Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes

ZEHAO YU, University of Tübingen, Tübingen AI Center, Germany
TORSTEN SATTLER, Czech Technical University in Prague, Czech Republic
ANDREAS GEIGER, University of Tübingen, Tübingen AI Center, Germany

https://niujinshuchong.github.io/gaussian-opacity-fields

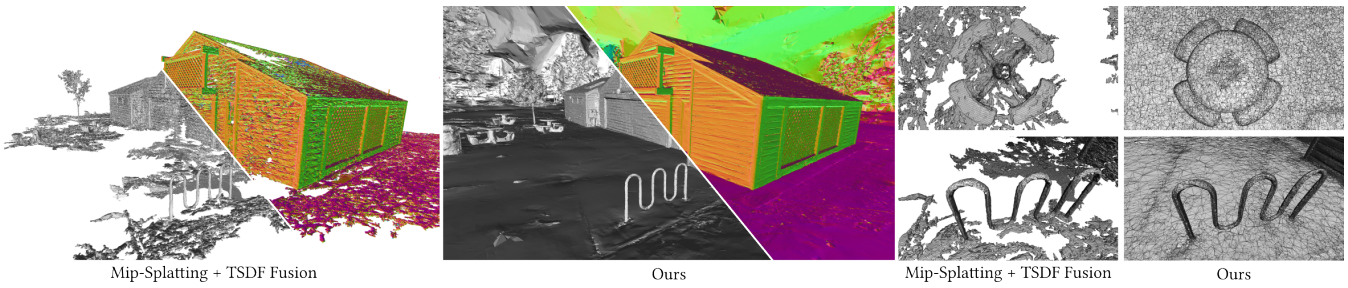


Fig. 1. Applying TSDF fusion with rendered depth maps from the state-of-the-art Mip-Splatting [Yu et al. 2024a] models results in noisy and incomplete meshes, while meshes extracted with our method are complete, smooth, and detailed. This is achieved by establishing Gaussian opacity fields from 3D Gaussians, which enables geometry extraction by directly identifying its level-set. Moreover, we generate tetrahedral meshes from 3D Gaussians and utilize Marching Tetrahedra to extract adaptive and compact meshes.

Recently, 3D Gaussian Splatting (3DGS) has demonstrated impressive novel view synthesis results, while allowing the rendering of high-resolution images in real-time. However, leveraging 3D Gaussians for surface reconstruction poses significant challenges due to the explicit and disconnected nature of 3D Gaussians. In this work, we present Gaussian Opacity Fields (GOF), a novel approach for efficient, high-quality, and adaptive surface reconstruction in unbounded scenes. Our GOF is derived from ray-tracing-based volume rendering of 3D Gaussians, enabling direct geometry extraction from 3D Gaussians by identifying its levelset, without resorting to Poisson reconstruction or TSDF fusion as in previous work. We approximate the surface normal of Gaussians as the normal of the ray-Gaussian intersection plane, enabling the application of regularization that significantly enhances geometry. Furthermore, we develop an efficient geometry extraction method utilizing Marching Tetrahedra, where the tetrahedral grids are induced from 3D Gaussians and thus adapt to the scene's complexity. Our evaluations reveal that GOF surpasses existing 3DGS-based methods in surface reconstruction and novel view synthesis. Further, it compares favorably to or even outperforms, neural implicit methods in both quality and speed.

CCS Concepts: • **Computing methodologies** → *Reconstruction*; **Rendering**; *Machine learning approaches*.

Additional Key Words and Phrases: Novel View Synthesis, Differentiable Rendering, Gaussian Splatting, Surface Reconstruction, Multi-view-to-3D

Authors' addresses: Zehao Yu, University of Tübingen, Tübingen AI Center, Germany; Torsten Sattler, Czech Technical University in Prague, Czech Republic; Andreas Geiger, University of Tübingen, Tübingen AI Center, Germany.

ACM 0730-0301/2024/12-ART
https://doi.org/10.1145/3687937

ACM Reference Format:
Zehao Yu, Torsten Sattler, and Andreas Geiger. 2024. Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. *ACM Trans. Graph.* 43, 6 (December 2024), 15 pages. https://doi.org/10.1145/3687937

1 INTRODUCTION

3D Reconstruction from multi-view images has been a long-standing goal in computer vision, with various applications in robotics, graphics, animation, virtual reality, and more. Since Neural Radiance Field (NeRF) [Mildenhall et al. 2020] demonstrated impressive novel view synthesis (NVS) results with implicit representations [Mescheder et al. 2019; Park et al. 2019] and volume rendering [Drebin et al. 1988; Kajiya and Von Herzen 1984; Levoy 1990], it has been extended to surface reconstruction with occupancy networks [Oechsle et al. 2021] and Signed Distance Functions (SDF) [Wang et al. 2021; Yariv et al. 2021]. While recent advancements [Li et al. 2023; Yariv et al. 2023; Yu et al. 2022a,b] have shown impressive reconstruction results, these methods are mostly limited to reconstructing foreground objects [Rosu and Behnke 2023] and computationally expensive to optimize [Yariv et al. 2023]. For instance, Neuralangelo [Li et al. 2023] models the background separately with NeRFs and necessitates approximately 128 GPU hours to reconstruct a single scene.

Another research avenue focuses on directly extracting surfaces from NeRF's opacity field for real-time rendering [Chen et al. 2023a; Rakotosaona et al. 2024; Reiser et al. 2024; Tang et al. 2022]. These methods employ a modular workflow including opacity field training, mesh extraction, simplification, and refinement. Particularly noteworthy is Binary Opacity Grids (BOG)[Reiser et al. 2024], which excels at capturing intricate details in unbounded scenes through super-sampling. To extract detailed surfaces, it renders depth maps

to generate a sparse high-resolution voxel grid and applies a heuristic fusion technique to label the voxels as inside or outside. Marching cube algorithm [Lorensen and Cline 1998] is applied to extract a high-resolution mesh with hundreds of millions of points and thousands of millions of triangles, which is then simplified using slow post-processing techniques [Garland and Heckbert 1997]. Notably, as it focuses on NVS rather than surface reconstruction, the extracted meshes are noisy and contain fewer details in the background region, probably due to lack of regularization and contracted space [Barron et al. 2022a] fusion.

More recently, 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] represents complex scenes as a set of 3D Gaussians, demonstrating photorealistic NVS results while trained efficiently and rendered in real-time. It has been quickly extended to surface reconstruction [Chen et al. 2023b; Guédon and Lepetit 2023; Huang et al. 2024; Yu et al. 2024b]. Notably, SuGaR [Guédon and Lepetit 2023] regularizes the 3D Gaussians to align with surfaces and employs Poisson surface reconstruction [Kazhdan and Hoppe 2013] to extract a mesh from rendered depth maps. 2D Gaussian Splatting (2DGS) [Huang et al. 2024] uses 2D Gaussians instead of 3D Gaussians as a scene representation for better surface representation and utilizes TSDF fusion to reconstruct a mesh. While these methods have shown improved reconstruction, they struggle with extracting fine-grained geometry [Guédon and Lepetit 2023] and reconstructing background regions [Huang et al. 2024]. A primary challenge is the inconsistency between mesh extraction and volume rendering during training. Specifically, Poisson reconstruction ignores the opacity and scale of Gaussian primitives and rendered depth maps are not sufficiently reliable. Moreover, TSDF fusion struggles to accurately model thin structures and to reconstruct unbounded scenes. Resorting to high-resolution voxel grids for TSDF leads to the creation of large meshes, similar to BOG [Reiser et al. 2024], due to the lack of adaptivity in the grid resolution relative to the scene's geometric complexity.

Contributions: In this paper, we propose *Gaussian Opacity Fields* (GOF), a novel approach to achieve efficient, high-quality, and adaptive surface reconstruction from 3D Gaussians directly. Our key insights are threefold: **First**, we establish a Gaussian opacity field from a set of 3D Gaussians. Specifically, unlike projection-based volume rendering, our method leverages an explicit ray-Gaussian intersection to determine a Gaussian's contribution during volume rendering. Our ray-tracing-inspired formula facilitates the evaluation of opacity values for any point along a ray. We then define the opacity of any 3D point as the minimal opacity among all training views that observed the point. Taking the minimum opacity across all views achieves view independence, making the opacity field solely a function of position. Our GOF is consistent with volume rendering during training and enables surface extraction from 3D Gaussians by directly identifying a level set, without resorting to Poisson reconstruction or TSDF fusion. **Second**, we approximate the surface normals of 3D Gaussians as the normals of intersection planes between the ray and Gaussians. This technique allows for the incorporation of regularizations [Huang et al. 2024] during training, thus enhancing the fidelity of geometry reconstruction. **Third**, we propose an efficient surface extraction technique based on tetrahedra-grids. Recognizing that 3D Gaussians effectively indicate potential surface locations, we focus opacity evaluations on these areas. In particular, we use the center and corners of 3D bounding boxes around the 3D Gaussian primitives as vertex sets for the tetrahedral mesh. Upon assessing the opacity at tetrahedral points, we utilize the Marching Tetrahedra algorithm for triangle mesh extraction. Given that our opacity fields challenge the assumption that opacity changes linearly, we further implement a binary search algorithm to accurately identify the opacity field's level set, substantially enhancing the quality of the resulting surfaces.

To demonstrate the effectiveness and efficiency of GOF, we carry out extensive experiments across three challenging datasets [Barron et al. 2022a; Jensen et al. 2014; Knapitsch et al. 2017]. Our results indicate that GOF not only matches but, in some cases, surpasses the performance of existing SDF-based methods, while being much faster. Moreover, GOF outperforms all other 3DGS-based methods in both surface reconstruction and novel view synthesis.

2 RELATED WORK

2.1 Novel view synthesis

NeRF [Mildenhall et al. 2020] utilizes a multi-layer perception (MLP) for scene representation, including geometry and view-dependent appearances. The MLP is optimized via a photometric loss through volume rendering [Drebin et al. 1988; Kajiya and Von Herzen 1984; Levoy 1990; Max 1995]. Subsequent enhancements have focused on optimizing NeRF's training using feature-grid representations [Chen et al. 2022; Fridovich-Keil et al. 2022; Kulhanek and Sattler 2023; Müller et al. 2022; Sun et al. 2022] and improving rendering speed via baking [Hedman et al. 2021; Reiser et al. 2021, 2023; Yariv et al. 2023]. Moreover, NeRF has been adapted to address challenges in anti-aliasing [Barron et al. 2022b, 2023] and unbounded scene modeling [Barron et al. 2022a; Zhang et al. 2020]. More recently, 3D Gaussian splatting [Kerbl et al. 2023] represents complex scenes with 3D Gaussians. It demonstrated impressive NVS results while being optimized efficiently and rendering high-resolution images in real-time. Subsequent works improved its rendering quality via anti-aliasing [Yu et al. 2024a] or extended it to dynamic scenes modeling [Zhou et al. 2024], and more [Chen and Wang 2024]. In this work, we extend 3DGS for high-quality surface reconstruction through the development of Gaussian Opacity Fields. We further introduce an efficient tetrahedron grid-based mesh extraction algorithm to extract scene adaptive and compact meshes.

2.2 3D reconstruction

3D Reconstruction from multi-view images is a fundamental problem in computer vision. Multi-view stereo methods [Schönberger et al. 2016; Yao et al. 2018; Yu and Gao 2020] often employ complex multi-stage pipelines that include feature matching, depth estimation, point clouds fusion, and ultimately, surface reconstruction from aggregated point clouds [Kazhdan and Hoppe 2013]. In contrast, neural implicit methods [Oechsle et al. 2021; Wang et al. 2021; Yariv et al. 2021] significantly simplify the pipeline by optimizing an implicit surface representation via volume rendering. After optimization, triangle meshes can be extracted easily with Marching Cubes [Lorensen and Cline 1998] at any resolution. Notable

advancements have been made through the adoption of more expressive scene representations [Li et al. 2023], advanced training strategies [Li et al. 2023], and the integration of monocular priors [Yu et al. 2022b]. Despite these advances, these methods are mostly limited in reconstructing foreground objects [Rosu and Behnke 2023] and are computationally expensive to optimize [Li et al. 2023; Yariv et al. 2023]. Furthermore, the use of high-resolution grids for capturing fine details often results in excessively large meshes. By contrast, we establish Gaussian Opacity Fields using 3DGS [Kerbl et al. 2023], which facilitates fast training. We utilize Marching Tetrahedra [Doi and Koide 1991; Shen et al. 2021] to extract adaptive, compact, and detailed meshes in unbounded scenes.

2.3 Surface Reconstruction with Gaussians

Inspired by the impressive NVS performance of 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023], researchers have proposed to utilize 3D Gaussians for surface reconstruction. Recent efforts [Chen et al. 2023b; Yu et al. 2024b] have integrated 3D Gaussians with neural implicit surfaces, optimizing a Signed Distance Function (SDF) network and 3D Gaussians jointly. While these approaches mark some advancements, they also inherit the shortcomings associated with implicit surfaces as previously discussed. Other studies have focused on surface extraction from optimized Gaussian primitives through post-processing techniques [Dai et al. 2024; Guédon and Lepetit 2023; Huang et al. 2024; Turkulainen et al. 2024]. Notably, SuGaR [Guédon and Lepetit 2023] and GaussianSurfels [Dai et al. 2024] adopt Poisson surface reconstruction [Kazhdan and Hoppe 2013] to extract meshes from rendered depth maps. Meanwhile, 2D Gaussian Splatting (2DGS) [Huang et al. 2024] employs TSDF fusion for this purpose. Though these methods achieve improved reconstructions, they face challenges in capturing fine-grained geometry and adequately reconstructing background areas. In this work, we derive Gaussian Opacity Fields *directly* from 3D Gaussians. Our GOF is consistent with the volume rendering process for rendering RGB images. It enables direct surface extraction by identifying a level set, without resorting to Poisson reconstruction and TSDF fusion. Additionally, we propose a tetrahedron grid-based technique for mesh extraction, resulting in adaptive and detailed meshes.

3 METHOD

Given multiple posed and calibrated images, our goal is to reconstruct the 3D scene efficiently while allowing detailed and compact surface extraction and photorealistic novel view synthesis. To this end, we first construct Gaussian Opacity Fields (GOF) from 3D Gaussians, enabling geometry extraction directly by identifying a level set, eliminating the need for Poisson reconstruction or TSDF fusion. Next, we extend two effective regularizers from 2DGS [Huang et al. 2024] to our 3D Gaussians, improving reconstruction quality. Finally, we propose a novel tetrahedra-based method to extract detailed and compact meshes from GOFs with marching tetrahedra.

3.1 Modeling

Similar to prior works [Kerbl et al. 2023; Zwicker et al. 2001], we represent the scene with a set of 3D Gaussian primitives $\{\mathcal{G}_k | k = 1, \cdots, K\}$. The geometry of each 3D Gaussian $\mathcal{G}_k$ is parameterized by center $\mathbf{p}_k \in \mathbb{R}^3$, scaling matrix $\mathbf{S}_k \in \mathbb{R}^{3\times3}$, and rotation $\mathbf{R}_k \in \mathbb{R}^{3\times3}$ parameterized by a quaternion:

$$\mathcal{G}_k(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p}_k)^T\Sigma_k^{-1}(\mathbf{x}-\mathbf{p}_k)} \tag{1}$$

where $\Sigma_k \in \mathbb{R}^{3\times3}$ is a covariance matrix defined as $\Sigma_k = \mathbf{R}_k\mathbf{S}_k\mathbf{S}_k^T\mathbf{R}_k^T$.

Ray Gaussian Intersection: Instead of projecting 3D Gaussians to 2D screen space and evaluating the Gaussian in 2D, we evaluate the contribution of a Gaussian to a ray with explicit ray-Gaussian intersection [Gao et al. 2023; Keselman and Hebert 2022]. As we will see in Section 3.2, this crucially enables evaluation of opacity values of *arbitrary* 3D points in contrast to the projection-based Gaussian splatting mechanism of [Kerbl et al. 2023] where 3D information is lost during the 3D-to-2D projection step. The ray-Gaussian intersection is defined as the point where the Gaussian reaches its maximum value along the ray. Specifically, given a camera center at $\mathbf{o} \in \mathbb{R}^3$ and a ray direction $\mathbf{r} \in \mathbb{R}^3$, any point $\mathbf{x} \in \mathbb{R}^3$ along the ray can be written as $\mathbf{x} = \mathbf{o} + t\mathbf{r}$, where t is depth of the ray. We first transform the point $\mathbf{x}$ to the local coordinate system of the 3D Gaussian and normalize the point by its scale:

$$\mathbf{o}_g = \mathbf{S}_k^{-1}\mathbf{R}_k(\mathbf{o} - \mathbf{p}_k) \tag{2}$$

$$\mathbf{r}_g = \mathbf{S}_k^{-1}\mathbf{R}_k\mathbf{r} \tag{3}$$

$$\mathbf{x}_g = \mathbf{o}_g + t\mathbf{r}_g \tag{4}$$

In this local coordinate system, the Gaussian value at any point along the ray becomes a 1D Gaussian, which is described by:

$$\mathcal{G}_k^{1D}(t) = e^{-\frac{1}{2}\mathbf{x}_g^T\mathbf{x}_g} = e^{-\frac{1}{2}(\mathbf{r}_g^T\mathbf{r}_g t^2 + 2\mathbf{o}_g^T\mathbf{r}_g t + \mathbf{o}_g^T\mathbf{o}_g)} \tag{5}$$

The maximum of Eq. 5 is achieved when the quadratic term reaches maximum. Therefore, the closed-form solution for Eq. 5 is

$$t^* = -\frac{B}{A} \tag{6}$$

where $A = \mathbf{r}_g^T\mathbf{r}_g$ and $B = \mathbf{o}_g^T\mathbf{r}_g$. Note that our formula is equivalent to the one presented in [Keselman and Hebert 2022], where the ray-Gaussian intersection is computed directly in world space. However, using the normalized Gaussian coordinate by transforming the ray offers a clearer geometric interpretation and simplifies defining the normal of the intersection plane, as we will show in Sec. 3.3.

Now, we define the contribution $\mathcal{E}$ of a Gaussian $\mathcal{G}_k$ for a given camera center $\mathbf{o}$ and ray direction $\mathbf{r}$ as:

$$\mathcal{E}(\mathcal{G}_k, \mathbf{o}, \mathbf{r}) = \mathcal{G}_k^{1D}(t^*) \tag{7}$$

Volume Rendering: Similar to 3DGS [Kerbl et al. 2023], the color of a camera ray is rendered via alpha blending according to the primitive's depth order $1, \ldots, K$:

$$\mathbf{c}(\mathbf{o}, \mathbf{r}) = \sum_{k=1}^{K} \mathbf{c}_k\, \alpha_k\, \mathcal{E}(\mathcal{G}_k, \mathbf{o}, \mathbf{r}) \prod_{j=1}^{k-1}(1 - \alpha_j\, \mathcal{E}(\mathcal{G}_j, \mathbf{o}, \mathbf{r})) \tag{8}$$

where $\mathbf{c}_k$ is the view-dependent color, modeled with spherical harmonics, and $\alpha_k \in [0, 1]$ is an additional parameter that influences the opacity of Gaussian k. To efficiently render an image, we utilize the same tile-based rendering process as in 3DGS [Kerbl et al. 2023].

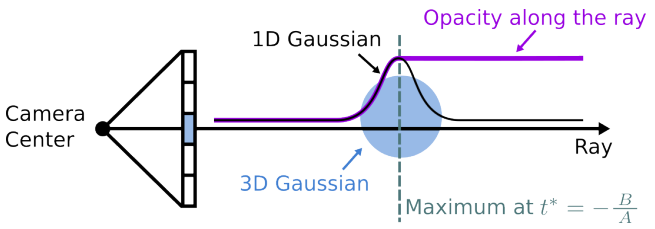


Fig. 2. **Illustration of ray tracing volume rendering**. Evaluating a 3D Gaussian along a ray results in a 1D Gaussian, which has a closed-form solution for when it reaches maximal value. We define the *opacity* along the ray as monotonously increasing until it reaches the maximal value and constant afterward.

3.2 Gaussian Opacity Fields

One significant benefit of using explicit ray-Gaussian intersection instead of projection is that it allows evaluating the opacity value or transmittance of *any* 3D point $\mathbf{x}$ along the ray. Let's first consider the case when there is only a single Gaussian $\mathcal{G}_k$ along the ray. In this case, we define the opacity of any 3D point along the ray as

$$\mathbf{O}_k(\mathcal{G}_k, \mathbf{o}, \mathbf{r}, t) = \begin{cases} \mathcal{G}_k^{1D}(t) & \text{if } t \le t^* \\ \mathcal{G}_k^{1D}(t^*) & \text{if } t > t^* \end{cases}$$

where $\mathbf{x} = \mathbf{o} + t\mathbf{r}$. Intuitively, the opacity (reverse of transmittance) increases until it reaches its maximal value, and remains constant afterwards as illustrated in Figure 2. Therefore, the opacity at any point along a ray given a set of Gaussians can be defined similar to the volume rendering process in Eq. 8 as:

$$\mathbf{O}(\mathbf{o}, \mathbf{r}, t) = \sum_{k=1}^{K} \alpha_k \, \mathbf{O}_k(\mathcal{G}_k, \mathbf{o}, \mathbf{r}, t) \prod_{j=1}^{k-1} (1 - \alpha_j \, \mathbf{O}_j(\mathcal{G}_j, \mathbf{o}, \mathbf{r}, t)) \tag{9}$$

As a 3D point might be visible by *any* training view, we define the opacity of a 3D point $\mathbf{x}$ as the minimal opacity value among all training views or viewing directions:

$$\mathbf{O}(\mathbf{x}) = \min_{(\mathbf{o},\mathbf{r})} \mathbf{O}(\mathbf{o}, \mathbf{r}, t) \tag{10}$$

We refer to $\mathbf{O}(\mathbf{x})$ as the *Gaussian Opacity Field* (GOF) since it is an opacity field derived from 3D Gaussians. Our GOF shares similarities to the visual hull [Laurentini 1994] or space carving [Kutulakos and Seitz 2000]. But instead of using a silhouette where the opacity value of a ray (all points on the ray) is either 1 or 0, GOF uses volume rendering to evaluate opacity for each point from 3D Gaussians.

Our GOF is consistent with the volume rendering process for RGB rendering during training. With GOF, we can extract surfaces directly by identifying their level sets, similar to UNISURF [Oechsle et al. 2021], without resorting to Poisson reconstruction [Guédon and Lepetit 2023] or TSDF fusion [Huang et al. 2024]. We will discuss our method for efficient and adaptive mesh extraction in Section 3.4.

We also note that GOFs are a general formula as long as the scene representation is a set of 3D Gaussians. For example, we can use it to extract a mesh from a pre-trained 3DGS [Kerbl et al. 2023] or Mip-Splatting [Yu et al. 2024a] model, where projection-based volume rendering is used for training, as we will show in the experiments.

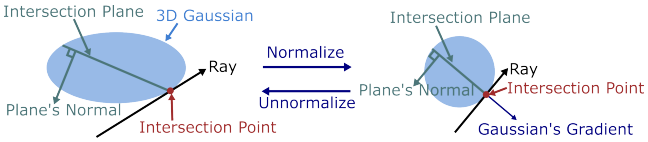


Fig. 3. **Definition of Gaussian's normal**. We approximate the Gaussian's normal as the normal of the ray-Gaussian intersection plane. First, we transform the ray into the Gaussian coordinate system and normalize it using the Gaussian's scales. In this normalized coordinate system, the ray-Gaussian intersection plane is perpendicular to the ray, making the normal the inverse of the ray direction. Finally, we transform the intersection plane back to world space by reversing the normalization process.

3.3 Optimization

Optimizing 3D Gaussians with pure photometric loss leads to noisy results as 3D reconstruction from multi-views is an underconstrained problem [Barron et al. 2022a; Zhang et al. 2020]. Therefore, we extend the regularization terms in 2DGS [Huang et al. 2024] to optimize our 3D Gaussians, including a depth distortion loss and a normal consistency loss.

Depth Distortion: We apply the depth distortion loss [Huang et al. 2024], which is originally proposed in Mip-NeRF 360 [Barron et al. 2022a], to the ray-Gaussian intersection to concentrate Gaussians along the ray:

$$\mathcal{L}_d = \sum_{i,j} \omega_i \omega_j |t_i - t_j| \tag{11}$$

where i, j index over Gaussians contributed to the ray and $\omega_i = \alpha_k \, \mathcal{E}(\mathcal{G}_k, \mathbf{o}, \mathbf{r}) \prod_{j=1}^{k-1} (1 - \alpha_j \, \mathcal{E}(\mathcal{G}_j, \mathbf{o}, \mathbf{r}))$ is the blending weight of the $i-$th Gaussian and t_i is the depth of the intersection point in Eq. 6. However, the distortion loss minimizes both the distance between Gaussians and the weights of each Gaussian whereas minimizing the weights of the Gaussians could lead to an increase in alpha values for Gaussians that are blended first, which results in exaggerated Gaussians, resulting in floaters. Therefore, we detach the gradient of weights ω_i and only minimize the distance between Gaussians.

Normal Consistency: A key challenge of applying 2DGS's normal consistency regularization [Huang et al. 2024] to 3D Gaussians is that the gradient of 3D Gaussians always points outwards from the centers. Consider a simple case when we render a single isotropic 3D Gaussian to image space. The rendered result is a 2D Gaussian in the image plane. The gradient of this 2D Gaussian always points outward from the projected 2D center, meaning the rendered normals at two different pixels will differ as long as the directions from the projected 2D Gaussian center to the pixel coordinates vary. Moreover, the normal at the projected 2D center is not well-defined. This ambiguity makes the optimization difficult.

To mitigate this issue, we define the normal of a 3D Gaussian as the normal of the intersection plane given a ray direction. An illustration is shown in Figure 3. Specifically, we begin by transforming the ray into the Gaussian coordinate system and normalizing it using the Gaussian's scales, as described in Eq. 3. In the normalized coordinate system, the ray-Gaussian intersection plane is perpendicular to the ray. Therefore, the normal is the inverse ray direction

Fig. 4. **Comparison of densification strategy on the Mip-NeRF 360 Dataset [Barron et al. 2022a].** Applying our densification to 3DGS [Kerbl et al. 2023] and Mip-Splatting [Yu et al. 2024a] significant improves the NVS results. Please note that the glass regions are blur in both 3DGS and Mip-Splatting, while our method renders the image faithfully.

$-\mathbf{r}_g$. Next, we reverse the normalization to transform the intersection plane back to world space, where the plane's normal is given by $-\mathbf{S}_k^{-1}\mathbf{r}_g$. Finally, we apply the inverse of the world-to-Gaussian rotation to transform the normal back to world space, yielding the normal direction of the intersection plane as $\mathbf{n}_i = -\mathbf{R}_k^T\mathbf{S}_k^{-1}\mathbf{r}_g$, which should then be normalized to ensure it is a valid unit vector. It is important to note that in world space, the intersection plane is not necessarily perpendicular to the ray direction, but the plane's normal is always perpendicular to the intersection plane.

After defining the normal of a Gaussian, we apply the depth-normal consistency regularisation:

$$\mathcal{L}_n = \sum_i \omega_i (1 - \mathbf{n}_i^\top \mathbf{N}) \tag{12}$$

where i indexes over intersected Gaussians along the ray, ω denotes the blending weight, and $\mathbf{N}$ is the normal estimated by the gradient of the depth map [Huang et al. 2024].

Final Loss: Finally, we optimize our model from an initial sparse point cloud using multiple posed images with the following loss:

$$\mathcal{L} = \mathcal{L}_c + \alpha\mathcal{L}_d + \beta\mathcal{L}_n \tag{13}$$

where $\mathcal{L}_c$ is an RGB reconstruction loss combining $\mathcal{L}_1$ with the D-SSIM term from [Kerbl et al. 2023], while $\mathcal{L}_d$ and $\mathcal{L}_n$ are regularization terms. Note that, we utilize the decoupled appearance modeling proposed in VastGaussian [Lin et al. 2024] to model the uneven illumination for the Tanks and Temples dataset [Knapitsch et al. 2017], where a small convolutional neural network is used to predict image-dependent colors such that the model will not fake inconsistent illumination with geometry. While other variants could also be applicable, we use VastGaussian's solution due to its demonstrated improvements in large-scale scenarios and its ease of reimplementation.

Improved Densification: As the optimization starts from a sparse point cloud, it is necessary to increase the number of 3D Gaussians to better reconstruct the scene. We follow the densification strategy in 3DGS [Kerbl et al. 2023]. Specifically, the densification of a Gaussian (either by cloning or splitting) is guided by the magnitude of the view-space position gradient $\frac{dL}{d\mathbf{x}}$, where $\mathbf{x}$ is the center of projected Gaussian. Mathematically, $\frac{dL}{d\mathbf{x}}$ sums over pixels $\mathbf{p}_i$ that the Gaussian contributed to:

$$\frac{dL}{d\mathbf{x}} = \sum_i \frac{dL}{d\mathbf{p}_i}\frac{d\mathbf{p}_i}{d\mathbf{x}} \tag{14}$$

If the norm of the gradient $\|\frac{dL}{d\mathbf{x}}\|_2$ is above a predefined threshold $\tau_\mathbf{x}$, the Gaussian is chosen as the candidate for densification.

However, we found that this metric is not effective in identifying overly blurred areas, due to its inability to distinguish between well-reconstructed regions and those where the gradient signals from different pixels negate each other, leading to minimal overall gradient magnitudes. Therefore, we propose a simple modification to the metric that accumulates the norms of the individual pixel gradients instead:

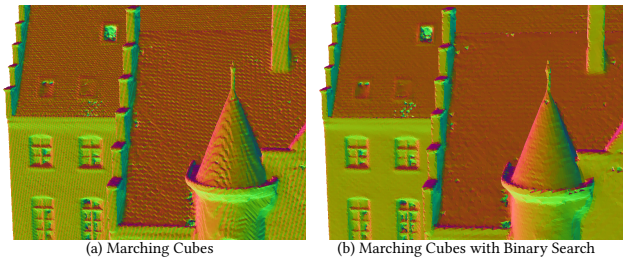
$$M = \sum_i \|\frac{dL}{d\mathbf{p}_i}\frac{d\mathbf{p}_i}{d\mathbf{x}}\| \tag{15}$$

Our metric M better indicates regions with significant reconstruction errors, resulting in better reconstruction and novel view synthesis results, as shown in Figure 4.

3.4 Surface Extraction

Post-training, the conventional step towards surface or triangle mesh extraction involves densely evaluating the opacity values within regions of interest, a technique well-suited to simple scenarios like the DTU Dataset [Jensen et al. 2014], as done in previous work [Oechsle et al. 2021; Wang et al. 2021; Yariv et al. 2021]. For a large-scale unbounded scene, some have adopted dense evaluation in a contracted space [Yariv et al. 2023]. However, dense evaluation for grids incurs substantial computational costs due to the cubic growth of complexity with grid resolution. Unfortunately, capturing fine details necessitates high-resolution grids, leading to significant overhead. Alternative sparse grids may reduce dense evaluation but still result in huge meshes, often comprising hundreds of millions of points and billions of faces. Simplifying such large and complex meshes typically requires a slow post-processing step. For instance, mesh simplification in BOG [Reiser et al. 2024] requires approximately 4 hours. To circumvent these challenges, we introduce a novel method for extracting adaptive and compact meshes using tetrahedral grids and marching tetrahedra.

Tetrahedral Grids Generation: Our primary insight is that the position and scale of 3D Gaussian primitives serve as reliable indicators for the presence of surfaces. To capitalize on this, we define a 3D bounding box around each Gaussian primitive, where the extent of the 3D box is 3 times the Gaussian scales. The 3D box's center has the highest opacity and its corners have the smallest opacity. Note that we do not consider the opacity α of Gaussian primitives but it could be used to filter out Gaussians with low opacity value. We



(a) Marching Cubes (b) Marching Cubes with Binary Search

Fig. 5. **Comparison of applying binary search to Marching cubes**. Strong step artifacts can be observed in Marching cubes results since the linear assumption does not hold in our GOF. Applying our binary search algorithm eliminates this artifact.

create tetrahedral grids with the center and corners of 3D bounding boxes. Inspired by Tetra-NeRF [Kulhanek and Sattler 2023], we employ the CGAL Library [Jamin et al. 2024] for Delaunay triangulation to construct tetrahedral cells. The generated tetrahedral cell might connect points across significant distances. Therefore, we employ a filtering step for the tetrahedral cells, removing any cell whose edges connect non-overlapping Gaussians. Gaussians are considered non-overlapping when the length of the edge connecting them exceeds the sum of their maximum scales, i.e., maximum 3-sigma extent dimension.

Efficient Opacity Evaluation: To efficiently evaluate the opacity of the vertices of the tetrahedral grid, we design a tile-based evaluation algorithm, inspired by 3DGS [Kerbl et al. 2023]. Specifically, we first project the vertices to image space and identify the corresponding tiles for these projections. These points are then organized according to their tile ID. For each tile, we retrieve the list of points that are projected within it, project these points again to identify the pixel that it falls inside and identify the Gaussians that contribute to this pixel. Finally, we enumerate all points to evaluate their opacity based on the pre-filtered list of Gaussians. Note that this process is iterated over all training images. Then we take the minimum opacity over all training images as the opacity for the vertices of the tetrahedral grid. An overview of the algorithm is provided in the supplementary material.

Binary Search of Level Set: Upon determining the opacity values for the tetrahedral grid, we proceed to extract triangle meshes using the Marching Tetrahedra method [Shen et al. 2021]. Traditional algorithms, including Marching Cubes [Lorensen and Cline 1998] and Marching Tetrahedra, typically rely on linear interpolation to approximate the level set, presuming the underlying field's linearity. This assumption, however, misaligns with the characteristics of our Gaussian Opacity Field, leading to artifacts due to linear interpolation, as shown in Figure 5 (a). To overcome this discrepancy and accurately identify the level set within our non-linear opacity field, we relax the linear assumption to a monotonically increasing assumption. This adjustment allows for the implementation of a binary search algorithm to precisely identify the level set. In practice, we found that conducting 8 iterations of binary search—effectively simulating 256 dense evaluations—yields consistent and reliable outcomes. A comparison highlighting the improvements of binary search is shown in Figure 5.

Table 1. **Quantitative results on the Tanks and Temples Dataset [Knapitsch et al. 2017]**. Reconstructions are evaluated with the official evaluation scripts and we report F1-score and average optimization time. The results of implicit methods are taken from Neuralangelo [Li et al. 2023] and the results of explicit methods are taken from 2DGS [Huang et al. 2024], where the mesh of 3DGS [Kerbl et al. 2023] and 2DGS [Huang et al. 2024] are extracted with TSDF fusion and SuGaR uses Poisson reconstruction. GOF outperforms all 3DGS-based surface reconstruction methods by a large margin and performs comparably with the SOTA neural implicit methods while optimizing significantly faster.

	Implicit			Explicit			
	NeuS	Geo-Neus	Neuralangelo	SuGaR	3DGS	2DGS	Ours
Barn	0.29	0.33	0.70	0.14	0.13	0.41	0.51
Caterpillar	0.29	0.26	0.36	0.16	0.08	0.23	0.41
Courthouse	0.17	0.12	0.28	0.08	0.09	0.16	0.28
Ignatius	0.83	0.72	0.89	0.33	0.04	0.51	0.68
Meetingroom	0.24	0.20	0.32	0.15	0.01	0.17	0.28
Truck	0.45	0.45	0.48	0.26	0.19	0.45	0.59
Mean	0.38	0.35	0.50	0.19	0.09	0.32	0.46
Time	>24h	>24h	>24h	>1h	14.3 m	15.5 m	24.2 m

4 EXPERIMENTS

We conduct a thorough evaluation of our Gaussian Opacity Fields (GOF), comparing its surface reconstruction and novel view synthesis against leading methods. We further validate the effectiveness of its key components through ablation studies.

4.1 Implementation Details

We build GOF upon the open-source 3DGS code base [Kerbl et al. 2023] and implement custom CUDA kernels for ray-tracing-based volume rendering, regularizations, and opacity evaluation. Regularization parameters are set to $\alpha = 1000$ for bounded scenes, $\alpha = 100$ for unbounded scenes, and $\beta = 0.05$ for all scenes, following 2DGS [Huang et al. 2024]. As our improved densification strategy increases the number of primitives for the same hyperparameters, we use a higher opacity threshold 0.05 instead of 0.005 for Gaussian pruning, resulting in a similar number of primitives post-training for a fair comparison. Similar to 3DGS, we stop densification at 15k iterations and optimize all of our models for 30k iterations. For mesh extraction, we adapt the Marching Tetrahedra algorithm [Shen et al. 2021] from the Kaolin library [Fuji Tsang et al. 2022] with our binary search algorithm and extract the mesh for the 0.5 level-set. All experiments are conducted on an NVIDIA A100 GPU.

4.2 Geometry Evaluation

We first compare against both SOTA implicit and explicit surface reconstruction methods on the Tanks and Temples Dataset. Reconstructions are evaluated only for the foreground objects since the ground truth point clouds do not contain background regions. As shown in Table 1, our method is competitive with the leading implicit approaches [Li et al. 2023] while being much more efficient to be optimized. Note that most implicit approaches [Li et al. 2023; Wang et al. 2021] only reconstruct the foreground objects, while our method can reconstruct detailed meshes also for the background regions, which is of great importance for mesh-based real-time rendering [Reiser et al. 2024]. Furthermore, while our method is slightly slower than 3DGS [Kerbl et al. 2023] and 2DGS [Huang et al.

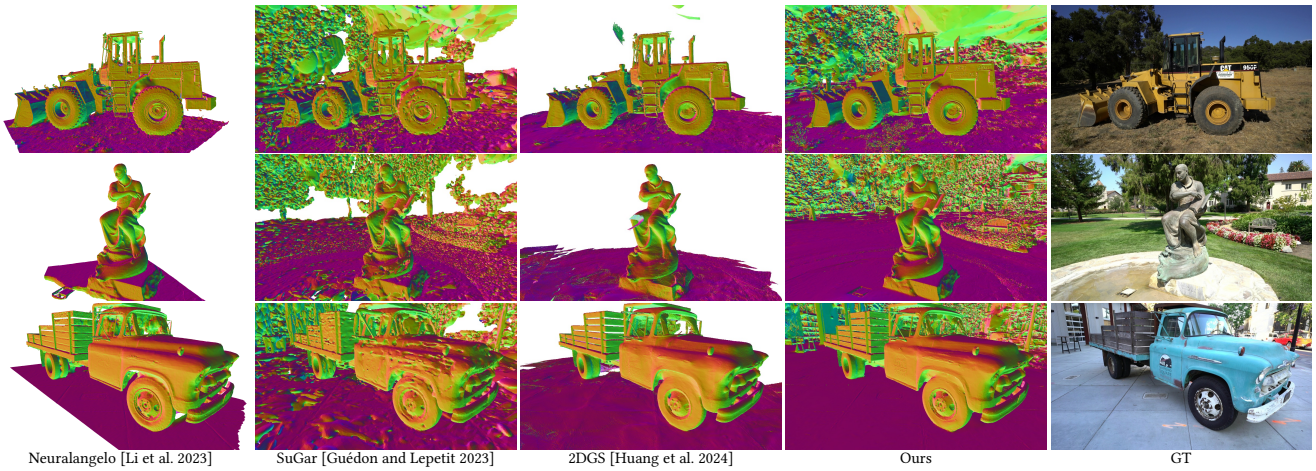


Fig. 6. **Surface Reconstruction on the Tanks and Temples Dataset [Knapitsch et al. 2017]**. We show the rendered normal maps from extract meshes and GT images for reference. Neural implicit methods, such as Neuralangelo [Li et al. 2023], model the foreground with SDF and the background with NeRF. Therefore, only the foreground mesh is extracted. SuGaR's mesh is noisy [Guédon and Lepetit 2023] and 2DGS fails at reconstructing background regions [Huang et al. 2024]. In contrast, our method can reconstruct detailed surfaces for both foreground objects and background regions.

Table 2. **Quantitative comparison on the DTU Dataset [Jensen et al. 2014]**. We report the Chamfer distance and average optimization time. The results of 3DGS [Kerbl et al. 2023] and SuGaR [Guédon and Lepetit 2023] are taken from 2DGS [Huang et al. 2024]. Our method achieves the highest reconstruction accuracy among other explicit methods.

		24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean	Time
implicit	NeRF [Mildenhall et al. 2021]	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	> 12h
	VolSDF [Yariv et al. 2021]	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 [Wang et al. 2021]	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
	Neuralangelo [Li et al. 2023]	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	> 12h
explicit	3DGS [Kerbl et al. 2023]	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	11.2 m
	SuGaR [Guédon and Lepetit 2023]	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
	GaussianSurfels [Dai et al. 2024]	0.66	0.93	0.54	0.41	1.06	1.14	0.85	1.29	1.53	0.79	0.82	1.58	0.45	0.66	0.53	0.88	6.7 m
	2DGS [Huang et al. 2024]	0.48	0.91	0.39	0.39	1.01	0.83	0.81	1.36	1.27	0.76	0.70	1.40	0.40	0.76	0.52	0.80	10.9 m
	Ours	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	18.4 m

2024] due to the ray-Gaussian intersection computation, it significantly outperforms all SOTA 3DGS-based methods in terms of reconstruction quality. A qualitative comparison is shown in Figure 6. GOF reconstructs fine-detailed surfaces for both foreground objects and the background regions. By contrast, meshes extracted from SuGaR [Guédon and Lepetit 2023] are noisy, while 2DGS [Huang et al. 2024] fails to extract geometry for the background regions.

We further compare against SOTA surface reconstruction methods on the DTU dataset [Jensen et al. 2014]. As shown in Table 2, our method outperforms all other 3DGS-based methods [Dai et al. 2024; Guédon and Lepetit 2023; Huang et al. 2024]. Despite a performance gap with the leading implicit reconstruction method [Li et al. 2023], GOF's optimization is much faster. This performance disparity is attributed to the DTU dataset's strong view-dependent appearance. Utilizing a better view-dependent appearance modeling [Verbin et al. 2022] or a coarse-to-fine training strategy [Li et al. 2023] could potentially improve the reconstructions.

4.3 Novel view Synthesis

To evaluate the NVS results of GOF, we further compare against SOTA NVS methods on the Mip-NeRF 360 dataset [Barron et al. 2022a]. The quantitative results are shown in Table 3. GOF not

Table 3. **Quantitative results on Mip-NeRF 360 [Barron et al. 2022b] dataset.** All results of the baseline methods are taken from their papers whenever available and we rerun SuGaR [Guédon and Lepetit 2023] with the same setting as ours. Our method achieved SOTA NVS results, especially in the outdoor scenes in terms of LPIPS.

	Outdoor Scene			Indoor scene		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF	21.46	0.458	0.515	26.84	0.790	0.370
Deep Blending	21.54	0.524	0.364	26.40	0.844	0.261
Instant NGP	22.90	0.566	0.371	29.15	0.880	0.216
MERF	23.19	0.616	0.343	27.80	0.855	0.271
MipNeRF360	24.47	0.691	0.283	31.72	0.917	0.180
Mobile-NeRF	21.95	0.470	0.470	-	-	-
BakedSDF	22.47	0.585	0.349	27.06	0.836	0.258
SuGaR	22.93	0.629	0.356	29.43	0.906	0.225
BOG	23.94	0.680	0.263	27.71	0873	0.227
3DGS	24.64	0.731	0.234	30.41	0.920	0.189
Mip-Splatting	24.65	0.729	0.245	30.90	0.921	0.194
2DGS	24.34	0.717	0.246	30.40	0.916	0.195
Ours	24.82	0.750	0.202	30.79	0.924	0.184

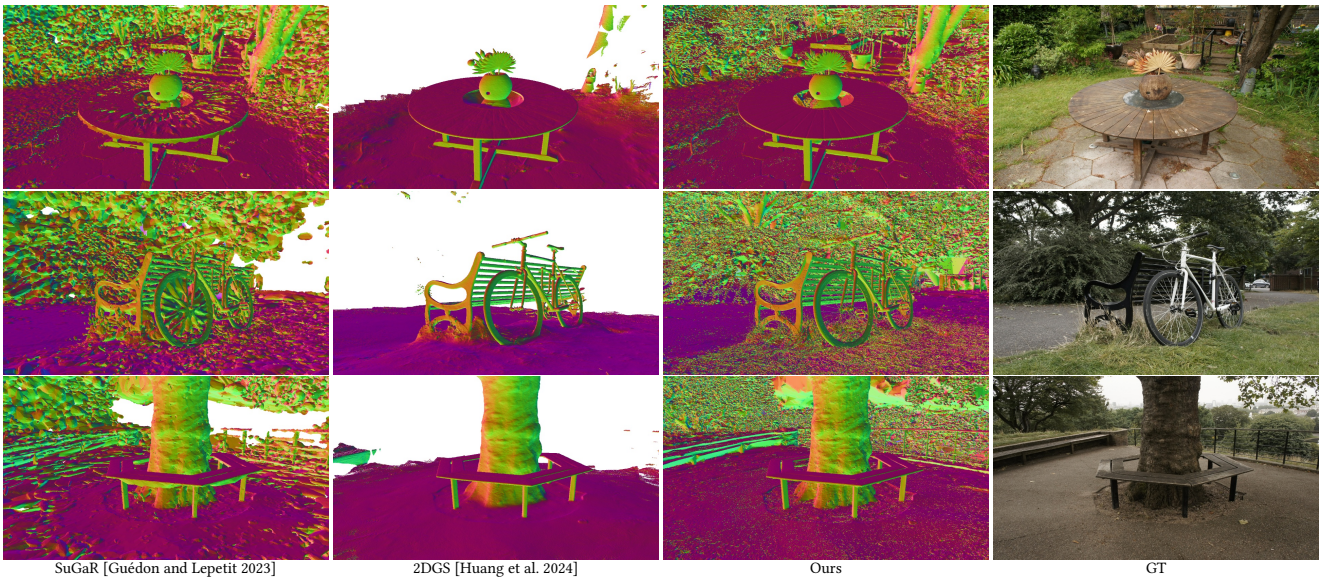


Fig. 7. **Reconstructions on the Mip-NeRF 360 Dataset [Barron et al. 2022a].** We show the rendered normal maps from extract meshes together with GT images for reference. Our method can reconstruct detailed surfaces for both foreground objects and background regions while meshes from previous work are noisy [Guédon and Lepetit 2023] or fail to reconstruct background regions and thin structures, such as the spokes in the bicycle scene [Huang et al. 2024].

Fig. 8. **Comparison of mesh extraction methods.** Applying GOF to SOTA Mip-Splatting [Yu et al. 2024a] yields significant improvements over TSDF, enabling complete mesh extraction for background regions.

only performs slightly better than all other 3DGS-based methods in terms of PSNR, but also outperforms all other methods significantly in terms of LPIPS [Zhang et al. 2018] in the outdoor scenes. The main improvements come from our improved densification strategy. In the ablation, we show that adding our densification strategy to 3DGS [Kerbl et al. 2023] and Mip-Splatting [Yu et al. 2024a] improves the NVS results by a large margin. For the indoor scenes, our results are similar to Mip-Splatting [Yu et al. 2024a], with less than 0.1 PSNR difference, which we attribute to our regularizations that trade-off NVS and surface reconstruction. Our method also outperforms Sugar [Guédon and Lepetit 2023] and 2DGS [Huang et al. 2024] in all metrics. We show a qualitative comparison of extracted meshes in Figure 7. Similar to our observations on the Tanks and Temples dataset [Knapitsch et al. 2017], GOF can reconstruct detailed surfaces for both foreground objects and background regions, while SuGaR's [Guédon and Lepetit 2023] meshes are noisy and have fewer details and 2DGS fails to extract meshes for the background regions. More qualitative results of our method can be found in Figure 13 and Figure 14.

Table 4. **Ablation on the Tanks and Temples Dataset [Knapitsch et al. 2017].** The metrics are computed with the official script from the dataset. GOF significantly improves surface reconstruction quality. Our regularization, decoupled appearance, and densification strategy modeling further improve the results.

	Precision ↑	Recall ↑	F-score ↑
A. Mip-Splatting w/ TSDF	0.15	0.25	0.16
B. Mip-Splatting w/ GOF	0.40	0.33	0.36
C. Ours w/o GOF	0.37	0.45	0.39
D. Ours w/o normal consistency	0.41	0.35	0.37
E. Ours w/o decoupled appearance	0.49	0.39	0.43
F. Ours w/ minimal axis's normal	0.46	0.36	0.40
G. Ours w/o improved densification	0.52	0.39	0.44
H. Ours	0.54	0.42	0.46

4.4 Ablation Study

Mesh Extraction: Our GOF enables mesh extraction from 3D Gaussians directly by identifying a level set without resorting to Poisson reconstruction or TSDF fusion. Compared to using TSDF fusion

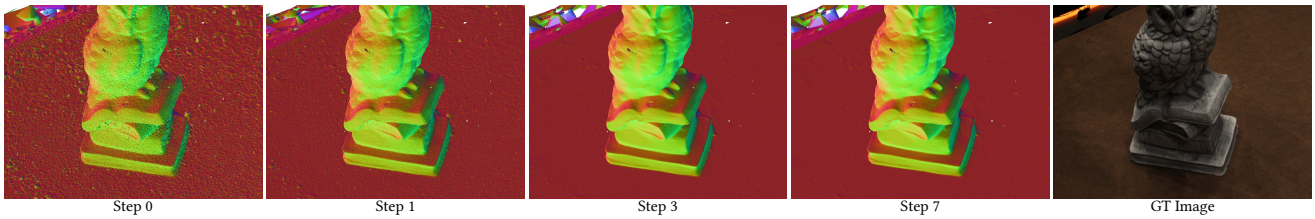


Fig. 9. **Different number of binary search steps with marching tetrahedra on the DTU dataset [Jensen et al. 2014].** Applying our binary search to Marching Tetrahedra [Shen et al. 2021] significantly improves the quality of extracted meshes in just few steps.

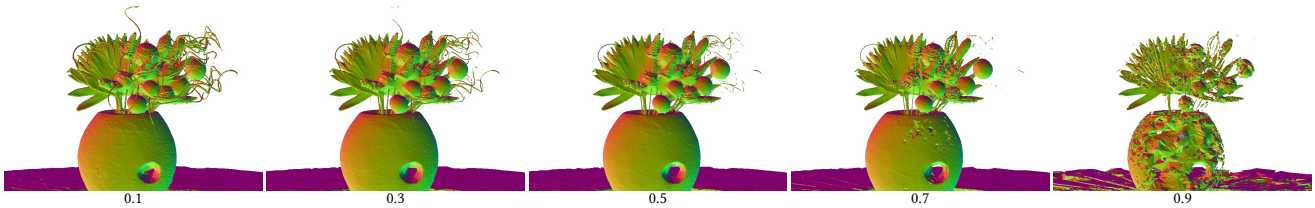


Fig. 10. **Meshes extracted with different level sets on the Mip-NeRF 360 dataset [Barron et al. 2022a].** Our method supports multi-layer meshes extraction by using different level sets for marching tetrahedra.

for mesh extraction (Table 4, C), our tetrahedra-based mesh extraction (Table 4, H) significantly improves the quality of the extracted meshes.

To further demonstrate the effectiveness and generalizability of our mesh extraction strategy, we apply it to a SOTA 3DGS-based model, Mip-Splatting [Yu et al. 2024a], and compare it with TSDF fusion. A comparison of the extracted meshes is shown in Figure 8. The mesh extracted from TSDF fusion is noisy and has a lot of holes on the ground, due to the inconsistency of depth. By contrast, our extracted mesh is more complete. Our method also extracts detailed surfaces for the background regions. Quantitative results in Table 4 (A vs. B) further indicate the effectiveness of our method.

Regularization and Appearance Modeling: As shown in Table 4 (D vs. H), using the normal regularization during training significantly improves reconstruction quality, which is consistent with the observation in 2DGS [Huang et al. 2024]. Including the decoupled appearance modeling [Lin et al. 2024] further improves the reconstruction results as shown in Table 4 (E vs. H), since the model is less likely to model view-dependent appearance with geometry.

Normal Definition: We approximate the normal of a Gaussian as the normal of the ray-Gaussian intersection plane. By contrast, SuGaR [Guédon and Lepetit 2023] approximates the normal as one of Ganssian's axis directions with minimal scale. We experimented by replacing our normal definition with the axis direction of the minimum scale and found that the F1-score decreases from 0.46 to 0.40 on the TNT dataset, as shown in Table 4 (F vs. H), validating the effectiveness of our normal definition.

Densification on Geometry: As shown in Table 4 (G vs. H), disabling our improved densification strategy leads to a decrease in the F1-score from 0.46 to 0.44 in the TNT dataset [Knapitsch et al. 2017], showing that our densification metric contributes positively to geometry reconstruction.

Table 5. **Ablation on the Mip-NeRF 360 [Barron et al. 2022b] dataset.** Our densification strategy improves the novel view synthesis results for 3DGS [Kerbl et al. 2023] and Mip-Splatting [Yu et al. 2024a] significantly.

	Outdoor Scene			Indoor scene		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS	**24.64**	0.731	0.234	30.41	0.920	0.189
w/ our densification	24.62	**0.743**	**0.199**	**31.10**	**0.928**	**0.174**
Mip-Splatting	24.65	0.729	0.245	30.90	0.921	0.194
w/ our densification	**24.77**	**0.745**	**0.205**	**31.18**	**0.926**	**0.180**
Ours	24.82	0.750	0.202	30.79	0.924	0.184

Overall Improvements: When using the same mesh extraction method, such as TSDF fusion (Table 4, A vs. C) or our tetrahedra-based mesh extraction (Table 4, B vs. H), the improvements of our method over Mip-Splatting [Yu et al. 2024a] are due to additional regularization, appearance modeling, and our improved densification strategy. Without these elements, the reconstruction quality between our method and rasterization-based methods is comparable. For instance, disabling our normal regularization results in similar reconstruction quality compared to Mip-Splatting with GOF (Table 4, D vs. B).

Densification on View Synthesis: To further evaluate the effectiveness of our improved densification metric on novel view synthesis, we apply it to 3DGS [Kerbl et al. 2023] and Mip-Splatting [Yu et al. 2024a]. The quantitative results, as shown in Table 5, demonstrate that our strategy improves NVS results significantly, especially in terms of LPIPS [Zhang et al. 2018]. Qualitative comparisons are also provided in Figure 4,where the glass regions are rendered with high fidelity using our densification metric.

It is important to note that our ray-Gaussian intersection formula has similar rendering quality to rasterization-based methods [Kerbl et al. 2023; Yu et al. 2024a]. For instance, our NVS results are similar to those of Mip-Splatting when augmented with our densification

strategy, as shown in Table 5. However, the ray-Gaussian intersection formula enables the establishment of opacity fields, improving mesh extraction compared to TSDF fusion. Additionally, our regularization and appearance modeling further improve the reconstruction quality, as shown previously.

Binary Search for Marching Tetrahedra: To demonstrate the effectiveness of applying binary search to Marching Tetrahedra [Shen et al. 2021], we apply binary search with different numbers of steps. As shown in Figure 9, using binary search significantly improves the quality of reconstructed meshes in just a few iterations.

Multi-layer Meshes: While we extract meshes for the 0.5 level set in the main paper, our method also supports extracting meshes with different level sets. As a proof of concept, we extract meshes with different level sets 0.1, 0.3, 0.5, 0.7, and 0.9 and show the rendered meshes in Figure 10. Finer structures can be extracted with a smaller level set, but it might result in expanded meshes.

5 LIMITATIONS

We now discuss some limitations and potential future extensions of our method.

Delaunay Triangulation Efficiency: We employ the CGAL Library [Jamin et al. 2024] for Delaunay triangulation to construct tetrahedral cells, which has $O(N \log N)$ complexity. It becomes a bottleneck particularly when the number of points increases. For example, it takes around 8 minutes to construct the tetrahedral cells for the bicycle scene in the Mip-NeRF dataset [Barron et al. 2022a]. This process could potentially be optimized by considering the spatial locality since the points are generated from 3D Gaussians, or by employing parallel processing techniques on GPUs. Additionally, optimizing the selection and number of Gaussian primitives used for constructing tetrahedral grids could further improve efficiency.

Opacity Evaluation Optimization: During the binary search of marching tetrahedra, our method evaluates the opacity of points using all training views, which may lead to redundant computations. Recognizing that a single view can determine a point's minimal opacity value suggests a more efficient approach could be developed by associating points with their respective influential training views.

View Dependent Appearance Modeling: Using spherical harmonics for view-dependent appearance modeling has limitations, such as potentially inaccurately representing reflections as geometric features. Incorporating a better view-dependent appearance model [Verbin et al. 2022] could potentially enhance the quality of reconstructions.

Mesh-based Rendering: While the current focus of GOF is on surface reconstruction and novel view synthesis, leveraging the extracted meshes for real-time rendering is an interesting future direction [Reiser et al. 2024]. This could potentially improve the quality of mesh-based rendering given the detailed and adaptive meshes extracted with GOF.

6 CONCLUSION

We have presented Gaussian Opacity Fields (GOF), a novel method for efficient, high-quality, and adaptive surface reconstruction in unbounded scenes. Our GOF is derived from ray-tracing-based volume rendering of 3D Gaussians, maintaining consistency with RGB rendering. Our GOF enables geometry extraction directly from 3D Gaussians by identifying its level set, without Poisson reconstruction or TSDF. We approximate the surface normal of Gaussians as the normal of the ray-Gaussian intersection plane and apply depth-normal consistency regularization to enhance geometry reconstructions. Furthermore, we propose an efficient and adaptive mesh extraction method utilizing Marching Tetrahedra, where the tetrahedral grids are induced from 3D Gaussians. Our evaluations reveal that GOF surpasses existing explicit methods in both surface reconstruction and novel view synthesis. Further, GOF achieves comparable surface reconstruction results with leading implicit methods while being able to reconstruct detailed meshes for the background regions in unbounded scenes.

Acknowledgement: We thank Christian Reiser and Binbin Huang for insightful discussions and valuable feedback throughout the project and proofreading. ZY and AG are supported by the ERC Starting Grant LEGO-3D (850533) and DFG EXC number 2064/1 - project number 390727645. TS is supported by a Czech Science Foundation (GACR) EXPRO grant (UNI-3D, grant no. 23-07973X).

REFERENCES

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

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022b. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5470–5479.

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

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. TensoRF: Tensorial Radiance Fields. In *European Conference on Computer Vision (ECCV)*.

Guikun Chen and Wenguan Wang. 2024. A survey on 3d gaussian splatting. *arXiv preprint arXiv:2401.03890* (2024).

Hanlin Chen, Chen Li, and Gim Hee Lee. 2023b. NeuSG: Neural Implicit Surface Reconstruction with 3D Gaussian Splatting Guidance. *arXiv preprint arXiv:2312.00846* (2023).

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2023a. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16569–16578.

Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. 2024. GaussianPro: 3D Gaussian Splatting with Progressive Propagation. *arXiv preprint arXiv:2402.14650* (2024).

Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. 2024. High-quality Surface Reconstruction using Gaussian Surfels. In *ACM SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, Article 22, 11 pages.

Akio Doi and Akio Koide. 1991. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE TRANSACTIONS on Information and Systems* 74, 1 (1991), 214–224.

Robert A Drebin, Loren Carpenter, and Pat Hanrahan. 1988. Volume rendering. *ACM Siggraph Computer Graphics* 22, 4 (1988), 65–74.

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

Clement Fuji Tsang, Maria Shugrina, Jean Francois Lafleche, Towaki Takikawa, Jiehan Wang, Charles Loop, Wenzheng Chen, Krishna Murthy Jatavallabhula, Edward Smith, Artem Rozantsev, Or Perel, Tianchang Shen, Jun Gao, Sanja Fidler, Gavriel State, Jason Gorski, Tommy Xiang, Jianing Li, Michael Li, and Rev Lebaredian. 2022. Kaolin: A Pytorch Library for Accelerating 3D Deep Learning Research. https://github.com/NVIDIAGameWorks/kaolin.

Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. 2023. Relightable 3D Gaussian: Real-time Point Cloud Relighting with BRDF Decomposition and Ray Tracing. *arXiv:2311.16043* (2023).

Michael Garland and Paul S Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 209–216.

Antoine Guédon and Vincent Lepetit. 2023. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. *arXiv preprint arXiv:2311.12775* (2023).
Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. 2021. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5875–5884.
Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery. https://doi.org/10.1145/3641519.3657428
Clément Jamin, Sylvain Pion, and Monique Teillaud. 2024. 3D Triangulation Data Structure. In *CGAL User and Reference Manual* (5.6.1 ed.). CGAL Editorial Board. https://doc.cgal.org/5.6.1/Manual/packages.html#PkgTDS3
Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. 2014. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 406–413.
James T Kajiya and Brian P Von Herzen. 1984. Ray tracing volume densities. *ACM SIGGRAPH computer graphics* 18, 3 (1984), 165–174.
Michael Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)* 32, 3 (2013), 1–13.
Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/
Leonid Keselman and Martial Hebert. 2022. Approximate Differentiable Rendering with Algebraic Surfaces. In *European Conference on Computer Vision (ECCV)*.
Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Transactions on Graphics* 36, 4 (2017).
Jonas Kulhanek and Torsten Sattler. 2023. Tetra-NeRF: Representing Neural Radiance Fields Using Tetrahedra. *arXiv preprint arXiv:2304.09987* (2023).
Kiriakos N Kutulakos and Steven M Seitz. 2000. A theory of shape by space carving. *International journal of computer vision* 38 (2000), 199–218.
Aldo Laurentini. 1994. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on pattern analysis and machine intelligence* 16, 2 (1994), 150–162.
Marc Levoy. 1990. Efficient ray tracing of volume data. *ACM Transactions on Graphics (TOG)* 9, 3 (1990), 245–261.
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 *IEEE Conference on Computer Vision and Pattern Recognition (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 *Conference on Computer Vision and Pattern Recognition (CVPR)*.
William E Lorensen and Harvey E Cline. 1998. Marching cubes: A high resolution 3D surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 347–353.
Nelson Max. 1995. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics* 1, 2 (1995), 99–108.
Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
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*.
Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.
Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages. https://doi.org/10.1145/3528223.3530127
Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *International Conference on Computer Vision (ICCV)*.
Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Marie-Julie Rakotosaona, Fabian Manhardt, Diego Martin Arroyo, Michael Niemeyer, Abhijit Kundu, and Federico Tombari. 2024. NeRFMeshing: Distilling Neural Radiance Fields into Geometrically-Accurate 3D Meshes. In *Proc. of the International Conf. on 3D Vision (3DV)*.
Christian Reiser, Stephan Garbin, Pratul P. Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan T. Barron, Peter Hedman, and Andreas Geiger. 2024. Binary Opacity Grids: Capturing Fine Geometric Detail for Mesh-Based View Synthesis. *arXiv* 2402.12377 (2024).
Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In *International Conference on Computer Vision (ICCV)*.
Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. 2023. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–12.
Radu Alexandru Rosu and Sven Behnke. 2023. PermutoSDF: Fast Multi-View Reconstruction with Implicit Surfaces using Permutohedral Lattices. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (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 *Advances in Neural Information Processing Systems (NeurIPS)*.
Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction.
Jiaxiang Tang, Hang Zhou, Xiaokang Chen, Tianshu Hu, Errui Ding, Jingdong Wang, and Gang Zeng. 2022. Delicate Textured Mesh Recovery from NeRF via Adaptive Surface Refinement. *arXiv preprint arXiv:2303.02091* (2022).
Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. 2024. DN-Splatter: Depth and Normal Priors for Gaussian Splatting and Meshing. *arXiv* 2403.17822 (2024).
Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. 2022. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. *CVPR* (2022).
Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *Advances in Neural Information Processing Systems* 34 (2021), 27171–27183.
Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. 2018. MVSNet: Depth Inference for Unstructured Multi-view Stereo. *European Conference on Computer Vision (ECCV)* (2018).
Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems* 34 (2021), 4805–4815.
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. *arXiv* (2023).
Mulin Yu, Tao Lu, Linning Xu, Lihan Jiang, Yuanbo Xiangli, and Bo Dai. 2024b. GSDF: 3DGS Meets SDF for Improved Rendering and Reconstruction. *arXiv* 2403.16964 (2024).
Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. 2022a. SDFStudio: A Unified Framework for Surface Reconstruction. https://github.com/autonomousvision/sdfstudio
Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. 2024a. Mip-Splatting: Alias-free 3D Gaussian Splatting. *Conference on Computer Vision and Pattern Recognition (CVPR)* (2024).
Zehao Yu and Shenghua Gao. 2020. Fast-MVSNet: Sparse-to-Dense Multi-View Stereo With Learned Propagation and Gauss-Newton Refinement. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. 2022b. MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)* (2022).
Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. NeRF++: Analyzing and Improving Neural Radiance Fields. *arXiv:2010.07492* (2020).
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*.
Hongyu Zhou, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. 2024. HUGS: Holistic Urban 3D Scene Understanding via Gaussian Splatting. *arXiv preprint arXiv:2403.12722* (2024).
Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. 2001. EWA volume splatting. In *Proceedings Visualization, 2001. VIS'01.* IEEE, 29–538.

A ADDITIONAL IMPLEMENTATION DETAILS

In this section, we describe our implementation details, including a minor modification to 3DGS's densification strategy to overcome the clustered issues resulting from clone operation and details for surface extraction.

Algorithm 2 Gaussian Opacity Field evaluation for a single view
w, h: width and height of training image
M, S, A: Gaussian means, covariances, and opacity
P: position of points
V: view configuration of current camera

function EvaluateSingleView(w, h, M, S, A, V, P)
  CullGaussian(M, V) ▷ Frustum Culling
  $M', S' \leftarrow$ ScreenspaceGaussians(M, S, V) ▷ Transform
  $T_g \leftarrow$ CreateTiles(w, h)
  $L_g, K_g \leftarrow$ DuplicateWithKeys(M', T_g) ▷ Indices and Keys
  SortByKeys(K_g, L_g) ▷ Globally Sort
  $R_g \leftarrow$ IdentifyTileRanges(T_g, K_g)
  CullPoints(P, V) ▷ Frustum Culling
  $P' \leftarrow$ ScreenspacePoints(P, V) ▷ Transform
  $T_p \leftarrow$ CreateTiles(w, h)
  $L_p, K_p \leftarrow$ CreateWithKeys(P', T_p) ▷ Indices and Keys
  SortByKeys(K_p, L_p) ▷ Globally Sort
  $R_p \leftarrow$ IdentifyTileRanges(T_p, K_p)
  $O \leftarrow \mathbf{1}$ ▷ Init Opacity
  for all Tiles t **in** I **do** ▷ I is the Canvas
    for all Pixels i **in** t **do**
      $r_g \leftarrow$ GetTileRange(R_g, t)
      $L'_g \leftarrow$ FilterGaussians(i, L_p, r_g, K_g, M, S, A) ▷ Select Gaussians that contributes to pixel i
      $r_p \leftarrow$ GetTileRange(R_p, t)
      $L'_p \leftarrow$ FilterPoints(i, L_p, r_p, K_p, P) ▷ Select Points that projected to pixel i
      for all Points p **in** L'_p **do**
        $O[p] \leftarrow$ EvaluateOpacity(i, L'_g, K_g, M, S, A)
      end for
    end for
  end for
  return O
end function

Algorithm 3 Marching Tetrahedra with Binary Search
M, S, A: Gaussian means, covariances, and opacity
P, C: Tetrahedra Points and Cells
V: view configurations
L: Level set value
N: Step of binary search

function MarchingTetrahedraBinarySearch(M, S, A, P, C, V)
  $O \leftarrow$ EvaluateOpacityField(M, S, A, V, P)
  $F, P_1, P_2, O_1, O_2 \leftarrow$ MarchingTetrahedra(P, C, O, S)
  for all i v **in** $\{1, 2, ..., N\}$ **do**
    $P_m \leftarrow$ MidPoint(P_1, P_2)
    $O_m \leftarrow$ EvaluateOpacityField(M, S, A, V, P_m)
    $P_1, P_2, O_1, O_2 \leftarrow$ ChangeEndPointsAndValues(P_1, P_2, O_1, O_2, L,P_m, O_m)
  end for
  $V \leftarrow$ LinearInterpolate(P_1, P_2, O_1, O_2, L) **return** F, V ▷ return faces and vertices for triangle mesh
end function

Clone with Sampling: We found that the position of Gaussian is relatively stable, which is also observed in Mip-Splatting [Yu et al. 2024a]. Therefore, Gaussians cloned from the same parents remain clustered. To address this issue, instead of using the same position for the cloned Gaussian, we sample a new Gaussian according to the Gaussian's parameter similar to the procedures for Gaussian split [Kerbl et al. 2023]. We found this simple strategy leads to more uniformly distributed Gaussians, as shown in Figure 11. However, we do not observe a significant impact in terms of NVS results.

Details for Surface Extraction: We provide a pseudo-code of our tile-based Gaussian Opacity Fields evaluation in Algorithm 1 and Algorithm 2. Note that the evaluation takes 3D Gaussians, training views, and 3D points as input and it does not rely on the tetrahedra cells. Therefore, the same algorithm also applies to the Marching Cubes Algorithm. The pseudo-code of our Marching Tetrahedra augmented with binary search is shown in Algorithm 3, the same idea could be applied to the Marching Cubes algorithm.

Algorithm 1 Gaussian Opacity Field evaluation
M, S, A: Gaussian means, covariances, and opacity
P: position of points
V: view configurations

function EvaluateOpacityField(M, S, A, V, P)
  $O \leftarrow \mathbf{1}$ ▷ Init Opacity
  for all Views v **in** V **do**
    $w, h \leftarrow$ GetImageSize(V)
    $O_v \leftarrow$ EvaluateSingleView(w, h, M, S, A, V, P)
    $O \leftarrow$ Min(O, O_v) ▷ Take Minimal Opacity
  end for
  return O
end function

B ADDITIONAL RESULTS

TSDF Fusion in Contraction Space: In the main paper, we compare GOF with 2DGS [Huang et al. 2024] with its default (bounded) settings. Specifically, 2DGS extracts meshes with TSDF fusion focusing on the foreground objects, resulting in missing geometry in the background regions. However, 2DGS also supports extracting meshes for the background regions by applying TSDF fusion in the contraction space [Barron et al. 2022a], which is referred as the unbounded setting. We extract unbounded meshes for 2DGS with a grid resolution of 2048 and the resulting meshes have similar number of vertices and faces with GOF's meshes. Then we compare GOF with both 2DGS's bounded and unbounded meshes. As shown in Figure 12, while 2DGS's unbounded setting can reconstruct meshes for the background regions, the meshes on the background regions are incomplete and lack details. By contrast, our method can reconstruct detailed surfaces for both foreground objects and background regions.

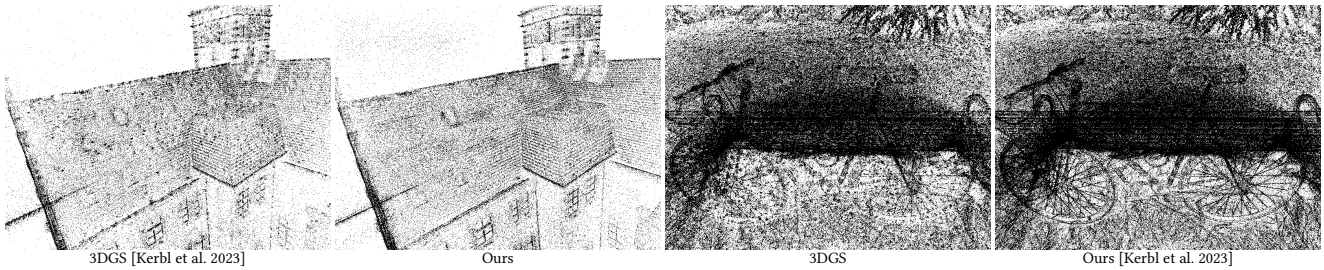


Fig. 11. **Comparison of clone strategy on the Mip-NeRF 360 Dataset [Barron et al. 2022a].** Our clone strategy leads to more uniformly distributed Gaussian primitives.

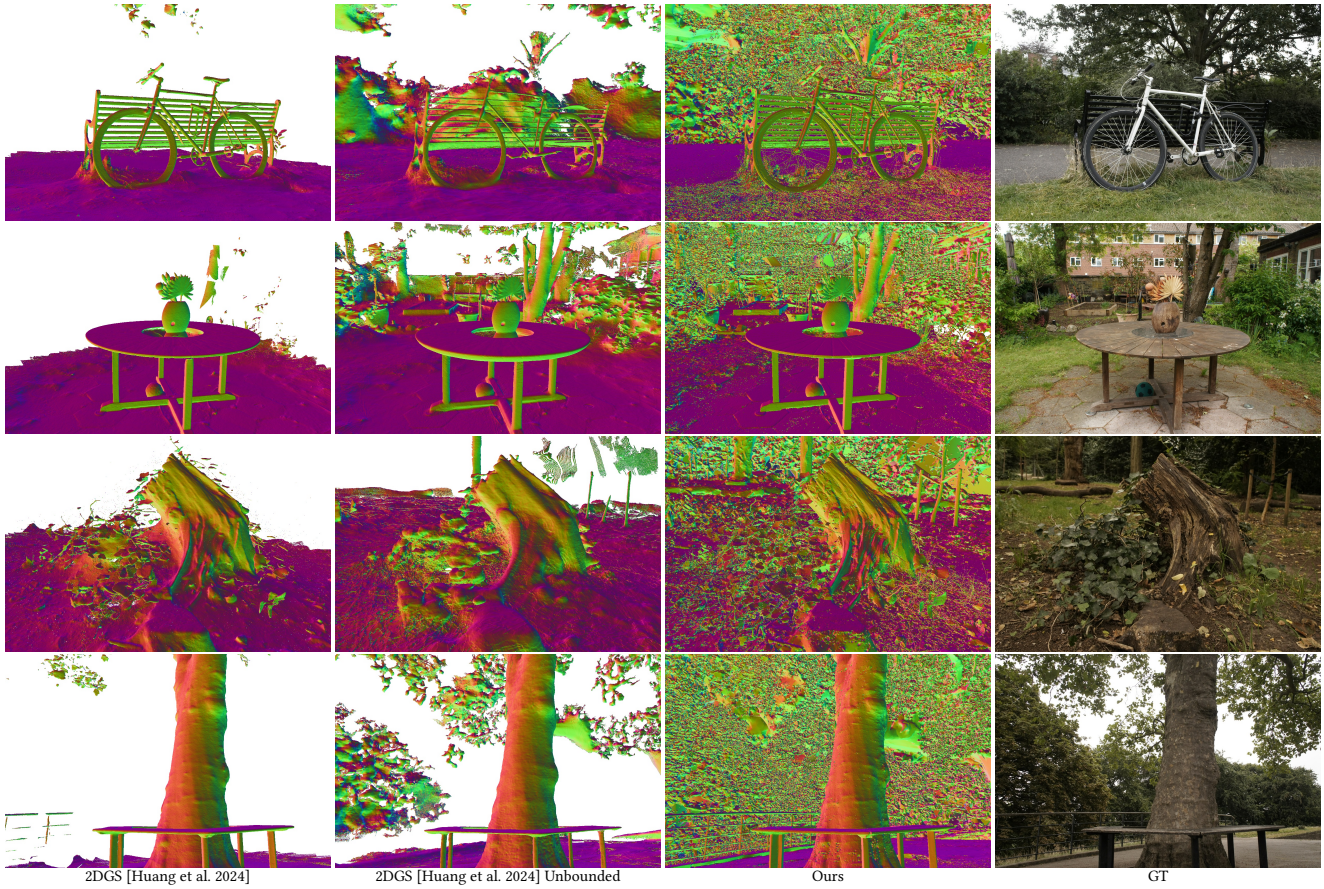


Fig. 12. **Reconstructions on the Mip-NeRF 360 Dataset [Barron et al. 2022a]**. We compare GOF with both 2DGS's bounded and unbounded mesh extraction. While 2DGS's unbounded setting can reconstruct mesh for the background region, its meshes are incomplete and lack of details. By contrast, GOF can reconstruct detailed meshes both for the foreground objects and background regions.

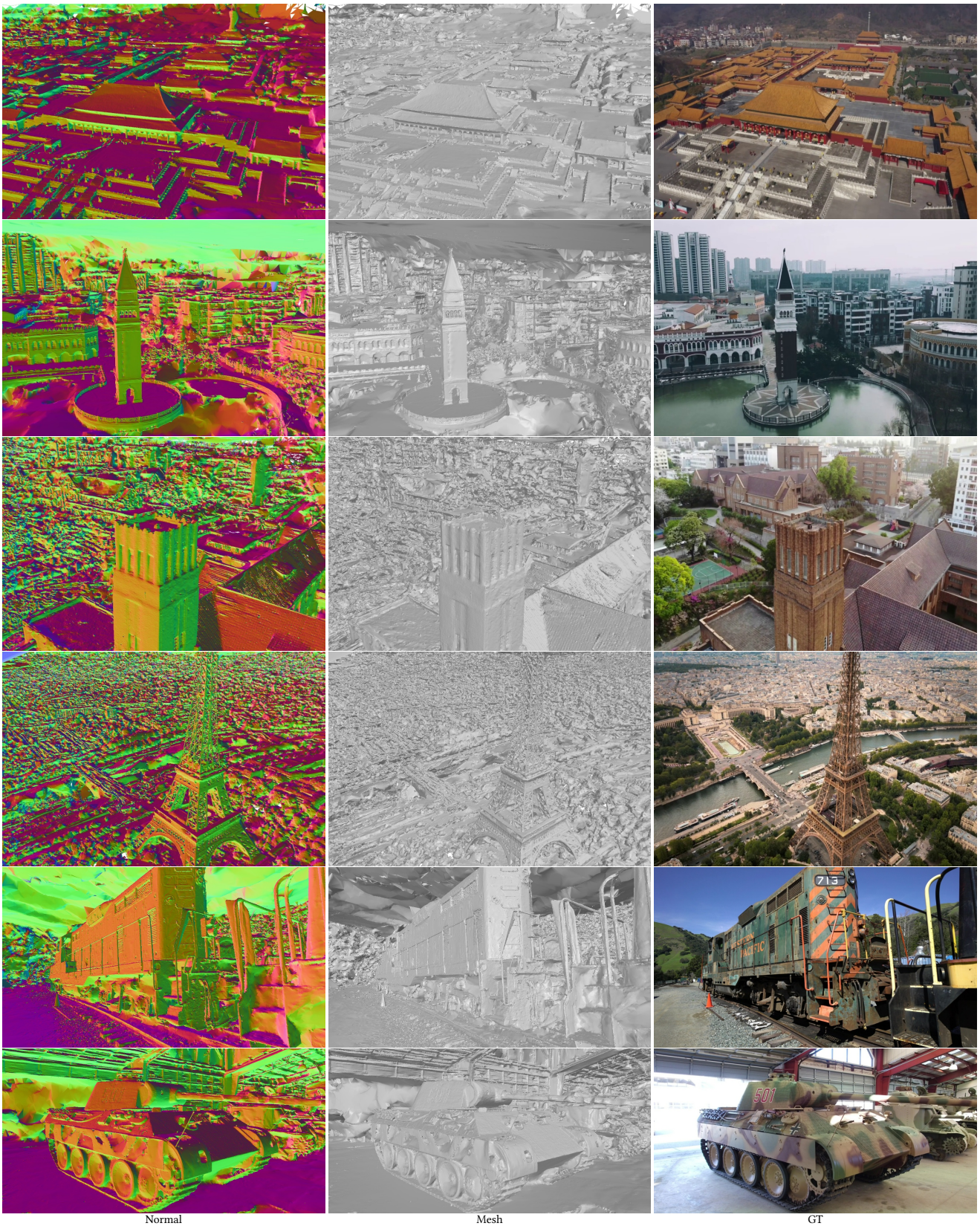


Fig. 13. **Reconstructions on the GaussianPro [Cheng et al. 2024] and the Tanks and Temples dataset [Knapitsch et al. 2017].**

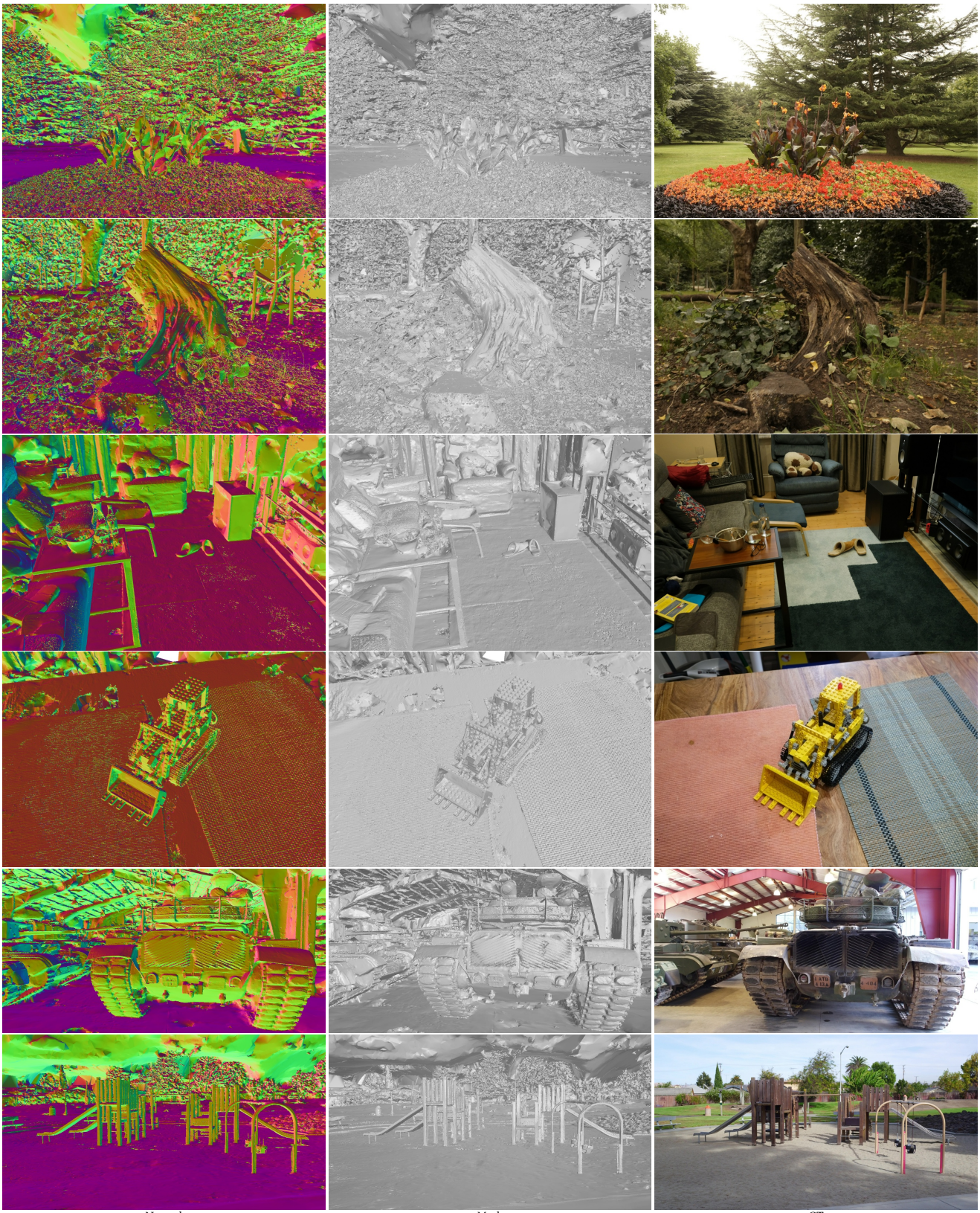


Fig. 14. **Reconstructions on the Mip-NeRF 360 [Barron et al. 2022a] and the Tanks and Temples dataset [Knapitsch et al. 2017].**