SOF: Sorted Opacity Fields for Fast Unbounded Surface Reconstruction

LUKAS RADL, Graz University of Technology, Austria
FELIX WINDISCH, Graz University of Technology, Austria
THOMAS DEIXELBERGER, Huawei Technologies, Austria
JOZEF HLADKY, Huawei Technologies, Germany
MICHAEL STEINER, Graz University of Technology, Austria
DIETER SCHMALSTIEG, Graz University of Technology, Austria and University of Stuttgart, Germany
MARKUS STEINBERGER, Graz University of Technology, Austria and Huawei Technologies, Austria

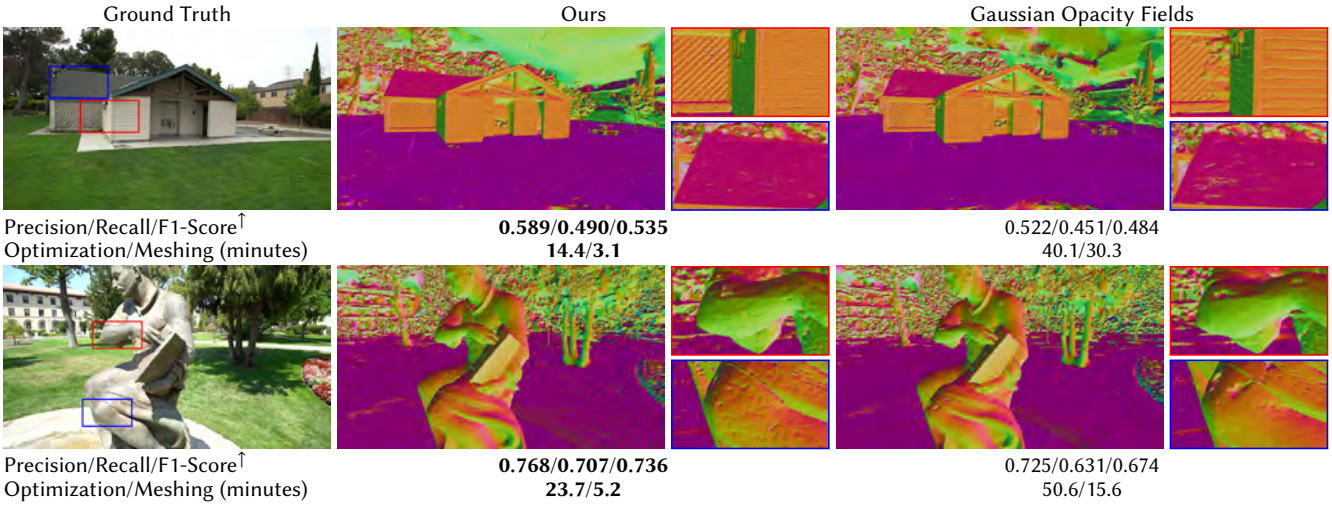

Fig. 1. We introduce Sorted Opacity Fields (SOF), which allow for swift extraction of high-quality unbounded meshes. Compared to current state-of-the-art Gaussian Opacity Fields [Yu et al. 2024b], our meshes are more detailed and have fewer artifacts. In addition, our approach accelerates optimization by over 3× and meshing by up to an order of magnitude.

Recent advances in 3D Gaussian representations have significantly improved the quality and efficiency of image-based scene reconstruction. Their explicit nature facilitates real-time rendering and fast optimization, yet extracting accurate surfaces—particularly in large-scale, unbounded environments—remains a difficult task. Many existing methods rely on approximate depth estimates and global sorting heuristics, which can introduce artifacts and limit the fidelity of the reconstructed mesh. In this paper, we present Sorted Opacity Fields (SOF), a method designed to recover detailed surfaces from 3D Gaussians with both speed and precision. Our approach improves upon prior work by introducing hierarchical resorting and a robust formulation of Gaussian depth, which better aligns with the level-set. To enhance mesh quality, we incorporate a level-set regularizer operating on the opacity field and introduce losses that encourage geometrically-consistent primitive shapes. In addition, we develop a parallelized Marching Tetrahedra algorithm tailored to our opacity formulation, reducing meshing time by up to an order of magnitude. As demonstrated by our quantitative evaluation, SOF achieves higher reconstruction accuracy while cutting total processing time by more than a factor of three. These results mark a step forward in turning efficient Gaussian-based rendering into equally efficient geometry extraction.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; *Graphics systems and interfaces.*

Additional Key Words and Phrases: Novel View Synthesis, Geometry Reconstruction, Rasterization

Authors' addresses: Lukas Radl, Graz University of Technology, Austria, lukas.radl@tugraz.at; Felix Windisch, Graz University of Technology, Austria, felix.windisch@tugraz.at; Thomas Deixelberger, thomas.deixelberger@huawei.com, Huawei Technologies, Austria; Jozef Hladky, jozef.hladky@huawei.com, Huawei Technologies, Germany; Michael Steiner, michael.steiner@tugraz.at, Graz University of Technology, Austria; Dieter Schmalstieg, Graz University of Technology, Austria and University of Stuttgart, Germany, dieter.schmalstieg@visus.uni-stuttgart.de; Markus Steinberger, Graz University of Technology, Austria and Huawei Technologies, Austria, steinberger@tugraz.at.

1 INTRODUCTION

Reconstruction of environments based on captured images alone has received widespread research attention in previous years, benefiting from recent advances in 3D scene representations and potential applications in telepresence, mixed- and virtual reality. These method either use implicit functions, explicit, optimizable primitives, or a combination of the aforementioned to enable learning-based scene optimization, which can subsequently used for novel view synthesis. Particularly noteworthy are Neural Radiance Fields (NeRF) [Mildenhall et al. 2020] and 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023]. While both methods build upon differentiable volume rendering, NeRF leverages large MLPs, evaluated at many sample positions in 3D space. On the other hand, 3DGS optimizes explicit 3D Gaussians,

which drastically reduces computational cost due to the complete removal of computations in empty space.

Since the advent of NeRF, surface reconstruction from implicit 3D scene representations has been investigated in various works, using signed distance fields [Yariv et al. 2021, 2023; Yu et al. 2022] or occupancy networks [Oechsle et al. 2021]. However, with the research community swiftly switching to 3DGS due to its rasterization-based rendering speed, a plethora of works have investigated conversion from trained point clouds to meshes. Most of these works either align 3D Gaussians to surfaces [Guédon and Lepetit 2024; Yu et al. 2024b] or work with Gaussian disks directly [Dai et al. 2024; Huang et al. 2024b]. While the aforementioned works significantly improve processing time whilst mostly retaining the quality, virtually all current surface reconstruction methods still rely on TSDF fusion [Curless and Levoy 1996], and are thus unable to extract detailed meshes for background regions.

Accurately identifying surfaces for large, unbounded environments remains a challenging task due to the much higher computational complexity and lack of supervision. Current methods such as Binary Opacity Grids [Reiser et al. 2024] apply marching cubes [Lorensen and Cline 1987], requiring dense evaluation and costly mesh simplification strategies. Recently, Gaussian Opacity Fields (GOF) [Yu et al. 2024b] has demonstrated fast unbounded surface reconstruction building on 3DGS. However, the resulting quality is still limited by imprecise depth estimates and approximate sorting. Additionally, the required processing time still exceeds an hour for large outdoor environments.

To this end, we present Sorted Opacity Fields (SOF), a method for rapid, high-fidelity unbounded mesh extraction from 3D Gaussians. Building on GOF, we first introduce hierarchical resorting [Radl et al. 2024b] to ensure robust depth estimates. Additionally, we propose a more precise depth estimate, which ensures close alignment with the 0.5 level set. We further enhance mesh quality by (1) encouraging foreground Gaussians towards surface-aligned disks whilst allowing background Gaussians to be more isotropic, and, (2) encouraging depth/level-set alignment. Finally, we propose a new parallelization scheme and several optimization for the Marching Tetrahedra algorithm from Yu et al. [2024b], cutting meshing time by up to an order of magnitude.

Summarized, our work makes the following contributions:

- We provide a comprehensive analysis of current 3DGS-based surface reconstruction methods and their shortcomings.
- We identify sorting as a culprit for incorrect depth computations and derive a more appropriate depth estimate.
- We propose an adaptive extent loss, combined with direct opacity field supervision, leading to high-fidelity unbounded meshes.
- We propose a new parallelization scheme and optimizations for Marching Tetrahedra, thereby improving meshing speed by up to 10×.

Overall, our method improves quality for unbounded mesh extraction whilst reducing overall processing time (optimization & meshing) by over 3× (cf. Fig. 1).

2 RELATED WORK

In this section, we revisit prior art in radiance fields and surface reconstruction.

Radiance Fields. The advent of Neural Radiance Fields (NeRF) [Mildenhall et al. 2020] marked a significant advancement in the field of novel view synthesis; leveraging large multi-layer perceptrons and differentiable volume rendering, novel views could be rendered at previously unattainable quality. Subsequent follow-up works have addressed reconstruction of unbounded scenes [Barron et al. 2022, 2023], faster inference [Chen et al. 2022; Fridovich-Keil et al. 2022; Müller et al. 2022], and rendering artifacts [Barron et al. 2021, 2023], and also applied NeRF in various other domains, such as 3D scene editing [Jambon et al. 2023; Nguyen-Phuoc et al. 2022; Radl et al. 2024a].

However, the field was again revolutionized by 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023], which instead models a scene as a set of emissive 3D Gaussian distributions. This work has resulted in a plethora of follow-up research, focusing on various rendering artifacts [Huang et al. 2024a; Radl et al. 2024b; Tu et al. 2025; Yu et al. 2024a], faster optimization [Mallick et al. 2024], and more lightweight representations [Fan et al. 2024; Papantonakis et al. 2024]. Many works [Kheradmand et al. 2024; Rota Bulò et al. 2024; Yu et al. 2024b] investigated and improved adaptive density control for Gaussians. While most works still adhere to the EWA splatting formulation [Zwicker et al. 2001], recent methods have explored 3D evaluation due to its increased robustness [Hahlbohm et al. 2025; Moenne-Loccoz et al. 2024; Steiner et al. 2025; Yu et al. 2024b].

Surface Reconstruction. Many works have investigated image-based surface reconstruction. Building on NeRF [Mildenhall et al. 2020], approaches have investigated occupancy networks [Oechsle et al. 2021] or signed distance fields (SDF) [Yariv et al. 2021; Yu et al. 2022]. More recent methods have improved the aforementioned in various ways [Li et al. 2023; Yariv et al. 2023]; however, the state-of-the-art Neuralangelo (NA) [Li et al. 2023] requires 100+ GPU hours for optimization. Particularly noteworthy is Binary Opacity Grids [Reiser et al. 2024], which extracts a mesh from a proposal network by forcing opacity along a ray towards a Dirac impulse for high-quality mesh rendering.

Benefiting from the explicit nature of 3D Gaussians, many works have investigated extraction of surfaces from the resulting point clouds. Most works either try to align 3D primitives with the reconstructed surface [Guédon and Lepetit 2024] or use 2D Gaussians directly [Dai et al. 2024; Huang et al. 2024b]. Recent methods have explored the use of multi-view stereo constrains to improve surface reconstruction [Chen et al. 2024; Wang et al. 2024]. All of the aforementioned method leverage TSDF fusion [Curless and Levoy 1996], which only extracts faithful meshes for foreground objects. Extraction of high-fidelity unbounded meshes from 3D Gaussians is a relatively underexplored topic. Gaussian Opacity Fields (GOF) [Yu et al. 2024b] directly extracts the level set leveraging a Delaunay Triangulation of 3D primitives [Kulhanek and Sattler 2023]; as we will demonstrate, the extraction step is costly, and the resulting meshes often lack intricate details. Our Sorted Opacity Fields extract more detailed meshes while drastically improving performance.

3 METHOD

We present Sorted Opacity Fields, a novel method for extracting high-quality, unbounded meshes from 3D Gaussians. Our work extends Gaussian Opacity Fields [Yu et al. 2024b] by incorporating hierarchical resorting [Radl et al. 2024b] and more precise depth estimates (cf. Sec. 3.2). We additionally incorporate novel losses to aid optimization (Sec. 3.3), and finally propose a fast Marching Tetrahedra implementation (Sec. 3.4).

3.1 Preliminaries

In this section, we revisit 3D Gaussian Splatting and its variants.

3D Gaussian Splatting. 3DGS [Kerbl et al. 2023] represents a scene as a collection of N 3D Gaussians, each characterized by a position $\boldsymbol{\mu}_i \in \mathbb{R}^3$, scale $\mathbf{s}_i \in \mathbb{R}^3_+$ and orientation $\mathbf{R}_i \in \mathbb{R}^{3\times3}$. The covariance matrix Σ_i is computed as $\mathbf{R}_i\mathbf{S}_i\mathbf{S}_i^T\mathbf{R}_i^T$, with $\mathbf{S}_i = \mathrm{diag}(\mathbf{s}_i)$. Each primitive is also equipped with a scalar opacity value $o_i \in [0,1]$, and a set of Spherical Harmonics coefficients $\mathbf{f}_i \in \mathbb{R}^{48}$. For rendering, the pixel color is accumulated via alpha blending:

$$\mathbf{C}(\mathbf{p}) = \sum_{i=0}^{N-1} o_i \mathcal{G}_i(\mathbf{p}) T_i, \tag{1}$$

with $\mathcal{G}_i$ the value of the i-th Gaussian at pixel $\mathbf{p}$ and $T_i = \prod_{j=0}^{i-1}(1 - o_j\mathcal{G}_j(\mathbf{p}))$ the transmittance. Further, Gaussians are assigned to *tiles* based on their image plane extent, significantly reducing the number of primitives per-pixel and thereby improving rendering speed.

StopThePop. Recently, Radl et al. [2024b] demonstrated significantly improved view-consistency by hierarchically resorting Gaussians for each pixel, avoiding so-called popping artifacts. Instead of relying on a global sort based on the view-space depth of $\boldsymbol{\mu}_i$, StopThePop performs per-pixel resorting based on each primitives point of maximum contribution along view rays $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, with

$$t_i = \frac{\mathbf{d}^T\Sigma_i^{-1}(\boldsymbol{\mu}_i - \mathbf{o})}{\mathbf{d}^T\Sigma_i^{-1}\mathbf{d}}, \tag{2}$$

thereby avoiding rendering artifacts. As 3DGS does, StopThePop also evaluates projected 2D Gaussians on the image plane; however, StopThePop exhibits smoother depth maps due to its improved view-consistency. To compensate for the computational overhead of resorting, StopThePop aggressively culls Gaussians from tiles; for instance, instead of using a 3σ bounding-box for each Gaussian, a tighter bound $\mathcal{E}_i$ can be computed by considering the opacity:

$$\mathcal{E}_i = \sqrt{2\ln(255\, o_i)}. \tag{3}$$

Gaussian Opacity Fields. Yu et al. [2024b] adhere to the same scene representation as [Kerbl et al. 2023; Radl et al. 2024b], but evaluate the Gaussians 3D, at the point of maximum contribution along each view ray $\mathbf{r}(t)$. Instead of performing evaluation in world space, GOF first maps $\mathbf{o}$ and $\mathbf{d}$ to Gaussian space with

$$\mathbf{o}_g = \mathbf{S}_i^{-1}\mathbf{R}_i(\mathbf{o} - \boldsymbol{\mu}_i), \tag{4}$$

$$\mathbf{d}_g = \mathbf{S}_i^{-1}\mathbf{R}_i\mathbf{d}, \tag{5}$$

$$\mathbf{r}_g(t) = \mathbf{o}_g + t\mathbf{d}_g. \tag{6}$$

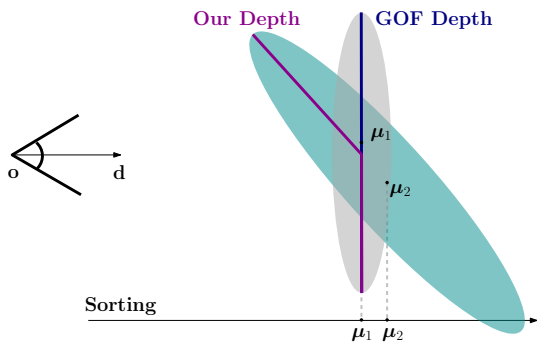

Fig. 2. **Sorting Motivation**: Sorting 3D Gaussians along view rays according to the z-depth in view space (as done by 3DGS) results in inaccurate depth values. We leverage hierarchical resorting [Radl et al. 2024b] for improved depth accuracy.

The contribution along the ray is now a 1D Gaussian, with

$$\mathcal{G}_i^{1\mathrm{D}}(t) = \exp\left(-\frac{1}{2}\mathbf{r}_g(t)^T\mathbf{r}_g(t)\right) \tag{7}$$

$$= \exp\left(-\frac{1}{2}\left(\mathbf{d}_g^T\mathbf{d}_g t^2 + 2\mathbf{d}_g^T\mathbf{o}_g + \mathbf{o}_g^T\mathbf{o}_g\right)\right). \tag{8}$$

Now, defining $A_i = \mathbf{d}_g^T\mathbf{d}_g$, $B_i = 2\mathbf{d}_g^T\mathbf{o}_g$ and $C_i = \mathbf{o}_g^T\mathbf{o}_g$, the maximum of Eqn. (7) is attained at

$$t_i^* = -\frac{B_i}{2A_i}, \tag{9}$$

which is equivalent to Eqn. (2). Combining Eqns. (7) and (9), we arrive at

$$\mathcal{G}_i^{1\mathrm{D}}(t_i^*) = \exp\left(-\frac{1}{2}\left(-\frac{B_i^2}{4A_i} + C_i\right)\right), \tag{10}$$

where t_i^* is the depth of the Gaussian. The color for a pixel $\mathbf{p}$ is now

$$\mathbf{C}(\mathbf{p}) = \sum_{i=0}^{N-1} \mathbf{c}_i\alpha_i T_i, \tag{11}$$

with $\alpha_i = o_i\mathcal{G}_i^{1\mathrm{D}}(t_i^*)$ and $\mathbf{c}_i$ the Gaussians color, derived from $\mathbf{f}_i$.

Equipped with the ability to evaluate 3D Gaussians at arbitrary positions by inserting t in Eqn. (8), GOF defines the opacity of a point $\mathbf{x} \in \mathbb{R}^3$ observed from a camera $\mathbf{o}$ as

$$O_N(\mathbf{x}) = \sum_{i=0}^{N-1} o_i\mathcal{G}_i^{1\mathrm{D}}(t_{\mathrm{eval}}) \prod_{j=1}^{i-1}\left(1 - o_j\mathcal{G}_i^{1\mathrm{D}}(t_{\mathrm{eval}})\right), \tag{12}$$

with $t_{\mathrm{eval}} = \min\left(t_i^*, t\right)$ for $\mathbf{x} = \mathbf{o} + t\mathbf{d}$. However, $\mathbf{x}$ may not be observed at all, *e.g.* due to occlusion or being outside the view frustum. Thus, we define the opacity of a point as the minimum opacity observed over all training views:

$$\mathbf{O}(\mathbf{x}) = \min_{\{\mathbf{o},\mathbf{d}\}} O_N(\mathbf{o} + t\mathbf{d}). \tag{13}$$

See App. A.1 for details on an efficient CUDA implementation.

3.2 Sorted Opacity Fields

While GOF [Yu et al. 2024b] manages to deliver convincing results for mesh extraction in bounded and unbounded scenes, we identify that incorrectly sorted Gaussians result in inconsistent depth estimates, causing a misalignment between depth and the underlying opacity field. Therefore, we incorporate hierarchical resorting [Radl et al. 2024b], which significantly improves depth accuracy (see Fig. 2). By considering the opacity field defined in Eqn. (12), we derive a more precise depth estimate, which provides closer alignment with the 0.5 level set.

Exact Depth. GOF [Yu et al. 2024b] defines the depth of a ray as

$$t_{\mathbf{r}} = \sum_{i=0}^{N-1} \mathcal{I}_{\{T_i>0.5, T_{i+1}<0.5\}}\, t_i^*, \tag{14}$$

with $\mathcal{I}_{\{.\}}$ denoting the indicator function. The opacity at this point is exactly $1 - T_{i+1}$, assuming primitives do not overlap and sorting is correct. While this already exceeds 0.5 in virtually all cases, Gaussians often overlap in regions with high primitives counts, for which $1 - T_{i+1}$ is now a lower bound; once again, assuming that sorting is correct.

To this end, we derive a more precise depth estimate. We want to find the exact depth where the transmittance T_i reaches 0.5; as above, this is only possible if $T_i > 0.5$ and $T_{i+1} < 0.5$. For the Gaussian which satisfies these conditions, we want to find a $t_{\mathbf{r}}^*$ such that

$$T_i \left(1 - o_i \mathcal{G}_i^{1D}\left(t_{\mathbf{r}}^*\right)\right) = 0.5. \tag{15}$$

Inserting Eqn. (7), it can be shown (cf. App. A.3 for the exact derivation) that the exact depth $t_{\mathbf{r}}^*$ is given

$$t_{\mathbf{r}}^* = t_{\mathbf{r}} - \frac{\sqrt{B_i^2 - 4A_i\left(C_i + 2\ln\left(\frac{T_i - 0.5}{\alpha_i}\right)\right)}}{2A}. \tag{16}$$

Under the assumption of non-overlapping and correctly sorted Gaussians, this exactly identifies the 0.5 level set of our opacity field. See Fig. 3 for a visual explanation.

While this assumption is violated even with our hierarchical resorting, we experimentally find that this depth formulation is much more appropriate and leads to improved results.

3.3 Optimization

In this section, we analyze the current losses included in GOF [Yu et al. 2024b] and their issues; we propose a novel extent loss which encourages flat, surface-aligned Gaussians for close objects while allowing for larger, less anistropic Gaussians required for unbounded meshes in the background. We also propose a method for direct opacity field supervision during optimization, and include a normal smoothness loss for better geometry reconstruction.

Distortion Loss and Depth-Normal Consistency Loss. Following both [Huang et al. 2024b; Yu et al. 2024b], we utilize the distortion loss and the depth-normal consistency loss. The distortion loss $\mathcal{L}_{\text{dist}}$ is defined as

$$\mathcal{L}_{\text{dist}} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} w_i w_j \left\|\text{NDC}\left(t_i\right) - \text{NDC}\left(t_j\right)\right\|^2, \tag{17}$$

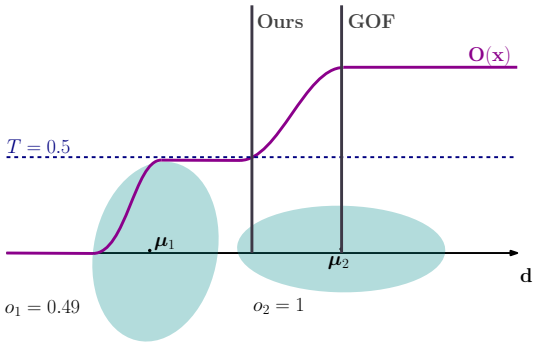

Fig. 3. **Illustration of our exact depth computation.** Whereas GOF consistently overestimates their depth to always be located at the maximum contribution of a Gaussian, we determine the depth by considering the 0.5 level set of our opacity field **O**.

with $w_i = \alpha_i T_i$ and NDC$(\cdot)$ the mapping to NDC space, where the near and far plane set to $\tau_n = 0.2$, $\tau_f = 100$, respectively.

However, in contrast to the standard 3DGS rasterizer [Kerbl et al. 2023], the StopThePop rasterizer performs its backward pass in front-to-back ordering to accomodate sorting. It can be shown (see App. C.1 for the detailed derivation) that the required gradients can be computed in this order as well. Finally, we find that detaching w_i, w_j, as done by GOF, results in worse surface reconstruction results given our improved sorting accuracy.

The depth-normal consistency loss $\mathcal{L}_{\text{normal}}$ is defined as

$$\mathcal{L}_{\text{normal}} = \sum_{i=0}^{N-1} w_i \left(1 - \mathbf{n}_i^T \mathbf{N}\right), \tag{18}$$

with $\mathbf{N}$ being the normal of the pixel, derived using finite differences from the rendered depth, and $\mathbf{n}_i$ the normal of the i-th Gaussian (as defined in Yu et al. [2024b]).

Extent Loss. Although the distortion loss drastically improves reconstruction quality, we observe that the scale of each Gaussian is not taken into consideration; for a Gaussian with $\mathbf{n}_i$ perpendicular to the view direction, the scale itself has no impact on $\mathcal{L}_{\text{dist}}$. Therefore, a distortion loss would be minimized by a setup of two closely aligned Gaussians with vastly different variances along view rays. One solution would be to minimize the scales of Gaussians along a single axis [Jiang et al. 2023] or the direct use of 2D Gaussian disks [Dai et al. 2024; Huang et al. 2024b]; however, this directly results in worse background reconstruction, as the number of primitives allocated in the background is significantly reduced.

To this end, we propose a novel *Extent Loss*, which penalizes the extent of Gaussians in NDC space (see Fig. 4 for an illustration). This encourages flat Gaussians in the foreground to properly represent surfaces, while allowing background Gaussians to be larger, enabling high-fidelity unbounded meshes.

Our loss can be defined as

$$\mathcal{L}_{\text{ext}} = \frac{\tau_n \tau_f}{\tau_f - \tau_n} \sum_{i=0}^{N-1} w_i \frac{2A_i\sqrt{B_i^2 - 4A_i\left(C_i - \mathcal{E}_i^2\right)}}{B_i^2}. \tag{19}$$

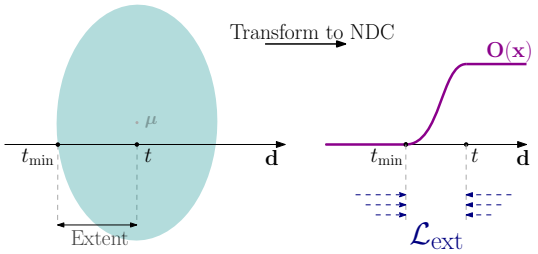

Fig. 4. **Illustration of our extent loss** $\mathcal{L}_{\text{ext}}$. We map both the mean and the minimum contribution t_{min} to NDC-space, where we penalize their extent; this results in a loss primarily focused on foreground Gaussians.

Note that this version corresponds to the linearized extent loss in NDC (cf. App. C.2 for the full derivation).

Direct Opacity Field Supervision. During meshing, the 0.5 level-set of the opacity field is extracted as the resulting surface; however, during optimization, no direct supervision to the opacity field is applied. Our key insight is that we can enforce alignment of the depth $t^*_{\mathbf{r}}$ and the 0.5 level set by minimizing

$$\mathcal{L}_{\text{opa}} = \left\| O_N\left(\mathbf{o} + t^*_{\mathbf{r}}\mathbf{d}\right) - 0.5 \right\|^2. \tag{20}$$

However, the depth is unknown until $T_{i+1} < 0.5$, and, by the definition of our exact depth, t_j may be larger than $t^*_{\mathbf{r}}$, even if $T_j > 0.5$.

Instead of using a straightforward, but inefficient two-pass solution, we compute O_N as

$$O_N = O_k + T_k \sum_{i=k}^{N-1} \alpha_i \prod_{j=k}^{i-1}(1-\alpha_j), \tag{21}$$

where k is the index of the Gaussian for which $T_k > 0.5$, $T_{k+1} < 0.5$ holds. This can be implemented efficiently by accumulating $\sum_{i=k}^{N-1} \alpha_i \prod_{j=k}^{i-1}(1-\alpha_j)$ after the depth is obtained, then computing the remaining O_k, T_k in a second pass, where we can re-use all results for Gaussian preprocessing. Note that the second pass is very efficient, as we do not need to perform alpha blending for pixel colors, and we only need to consider Gaussian up to index k.

Normal Smoothness. Finally, we incorporate a normal smoothness loss following Gao et al. [2024], formally defined as

$$\mathcal{L}_{\text{smooth}} = \|\nabla \mathbf{N}\| \exp\left(-\|\nabla \mathbf{I}\|\right), \tag{22}$$

with ∇ denoting finite-differences and $\mathbf{I}$ the ground-truth image.

Our final loss $\mathcal{L}$ is thus composed as a combinations of losses from GOF ($\mathcal{L}_{\text{GOF}}$) complemented with our novel losses:

$$\mathcal{L}_{\text{GOF}} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}} + \lambda_{\text{normal}}\mathcal{L}_{\text{normal}}, \tag{23}$$

$$\mathcal{L} = \mathcal{L}_{\text{GOF}} + \lambda_{\text{ext}}\mathcal{L}_{\text{ext}} + \lambda_{\text{opa}}\mathcal{L}_{\text{opa}} + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}}. \tag{24}$$

3.4 Fast Marching Tetrahedra

For mesh extraction in unbounded scenes, we follow GOF [Yu et al. 2024b] and use Deep Marching Tetrahedra [Shen et al. 2021] with 8 binary search iterations to identify the 0.5 level set.

The current Marching Tetrahedra algorithm first processes all Gaussians, as done in the standard 3DGS rendering pipeline, then processes all 3D points $\mathbf{x}_i \in \mathbb{R}^3$, assigning them to individual pixels

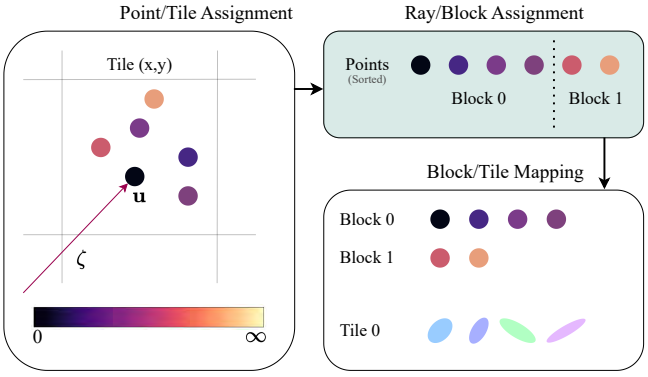

Fig. 5. **Pipeline Overview** for our Fast Marching Tetrahedra Ray/Tile mapping setup, assuming a block size of 4 rays. Instead of mapping points to pixels, we directly map points to tiles, thereby ensuring a balanced load.

based on their projected position $\mathbf{u}_i \in \mathbb{R}^2$. Next, for each pixel, all points assigned to this pixel are evaluated according to Eqn. (12), and the results are stored. This setup results in (1) an uncontrollable number of Gaussians assigned to individual pixels, as well as (2) an unbalanced per-pixel load.

Ray/Tile Scheduling. To alleviate the aforementioned issues, we propose a different parallelization strategy: Instead of assigning points to pixels and finding the contributing Gaussians via the pixel's tile, we opt for directly assigning points to individual tiles.

First, depending on $\mathbf{u}_i$, we assign points to individual tiles; clearly, each point may only belong to a single tile, but each tile may have an arbitrary number of points. Next, we create key combinations of `[tile_id, depth]` for each point, which we sort, ensuring a balanced per-block workload. The number of blocks to launch for a tile is

$$N_{\text{blocks}} = \lceil N_P/256 \rceil, \tag{25}$$

with N_P denoting the number of points per tile. An inclusive sum over all N_{blocks} gives the total number of blocks to launch. Finally, we create a look-up-table, which yields the correct tile id for each block, used for accessing the contributing Gaussians.

The per-thread workload is now evenly distributed across blocks due to evaluating a single point $\mathbf{x}_i$ per thread and previous depth sorting, and the algorithm works efficiently for any set of points. For an overview of our parallelization scheme, cf. Fig. 5.

Tighter Gaussian Bounding. In contrast to the fixed 3σ cutoff, we use an adaptive, opacity-based Gaussian cutoff $\mathcal{E}_i$ (cf. Eqn. (3)). We also observe that Gaussians with $o_i < \frac{1}{255}$ are never rendered, and thus no longer receive gradients; the number of unrendered primitives is $\approx 9\%$ after optimization due to the 3D filter proposed by Yu et al. [2024a]. We do not consider these points for the tetrahedral grid generation, which boosts performance while retaining reconstruction quality; see App. B.2 for more details.

Early Stopping. We also propose additional early stopping strategies to further improve performance. Our primary insight is that the sort order is irrelevant when evaluating Eqn. (12) (see App. A.2 for a proof). Therefore, we perform sorting based on the minimal

Table 1. **Full geometry reconstruction results for the DTU dataset** [Jensen et al. 2014]. Results with † were taken from GOF [Yu et al. 2024b], results with ‡ were reproduced with the current codebase. We report Chamfer Distance (lower scores are better).

Method	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Avg	Time
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
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	>12h
NA†	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	0.60	>12h
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	11.2m
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	≈1h
2DGS‡	0.52	0.80	0.34	0.42	0.97	0.89	0.82	1.24	1.25	0.63	0.65	1.97	0.42	0.69	0.49	0.81	10.9m
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
GOF‡	0.53	0.89	0.43	0.38	1.33	0.87	0.77	1.28	1.29	0.79	0.77	1.15	0.46	0.70	0.54	0.81	≈1h
Ours	0.55	0.70	0.41	0.39	1.12	0.73	0.66	1.11	1.47	0.60	0.63	1.05	0.58	0.62	0.48	0.74	22.8m

z-depth at which the Gaussian contributes, which is given as

$$z_{\min} = (\mathbf{W}\boldsymbol{\mu}_i)_z - \mathcal{E}_i\lambda_3, \tag{26}$$

with λ_3 the eigenvalue of the z-axis of $\boldsymbol{\Sigma}_i$ in view space and $\mathbf{W}$ the camera matrix. We then perform early stopping if $z_{\min} > t$.

Finally, during the binary search, we iteratively refine the mesh to better align with the 0.5 level set. However, for each point, we simply need to know whether $\mathbf{O}(\mathbf{x}_i)$ exceeds 0.5, or not. Thus, we perform early stopping once $O_N(\mathbf{x}_i) > 0.5$, as the exact value is not required. We can also prune points for which we have observed that $O_N(\mathbf{x}_i) < 0.5$, as $\mathbf{O}(\mathbf{x}_i)$ will never exceed 0.5 thereafter.

4 RESULTS AND EVALUATION

In this section, we evaluate our Sorted Opacity Fields for Surface Reconstruction and Novel View Synthesis.

Implementation Details. We base our implementation on StopThePop [Radl et al. 2024b], and integrate 3D evaluation, densification, normal and distortion losses following GOF [Yu et al. 2024b]. We also adopt the appearance model from VastGaussian [Lin et al. 2024] for all geometry reconstruction results. For faster optimization, we use the *fused-ssim* implementation from Mallick et al. [2024], and additionally compute the 3D filter proposed by Yu et al. [2024a] in CUDA. We also use the same densification strategy as GOF; however, note that the choice of densification strategy significantly impacts the resulting mesh/reconstruction quality (cf. App. A.4).

We use $\lambda_{\text{dist}} = 100$ for unbounded scenes and $\lambda_{\text{dist}} = 1000$ for bounded scenes. We set $\lambda_{\text{normal}} = 0.05$ for all scenes, and let $\lambda_{\text{smooth}} = 0.01$, $\lambda_{\text{ext}} = 0.1$ and $\lambda_{\text{opa}} = 0.04$. Note that all losses, except for $\mathcal{L}_{\text{rgb}}$, are inactive for the first 15K iterations.

4.1 Surface Reconstruction

For geometry reconstruction, we used both the DTU dataset [Jensen et al. 2014] and the Tanks & Temples dataset [Knapitsch et al. 2017] to test our method for both bounded and unbounded scenes.

Bounded Meshes. We present our results for the DTU dataset in Table 1. We re-evaluated GOF (cf. Table 1, GOF‡) using TSDF fusion [Curless and Levoy 1996]; we also include the reported numbers. As we can see, our method performs comparably to other explicit based methods. Note that we only used the losses in $\mathcal{L}_{\text{GOF}}$, and disabled all other losses; as demonstrated in App. B.1, our other losses result in slightly decreased surface reconstruction quality.

Table 2. **Full geometry reconstruction results for the Tanks & Temples dataset** [Knapitsch et al. 2017]. We evaluate the F1-score (higher is better) and include results for other methods from GOF [Yu et al. 2024b] (marked with†). Overall, our method achieves the best reconstruction quality of all tested 3DGS-based methods, combined with very fast optimization.

Method	Barn	Caterp	Courth	Ignatius	Meetingr	Truck	Avg	Time
Geo-NeuS†	0.33	0.26	0.12	0.72	0.20	0.45	0.38	>24h
NeuS†	0.29	0.29	0.17	0.83	0.24	0.45	0.38	>24h
NA†	0.70	0.36	0.28	0.89	0.32	0.48	0.50	>24h
3DGS†	0.13	0.08	0.09	0.04	0.01	0.19	0.09	14.3m
SuGaR†	0.14	0.16	0.08	0.33	0.15	0.26	0.19	>1h
2DGS†	0.36	0.23	0.13	0.44	0.16	0.26	0.30	15.5m
GOF†	0.51	0.41	0.28	0.68	0.28	0.59	0.46	24.2m
GOF	0.48	0.40	0.29	0.67	0.27	0.60	0.45	>1h
Ours	0.54	0.41	0.30	0.74	0.31	0.56	0.47	16.7m

Unbounded Meshes. We show the results for unbounded meshes, using our Fast Marching Tetrahedra algorithm, in Table 2. Our method quantitatively outperforms GOF and other explicit, 3DGS-based methods. Particularly for *Barn* and *Ignatius*, our method attains much higher reconstruction quality than GOF. We additionally provide comparisons with GOF for unbounded meshes in Fig. 8. As we can see, our method results in fewer artifacts compared to GOF, while recovering intricate geometric details (cf. Fig. 7 (b)).

4.2 Novel View Synthesis

We also evaluate our method in Novel View Synthesis, where we utilize the Mip-NeRF 360 dataset [Barron et al. 2022]. For our comparison, we choose state-of-the-art reconstruction methods [Kerbl et al. 2023; Kheradmand et al. 2024; Mallick et al. 2024; Radl et al. 2024b; Yu et al. 2024a] as well as recent works in surface reconstruction [Huang et al. 2024b; Yu et al. 2024b]. For Ours and GOF, we disabled decoupled appearance modeling.

We use PSNR, SSIM, LPIPS [Zhang et al. 2018] and ꟻLIP [Andersson et al. 2020] for our evaluation; see the results in Table 3. Our method remains competitive with GOF, with slightly reduced image quality metrics. This aligns with the findings for StopThePop, where the indoor image metrics are worse compared to 3DGS, as the hierarchical rasterizer prevents "cheating" view-dependent effects.

Table 3. **Full Novel View Synthesis results for the Mip-NeRF 360 dataset** [Barron et al. 2022]. Our method performs similarly to GOF, while experiencing slightly lower scores for indoor scenes.

	Mip-NeRF 360 Outdoor				Mip-NeRF 360 Indoor			
Method	PSNR↑	SSIM↑	LPIPS↓	ꟻLIP↓	PSNR↑	SSIM↑	LPIPS↓	ꟻLIP↓
3DGS	24.59	0.727	0.240	0.167	30.98	0.922	0.189	0.094
Mip-Splatting	24.72	0.731	0.240	0.168	31.06	0.925	0.187	0.094
StopThePop	24.60	0.728	0.235	0.167	30.62	0.921	0.186	0.099
Taming-3DGS	25.01	0.740	0.227	0.163	31.34	0.927	0.183	0.090
3DGS-MCMC	25.17	0.759	0.198	0.160	31.59	0.932	0.174	0.091
2DGS	24.22	0.705	0.285	0.174	30.10	0.912	0.212	0.103
GOF	24.79	0.745	0.208	0.165	30.47	0.918	0.189	0.100
Ours	24.78	0.745	0.208	0.165	30.13	0.916	0.188	0.104
Ours (MCMC)	24.70	0.738	0.214	0.168	30.41	0.921	0.184	0.103

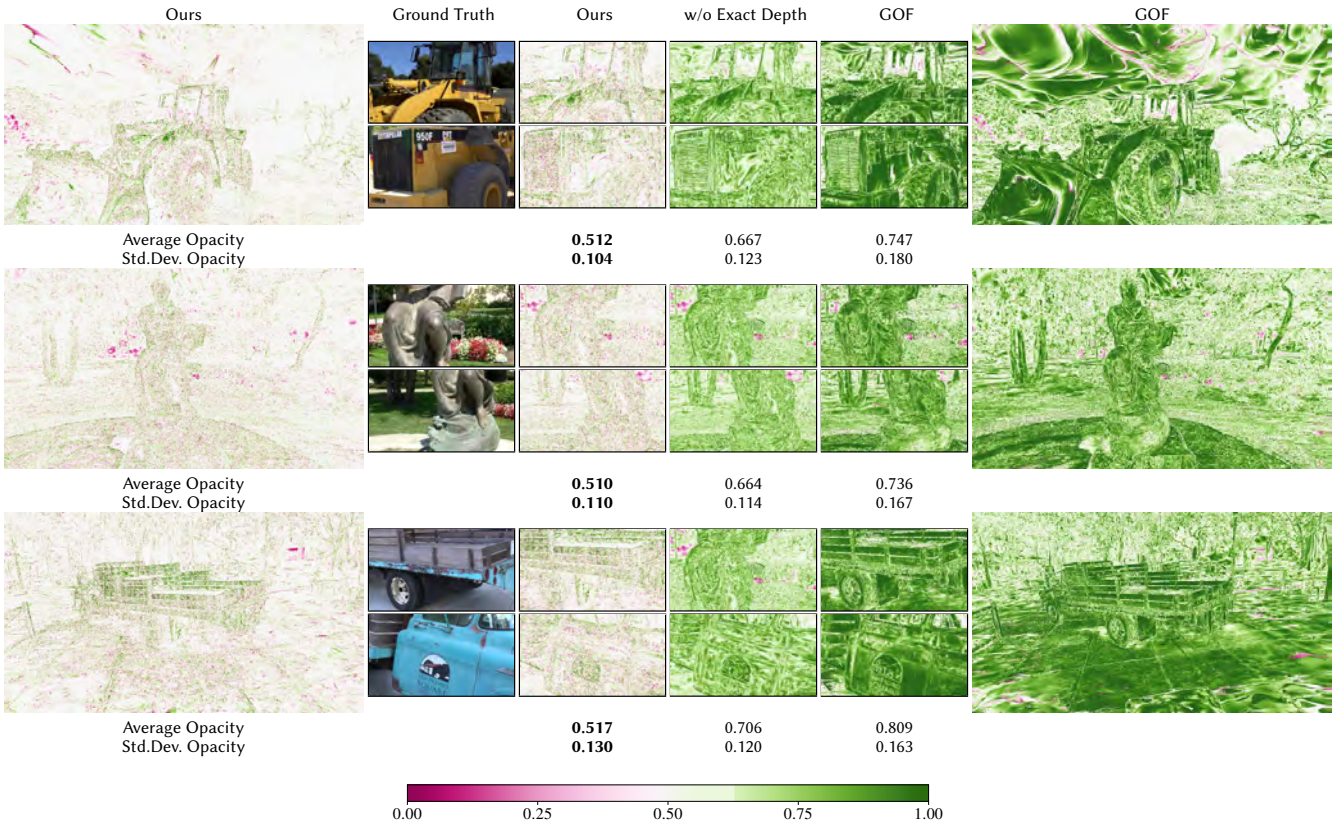

Fig. 6. **Level-Set Alignment Comparison**: We render $O_N(t_\mathbf{r})$ for our method and GOF, as well as our method with exact depth (white is 0.5, see the inset colorbar). Overall, as we can see by the images as well as the reported statistics, the depth and the 0.5 level set are closely aligned; GOF vastly overestimates the depth values, leading to high opacity values. The reported statistics were averaged over the entire test set.

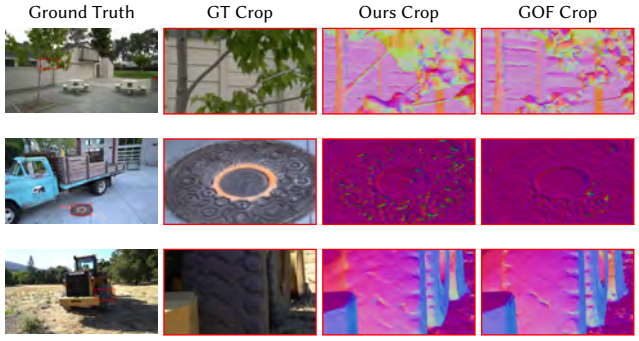

Fig. 7. **Quality Comparison:** Compared to GOF, which tends to create overly smooth geometry, our method recovers high-quality meshes with finer geometric detail.

4.3 Performance Analysis

We conduct a thorough performance evaluation of our method compared to GOF [Yu et al. 2024b], where we use the Tanks & Temples dataset [Knapitsch et al. 2017] throughout. For all presented results, we used an NVIDIA RTX 4090 with CUDA 12.2.

Training. Despite the required computational overhead necessary for per-pixel resorting [Radl et al. 2024b] and our newly introduced losses, we are still $\approx 3\times$ faster than GOF for optimization. Concretely, we find that the advanced culling strategies from StopThePop, combined with our additional optimizations alone completely negate the overhead from resorting.

Meshing. We conduct a thorough performance evaluation for our Fast Marching Tetrahedra algorithm. We take trained models from GOF for all scenes in Tanks & Temples [Knapitsch et al. 2017] to ensure a fair, point-cloud independent comparison. We average timings for point-preprocessing, Gaussian-processing and opacity field evaluation. The results are shown in Table 4.

As we can see, our full approach is $\approx 6.8\times$ faster than GOF, in terms of average iteration time. Introducing our Tile/Ray Scheduling already improves performance, benefiting from a more evenly distributed workload across thread blocks. With Min Z Bounding, performance once again improves drastically by removing unnecessary Gaussian evaluations.

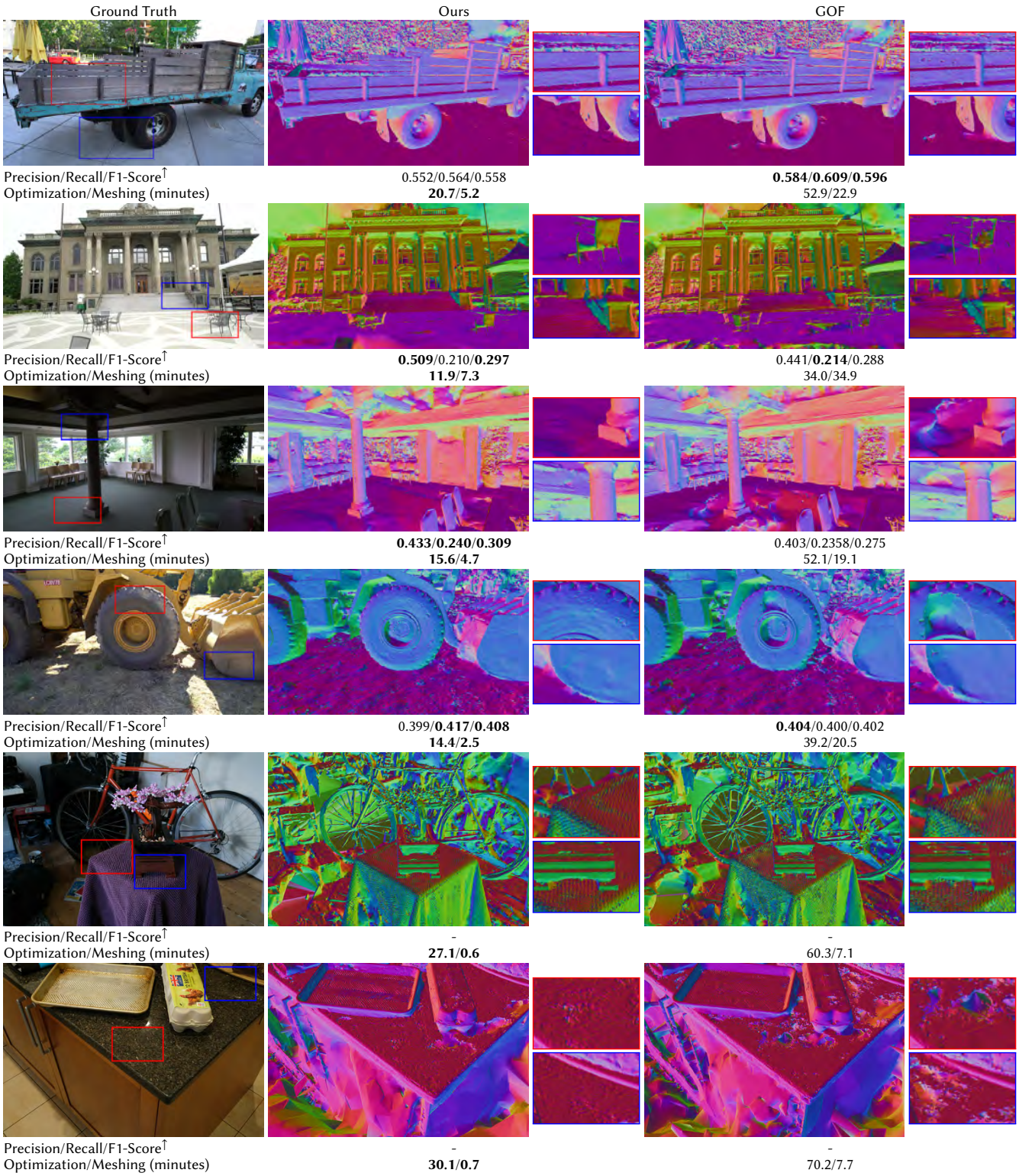

Fig. 8. **Qualitative Evaluation for Unbounded Mesh Extraction**: We compare our unbounded meshes with those extracted by GOF using scenes from Tanks & Temples [Knapitsch et al. 2017] as well as Mip-NeRF 360 [Barron et al. 2022]. We report surface reconstruction metrics (if available) as well as optimization/meshing times (not including Tetrahedralization). As we can see, our meshes are more detailed and exhibit fewer artifacts.

Table 4. **Marching Tetrahedra Performance Analysis**: Our Fast Marching Tetrahedra algorithm outperforms GOF by a factor of ≈ 6.8×. Timings were averaged over all scenes from the Tanks & Temples dataset [Knapitsch et al. 2017], using the same Gaussian point cloud (from GOF) for fairness.

Timings in ms	Process Points	Process Gaussians	Integrate	Total
GOF	1.17	1.07	501.12	503.38
+ Tile/Ray Scheduling	1.15	0.99	415.99	418.13
+ Min Z Bounding	1.15	0.99	257.29	259.44
+ Early Stopping	1.15	0.99	116.32	118.27
+ Point Pruning	0.91	0.99	72.41	74.40

Early stopping (as performed here for the 0.5 level set) once again doubles performance, and point pruning leads to another large reduction in meshing times.

When adding up training and meshing time, our approach is more than 3× faster compared to GOF (see App. B.3 for a detailed breakdown), and results in better meshes, as seen previously.

4.4 Ablation Studies

In this section, we provide further experiments and analyze each introduced component; for more experiments, cf. App. B.1.

Surface Reconstruction. We investigate the impact of our proposed components for unbounded scenes in Table 5. As we can see in *(A) w/o Exact Depth*, disabling Exact Depth has the largest impact on reconstruction quality. Besides, disabling $\mathcal{L}_{\text{opa}}$, $\mathcal{L}_{\text{ext}}$, $\mathcal{L}_{\text{smooth}}$ has a minor impact on final reconstruction quality. As can be seen for *(F)*, disabling both $\mathcal{L}_{\text{opa}}$, $\mathcal{L}_{\text{ext}}$ has a larger impact on reconstruction, highlighting that the losses benefit from each other.

Table 5. **Surface Reconstruction Ablation**: We remove individual components of our method and evaluate F1-score for the Tanks & Temples dataset [Knapitsch et al. 2017]. As we gradually remove individual parts of SOF, the performance degrades. Removing exact depth decreases quality drastically, highlighting the need for robust depth estimates.

Method	Barn	Caterp	Courth	Ignatius	Meetingr	Truck	Avg
Ours	0.535	0.408	0.297	0.736	0.309	0.558	0.474
(A) w/o Exact Depth	0.468	0.396	0.272	0.706	0.286	0.545	0.445
(B) w/o Attach Grad	0.508	0.407	0.292	0.724	0.305	0.551	0.465
(C) w/o $\mathcal{L}_{\text{opa}}$	0.535	0.408	0.292	0.734	0.305	0.552	0.471
(D) w/o $\mathcal{L}_{\text{extent}}$	0.534	0.407	0.288	0.736	0.308	0.547	0.470
(E) w/o $\mathcal{L}_{\text{smooth}}$	0.523	0.406	0.299	0.738	0.307	0.553	0.471
(F) w/o $\mathcal{L}_{\text{opa}}$, $\mathcal{L}_{\text{extent}}$	0.527	0.405	0.283	0.733	0.304	0.544	0.466

Exact Depth Computation. To compare our depth rendering strategy with the one from GOF, we evaluate the opacity according to Eqn. (12), using median depth as well as our exact depth.

As can be seen in Fig. 6, the opacity is much closer to 0.5 for our exact depth computation, as median depth vastly overestimates the depth. Disabling exact depth still leads to closer alignment compared to GOF, which can be attributed to $\mathcal{L}_{\text{opa}}$ and hierarchical resorting.

5 DISCUSSION, LIMITATIONS AND FUTURE WORK

Generally, 3D Gaussians remain a poor target to extract a level set from, as identifying their exact level set requires expensive iterative evaluations (as performed here). Another alternative is to use constant-density ellipsoids [Mai et al. 2024] or constant-density tetrahedral cells [Govindarajan et al. 2025], for which the exact level set can be computed efficiently; however, these methods are imply longer optimization times. Therefore, moving towards fully correct volumetric rendering [Condor et al. 2025; Talegaonkar et al. 2025] may offer a practical compromise.

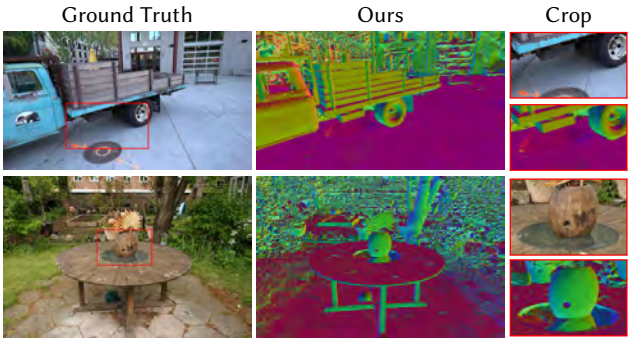

Fig. 9. **Failure Cases:** 3DGS struggles to model surfaces with view-dependent effects, resulting in holes due to a lack of primitive density, as well as inconsistent Gaussian orientation.

Another challenging current limitation is the use of low-order spherical harmonics, which do not allow for accurate modeling of complex view-dependent effects. Thus, 3DGS resorts to incorrect Gaussian placement to "fake" correct view-dependent appearance with additional geometry, in turn posing a challenge to surface reconstruction methods (cf. Fig. 9). While recent work has already investigated improved approaches for view-dependent appearance [Liu et al. 2025; Yang et al. 2024], these methods still fall short in extreme scenarios, highlighting the need for a more robust solution.

Finally, while we demonstrated improved quality for unbounded surface reconstruction, quality generally degrades compared to state-of-the-art implicit methods [Li et al. 2023] or recent preprints which explore bounded meshing with MVS-constrains [Chen et al. 2024; Wang et al. 2024]. Future work could further improve the scene representation to bridge the current quality gap.

6 CONCLUSION

In this work, we rigorously analyzed Gaussian Opacity Fields and its shortcomings. First, we show how GOF consistently overestimates the depth along a view ray, which leads to worse geometry reconstruction results; we present a more exact depth estimate, which remedies these issues and enables a better estimate of the 0.5 level set during training. In addition, we presented a novel extent loss along with direct opacity field evaluation to improve quality for meshing of unbounded scenes. Although these losses incur a slight computational overhead, this is remedied by smart performance optimization, leading to reduced training times overall. We also presented a novel *Marching Tetrahedra* parallelization scheme, which is 10× faster compared to GOF. Our full method outperforms related work both qualitatively and quantitatively.

REFERENCES

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proc. ACM Comput. Graph. Interact. Tech* 3, 2, Article 15 (2020), 23 pages.

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *ICCV*.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *CVPR*.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2023. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. In *ICCV*.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. TensoRF: Tensorial Radiance Fields. In *ECCV*.

Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. 2024. PGSR: Planar-based Gaussian Splatting for Efficient and High-Fidelity Surface Reconstruction. *arxiv preprint arxiv:2406.06521* (2024).

Jorge Condor, Sebastien Speierer, Lukas Bode, Aljaz Bozic, Simon Green, Piotr Didyk, and Adrian Jarabo. 2025. Don't Splat your Gaussians: Volumetric Ray-Traced Primitives for Modeling and Rendering Scattering and Emissive Media. *ACM TOG* (2025).

Brian Curless and Marc Levoy. 1996. A Volumetric Method for Building Complex Models from Range Images. In *SIGGRAPH*.

Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. 2024. High-quality Surface Reconstruction using Gaussian Surfels. In *SIGGRAPH Asia*.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. 2024. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. In *NeurIPS*.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance Fields without Neural Networks. In *CVPR*.

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

Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. 2025. Radiant Foam: Real-Time Differentiable Ray Tracing. arXiv:2502.01157 [cs.CV] https://arxiv.org/abs/2502.01157

Antoine Guédon and Vincent Lepetit. 2024. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. In *CVPR*.

Florian Hahlbohm, Fabian Friederichs, Tim Weyrich, Linus Franke, Moritz Kappel, Susana Castillo, Marc Stamminger, Martin Eisemann, and Marcus Magnor. 2025. Efficient Perspective-Correct 3D Gaussian Splatting Using Hybrid Transparency. *Comput. Graph. Forum* 44, 2 (2025).

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep Blending for Free-viewpoint Image-based Rendering. *ACM TOG* 37, 6, Article 257 (2018), 15 pages.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024b. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH*.

Letian Huang, Jiayang Bai, Jie Guo, Yuanqi Li, and Yanwen Guo. 2024a. On the Error Analysis of 3D Gaussian Splatting and an Optimal Projection Strategy. In *ECCV*.

Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, George Drettakis, and Thomas Leimkühler. 2023. NeRFshop: Interactive Editing of Neural Radiance Fields. *Proc. ACM Comput. Graph. Interact. Tech* 6, 1 (2023), 1:1–1:21.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. 2014. Large Scale Multi-View Stereopsis Evaluation. In *CVPR*.

Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. 2023. GaussianShader: 3D Gaussian Splatting with Shading Functions for Reflective Surfaces. In *CVPR*.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM 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. 2024. 3D Gaussian Splatting as Markov Chain Monte Carlo. In *NeurIPS*.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM TOG* 36, 4, Article 78 (2017), 13 pages.

Jonas Kulhanek and Torsten Sattler. 2023. Tetra-NeRF: Representing Neural Radiance Fields Using Tetrahedra. In *ICCV*.

Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. 2023. Neuralangelo: High-Fidelity Neural Surface Reconstruction. In *CVPR*.

Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, and Wenming Yang. 2024. VastGaussian: Vast 3D Gaussians for Large Scene Reconstruction. In *CVPR*.

Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. 2025. Deformable Beta Splatting. arXiv:2501.18630 [cs.CV] https://arxiv.org/abs/2501.18630

William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH*.

Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T. Barron, and Yinda Zhang. 2024. EVER: Exact Volumetric Ellipsoid Rendering for Real-time View Synthesis. arXiv:2410.01804 [cs.CV] https://arxiv.org/abs/2410.01804

Saswat Mallick, Rahul Goel, Bernhard Kerbl, Francisco Vicente Carrasco, Markus Steinberger, and Fernando De La Torre. 2024. Taming 3DGS: High-Quality Radiance Fields with Limited Resources. In *SIGGRAPH Asia*.

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

Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 2024. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM TOG* 43, 6, Article 232 (2024), 19 pages.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM TOG* 41, 4, Article 102 (2022), 15 pages.

Thu Nguyen-Phuoc, Feng Liu, and Lei Xiao. 2022. SNeRF: Stylized Neural Implicit Representations for 3D Scenes. *ACM TOG* 41, 4 (2022), 142:1–142:11.

Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *ICCV*.

Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. 2024. Reducing the Memory Footprint of 3D Gaussian Splatting. *Proc. ACM Comput. Graph. Interact. Tech* 7, 1 (2024).

Lukas Radl, Michael Steiner, Andreas Kurz, and Markus Steinberger. 2024a. LAENeRF: Local Appearance Editing for Neural Radiance Fields. In *CVPR*.

Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. 2024b. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM TOG* 43, 3, Article 64 (2024).

Christian Reiser, Stephan Garbin, Pratul Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan Barron, Peter Hedman, and Andreas Geiger. 2024. Binary Opacity Grids: Capturing Fine Geometric Detail for Mesh-Based View Synthesis. *ACM TOG*, Article 149 (July 2024), 14 pages.

Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. 2024. Revising Densification in Gaussian Splatting. In *ECCV*.

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. 2021. Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis. In *NeurIPS*.

Michael Steiner, Thomas Köhler, Lukas Radl, Felix Windisch, Dieter Schmalstieg, and Markus Steinberger. 2025. AAA-Gaussians: Anti-Aliased and Artifact-Free 3D Gaussian Rendering. arXiv:2504.12811 [cs.GR] https://arxiv.org/abs/2504.12811

Chinmay Talegaonkar, Yash Belhe, Ravi Ramamoorthi, and Nicholas Antipa. 2025. Volumetrically Consistent 3D Gaussian Rasterization. arXiv:2412.03378 [cs.CV] https://arxiv.org/abs/2412.03378

Xuechang Tu, Lukas Radl, Michael Steiner, Markus Steinberger, Bernhard Kerbl, and Fornando de la Torre. 2025. VRsplat: Fast and Robust Gaussian Splatting for Virtual Reality. *Proc. ACM Comput. Graph. Interact. Tech* 8, 1, Article 1 (2025).

Jiepeng Wang, Yuan Liu, Peng Wang, Cheng Lin, Junhui Hou, Xin Li, Taku Komura, and Wenping Wang. 2024. GausSurf: Geometry-Guided 3D Gaussian Splatting for Surface Reconstruction. *arXiv preprint arXiv:2411.19454* (2024).

Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. 2024. Spec-Gaussian: Anisotropic View-Dependent Appearance for 3D Gaussian Splatting. In *NeurIPS*.

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume Rendering of Neural Implicit Surfaces. In *NeurIPS*.

Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. 2023. BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis. In *SIGGRAPH*.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. 2024a. Mip-Splatting: Alias-free 3D Gaussian Splatting. In *CVPR*.

Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. 2022. MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction. In *NeurIPS*.

Zehao Yu, Torsten Sattler, and Andreas Geiger. 2024b. Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. *ACM TOG* 43, 6, Article 271 (Nov. 2024), 13 pages.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.

Mathias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. 2001. EWA Volume Splatting. In *IEEE Visualization*.

A ADDITIONAL METHOD DETAILS

In this section, we provide additional implementation details and mathematical derivations for our method.

A.1 Efficient 3D Gaussian Evaluation

The following performance optimization is already present in the current GOF codebase[1] [Yu et al. 2024b]; for completeness, we describe it here. Note that we re-evaluate GOF using the current implementation for all presented results (unless indicated otherwise).

Recall the definition of A_i, B_i, C_i from Sec. 3.1:

$$A_i = \mathbf{d}_g^T \mathbf{d}_g \tag{27}$$

$$B_i = 2\mathbf{d}_g^T \mathbf{o}_g \tag{28}$$

$$C_i = \mathbf{o}_g^T \mathbf{o}_g. \tag{29}$$

Decomposing $A_i = \mathbf{d}^T \mathbf{R}_i^T \mathbf{S}_i^{-1} \mathbf{S}_i^{-1} \mathbf{R}_i \mathbf{d}$, we see that this can also be rewritten as $\mathbf{d}^T \Sigma^{-1} \mathbf{d}$. Thus, as Σ^{-1} is a 3×3 symmetric matrix and independent of $\mathbf{d}$, we can precompute it and store it in 6 values per Gaussian.

Similarly, we can precompute $\mathbf{o}^T \mathbf{S}_i^{-1} \mathbf{S}_i^{-1} \mathbf{R}_i$ for B_i (which can be stored in 3 values) and precompute C_i directly, as a scalar. As a result, we can compute these 10 values during Gaussian preprocessing instead of computing them on-the-fly, leading to improved performance overall.

A.2 Sorting for Opacity Field Evaluation

Here, we mathematically demonstrate why sorting is not required for correct opacity field evaluation.

Proposition A.1. *Let $\alpha_i = o_i \mathcal{G}_i(t_i)$. Now,*

$$O_N = \sum_{i=0}^{N-1} \alpha_i \prod_{j=0}^{i-1} (1 - \alpha_j)$$

is independent of the sorting order.

Proof. We want to show that $O_N = 1 - \prod_{i=0}^{N-1}(1 - \alpha_j)$, which is order-independent due to the product. To this end, we have

$$O_N = \alpha_0 + (1 - \alpha_0) \sum_{i=1}^{N-1} \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j).$$

We proceed by induction.

Base case ($N = 1$): In both cases, we clearly have $O_1 = \alpha_0$.

Inductive step: Assume the identity holds for $N - 1$, *i.e.*,

$$\sum_{i=1}^{N-1} \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) = 1 - \prod_{i=1}^{N-1} (1 - \alpha_i). \tag{a}$$

We now consider N:

$$\begin{aligned}
\sum_{i=0}^{N-1} \alpha_i \prod_{j=0}^{i-1} (1 - \alpha_j) &= \alpha_0 + (1 - \alpha_0) \sum_{i=1}^{N-1} \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \\
&\overset{(a)}{=} \alpha_0 + (1 - \alpha_0) \left(1 - \prod_{i=1}^{N-1} (1 - \alpha_i) \right) \\
&= \alpha_0 + (1 - \alpha_0) - (1 - \alpha_0) \prod_{i=1}^{N-1} (1 - \alpha_i) \\
&= 1 - \prod_{i=0}^{N-1} (1 - \alpha_i).
\end{aligned}$$

Thus, O_N is a product of $(1 - \alpha_i)$, and thus order-independent. □

A.3 Exact Depth Details

Exact Depth Derivation. We want to find a t such that

$$T_i \left(1 - o_i \mathcal{G}_i^{1D}(t) \right) = 0.5. \tag{30}$$

First, we isolate $\mathcal{G}_i^{1D}(t)$:

$$\begin{aligned}
T_i - T_i o_i \mathcal{G}_i^{1D}(t) &= 0.5 \\
\mathcal{G}_i^{1D}(t) &= \frac{T_i - 0.5}{\alpha_i}.
\end{aligned}$$

Next, we insert Eqn. (8) and continue reformulating

$$\begin{aligned}
\exp\left(-\frac{1}{2} \left(A_i t^2 + B_i t + C_i \right) \right) &= \frac{T_i - 0.5}{\alpha_i} \\
A_i t^2 + B_i t + C_i + 2 \ln \left(\frac{T_i - 0.5}{\alpha_i} \right) &= 0,
\end{aligned}$$

which is a quadratic equation in t. The solution is thus

$$t_{1,2} = t_{\mathbf{r}} \pm \frac{\sqrt{B_i^2 - 4A_i \left(C_i + 2 \ln \left(\frac{T_i - 0.5}{\alpha_i} \right) \right)}}{2A_i}. \tag{31}$$

Because we defined the Gaussian contribution to be maximal at $t_{\mathbf{r}}$, our exact depth is

$$t = t_{\mathbf{r}} - \frac{\sqrt{B_i^2 - 4A_i \left(C_i + 2 \ln \left(\frac{T_i - 0.5}{\alpha_i} \right) \right)}}{2A_i}. \tag{32}$$

Backward Details. It can be shown that the additional gradients due to the exact depth computations are given as

$$\frac{\partial \delta_t}{\partial A_i} = \frac{B_i^2 - 2A_i \left(C_i + 2 \ln \left(\frac{T_i - 0.5}{\alpha_i} \right) \right)}{2A_i^2 \delta_t}, \tag{33}$$

$$\frac{\partial \delta_t}{\partial B_i} = \frac{t_{\mathbf{r}}}{\delta_t}, \tag{34}$$

$$\frac{\partial \delta_t}{\partial C_i} = \frac{1}{\delta_t}, \tag{35}$$

where $\delta_t = -\frac{\sqrt{B_i^2 - 4A_i \left(C_i + 2 \ln \left(\frac{T_i - 0.5 +}{\alpha_i} \right) \right)}}{2A_i}$. In our implementation, we detached these gradients, as they resulted in worse metrics overall (cf. Table A.1, *(A) Exact Depth Grad*).

[1] https://github.com/autonomousvision/gaussian-opacity-fields

Table A.1. **Additional Ablation Studies**: We introduce different component to our method and report the F1-score for the Tanks & Temples dataset [Knapitsch et al. 2017].

Method	Barn	Caterp	Courth	Ignatius	Meetingr	Truck	Avg
GOF	0.484	0.402	0.288	0.674	0.275	0.596	0.453
Ours	0.535	0.408	0.297	0.736	0.309	0.558	0.474
(A) Exact Depth Grad	0.542	0.396	0.280	0.725	0.307	0.550	0.467
(B) MCMC	0.487	0.375	0.284	0.674	0.286	0.531	0.439
(C) Recycling	0.531	0.402	0.288	0.721	0.297	0.560	0.467
(D) Recloning	0.535	0.400	0.283	0.720	0.294	0.549	0.463
(E) PGSR Appearance	0.532	0.398	0.288	0.702	0.284	0.552	0.459
(F) Scale Minimization	0.535	0.400	0.283	0.720	0.294	0.549	0.463

A.4 Densification

We identify that densification is actually a significant factor for the resulting mesh quality. To this end, we integrate 3DGS-MCMC [Kheradmand et al. 2024], a recent state-of-the-art densification method for 3DGS, and present the results in Table A.1, variant *(B) MCMC Densification*.

As we can see, the results are worse when considering the same number of primitives. Intuitively, 3DGS-MCMC simply moves low-opacity Gaussians to the location of high-opacity Gaussians, and modifies their opacity such that the change in output is minimal. Compared to the densification strategy in GOF, this results in a more non-uniform distribution of primitives, which poses issues for the subsequent Marching Tetrahedra algorithm.

A.5 Dead Gaussians.

Gaussians with $o_i < \frac{1}{255}$ are no longer visible from any view (as their corresponding α_i can no longer exceed this threshold), and can thus no longer receive gradients. When using the 3D filter from Yu et al. [2024a], this applies to even more primitives, as the opacity with the 3D filter applied $\hat{o}_i$ is

$$\hat{o}_i = o_i \sqrt{\frac{|\Sigma_i|}{|\Sigma_i + s\mathbf{I}|}}, \tag{36}$$

with $s > 0$, thus the scaling factor is smaller than 1 (cf. Yu et al. [2024a] for details). In our experiments, we find that $\approx 9\%$ of Gaussians are no longer visible by the end of training (cf. Table A.3 for details). To this end, we experimentally devise two strategies to make full use of our Gaussian budget, after densification.

Recycling. Inspired by 3DGS-MCMC [Kheradmand et al. 2024], we repurpose their cloning strategy to move dead Gaussians towards Gaussians with high opacity.

Recloning. Due to the tendency of 3DGS-MCMC to create a more non-uniform distribution of Gaussians, we also design a recloning strategy, inspired by GOF densification [Yu et al. 2024b]. As done previously, we sample the alive Gaussians with their opacity as their probability. However, instead of directly placing the Gaussians at the mean of the sampled Gaussians, we sample another position using the selected Gaussian as the PDF.

As we can see in Table A.1 (cf. *(C) Recycling*, *(D) Recloning*), both variants perform worse compared to simply ignoring these primitives altogether.

B ABLATION STUDIES

In this section, we present additional ablation studies and more experimental results.

B.1 Additional Experiments

PGSR Appearance. We additionally test the appearance model proposed by PGSR [Chen et al. 2024]. Here, each image i is assigned two learnable scalars $a_i, b_i \in \mathbb{R}$, and $\mathcal{L}_{\text{rgb}}$ is computed with the adjusted rendered image $\hat{\mathbf{I}}_{\text{PGSR}}$

$$\hat{\mathbf{I}}_{\text{PGSR}} = \exp(a_i)\,\hat{\mathbf{I}} + b_i, \tag{37}$$

with $\hat{\mathbf{I}}$ the rendered image. We provide results for both DTU and Tanks & Temples (cf. Table A.8, *(G)* and Table A.1, *(E)*). As can be seen, we find that the appearance model based on VastGaussian [Lin et al. 2024], as used in our final model, performs better compared to the PGSR appearance embedding.

Scale Minimization. We also experiment with minimizing the smallest scale of each Gaussian [Jiang et al. 2023], *i.e.* using

$$\mathcal{L}_{\text{scale}} = \frac{1}{N} \sum_{i=0}^{N-1} \min(\mathbf{s}_i), \tag{38}$$

with $\min(\mathbf{s}_i)$ denoting the minimum scalar element of the vector $\mathbf{s}_i \in \mathbb{R}^3_+$. As can be seen in Table A.1 (cf. *F*), our full configuration, using our novel extent loss, achieves better results.

DTU Ablation. We present additional ablation studies for the DTU dataset [Jensen et al. 2014] in Table A.8; we also included the results for our re-evaluation of GOF and 2DGS. We also show a qualitative comparison for bounded meshes in Fig. A.2.

Novel View Synthesis. In addition to our results for the Mip-NeRF 360 dataset [Barron et al. 2022], we additionally include image quality metrics for 4 scenes sourced from Tanks & Temples [Knapitsch et al. 2017] and Deep Blending [Hedman et al. 2018]. We present the results in Table A.2.

We also provide a qualitative comparison in Fig. A.1. As we can see, our method produces similar rendering compared to GOF.

B.2 Bounding Strategies for Marching Tetrahedra

As discussed in the main material, the 3σ cutoff is not correct, and results in bounding-boxes which are too small (if $o \approx 1$) or overly large bounding boxes otherwise. As also discussed in Appendix A.4, we observe a large number of "dead" primitives, which are no longer rendered due to their low opacity; thus, they also no longer receive gradients. These Gaussians might correspond to floaters which were eliminated by the opacity resets [Kerbl et al. 2023] or optimization; hence, they are not necessarily a good indicator of surfaces.

To this end, we perform an additional ablation study with different cutoff values and Gaussian culling. In terms of grid generation strategies, we evaluate 3σ, as used by GOF [Yu et al. 2024b], 3.33σ (obtained by letting $o = 1$ in Eqn. (3) , which produces overly large bounding boxes) as well as our proposed bounding strategy. For culling, we evaluate the standard approach of considering all Gaussians for tetrahedral grid generation as well as removing all Gaussians with $o < o_{\text{min}}$. We show the results in Table A.4.

Table A.2. **Additional Novel View Synthesis results** for Tanks & Temples [Knapitsch et al. 2017] and Deep Blending [Hedman et al. 2018]; we use the same scenes as done in Kerbl et al. [2023]. As can be seen, our method, as well as our MCMC-variant, perform very well for the Deep Blending dataset.

Method	DrJohnson				Playroom				Train				Truck			
	PSNR↑	SSIM↑	LPIPS↓	ꟻLIP ↓	PSNR↑	SSIM↑	LPIPS↓	ꟻLIP ↓	PSNR↑	SSIM↑	LPIPS↓	ꟻLIP ↓	PSNR↑	SSIM↑	LPIPS↓	ꟻLIP ↓
3DGS	29.060	0.898	0.247	0.119	29.860	0.901	0.246	0.143	22.040	0.813	0.208	0.250	25.390	0.878	0.148	0.148
Mip-Splatting	29.150	0.902	0.243	0.118	30.170	0.909	0.243	0.142	22.160	0.818	0.205	0.249	25.480	0.886	0.147	0.149
StopThePop	29.420	0.903	0.234	0.115	30.300	0.905	0.235	0.138	21.480	0.808	0.204	0.267	24.940	0.878	0.142	0.164
Taming-3DGS	29.510	0.909	0.234	0.117	30.240	0.911	0.235	0.151	22.250	0.819	0.208	0.253	25.880	0.892	0.128	0.143
3DGS-MCMC	29.520	0.904	0.234	0.118	29.930	0.908	0.233	0.151	22.830	0.843	0.183	0.241	26.450	0.900	0.112	0.135
2DGS	28.950	0.900	0.256	0.121	30.230	0.907	0.255	0.142	21.190	0.791	0.250	0.268	25.100	0.873	0.173	0.154
GOF	28.760	0.900	0.247	0.121	30.280	0.911	0.239	0.140	21.590	0.817	0.203	0.265	25.520	0.888	0.133	0.150
Ours	29.380	0.907	0.232	0.114	30.650	0.915	0.230	0.136	20.800	0.804	0.212	0.284	25.100	0.890	0.127	0.163
Ours (MCMC)	29.880	0.908	0.234	0.113	31.030	0.915	0.242	0.136	20.910	0.810	0.201	0.279	25.260	0.893	0.119	0.162

Table A.3. **Point Cloud Analysis**: We report the number of primitives, percentage of unrendered primitives as well as the final number of mesh vertices for Ours and GOF.

Method		Barn	Caterp	Courth	Ignatius	Meetingr	Truck	Avg
Ours	#Gaussians	847K	1.06M	386K	2.65M	942K	2.19M	1.35M
	% dead	5.02%	8.54%	7.86%	8.07%	11.91%	8.94%	8.39%
	#Vertices	10.19M	11.58M	3.30M	27.77M	9.14M	22.45M	14.07M
GOF	#Gaussians	835K	1.06M	390K	2.63M	962K	2.16M	1.34M
	% dead	5.48%	7.59%	11.26%	8.77%	12.33%	10.61%	9.34%
	#Vertices	10.05M	11.55M	4.02M	27.76M	8.61M	22.68M	14.03M

As we can see, these strategies have a very minor impact on the final quality when averaged over many scenes. However, note that removing primitives for the initial tetrahedral grid generation positively impacts meshing times, which justifies our use of the 0.0039 cutoff. Conversely, while this results in a reduced number of initial points for the tetrahedral grid generation, this does not impact the vertex count, as can be seen in Table A.3.

Table A.4. **Marching Tetrahedra Bounding Strategies**: We evaluate 3 types of bounding boxes (StopThePop-bounding, 3σ, 3.33σ) and 2 types of culling strategies (no culling, $\frac{1}{255}$) and report the results for precision, recall and F1-score.

Bounding	Cutoff	Precision↑	Recall↑	F1-score↑
StopThePop	-	0.5404	0.4376	0.4732
	0.0039	0.5405	0.4376	0.4733
3σ	-	0.5374	0.4392	0.4733
	0.0039	0.5388	0.4384	0.4732
3.33σ	-	0.5366	0.4388	0.4728
	0.0039	0.5389	0.4384	0.4732

B.3 Full Timing Breakdown

We provide a detailed total timings comparison for our method and GOF in Table A.6. Note that both methods use the Delaunay Triangulation from Tetra-NeRF [Kulhanek and Sattler 2023], which uses the CGAL library. We explicitly isolated this stage, as we did not optimize it and still use the same implementation; however, note this the performance of this stage is influenced by the number of primitives.

As we can see, our method is much faster compared to GOF, particularly for the binary search, where our method reduces the average time from 30 to just 7 minutes. The tetrahedralization stage also benefits from our culling strategies for tetrahedral grid generation, particularly if the number of "dead" Gaussians is high; however, we observed a constant percentage of dead Gaussians for Tanks & Temples.

Table A.5. **Detailed per-scene results for the Tanks & Temples dataset** [Knapitsch et al. 2017].

Metric	Method	Barn	Caterp	Courth	Ignatius	Meetingr	Truck	Avg
Precision	Ours	0.589	0.399	0.509	0.768	0.433	0.552	0.542
	GOF	0.522	0.404	0.441	0.725	0.403	0.584	0.513
Recall	Ours	0.490	0.417	0.210	0.707	0.240	0.564	0.438
	GOF	0.451	0.400	0.214	0.631	0.208	0.609	0.419
F1-score	Ours	0.535	0.408	0.297	0.736	0.309	0.558	0.474
	GOF	0.484	0.402	0.288	0.674	0.275	0.596	0.453

B.4 Per Scene Results

In this section, we present detailed per-scene results for surface reconstruction and novel view synthesis.

TNT Dataset. We present detailed per-scene results for the Tanks & Temples dataset [Knapitsch et al. 2017], including precision and recall, compared to GOF in Table A.5. As we can see, our method achieves better reconstruction results compared to GOF for all scenes, except for *Truck*. Particularly for this scene, we find that our method struggles to reconstruct a smooth ground (cf. Fig. 8), resulting in a reduction for all surface reconstruction metrics.

Mip-NeRF 360 Dataset. We present detailed per-scene results for the Mip-NeRF 360 dataset [Barron et al. 2022] in Table A.7.

C DERIVATIONS

In this section, we provide additional mathematical derivations.

Table A.6. **Detailed Timing Comparisons** for Ours and GOF [Yu et al. 2024b] on the Tanks & Temples dataset [Knapitsch et al. 2017], including Optimization, Tetrahedralization [Kulhanek and Sattler 2023] and Binary Search (in minutes). Overall, our method is about 2× faster on average, with improved mesh quality.

Timings in minutes	Barn (837K)		Caterpillar (1.06M)		Courthouse (384K)		Ignatius (2.74M)		Meetingroom (938K)		Truck (2.15M)		Average	
	GOF	Ours	GOF	Ours	GOF	Ours	GOF	Ours	GOF	Ours	GOF	Ours	GOF	Ours
Optimization	40.1m	14.3m	39.2m	14.4m	34.0m	11.5m	50.6m	23.7m	52.1m	15.6m	52.9m	20.7m	44.8m	16.7m
Tetrahedralization [2023]	1.2m	1.2m	1.6m	1.5m	0.6m	0.5m	4.3m	3.8m	1.4m	1.3m	3.1m	2.9m	2.2m	1.9m
Binary Search	30.3m	3.1m	20.5m	2.5m	34.9m	7.3m	15.6m	5.2m	19.1m	4.7m	22.9m	5.2m	23.9m	4.6m
Total	71.6m	18.6m	61.3m	18.4m	69.5m	19.3m	70.5m	32.7m	72.6m	21.6m	78.9m	28.8m	70.7m	23.2m

C.1 Distortion Loss Derivation

The distortion loss [Huang et al. 2024b] is formally defined as

$$\mathcal{L}_{\text{dist}} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} w_i w_j (d_i - d_j)^2, \tag{39}$$

with $d_i = \text{NDC}(t_i)$, and t_i the depth of the i-th Gaussian. As demonstrated by Huang et al. [2024b], the loss can be reformulated as

$$\mathcal{L}_{\text{dist}} = \sum_{i=0}^{N-1} w_i (d_i^2 A_{i-1} + \mathcal{D}_{i-1} - 2 d_i D_{i-1}), \tag{40}$$

with $A_i = \sum_{j=0}^{i} w_j$, $D_i = \sum_{j=0}^{i} w_j d_j$, and $\mathcal{D}_i = \sum_{j=0}^{i} w_j d_j^2$.

It can be shown that the required derivatives $\frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_k}$, $\frac{\partial \mathcal{L}_{\text{dist}}}{\partial d_k}$ are

$$\frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_k} = d_i^2 A_{N-1} + \mathcal{D}_{N-1} - 2 d_i D_{N-1}, \tag{41}$$

$$\frac{\partial \mathcal{L}_{\text{dist}}}{\partial d_k} = 2\alpha_k \left(d_k A_{N-1} - D_{N-1}\right). \tag{42}$$

However, due to occlusion (*i.e.* α_k being large reduces the loss for α_{k+1} due to T_{k+1}), the derivative w.r.t. α_k is actually (in back-to-front order) [Huang et al. 2024b]

$$\frac{\partial \mathcal{L}_{\text{dist}}}{\partial \alpha_k} = \alpha_k \frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_k} + (1-\alpha_k)\frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_{k+1}}. \tag{43}$$

However, StopThePop [Radl et al. 2024b] performs the backward pass in front-to-back ordering, avoiding the otherwise implied memory overhead of storing per-ray sorted lists of Gaussians.

We can write $\frac{\partial \mathcal{L}}{\partial \alpha_k}$ as

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \alpha_k} &= \left(\frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_k} - \alpha_{k+1}\frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_{k+1}} - \sum_{i=k+2}^{N-1}\alpha_i \prod_{j=k+1}^{i-1}(1-\alpha_j)\frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_i}\right) T_k \\ &= \left(\frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_k} - \frac{1}{T_{k+1}}\sum_{i=k+1}^{N-1} w_i \frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_i}\right) T_k.\end{aligned}$$

Now, we analyze the term $\sum_{i=k+1}^{N-1} w_i \frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_i}$:

$$\begin{aligned}\sum_{i=k+1}^{N-1} w_i \frac{\partial \mathcal{L}_{\text{dist}}}{\partial w_i} &= \sum_{i=k+1}^{N-1} w_i (d_i^2 A + \mathcal{D} - 2 d_i D), \\ &= A\sum_{j=k+1}^{N-1} w_i d_i^2 + \mathcal{D}\sum_{j=1}^{N-1} w_i - 2D\sum_{j=k+1}^{N-1} w_i d_i, \\ &= A\left(\mathcal{D} - \sum_{j=0}^{k} w_j d_j^2\right) + \mathcal{D}\left(A - \sum_{j=0}^{k} w_j\right) - 2D\left(D - \sum_{j=0}^{k} w_j d_j\right)\end{aligned}$$

where we omitted the subscript $N-1$ for notational convenience.

Thus, we conclude that we can in fact compute $\frac{\partial \mathcal{L}}{\partial \alpha_k}$ on the fly in *front-to-back ordering*, where we prepare for the next iteration by repeated subtraction starting from $A_{N-1}, D_{N-1}, \mathcal{D}_{N-1}$.

C.2 Extent Loss Derivation

The extent of a Gaussian along a view ray can be defined as

$$\epsilon_i = \frac{\sqrt{B_i^2 - 4A_i\left(C_i - 2\ln\left(255 o_i\right)\right)}}{2A_i}, \tag{44}$$

which is easily obtained by letting $\alpha_i = \frac{1}{255}$ and solving for t. However, instead of mapping both t^* and $t^* - \epsilon_i$ to NDC and compute the distance there, we linearize the NDC-mapping:

$$\text{NDC}(t^* - \epsilon_i) \approx \text{NDC}(t^*) - \text{NDC}'(t^*)(t^* - (t^* - \epsilon_i)). \tag{45}$$

Inserting $\frac{\partial \text{NDC}}{\partial z}$ and reformulating, we obtain

$$\text{NDC}(t^*) - \text{NDC}(t^* - \epsilon_i) \approx \frac{\tau_f \tau_n}{(\tau_f - \tau_n)}\frac{\epsilon_i}{(t^*)^2} \tag{46}$$

$$= \frac{\tau_f \tau_n}{(\tau_f - \tau_n)}\frac{2A_i\sqrt{B_i^2 - 4A_i\left(C_i - \mathcal{E}_i^2\right)}}{B_i^2}. \tag{47}$$

We now define our extent loss as the alpha-blended sum of individual NDC distances:

$$\mathcal{L}_{\text{ext}} = \frac{\tau_f \tau_n}{\left(\tau_f - \tau_n\right)}\sum_{i=0}^{N-1} w_i \frac{2A_i\sqrt{B_i^2 - 4A_i\left(C_i - \mathcal{E}_i^2\right)}}{B_i^2}. \tag{48}$$

Computing the gradients for this term is straightforward.

Table A.7. **Full per-scene results** for Novel View Synthesis for the Mip-NeRF 360 dataset [Barron et al. 2021].

Method	bicycle	bonsai	counter	flowers	garden	stump	treehill	kitchen	room	Average
					PSNR↑					
3DGS [Kerbl et al. 2023]	25.183	32.102	28.982	21.478	27.240	26.620	22.452	31.352	31.494	27.434
Mip-Splatting [Yu et al. 2024a]	25.320	32.134	29.003	21.635	27.484	26.581	22.583	31.343	31.779	27.540
StopThePop [Radl et al. 2024b]	25.206	31.904	28.703	21.454	27.174	26.677	22.472	31.230	30.844	27.296
Taming-3DGS [Mallick et al. 2024]	25.474	32.474	29.057	21.869	27.756	27.052	22.906	31.762	32.087	27.826
3DGS-MCMC [Kheradmand et al. 2024]	25.686	32.654	29.376	22.014	27.867	27.358	22.944	32.089	32.252	28.027
2DGS [Huang et al. 2024b]	24.739	30.723	28.115	21.135	26.702	26.165	22.355	30.325	31.246	26.834
GOF [Yu et al. 2024b]	25.467	31.598	28.681	21.644	27.426	26.989	22.406	30.787	30.812	27.312
Ours	25.457	31.246	28.419	21.780	27.245	26.925	22.508	30.502	30.345	27.159
Ours (MCMC)	25.232	31.388	28.585	21.622	27.182	26.600	22.843	30.492	31.174	27.235
Method	bicycle	bonsai	counter	flowers	garden	stump	treehill	kitchen	room	Average
					SSIM↑					
3DGS [Kerbl et al. 2023]	0.763	0.939	0.906	0.603	0.862	0.772	0.632	0.925	0.917	0.813
Mip-Splatting [Yu et al. 2024a]	0.768	0.942	0.909	0.608	0.869	0.773	0.638	0.928	0.920	0.817
StopThePop [Radl et al. 2024b]	0.767	0.939	0.904	0.603	0.862	0.775	0.635	0.925	0.917	0.814
Taming-3DGS [Mallick et al. 2024]	0.780	0.944	0.910	0.614	0.873	0.788	0.646	0.931	0.924	0.823
3DGS-MCMC [Kheradmand et al. 2024]	0.799	0.948	0.917	0.645	0.878	0.811	0.659	0.934	0.930	0.836
2DGS [Huang et al. 2024b]	0.733	0.907	0.893	0.576	0.842	0.757	0.616	0.916	0.931	0.797
GOF [Yu et al. 2024b]	0.787	0.937	0.902	0.637	0.868	0.794	0.641	0.917	0.916	0.822
Ours	0.786	0.936	0.900	0.638	0.866	0.791	0.645	0.914	0.915	0.821
Ours (MCMC)	0.779	0.940	0.905	0.625	0.863	0.781	0.643	0.917	0.922	0.819
Method	bicycle	bonsai	counter	flowers	garden	stump	treehill	kitchen	room	Average
					LPIPS↓					
3DGS [Kerbl et al. 2023]	0.213	0.206	0.202	0.338	0.109	0.216	0.327	0.127	0.221	0.218
Mip-Splatting [Yu et al. 2024a]	0.212	0.204	0.200	0.339	0.108	0.216	0.326	0.126	0.218	0.216
StopThePop [Radl et al. 2024b]	0.205	0.203	0.199	0.335	0.107	0.210	0.319	0.126	0.216	0.213
Taming-3DGS [Mallick et al. 2024]	0.192	0.201	0.198	0.332	0.100	0.196	0.313	0.122	0.210	0.207
3DGS-MCMC [Kheradmand et al. 2024]	0.168	0.191	0.185	0.284	0.094	0.171	0.272	0.121	0.198	0.187
2DGS [Huang et al. 2024b]	0.269	0.243	0.230	0.373	0.147	0.258	0.377	0.147	0.227	0.252
GOF [Yu et al. 2024b]	0.180	0.198	0.204	0.279	0.107	0.195	0.279	0.137	0.217	0.200
Ours	0.183	0.195	0.202	0.276	0.107	0.197	0.276	0.139	0.214	0.199
Ours (MCMC)	0.182	0.195	0.196	0.295	0.111	0.200	0.282	0.141	0.204	0.201
Method	bicycle	bonsai	counter	flowers	garden	stump	treehill	kitchen	room	Average
					ꟻLIP ↓					
3DGS [Kerbl et al. 2023]	0.158	0.082	0.105	0.225	0.118	0.150	0.186	0.096	0.093	0.135
Mip-Splatting [Yu et al. 2024a]	0.160	0.083	0.104	0.225	0.119	0.152	0.184	0.096	0.092	0.135
StopThePop [Radl et al. 2024b]	0.160	0.085	0.110	0.225	0.120	0.149	0.183	0.099	0.101	0.137
Taming-3DGS [Mallick et al. 2024]	0.157	0.079	0.103	0.214	0.113	0.145	0.186	0.091	0.089	0.131
3DGS-MCMC [Kheradmand et al. 2024]	0.156	0.079	0.101	0.209	0.115	0.142	0.176	0.093	0.090	0.129
2DGS [Huang et al. 2024b]	0.167	0.184	0.101	0.236	0.127	0.159	0.114	0.106	0.090	0.143
GOF [Yu et al. 2024b]	0.157	0.086	0.109	0.217	0.121	0.147	0.184	0.104	0.102	0.136
Ours	0.157	0.091	0.111	0.214	0.124	0.146	0.183	0.107	0.106	0.138
Ours (MCMC)	0.161	0.092	0.110	0.219	0.125	0.151	0.181	0.107	0.101	0.139

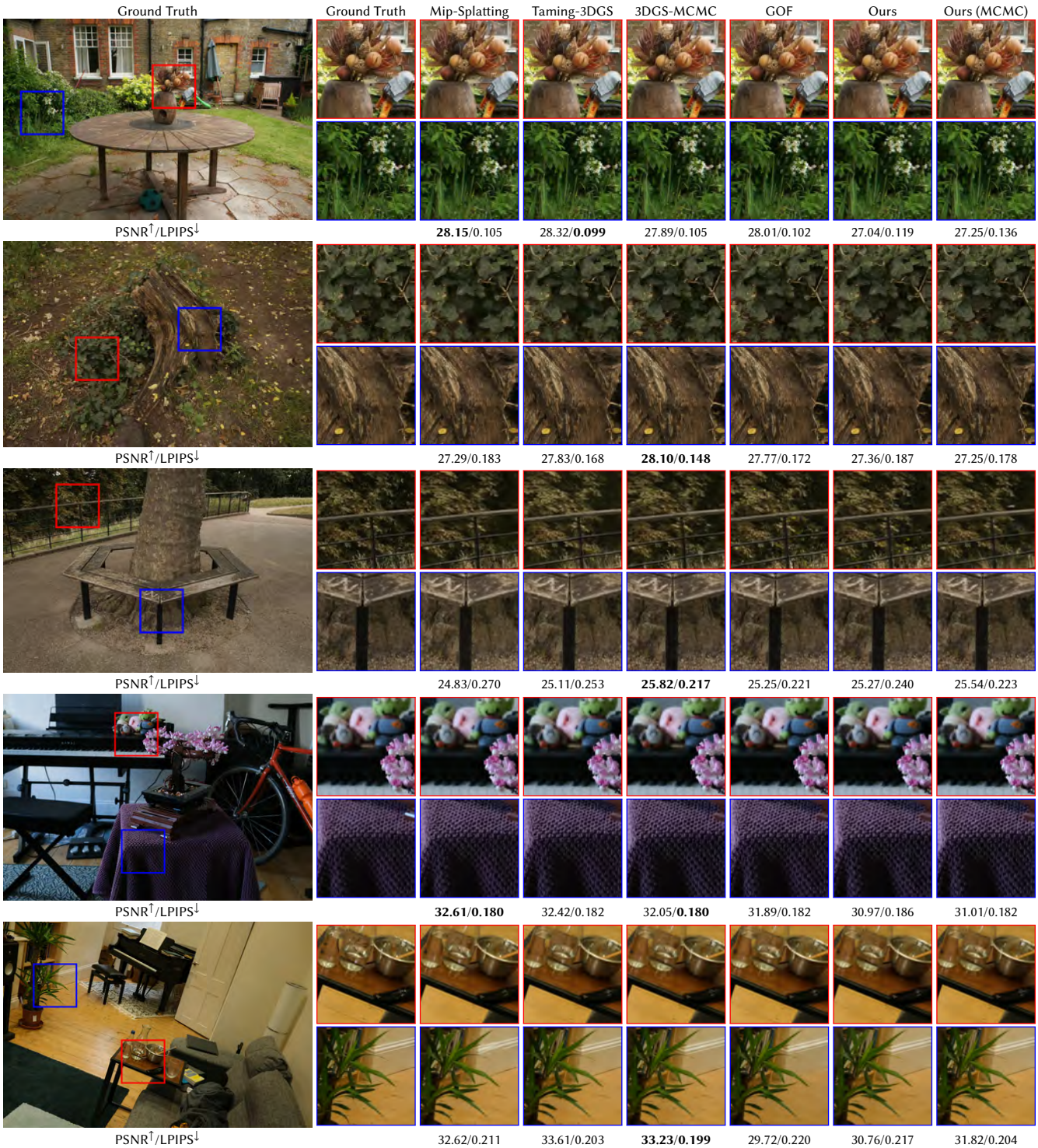

Fig. A.1. **Qualitative Comparison for Novel View Synthesis**. We show small zoom-ins of various scenes from the Mip-NeRF 360 dataset [Barron et al. 2022]. PSNR/LPIPS scores for this view are inset. Overall, 3DGS-MCMC [Kheradmand et al. 2024] achieves the best results, while our results are on par with GOF.

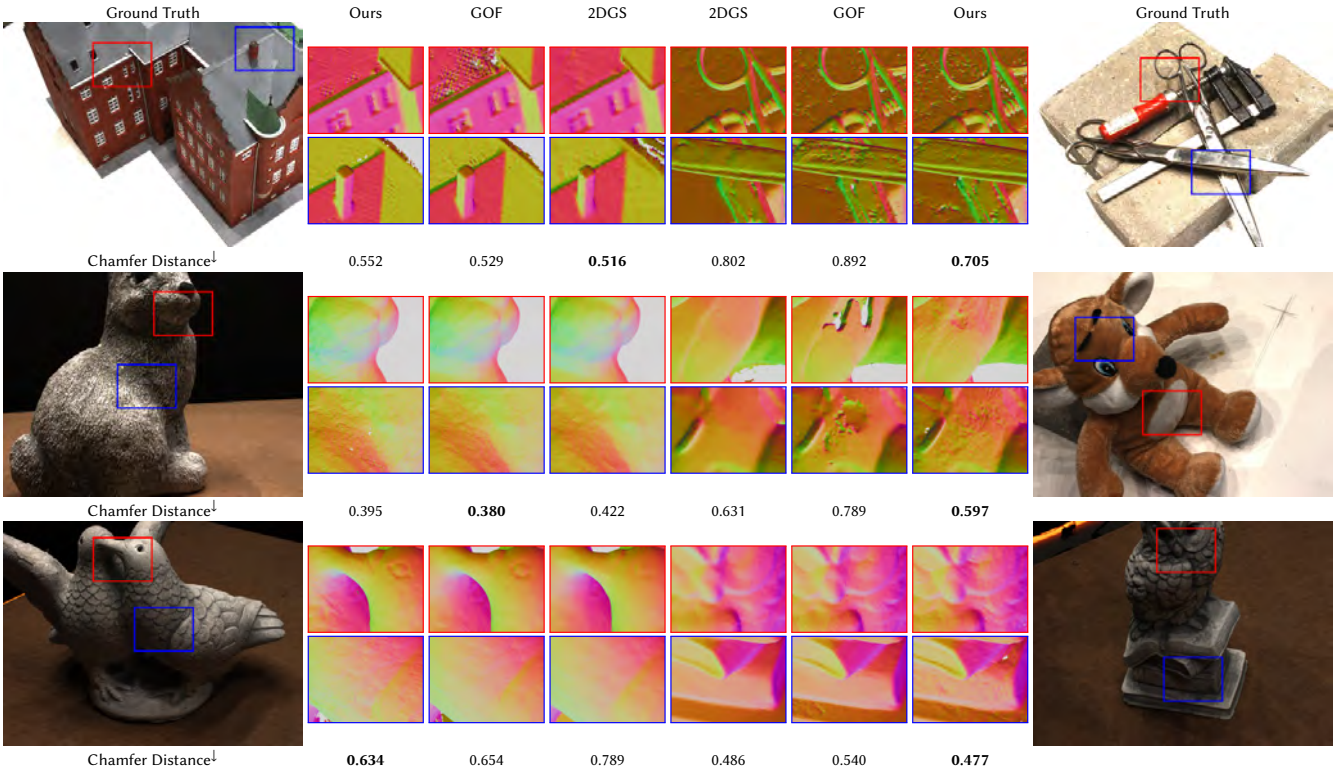

Fig. A.2. **Qualitative Comparison for Bounded Meshes**: We compare our bounded meshes with those from GOF [Yu et al. 2024b] and 2DGS [Huang et al. 2024b]. Compared to GOF, our meshes have fewer artifacts, while meshes extracted using 2DGS are overly smooth and lack intricate details. However, all methods still exhibit holes and cavities due to the limitations of current view-dependent appearance.

Table A.8. **Ablation study for bounded meshes on the DTU dataset** [Jensen et al. 2014]. Our method achieves better results compared to both 2DGS [Huang et al. 2024b] and GOF [Yu et al. 2024b]. Adding the losses designed for unbounded mesh extraction reduces the overall surface reconstruction quality.

Method	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Avg
GOF	0.529	0.892	0.434	0.380	1.332	0.866	0.770	1.284	1.289	0.789	0.767	1.148	0.457	0.696	0.540	0.812
2DGS	0.516	0.802	0.344	0.422	0.972	0.890	0.823	1.235	1.246	0.631	0.654	1.970	0.416	0.690	0.486	0.806
Ours	0.552	0.705	0.408	0.395	1.120	0.735	0.657	1.112	1.467	0.597	0.634	1.046	0.581	0.615	0.477	0.740
(A) w/o Exact Depth	0.585	0.882	0.469	0.423	1.269	0.973	0.858	1.188	1.365	1.062	0.686	1.272	0.718	1.009	0.546	0.887
(B) w/ $\mathcal{L}_{\text{smooth}}$	0.585	0.716	0.410	0.399	1.310	0.774	0.795	1.176	1.253	0.627	0.837	1.188	0.635	0.667	0.553	0.795
(C) w/ $\mathcal{L}_{\text{opa}}$	0.579	0.749	0.449	0.414	1.178	0.669	0.769	1.151	1.271	0.643	0.678	1.335	0.631	0.683	0.531	0.782
(D) w/ $\mathcal{L}_{\text{extent}}$	0.590	0.773	0.486	0.381	1.253	0.779	0.766	1.190	1.325	0.755	0.654	1.227	0.535	0.667	0.524	0.794
(E) w/ Attached Grad	0.646	0.675	0.838	0.279	1.429	0.555	0.619	0.878	1.659	0.517	0.437	1.160	1.545	0.428	0.422	0.806
(F) w/ $F = 1000$	0.557	0.738	0.400	0.393	1.263	0.712	0.742	1.188	1.298	0.810	0.706	1.311	0.549	0.698	0.579	0.796
(G) w/ PGSR Appearance	0.585	0.854	0.577	0.413	1.300	0.966	0.799	1.313	1.556	1.238	0.750	1.333	0.863	1.097	0.532	0.945