |
| 0.4 | 418 (10.90%) | 366 (4.85%) | 3,182 (22.03%) | 18,004 (67.11%) | 21,970 | 14.47 | 27.65 |
| 0.2 | 2.5 (0.07%) | 32.5 (0.43%) | 523 (3.62%) | 25,637 (95.57%) | 26,195 | 17.86 | 27.78 |
| **GGGS** | | | | | | | |
| 0.8 | 12,907 (94.57%) | 37.1 (0.15%) | 4,535 (7.59%) | 17,601 (14.74%) | 35,080 | 32.31 | 25.11 |
| 0.6 | 9,196 (67.38%) | 582 (2.27%) | 8,030 (13.45%) | 46,937 (39.30%) | 64,745 | 60.16 | 25.70 |
| 0.4 | 1,429 (10.47%) | 328 (1.28%) | 10,739 (17.98%) | 90,857 (76.08%) | 103,352 | 88.95 | 25.96 |
| 0.2 | 15.8 (0.12%) | 60.1 (0.23%) | 1,110 (1.86%) | 117,080 (98.04%) | 118,266 | 100.47 | 25.99 |

## B Analysis of Threshold-Controlled Gaussian Composition

**Effect of the relative-depth-gain threshold.** Table 6 reports Gaussian selection at different thresholds $\gamma$ on the NeRF Synthetic dataset. Across all three reconstruction methods, decreasing $\gamma$ increases the fraction of the original $\mathcal{G}_4$ retained in the final composite and decreases the corresponding fraction for $\mathcal{G}_1$. At intermediate thresholds, the composites contain Gaussians from multiple candidate representations.

As $\gamma$ decreases, the total Gaussian count, extracted mesh size, and mesh rendering PSNR increase across all three methods. For Gaussian Wrapping, reducing $\gamma$ from $0.8$ to $0.2$ increases the mean mesh size from $9.65$ to $21.12$ MB. The corresponding PSNR increases from $26.95$ to $27.33$ dB. MILo and GGGS show similar trends. These results demonstrate that varying $\gamma$ controls the trade-off between mesh size and rendering quality while keeping the candidate representations fixed.

**Spatial distribution of selected Gaussians.** Fig. 6 visualizes composites constructed from GGGS representations for two representative scenes. Gaussians are colored by their original representation levels. At $\gamma = 0.8$, Gaussians from lower-budget representations cover most of the visible surface. At intermediate thresholds, different regions are represented by Gaussians from different levels. As $\gamma$ decreases, Gaussians from higher-budget representations cover larger regions. At $\gamma = 0.2$, most of the visible surface is represented by Gaussians from $\mathcal{G}_4$. The visualization shows how $\gamma$ changes the

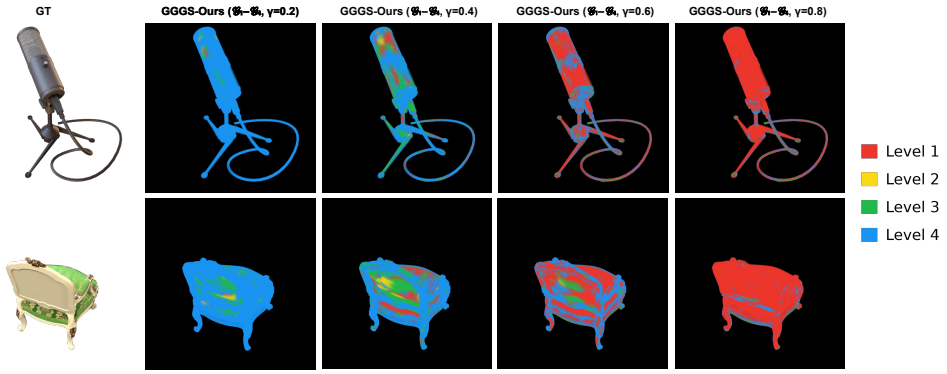


Figure 6: **Spatial Gaussian composition across relative-depth-gain thresholds.** We visualize composites constructed from GGGS representations, with Gaussians colored by their original candidate levels $\mathcal{G}_1$–$\mathcal{G}_4$. A larger $\gamma$ retains lower-budget Gaussians across more regions. As $\gamma$ decreases, Gaussians from higher-budget representations cover larger regions. At intermediate thresholds, different regions are represented by Gaussians from different candidate levels, illustrating the spatial selection in our composition procedure.

Table 7: **Effect of including the *Box* scene in the DeMapGS evaluation.** We report Chamfer distance (CD) averaged over the five scenes excluding *Box*, for *Box* alone, and averaged over all six scenes. Bold indicates the lowest CD in each column within each group: individual representations ($\mathcal{G}_1$–$\mathcal{G}_5$) and our composites evaluated at different $\gamma$ values.

| | MILo | | | Gaussian Wrapping | | |
|---|---|---|---|---|---|---|
| Representation | Five scenes ↓ | Box ↓ | Six scenes ↓ | Five scenes ↓ | Box ↓ | Six scenes ↓ |
| $\mathcal{G}_1$ | 0.27540 | 3.32665 | 0.78394 | 0.99343 | 7.34016 | 2.05122 |
| $\mathcal{G}_2$ | 0.19173 | 2.66626 | 0.60415 | 0.76093 | 6.89595 | 1.78343 |
| $\mathcal{G}_3$ | 0.15932 | **1.93717** | **0.45563** | 0.50745 | 4.31539 | 1.14211 |
| $\mathcal{G}_4$ | 0.13808 | 4.10786 | 0.79971 | 0.15480 | **3.52287** | **0.71614** |
| $\mathcal{G}_5$ | **0.13122** | 4.42256 | 0.84644 | **0.15479** | 3.52931 | 0.71721 |
| Ours ($\gamma = 0.8$) | 0.20168 | **3.46040** | **0.74480** | 0.16368 | **2.97372** | **0.63202** |
| Ours ($\gamma = 0.6$) | 0.15788 | 3.83743 | 0.77114 | 0.15524 | 3.25930 | 0.67258 |
| Ours ($\gamma = 0.4$) | 0.13601 | 4.20786 | 0.81465 | 0.15453 | 3.52171 | 0.71573 |
| Ours ($\gamma = 0.2$) | **0.13408** | 4.34694 | 0.83622 | **0.15417** | 3.53681 | 0.71794 |

spatial distribution of selected Gaussians, complementing the level-wise selection rates and counts reported in Table 6.

The relative-depth-gain threshold $\gamma$ therefore controls the extent of replacement across representation levels and the resulting mesh complexity.

## C Datasets

**DeMapGS.** Our main results are averaged over five object-centric scenes: *Book, Buddha, Bust, Cat, and Tree*. The remaining scene, *Box*, is excluded from the main averages and reported separately. For MILo and Gaussian Wrapping, reconstruction errors on *Box* are substantially larger than those on the other scenes and dominate the six-scene means. Reporting *Box* separately allows us to distinguish its influence from performance trends across the remaining scenes. Table 7 reports the *Box* results alongside the five-scene and six-scene averages, showing how its inclusion affects the aggregate results. We further examine the non-monotonic relationship between Gaussian budget and geometric accuracy observed for MILo on *Box* in Appendix J.

Table 8: **Surface reconstruction results on the DTU dataset.** We report Chamfer distance (CD) for 15 scans. We reproduce the GGGS results and evaluate Ours on GGGS representations at $\gamma = 0.2$. All other results are taken from the Gaussian Wrapping paper. GSWrap denotes Gaussian Wrapping. Red, orange, and yellow indicate the best, second-best, and third-best results in each column, respectively. Equal values share the same rank and color. The reported GGGS results are included for reference and excluded from the ranking.

| | Method | 24 | 37 | 40 | 55 | 63 | 65 | 69 | 83 | 97 | 105 | 106 | 110 | 114 | 118 | 122 | Mean ↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Implicit | NeRF | 1.90 | 1.60 | 1.85 | 0.58 | 2.28 | 1.27 | 1.47 | 1.67 | 2.05 | 1.07 | 0.88 | 2.53 | 1.06 | 1.15 | 0.96 | 1.49 |
| | 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 |
| | NeuS | 1.00 | 1.37 | 0.93 | 0.43 | 1.10 | 0.65 | 0.57 | 1.48 | 1.09 | 0.83 | 0.52 | 1.20 | 0.35 | 0.49 | 0.54 | 0.84 |
| | 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 |
| Explicit | 3DGS | 2.14 | 1.53 | 2.08 | 1.68 | 3.49 | 2.21 | 1.43 | 2.07 | 2.22 | 1.75 | 1.79 | 2.55 | 1.53 | 1.52 | 1.50 | 1.96 |
| | 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 |
| | 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 |
| | 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 |
| | RaDe-GS | 0.46 | 0.73 | 0.33 | 0.38 | 0.79 | 0.75 | 0.76 | 1.19 | 1.22 | 0.62 | 0.70 | 0.78 | 0.36 | 0.68 | 0.47 | 0.68 |
| | MILo | 0.43 | 0.74 | 0.34 | 0.37 | 0.80 | 0.74 | 0.70 | 1.21 | 1.22 | 0.66 | 0.62 | 0.80 | 0.37 | 0.76 | 0.48 | 0.68 |
| | PGSR | 0.34 | 0.54 | 0.44 | 0.37 | 0.78 | 0.57 | 0.49 | 1.06 | 0.63 | 0.59 | 0.47 | 0.50 | 0.30 | 0.37 | 0.34 | 0.52 |
| | GSWrap (30k) | 0.40 | 0.51 | 0.27 | 0.32 | 0.81 | 0.46 | 0.45 | 1.04 | 0.65 | 0.52 | 0.33 | 0.59 | 0.32 | 0.33 | 0.34 | 0.49 |
| | GGGS (reported, 30k) | 0.37 | 0.51 | 0.27 | 0.31 | 0.81 | 0.43 | 0.42 | 1.05 | 0.64 | 0.52 | 0.32 | 0.58 | 0.30 | 0.31 | 0.33 | 0.48 |
| | GGGS (reproduced) | 0.38 | 0.50 | 0.27 | 0.31 | 0.81 | 0.43 | 0.42 | 1.03 | 0.64 | 0.51 | 0.32 | 0.58 | 0.30 | 0.31 | 0.33 | 0.48 |
| | Ours (GGGS, $\gamma = 0.2$) | 0.38 | 0.50 | 0.27 | 0.32 | 0.81 | 0.44 | 0.42 | 1.03 | 0.64 | 0.51 | 0.33 | 0.58 | 0.31 | 0.32 | 0.33 | 0.48 |

Table 9: **Gaussian representation efficiency on DTU under TSDF-based mesh extraction.** We compare GGGS representations and our composites in terms of Gaussian count and Chamfer distance (CD). Results are averaged over 15 scans. Gaussian counts are reported in thousands (K).

| Method | Setting | #Gaussians (K) ↓ | CD ↓ |
|---|---|---|---|
| GGGS | $\mathcal{G}_1$ | 16.63 | 0.507 |
| | $\mathcal{G}_2$ | 31.17 | 0.489 |
| | $\mathcal{G}_3$ | 72.69 | 0.485 |
| | $\mathcal{G}_4$ | 145.05 | 0.481 |
| | $\mathcal{G}_5$ | 606.28 | 0.477 |
| Ours ($\mathcal{G}_1$–$\mathcal{G}_4$) | $\gamma = 0.8$ | 34.61 | 0.493 |
| | $\gamma = 0.6$ | 66.73 | 0.484 |
| | $\gamma = 0.4$ | 97.95 | 0.480 |
| | $\gamma = 0.2$ | 132.60 | 0.479 |

**NeRF Synthetic.** We use all eight scenes: *Chair, Drums, Ficus, Hotdog, Lego, Materials, Mic, and Ship*. We use images at their original resolution and follow the provided training and test splits.

**Mip-NeRF 360.** We use all nine scenes: five outdoor scenes (*Bicycle, Flowers, Garden, Stump, and Treehill*) and four indoor scenes (*Bonsai, Counter, Kitchen, and Room*). Images are downsampled by factors of 4 and 2 for outdoor and indoor scenes, respectively.

**Tanks and Temples.** We use six scenes: *Barn, Caterpillar, Courthouse, Ignatius, Meetingroom, and Truck*.

**DTU.** We use 15 scans: 24, 37, 40, 55, 63, 65, 69, 83, 97, 105, 106, 110, 114, 118, and 122.

## D Evaluation on DTU

The Gaussian-based reconstruction methods considered in this work use TSDF fusion of rendered depth maps for mesh extraction on DTU. With fixed TSDF parameters, changes in Gaussian count



Figure 7: **Qualitative mesh reconstruction results on DTU.** Meshes are extracted using TSDF fusion from our composites constructed from GGGS representations at $\gamma = 0.2$.

Table 10: **Published F1 scores on the Tanks and Temples dataset.** All results are taken from the Gaussian Wrapping paper (Gomez et al., 2026) and included for reference. GSWrap denotes Gaussian Wrapping.

| | Implicit | | | Explicit | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Scene | NeuS | Geo-NeuS | Neuralangelo | SuGaR | 2DGS | GOF | RaDe-GS | QGS | MILo | GSWrap |
| Barn | 0.29 | 0.33 | 0.70 | 0.14 | 0.36 | 0.51 | 0.43 | 0.43 | 0.59 | 0.58 |
| Caterpillar | 0.29 | 0.26 | 0.36 | 0.16 | 0.23 | 0.41 | 0.32 | 0.31 | 0.39 | 0.48 |
| Courthouse | 0.17 | 0.12 | 0.28 | 0.08 | 0.13 | 0.28 | 0.21 | 0.26 | 0.29 | 0.30 |
| Ignatius | 0.83 | 0.72 | 0.89 | 0.33 | 0.44 | 0.68 | 0.69 | 0.79 | 0.78 | 0.70 |
| Meetingroom | 0.24 | 0.20 | 0.32 | 0.15 | 0.16 | 0.28 | 0.25 | 0.25 | 0.26 | 0.33 |
| Truck | 0.45 | 0.45 | 0.48 | 0.26 | 0.26 | 0.59 | 0.51 | 0.60 | 0.62 | 0.58 |
| Mean ↑ | 0.38 | 0.35 | 0.50 | 0.19 | 0.30 | 0.46 | 0.40 | 0.44 | 0.49 | 0.50 |

do not directly translate into changes in mesh complexity, and the resulting mesh sizes are similar across budgets in our experiments. We therefore present DTU as a supplementary evaluation of Gaussian representation efficiency, rather than as a main benchmark for the geometry–mesh-size trade-off.

We evaluate our method on GGGS representations, as GGGS is the only method for which we successfully reproduced the reported DTU performance. Table 8 reports per-scene Chamfer distances for the reproduced baseline and Ours, alongside published results of other methods for reference. Ours maintains comparable reconstruction accuracy while using substantially fewer Gaussians, as quantified in Table 9.

Specifically, at $\gamma = 0.2$, Ours reduces the average Gaussian count from 606.28K to 132.60K relative to $\mathcal{G}_5$, while the mean Chamfer distance increases only from 0.477 to 0.479. Relative to $\mathcal{G}_4$, Ours achieves a lower Chamfer distance with fewer Gaussians. These results show that our method maintains comparable reconstruction accuracy with substantially fewer Gaussians and is also applicable to TSDF-based mesh extraction. Fig. 7 provides qualitative examples of meshes extracted from our GGGS composites on DTU.

## E Evaluation on Tanks and Temples

We evaluate our method on Gaussian Wrapping and GGGS representations across six scenes from Tanks and Temples. Table 10 reports published F1 scores for reference. Table 11 compares meshes extracted from our composites with those extracted from the reproduced $\mathcal{G}_4$ and $\mathcal{G}_5$ representations. For both reconstruction methods, Ours composes $\mathcal{G}_1$–$\mathcal{G}_4$ at $\gamma = 0.1$, using $\mathcal{G}_5$ as the reference.

We use the same input data for both reconstruction methods and our compositions. For GGGS, this input preparation differs from the PGSR-based preprocessing recommended by its official implementation. We omit coverage-guided Gaussian addition because foreground masks are unavailable, while retaining relative-depth-gain Gaussian replacement and subsequent Gaussian optimization.

Compared with $\mathcal{G}_4$, our composites produce smaller meshes with similar geometric accuracy for both reconstruction methods. For Gaussian Wrapping, the mean mesh size decreases from 104.3 to

Table 11: **Reconstruction accuracy and mesh size on the Tanks and Temples dataset.** We compare meshes extracted from the reproduced $\mathcal{G}_4$ and $\mathcal{G}_5$ representations with those extracted from our composites. Both reconstruction methods use the same input data. For GGGS, our input preparation differs from the PGSR-based preprocessing recommended by its official implementation. For both methods, Ours composes $\mathcal{G}_1$–$\mathcal{G}_4$ at $\gamma = 0.1$, using $\mathcal{G}_5$ as the reference. We report per-scene F1 scores, along with mean F1 scores and mesh sizes over the six scenes.

| | Gaussian Wrapping | | | GGGS | | |
|---|---|---|---|---|---|---|
| Scene | $\mathcal{G}_5$ | $\mathcal{G}_4$ | Ours | $\mathcal{G}_5$ | $\mathcal{G}_4$ | Ours |
| Barn | 0.580 | 0.507 | 0.507 | 0.671 | 0.652 | 0.643 |
| Caterpillar | 0.479 | 0.444 | 0.444 | 0.530 | 0.532 | 0.533 |
| Courthouse | 0.305 | 0.273 | 0.273 | 0.359 | 0.363 | 0.363 |
| Ignatius | 0.700 | 0.676 | 0.676 | 0.769 | 0.756 | 0.756 |
| Meetingroom | 0.337 | 0.295 | 0.292 | 0.410 | 0.404 | 0.402 |
| Truck | 0.585 | 0.538 | 0.537 | 0.682 | 0.687 | 0.688 |
| Mean F1 ↑ | 0.498 | 0.456 | 0.455 | 0.570 | 0.566 | 0.564 |
| Mean mesh size (MB) ↓ | 214.6 | 104.3 | 78.4 | 877.2 | 429.4 | 331.9 |

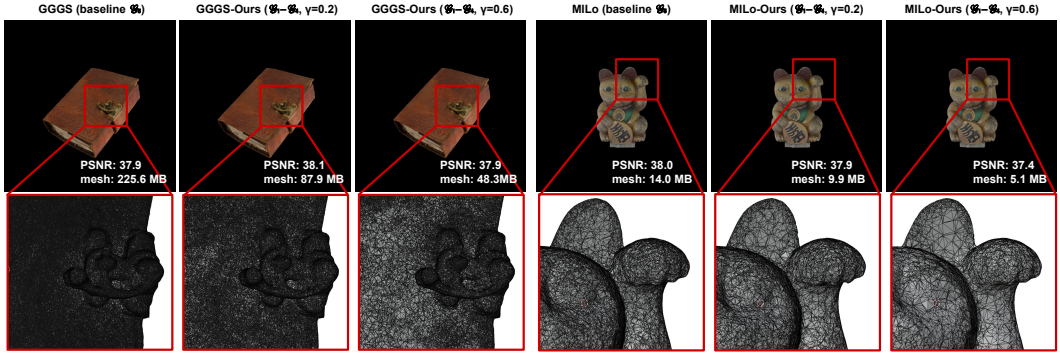


Figure 8: **Qualitative comparisons on the DeMapGS dataset.** We compare meshes extracted from $\mathcal{G}_5$ with those extracted from our compositions of $\mathcal{G}_1$–$\mathcal{G}_4$ at $\gamma = 0.2$ and 0.6. Results are shown for GGGS on Book (left) and MILo on Cat (right). The top row shows mesh renderings, and the bottom row shows enlarged wireframe views of the marked regions, highlighting coarser triangulation in smooth regions and relatively denser triangulation around detailed structures.

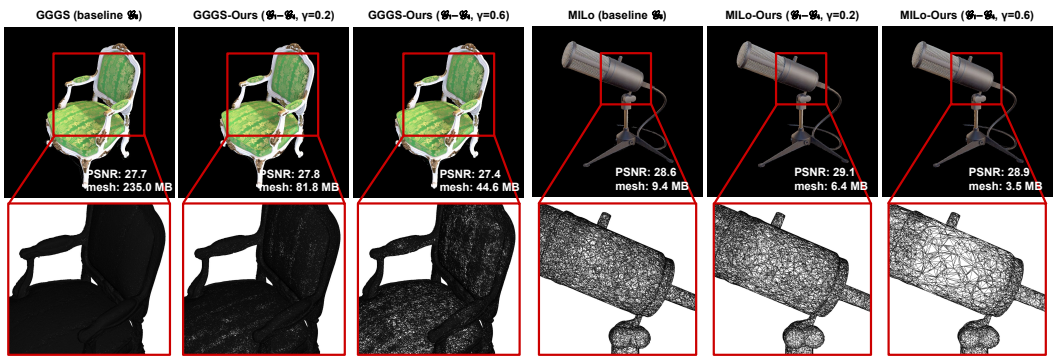


Figure 9: **Qualitative comparisons on the NeRF Synthetic dataset.** We compare meshes extracted from $\mathcal{G}_5$ with those extracted from our compositions of $\mathcal{G}_1$–$\mathcal{G}_4$ at $\gamma = 0.2$ and 0.6. Results are shown for GGGS on Chair (left) and MILo on Mic (right).

78.4 MB, a reduction of approximately 25%. GGGS shows a similar reduction of approximately 23%, from 429.4 to 331.9 MB. These reductions are accompanied by only small changes in mean

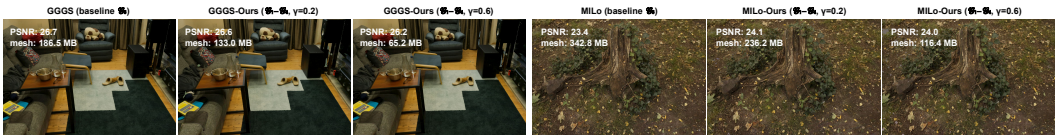


Figure 10: **Qualitative comparisons on the Mip-NeRF 360 dataset.** We compare mesh novel-view renderings from $\mathcal{G}_5$ and our compositions of $\mathcal{G}_1$–$\mathcal{G}_4$ at $\gamma = 0.2$ and $0.6$. Results are shown for GGGS on Room (left) and MILo on Stump (right).

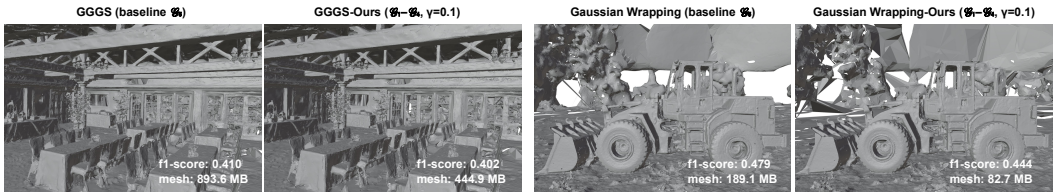


Figure 11: **Qualitative comparisons on the Tanks and Temples dataset.** We compare meshes extracted from $\mathcal{G}_5$ and our compositions of $\mathcal{G}_1$–$\mathcal{G}_4$ at $\gamma = 0.1$. Results are shown for GGGS on Meetingroom (left) and Gaussian Wrapping on Caterpillar (right).

F1 score: from 0.456 to 0.455 for Gaussian Wrapping and from 0.566 to 0.564 for GGGS. Although $\mathcal{G}_5$ achieves higher mean F1 scores, it yields substantially larger meshes. These results demonstrate that Ours offers a trade-off between mesh size and geometric accuracy.

## F Qualitative Results

**DeMapGS dataset.** Fig. 8 compares mesh renderings and local triangulation patterns on the DeMapGS dataset. Our composites produce smaller meshes while preserving the overall surface structure and visual details. The enlarged wireframe views show reduced triangle density in smooth, simple regions while retaining relatively dense triangulation around detailed structures, illustrating a more efficient spatial allocation of mesh complexity.

**NeRF Synthetic dataset.** Fig. 9 presents additional comparisons on the NeRF Synthetic dataset. Our composites preserve visual details with smaller mesh sizes and comparable mesh rendering PSNR. The wireframe views show how this is achieved: coarser triangulation in smooth regions and relatively denser triangulation around detailed structures.

**Mip-NeRF 360 dataset.** Fig. 10 shows mesh renderings from GGGS and MILo. In these examples, our composites substantially reduce mesh size while maintaining comparable rendering quality.

**Tanks and Temples dataset.** Fig. 11 shows extracted meshes from GGGS and Gaussian Wrapping. We visualize the mesh surfaces without color to compare the reconstructed scene structures. In these examples, our composites substantially reduce mesh size while preserving the main structures, with modest reductions in F1 score.

## G Mesh Size and Rendering Quality Trade-offs

We further show the trade-off between mesh size and novel-view rendering quality across datasets and reconstruction methods. Figs. 12 and 13 compare meshes extracted from individual Gaussian representations with those extracted from our composites. We report PSNR, SSIM, and LPIPS against mesh size. Our composites generally improve these trade-offs, with gains varying across methods and metrics.

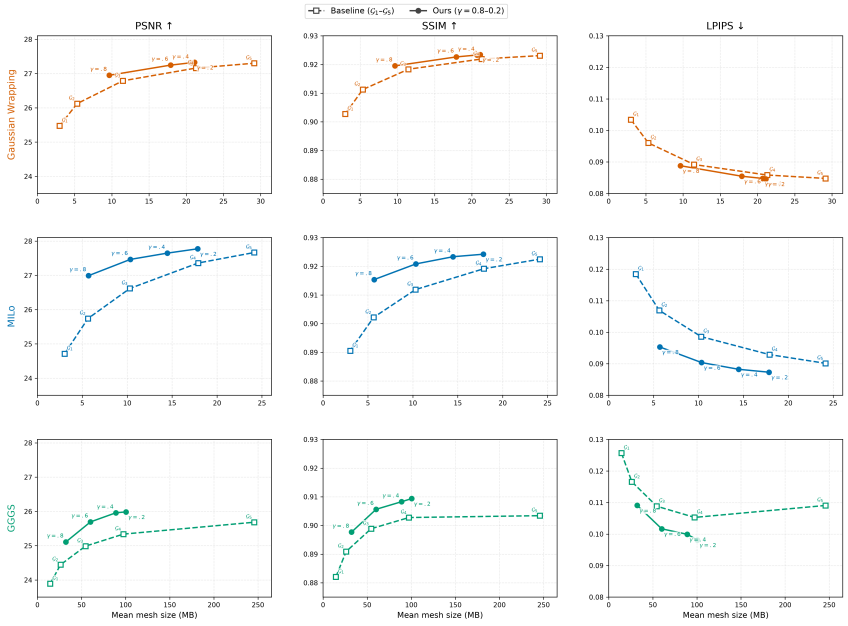


Figure 12: **Trade-offs between mesh size and rendering quality on the NeRF Synthetic dataset.** Dashed lines with square markers show baselines extracted from $\mathcal{G}_1$–$\mathcal{G}_5$. Solid lines with circular markers show Ours, which composes $\mathcal{G}_1$–$\mathcal{G}_4$ at $\gamma \in \{0.2, 0.4, 0.6, 0.8\}$. Mesh sizes and mesh rendering metrics are averaged over all eight scenes.

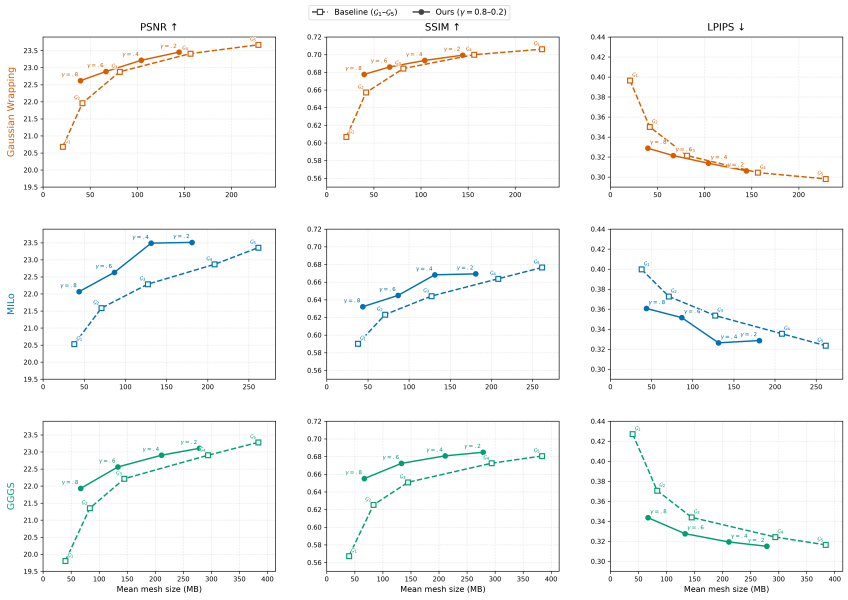


Figure 13: **Trade-offs between mesh size and rendering quality on the Mip-NeRF 360 dataset.** Dashed lines with square markers show baselines extracted from $\mathcal{G}_1$–$\mathcal{G}_5$. Solid lines with circular markers show Ours, which composes $\mathcal{G}_1$–$\mathcal{G}_4$ at $\gamma \in \{0.2, 0.4, 0.6, 0.8\}$. Mesh sizes and mesh rendering metrics are averaged over all nine scenes.

## H Comparison with Mesh Simplification

We compare our pre-extraction Gaussian composition with post-extraction mesh simplification using quadric error metrics (QEM) on the DeMapGS and NeRF Synthetic datasets. For each scene and

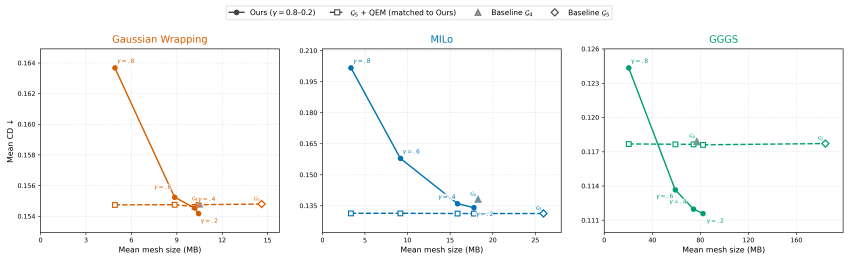


Figure 14: **Comparison with mesh simplification on the DeMapGS dataset.** We compare Ours with QEM simplification of the $\mathcal{G}_5$ mesh, matched to the face count of each Ours result. Direct extraction from $\mathcal{G}_4$ and $\mathcal{G}_5$ is included for reference. Lower Chamfer distance (CD) is better.

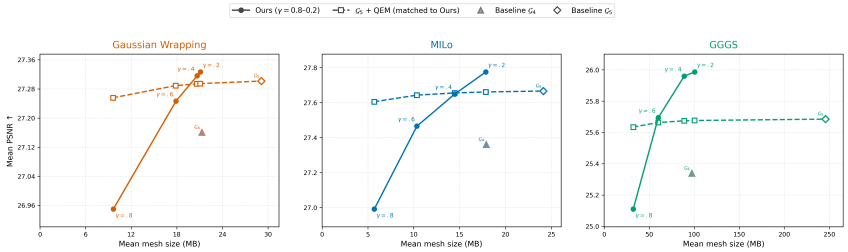


Figure 15: **Comparison with mesh simplification on the NeRF Synthetic dataset.** We compare Ours with QEM simplification of the $\mathcal{G}_5$ mesh, matched to the face count of each Ours result. Direct extraction from $\mathcal{G}_4$ and $\mathcal{G}_5$ is included for reference. Higher mesh rendering PSNR is better.

threshold $\gamma$, we simplify the mesh extracted from $\mathcal{G}_5$ to match the face count of the corresponding Ours mesh, yielding nearly identical mesh sizes. We report Chamfer distance on the DeMapGS dataset and mesh rendering PSNR on the NeRF Synthetic dataset. Meshes directly extracted from $\mathcal{G}_4$ and $\mathcal{G}_5$ are included for reference.

As shown in Fig. 14, the QEM curves on the DeMapGS dataset remain nearly flat over the evaluated mesh sizes indicating that simplification largely preserves the geometric accuracy of the original $\mathcal{G}_5$ meshes. For MILo, QEM achieves lower Chamfer distances than Ours throughout the evaluated range. Nevertheless, at $\gamma = 0.4$ and $0.2$, Ours produces smaller meshes with lower Chamfer distances than direct extraction from $\mathcal{G}_4$. Thus, our composition improves the size–accuracy trade-off over the lower-budget baseline, although simplifying the finest-budget mesh remains more accurate in this case.

For GGGS, Ours achieves lower Chamfer distances than both QEM and the original $\mathcal{G}_5$ mesh at $\gamma = 0.2$ and $0.4$, with this advantage maintained even at the smaller mesh size obtained with $\gamma = 0.6$. At these settings, Ours also produces smaller and more accurate meshes than the $\mathcal{G}_4$ baseline. For Gaussian Wrapping, Ours improves upon both QEM and the original $\mathcal{G}_5$ mesh at $\gamma = 0.2$ and retains this advantage at the smaller mesh size obtained with $\gamma = 0.4$. At larger $\gamma$, Ours produces more compact meshes, while QEM achieves lower Chamfer distances at comparable mesh sizes.

As shown in Fig. 15, QEM largely preserves the rendering quality of the original $\mathcal{G}_5$ meshes on the NeRF Synthetic dataset as mesh size decreases. Ours achieves higher PSNR at comparable mesh sizes across all three methods at $\gamma = 0.2$, with the clearest gains observed for GGGS. This advantage extends to smaller meshes at $\gamma = 0.4$ for Gaussian Wrapping and $\gamma = 0.6$ for GGGS. The advantage of Ours diminishes at the smallest mesh sizes, where QEM achieves higher PSNR.

These results show that pre-extraction composition can improve mesh size–quality trade-offs over direct extraction from $\mathcal{G}_4$ and, in several settings, simplification of $\mathcal{G}_5$ meshes. The relative advantage over QEM varies with the reconstruction method, evaluation metric, and mesh budget.

While the results above compare final mesh quality, Ours and QEM also differ in the stage at which complexity is reduced. Ours reduces complexity at the Gaussian-representation level before extrac-

Table 12: **Effect of Gaussian optimization.** We compare direct extraction from $\mathcal{G}_4$ with our composition at $\gamma = 0.2$, with and without the same 2K Gaussian optimization. Results are averaged over the five DeMapGS scenes.

| Representation | GS optim. | Mesh (MB) ↓ | CD ↓ |
|---|---|---|---|
| $\mathcal{G}_4$ | × | 18.29 | 0.1381 |
| $\mathcal{G}_4$ | ✓ | 18.11 | 0.1342 |
| Ours ($\gamma = 0.2$) | × | 17.96 | 0.1370 |
| Ours ($\gamma = 0.2$) | ✓ | 17.78 | 0.1341 |

tion, whereas QEM operates on an already extracted finest-budget mesh. Consequently, Ours avoids the memory cost associated with extracting the finest-budget mesh prior to simplification.

## I Comparison with Optimized Single-Budget Representations

For MILo baseline, we isolate the contribution of multi-budget composition by applying the same 2K Gaussian optimization to the single-budget $\mathcal{G}_4$ representation. Table 12 shows that composition already improves upon direct extraction from $\mathcal{G}_4$ before optimization, producing a smaller mesh with lower CD. Under identical optimization, Ours retains comparable geometric accuracy with a smaller mesh. These results indicate that composition contributes to the mesh-size–accuracy trade-off before optimization and that this advantage is retained under identical optimization.

## J Limitations

**Dependence on reference quality.** Our method uses depth maps rendered from the highest-budget representation as a geometric reference. However, a larger Gaussian budget does not necessarily yield more accurate geometry. As shown in Table 7, MILo's $\mathcal{G}_5$ produces a mesh with a higher Chamfer distance than the mesh extracted from $\mathcal{G}_3$ on *Box*. This suggests that the highest-budget representation is not always the most reliable geometric reference. When the reference depth is inaccurate, reducing reference-depth error does not necessarily improve geometric accuracy. This can limit the effectiveness of both relative-depth-gain selection and Gaussian optimization.

**Cost of multi-budget inputs.** Our method requires Gaussian representations at multiple budgets, introducing additional input-preparation cost. For GGGS and Gaussian Wrapping, we obtain these representations by subsampling a single trained model and fine-tuning each subset for 10K iterations. For MILo, each budget is trained separately; the five pre-joint runs require approximately 2.5 hours per scene in total, compared with about 3 hours for a single full MILo run. The composition stage itself adds only 41–91 seconds per scene (Table 5). Reconstruction methods that produce multiple budgets within a single training run could substantially reduce this overhead.

## K Future Work

Future work could extend our Gaussian composition framework by incorporating the mesh complexity induced by each replacement into the selection criterion, enabling more direct control over the trade-off between mesh size and geometric accuracy.