Geometry Field Splatting with Gaussian Surfels

Supplementary Material

A. Derivations

A.1. Revisit Volume Splatting

A.1.1 Original Derivation

Given a density field $\sigma(\mathbf{x})$ and a color field $\mathbf{c}(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^3$, the volume splatting algorithm [28] proposes to decompose the density field into the weighted sum of a set of n independent kernels $\{\mathcal{K}_1, \mathcal{K}_2, ..., \mathcal{K}_n\}$, each of which is a function mapping from x to a scalar and associated with a weight $\omega_i \in \mathbb{R}_+, i = 1, 2, ..., n$.

Formally, the density field is expressed as:

$$\sigma(\mathbf{x}) = \sum_{i=1}^{n} \omega_i \mathcal{K}_i(\mathbf{x}). \tag{1}$$

This decomposition helps simplify the volume rendering equation, which does not consider the scattering, for efficient rendering.

Specifically, given a ray shooting from the camera origin $\mathbf{o}$ with direction $\mathbf{d}$, the location of a point $\mathbf{p}$ on the ray can be expressed as $\mathbf{p}(l) = \mathbf{o} + \mathbf{d}l$, where $l \in \mathbb{R}_+$ denotes the depth. The rendered color $\mathbf{C}$ with the exponential falloff is given by the following equation:

$$\mathbf{C} = \int_0^\infty \mathbf{c}(\mathbf{p}(l))\sigma(\mathbf{p}(l)) \exp(-\int_0^l \sigma(\mathbf{p}(l'))dl')dl. \tag{2}$$

To simplify the notation without losing generality, we rewrite the Eqn. 2 as:

$$\mathbf{C} = \int_0^\infty \mathbf{c}(l)\sigma(l) \exp(-\int_0^l \sigma(l')dl')dl. \tag{3}$$

Zwicker et al. [28] propose to choose each of the kernels which compose the density field to have finite intersection intervals on the ray and assume that there is no overlapping between intersection intervals of any two of them. These kernels can then be sorted based on their intersection interval along the ray. By plugging the Eqn. 1 into the Eqn. 3, we get:

$$\begin{aligned}\mathbf{C} = \int_0^\infty \sum_{i=1}^{n} \mathbf{c}(l)\omega_i\mathcal{K}_i(l) \exp(-\int_0^l \omega_i\mathcal{K}_i(l')dl') \\ \prod_{j=1}^{i-1} \exp(-\int_0^l \omega_j\mathcal{K}_j(l')dl')dl.\end{aligned} \tag{4}$$

Relying on the assumption that each kernel has finite intersection interval and there is no overlapping, we can further rewrite the Eqn. 4 as:

$$\begin{aligned}\mathbf{C} = \sum_{i=1}^{n}(\prod_{j=1}^{i-1} \exp(-\int_0^\infty \omega_j\mathcal{K}_j(l')dl')) \\ (\int_0^\infty \mathbf{c}(l)\omega_i\mathcal{K}_i(l) \exp(-\int_0^l \omega_i\mathcal{K}_i(l')dl')dl).\end{aligned} \tag{5}$$

Zwicker et al. [28] then assume the color is constant within the intersection interval of each individual kernel and ignores the self-occlusion. Eqn. 5 can then be written as:

$$\mathbf{C} \approx \sum_{i=1}^{n}(\prod_{j=1}^{i-1} \exp(-\int_0^\infty \omega_j\mathcal{K}_j(l')dl'))(\mathbf{c}_i \int_0^\infty \omega_i\mathcal{K}_i(l)dl), \tag{6}$$

where $\mathbf{c}_i$ denotes the constant color value within the intersection interval of i^{th} kernel.

Zwicker et al. [28] then propose to expand the exponential term using the Taylor series and defines a value $\rho_i = \omega_i \int_0^\infty \mathcal{K}_i(l)dl$, which is called the footprint function with respect to the current ray. We can then reach:

$$\begin{aligned}\mathbf{C} &\approx \sum_{i=1}^{n}(\prod_{j=1}^{i-1}(1-\rho_j))(\mathbf{c}_i\rho_i) \\ &= \sum_{i=1}^{n} \mathbf{c}_i\rho_i(\prod_{j=1}^{i-1}(1-\rho_j)),\end{aligned} \tag{7}$$

which corresponds to the Eqn. 2 in the main paper. Notice that, even though the kernels are used to decompose the density field, due to the Taylor series expansion, $\rho_i, i = 1, 2, ..., n$ has to be within the range $[0, 1]$. Also, notice that, in a more general sense, the footprint function is the integration of the density along the intersection interval, but it is assumed that there is only one kernel constituting the density there, therefore, the footprint function is simplified as the integration of the kernel values.

Therefore, as long as we can evaluate the footprint function ρ easily and even differentiably, we can then reach an algorithm for efficient volume rendering, and even inverse rendering [9] under certain approximations.

As a summary, we identify following factors that make the volume splatting an approximate rendering algorithm:

- Self-occlusion is ignored.
- Transmittance term is approximated through the Taylor expansion.
- It is assumed that there is no overlapping between the intersection intervals of any two kernels, but in practice, it is not the case, which leads to the sorting not clearly defined, as observed in [15].

A common choice of kernel is to use 3D Gaussian kernel as in [9, 28]. In [9], the footprint function of 3D Gaussian kernel is further approximated due to the perspective transform, which introduces additional bias.

A recent emerging interest [4, 6] is to use 2D Gaussian kernel, which is then known as Gaussian surfel [14]. In this case, perspective projection is no longer a problem because the intersection between the 2D Gaussian kernel and the ray can be efficiently calculated [6]. And the sorting is always well-defined because the intersection interval along the ray is in general a point instead of an interval in the case of 3D Gaussian. However, the footprint function is then undefined because the integrand is a discontinuous function which has a finite value at the place where ray intersects with the 2D Gaussian kernel, and zero otherwise. The workaround proposed in [6] is to simply use the finite value as the footprint function, which still introduces further bias.

A rigorous exact rendering algorithm with efficient splatting is still an open problem, and we manage to solve it.

A.1.2 Derivation of Refined Splatting Algorithm

To move towards the exact rendering, we first address two approximations in the aforementioned derivation. We do not ignore the self-occlusion and do not expand the transmittance term. This corresponds to the Eqn. 7 in the main paper.

Specifically, from the Eqn. 5, we have:

$$\begin{aligned}\mathbf{C} = \sum_{i=1}^{n}(\prod_{j=1}^{i-1}\exp(-\rho_j))(\mathbf{c}_i\int_0^{\infty}\omega_i\mathcal{K}_i(l)\\ \exp(-\int_0^{l}\omega_i\mathcal{K}_i(l')dl')dl).\end{aligned} \tag{8}$$

Notice that:

$$\frac{d}{dl}\exp(-\int_0^{l}\omega_i\mathcal{K}_i(l')dl') = -\omega_i\mathcal{K}_i(l)\exp(-\int_0^{l}\omega_i\mathcal{K}_i(l')dl') \tag{9}$$

Therefore, from Eqn. 8 we have:

$$\begin{aligned}\mathbf{C} &= \sum_{i=1}^{n}(\prod_{j=1}^{i-1}\exp(-\rho_j))(\mathbf{c}_i(-\exp(-\int_0^{l}\omega_i\mathcal{K}_i(l')dl'))|_0^{\infty})\\ &= \sum_{i=1}^{n}(\prod_{j=1}^{i-1}\exp(-\rho_j))(\mathbf{c}_i(1-\exp(-\int_0^{\infty}\omega_i\mathcal{K}_i(l')dl')))\\ &= \sum_{i=1}^{n}(\prod_{j=1}^{i-1}\exp(-\rho_j))(\mathbf{c}_i(1-\exp(-\rho_i)))\\ &= \sum_{i=1}^{n}\mathbf{c}_i(1-\exp(-\rho_i))\prod_{j=1}^{i-1}\exp(-\rho_j),\end{aligned} \tag{10}$$

which corresponds to the Eqn. 7 in the main paper. Therefore, there is **no restriction on the value range** of $\rho_i, i = 1, 2, ..., n$.

A.2. Parameterize and Render Geometry Field with Gaussian Surfels.

We focus on deriving the Eqn. 15 in the paper here. Recall that our goal is to evaluate the following equation:

$$\rho_i = \int_{t_i-h/\cos\theta_i}^{t_i+h/\cos\theta_i}\frac{\psi(-F(\mathbf{x}(t)))}{\Psi(-F(\mathbf{x}(t)))}||\nabla F(\mathbf{x}(t))||\cdot|\boldsymbol{\omega}\cdot\mathbf{n}(\mathbf{x}(t))|dt, \tag{11}$$

where $\Psi(\cdot)$ is the CDF of the standard normal distribution and $\psi(\cdot) = \Psi'(\cdot)$. Besides, when $|t-t_i| < h/\cos\theta_i$,

$$\begin{aligned}F(\mathbf{x}(t)) &= f_i \times \left(1 - \frac{|t-t_i|}{h/\cos\theta_i}\right) - c,\\ ||\nabla F(\mathbf{x}(t))|| &= f_i/h,\\ \cos\theta_i &= |\boldsymbol{\omega}\cdot\mathbf{n}_i|.\end{aligned} \tag{12}$$

Recall that $\nabla F(\mathbf{x}(t))$ is parallel to $\mathbf{n}_i$ and we have $\mathbf{n}(\mathbf{x}(t)) = \nabla F(\mathbf{x}(t))/||\nabla F(\mathbf{x}(t))||$, which implies that $\cos\theta_i = |\boldsymbol{\omega}\cdot\mathbf{n}(\mathbf{x}(t))|$.

Notice that Eqn. (11) is a symmetric integration with respect to t_i, therefore,

$$\begin{aligned}\rho_i &= 2\int_{t_i}^{t_i+h/\cos\theta_i}\frac{\psi(-F(\mathbf{x}(t)))}{\Psi(-F(\mathbf{x}(t)))}||\nabla F(\mathbf{x}(t))||\cdot|\boldsymbol{\omega}\cdot\mathbf{n}(\mathbf{x}(t))|dt\\ &= 2\int_{t_i}^{t_i+h/\cos\theta_i}\frac{\psi(-F(\mathbf{x}(t)))}{\Psi(-F(\mathbf{x}(t)))}\cdot(f_i/h)\cdot(\cos\theta_i)dt.\end{aligned} \tag{13}$$

Notice that, when $t \in [t_i, t_i + h/\cos\theta_i]$:

$$\begin{aligned}\frac{d}{dt}\ln\Psi(-F(\mathbf{x}(t))) &= \frac{\psi(-F(\mathbf{x}(t)))}{\Psi(-F(\mathbf{x}(t)))}\cdot(-F(\mathbf{x}(t)))'\\ &= \frac{\psi(-F(\mathbf{x}(t)))}{\Psi(-F(\mathbf{x}(t)))}\cdot(c - f_i(1-\frac{t-t_i}{h/\cos\theta_i}))'\\ &= \frac{\psi(-F(\mathbf{x}(t)))}{\Psi(-F(\mathbf{x}(t)))}\cdot\frac{f_i}{h/\cos\theta_i}\\ &= \frac{\psi(-F(\mathbf{x}(t)))}{\Psi(-F(\mathbf{x}(t)))}\cdot\frac{f_i}{h}\cdot\cos\theta_i.\end{aligned} \tag{14}$$

Therefore, Eqn. (13) is then:

$$\begin{aligned}\rho_i &= 2\int_{t_i}^{t_i+h/\cos\theta_i}\frac{\psi(-F(\mathbf{x}(t)))}{\Psi(-F(\mathbf{x}(t)))}\cdot(f_i/h)\cdot(\cos\theta_i)dt\\ &= 2\ln\Psi(-F(\mathbf{x}(t)))|_{t_i}^{t_i+h/\cos\theta_i}\\ &= 2(\ln\Psi(-(-c)) - \ln\Psi(-(f_i-c)))\\ &= 2(\ln\Psi(c) - \ln\Psi(c-f_i))\\ &= -2\ln\Psi(c-f_i),\end{aligned} \tag{15}$$

where c is a large positive number such that $\Psi(c) = 1$, and therefore $\ln \Psi(c) = 0$.

Notice that we enable the calculation of footprint function by using the extrusion with width $2h$, and it can be seen as equivalent to the original case without the extrusion, by letting $h \to 0$.

Discussion about Overlapping. It is less obvious that the intersections between Gaussian surfels and the ray could overlap, i.e., the intersection points coincide, but we argue that it is indeed the case in practice.

First of all, we explicitly utilize the depth distortion loss to promote the ray to intersect the visible surface exactly once. Additionally, assume we have a flattened surface in the space. It then requires certain number of Gaussian surfels to compose it. Since we explicitly enforce the depth-normal consistency to smooth the geometry, these Gaussian surfels could become coplanar and partially overlap with each other within the accuracy of floating point numbers. Notice that the partial overlapping here refers to the overlapping of Gaussian surfels in the 3D space instead of the overlapping of intersections with respect to the ray which we mainly talk about. Therefore, when a ray falls into the area where Gaussian surfels overlap, the intersections between Gaussian surfels and the ray then coincide.

B. Additional Implementation Details

As to the depth distortion loss, the regularization weight is set to 1000 for the DTU dataset, 10 for the BlendedMVS dataset, and 0 for the Mip-NeRF 360 dataset.

B.1. Geometry Field Splatting

We base our rasterizer implementation on that of 2DGS [6] using CUDA. Given a ray, the footprint function for the i^{th} Gaussian surfel is defined as:

$$\begin{aligned} \rho_i &= -2 \ln \Psi(c - f_i) \\ &= -2 \ln(0.5 + 0.5\text{erf}(\frac{c - f_i}{\sqrt{2}})). \end{aligned} \tag{16}$$

We choose $c = 3$, because $\Psi(3) \approx 0.999$. Therefore,

$$\rho_i = -2 \ln(0.5 + 0.5\text{erf}(\frac{3 - f_i}{\sqrt{2}})), \tag{17}$$

We further clamp the maximum of f_i to be 4.28 such that the converted opacity $1 - \exp(-\rho_i) \approx 0.99$, because as in the original 3DGS [9] and 2DGS [6] implementations, the maximum of opacity is clamped to be 0.99. Namely,

$$\rho_i = -2 \ln(0.5 + 0.5\text{erf}(\frac{3 - \min\{f_i, 4.28\}}{\sqrt{2}})). \tag{18}$$

However, we find that when f_i is small, the derivative of ρ_i is close to zero and propagates almost zero gradients back to the parameter due to the numerical error, which hinders the optimization.

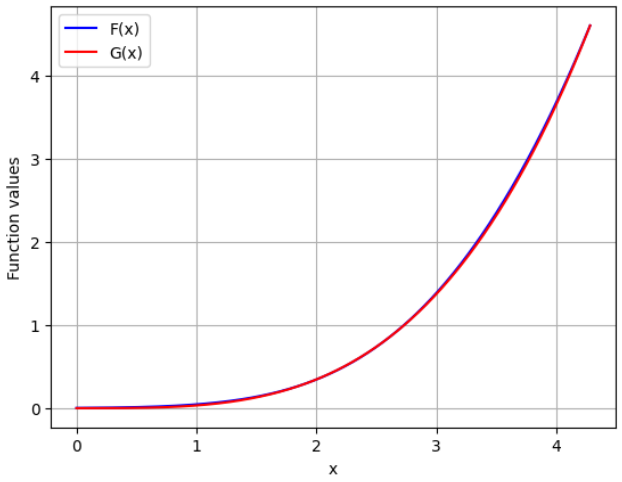

Figure 1. Plotting of function values of $F(x) = -2\ln(0.5 + 0.5\text{erf}(\frac{3-x}{\sqrt{2}}))$ and $G(x) = 0.03279x^{3.4}$ over the range $[0, 4.28]$.

To alleviate the numerical error, we approximate Eqn. (18) with a polynomial function:

$$\rho_i \approx 0.03279(\min\{f_i, 4.28\})^{3.4}, \tag{19}$$

We plot functions $F(x) = -2\ln(0.5 + 0.5\text{erf}(\frac{3-x}{\sqrt{2}}))$ and $G(x) = 0.03279x^{3.4}$ in Fig. 1, and it can be seen that these two functions are close.

B.2. Remedy Loss Landscape Defects

The color propagation per ray is approximated by propagating color in the $\mathbb{R}^3$ space. However, propagating the colors of all Gaussian surfels into the color of every Gaussian surfel is impractical.

Therefore, we use k-closest-point [16] algorithm based on the centers of Gaussian surfels to identify 10 closest Gaussian surfels to each Gaussian surfel every 100 iterations. The color of every Gaussian surfel is then blended based on these 10 closest Gaussian surfels which include itself. The choice of number of closest Gaussian surfels is made by ensuring the program does not cause out of memory error with either of our two color representations on a standard consumer-level graphics card with around 10 GB memory, on the DTU dataset [7].

B.3. Improve Color Representation

Following the default setting of [19], the shallow MLP is implemented with 2 hidden layers, and the spherical harmonics encoding of directions has degree 4.

C. Additional Evaluation Results

Evaluation Details. We find that the evaluation of mesh quality is also dependent on how well the mesh is cleaned

Methods	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Avg. ↓
Ours (SH)	0.38	0.63	0.30	0.35	0.79	0.63	0.65	1.10	1.21	0.62	0.48	1.18	0.33	0.42	0.38	0.63
- Geometry Field Splatting	0.50	0.67	0.35	0.43	0.94	0.89	0.70	1.17	1.18	0.65	0.57	1.02	0.39	0.61	0.48	0.70
- Remedy Loss Landscape Defects	0.42	0.67	0.35	0.36	0.85	0.69	0.70	1.19	1.19	0.65	0.49	1.24	0.36	0.46	0.40	0.67
+ Per-Ray Sorting	0.39	0.65	0.31	0.33	0.78	0.49	0.61	1.14	1.20	0.64	0.46	1.07	0.29	0.44	0.37	0.61
Ours (Latent)	0.40	0.59	0.39	0.38	0.72	0.59	0.65	1.08	0.93	0.59	0.50	0.67	0.34	0.47	0.40	0.58
- Geometry Field Splatting	0.52	0.79	0.47	0.52	0.91	0.71	0.84	1.17	1.07	0.63	0.65	0.82	0.42	0.75	0.53	0.72
- Remedy Loss Landscape Defects	0.38	0.56	0.39	0.36	0.80	0.60	0.68	1.13	0.95	0.64	0.50	0.65	0.35	0.48	0.40	0.59
+ Per-Ray Sorting	0.43	0.64	0.38	0.34	0.73	0.50	0.57	1.13	1.00	0.65	0.49	0.59	0.30	0.48	0.38	0.57
- Ray Direction Conditioning	0.47	0.61	0.42	0.39	0.81	0.78	0.75	1.13	0.98	0.60	0.56	0.67	0.39	0.55	0.44	0.64
- Reflected Ray Direction Conditioning	0.39	0.61	0.40	0.37	0.78	0.60	0.62	1.09	0.98	0.61	0.50	0.80	0.35	0.47	0.39	0.60

Table 1. Quantitative evalution on the DTU dataset based on the Chamfer Distance for different ablation models. The best metric is highlighted in red, the second best metric is highlighted in orange, and the third best metric is highlighted in yellow.

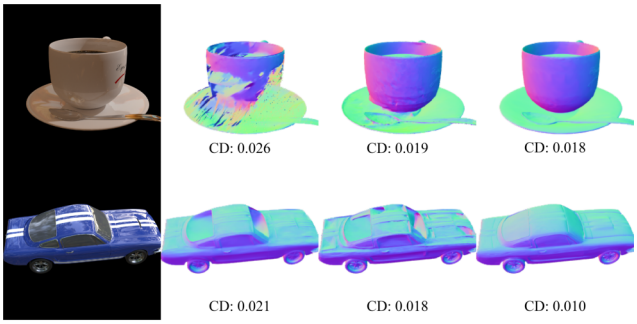

Figure 2. Comparisons of normal maps and chamfer distances (denoted as "CD"; lower is better) of reconstructed geometry for highly specular surfaces (first column) from the ShinyBlender dataset [20].

Methods	Outdoor Scenes			Indoor Scenes		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF	21.46	0.458	0.515	26.84	0.790	0.370
INGP	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
BakedSDF	22.47	0.585	0.349	27.06	0.836	0.258
MipNeRF 360	24.47	0.691	0.283	31.72	0.917	0.180
BOG	23.94	0.680	0.263	27.71	0.873	0.227
3DGS	24.64	0.731	0.234	30.41	0.920	0.189
SuGaR	22.93	0.629	0.356	29.43	0.906	0.225
MipSplatting	24.65	0.729	0.245	30.90	0.921	0.194
2DGS	24.18	0.703	0.287	30.06	0.909	0.213
GOF	24.82	0.750	0.202	30.79	0.924	0.184
RaDe-GS	25.17	0.764	0.199	30.74	0.928	0.165
Ours (SH)	24.40	0.734	0.224	29.93	0.916	0.194
Ours (Latent)	23.76	0.693	0.293	29.92	0.906	0.219

Table 2. Quantitative comparison on the Mip-NeRF 360 dataset based on the view synthesis.

after extraction. Typically, SDF-based approaches use the marching cube algorithm to produce closed-surface meshes which differ from the open-surface ground-truth and then contain redundant parts. In contrast, splatting-based approaches typically use the TSDF fusion algorithm to produce open-surface meshes based on the depth maps, which are more concise. It then becomes necessary to clean triangles, which are invisible from all training views, to have a fair unified evaluation protocol. In practice, we combine the scripts from [11] and [6] to clean the mesh before evaluation. As to the Neuralangelo, we find that its extracted mesh sometimes is enclosed in a sphere, which makes the mesh cleaning fail. Therefore, we manually remove the enclosing spheres if they exist for the extracted meshes of Neuralangelo before passing them into the evaluation.

C.1. Quantitative Evaluation

We evaluate the view synthesis quality of our method on the Mip-NeRF 360 dataset [1], while comparing with NeRF [12], INGP [13], MERF [17], BakedSDF [22], MipNeRF 360 [1], BOG [18], 3DGS [9], SuGaR [5], MipSplatting [24], 2DGS [6], GOF [25], and RaDe-GS [27]. We find that with geometry clearly defined and its corresponding regularization, the view synthesis is harmed.

As to the rendering speed during inference, since our proposed color propagation is irrelevant to the view, it can be achieved and cached before rendering, thus does not impact the rendering speed.

C.2. Qualitative Evaluation

We show complete rendering of all cases on the DTU dataset [7] and BlendedMVS dataset [21] between our method and 2DGS [6] in Fig. 3, Fig. 4, and Fig. 5. In the 2DGS algorithm, using the median depth in the regularization leads to better quantitative results, while using the mean depth in the regularization produces smoother geometry and more visually pleasing results. In Fig. 3, we compare with 2DGS trained with both settings. We capture more geometric details than those trained with mean depth in 2DGS, and are free of cracks and holes compared to those trained with median depth in 2DGS. We also show the extracted mesh on a few scenes of MipNeRF 360 dataset in Fig. 7 using our method and TSDF fusion. We further provide preliminary comparison results for two selected cases on the ShinyBlender dataset [20] in Fig. 2 to demonstrate the capability of our method to generalize to highly specular surfaces.

We also test our method on the Tanks&Temples dataset [10] but this dataset does not provide ground-truth camera parameters, which requires an iterative-closest-point procedure to align the extracted mesh and ground-truth mesh. After manual inspection, we find that the metrics heavily depend on how well this off-the-shelf alignment algorithm aligns the extracted mesh and ground-truth mesh, and it actually

fails on two scenes. Therefore, we choose to only show our qualitative results in Fig. 6.

C.3. Ablation Study

We provide the complete evaluation results on the DTU dataset for all our ablation models in Tab. 1. Specifically, we evaluate with our two color representations, i.e., SH representation and latent representation. As to the latent representation, we also ablate on the choice of conditioning for MLP. We conduct all the ablation experiments with the same hyper-parameters with the baseline.

By comparing the baseline with the ablation model without geometry field splatting, it is clear that the geometry field splatting instead of the original approximate rendering formulation in 2DGS significantly boosts the performance. However, we also find that the geometry field splatting makes the SH representation more sensitive to the specular surfaces, which is reflected in the "scan110". By comparing the baseline with the ablation model without the remedy to loss landscape defects, we can see that our proposed color blending consistently improves the performance on almost every case, especially for the SH representation. The ablation model which implements per-ray sorting achieves better averaged performance, but does not always achieve the best performance on every case. We argue that it may be due to the fact that the hyper-parameters are tailored for the global sorting approximation, and not tuned for the per-ray sorting case. Implementing [15] with our algorithm while tuning relevant hyper-parameters may further improve our performance.

As to the latent representation, without ray direction conditioning or reflected ray direction conditioning, the reconstructed geometry suffers.

D. Additional Discussions

In this paper, we focus on evaluating our proposed geometry field representation and remedy to loss landscape defects, and therefore do not incorporate many other methods which could further improve the geometry reconstruction. However, we identify a few potential ways here which may further improve the geometry reconstruction with our method.

Anti-aliasing. As discussed in [6], a 2D Gaussian is degenerated into a line in the screen space while being observed from a slanted viewpoint. Besides, as discussed in [28], the output image has to be band-limited to avoid aliasing. We follow [6] to only apply an object-space low-pass filter [2], which actually does not follow our rendering model. Similar to [24], it would be useful to design an anti-aliasing method which is tailored for our rendering model.

Appearance Embedding. We assume that the scene is static and the images are captured in the same lighting condition. However, there could be slight variance of lighting conditions, and brightness of the images, etc., due to the capture. Therefore, it would be useful to have an appearance embedding per image as in [3, 25, 27].

Color Representation. We choose to use a latent representation for color that is as simple as possible to improve on the specular cases and emphasize more on our proposed geometric representation. In the future, it will be beneficial to further extend our method with better latent representations as in [8] to work on highly specular surfaces.

Multi-view Stereo Constraint. Concurrently, Chen et al. [3] shows that the multi-view stereo constraint could be helpful for the geometry reconstruction quality using the volume splatting representation. It would be useful to apply the multi-view constraint with our clearly defined geometry, which may further improve the reconstruction.

Incorporate Monocular Priors. In the dense-views geometry reconstruction task, the captured images are assumed to be sufficient to recover the geometry. However, there may not be clear standard to decide whether the captured images are sufficient to define a unique solution, and, in practice, the captured images are usually not enough. As in [4], it would be useful to incorporate modern monocular priors, such as the estimated monocular normal (e.g., [26]), to compensate for the deficiency of input images, especially for the scene-level cases.

Densification Strategies. We observe that, due to our refined splatting algorithm, a Gaussian surfel could have a much larger fully opaque area. Even though this property benefits the geometry reconstruction overall, it makes the optimization and densification harder, which is also reflected in the affected view synthesis quality. Since the optimization is stochastic, a Gaussian surfel could have a large fully opaque area. Besides, the enforced depth-normal consistency loss effectively squeezes the Gaussian surfel, which makes the large fully opaque area difficult to reduce. It then blocks the optimization towards the other Gaussian surfels it occludes. Even though we periodically reset the geometry value on the Gaussian surfel to make it less opaque, we find that its geometry value will still recover fast instead of shrinking its size.

We find that using the densification strategy proposed by [23] leads to smaller Gaussian surfels in general without losing the quality, which alleviates such a problem. However, a different strategy which may fully solve this problem is welcome.

Mesh Extraction. We follow [6] to use the TSDF fusion to extract the mesh. Even though it gives reasonable results, it is slow as it operates on the CPU and it also cannot handle thin structures such as the strips on the wheels of the bicycle well. In contrast, we also find the marching tetrahedra proposed by [25] works sub-optimally on the object-centric cases. With the advance of explicit primitives, i.e., Gaussian kernels, it would be useful to have a mesh extraction method that may directly convert primitives of different representations.

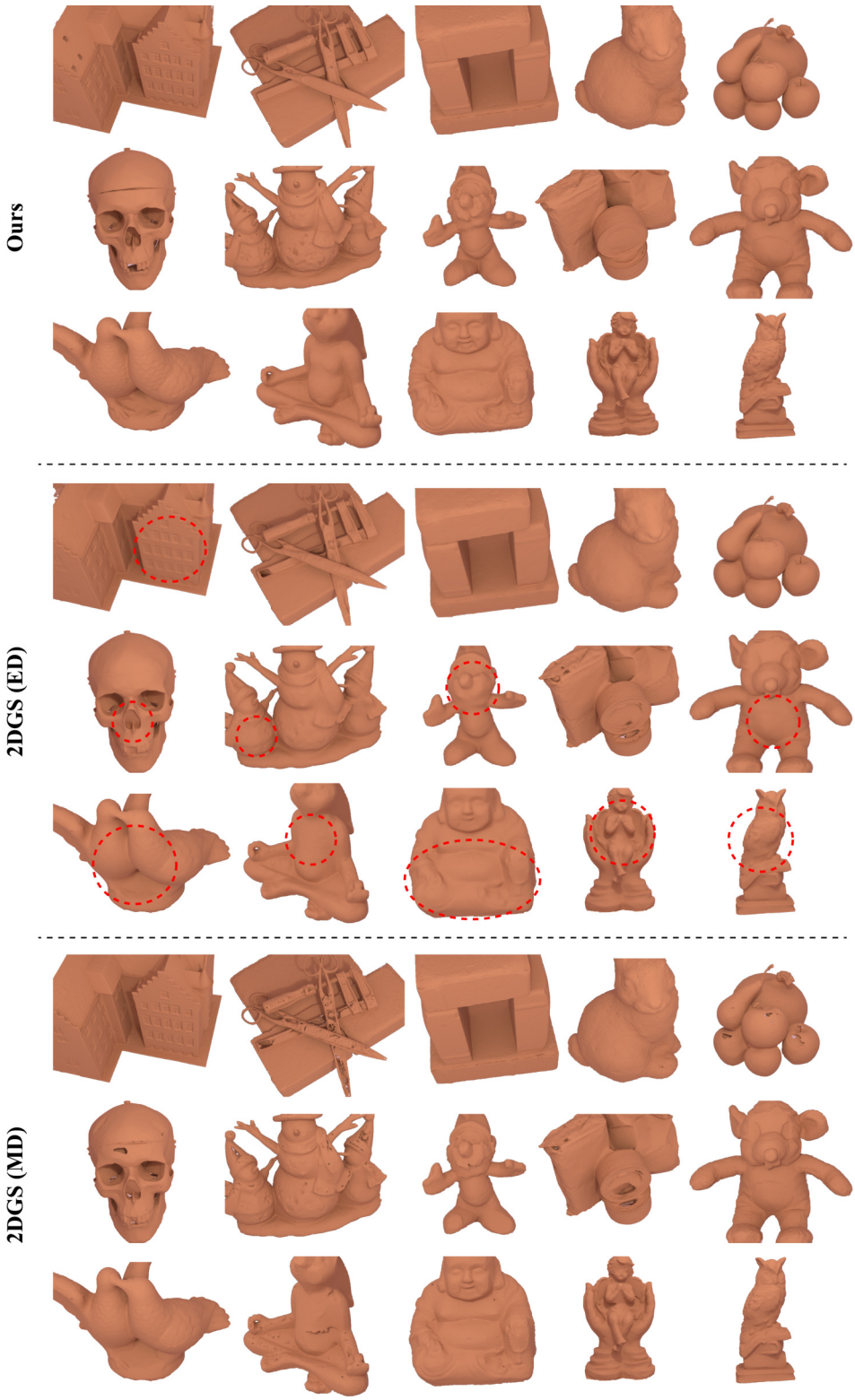

Figure 3. Comparison of rendering of meshes extracted by our method (at the top), 2DGS trained with mean depth in the regularization (in the middle), and 2DGS trained with median depth in the regularization (at the bottom) on the DTU dataset. Non-obvious differences are highlighted in red circles. The reader may wish to zoom into the electronic version in the figures.

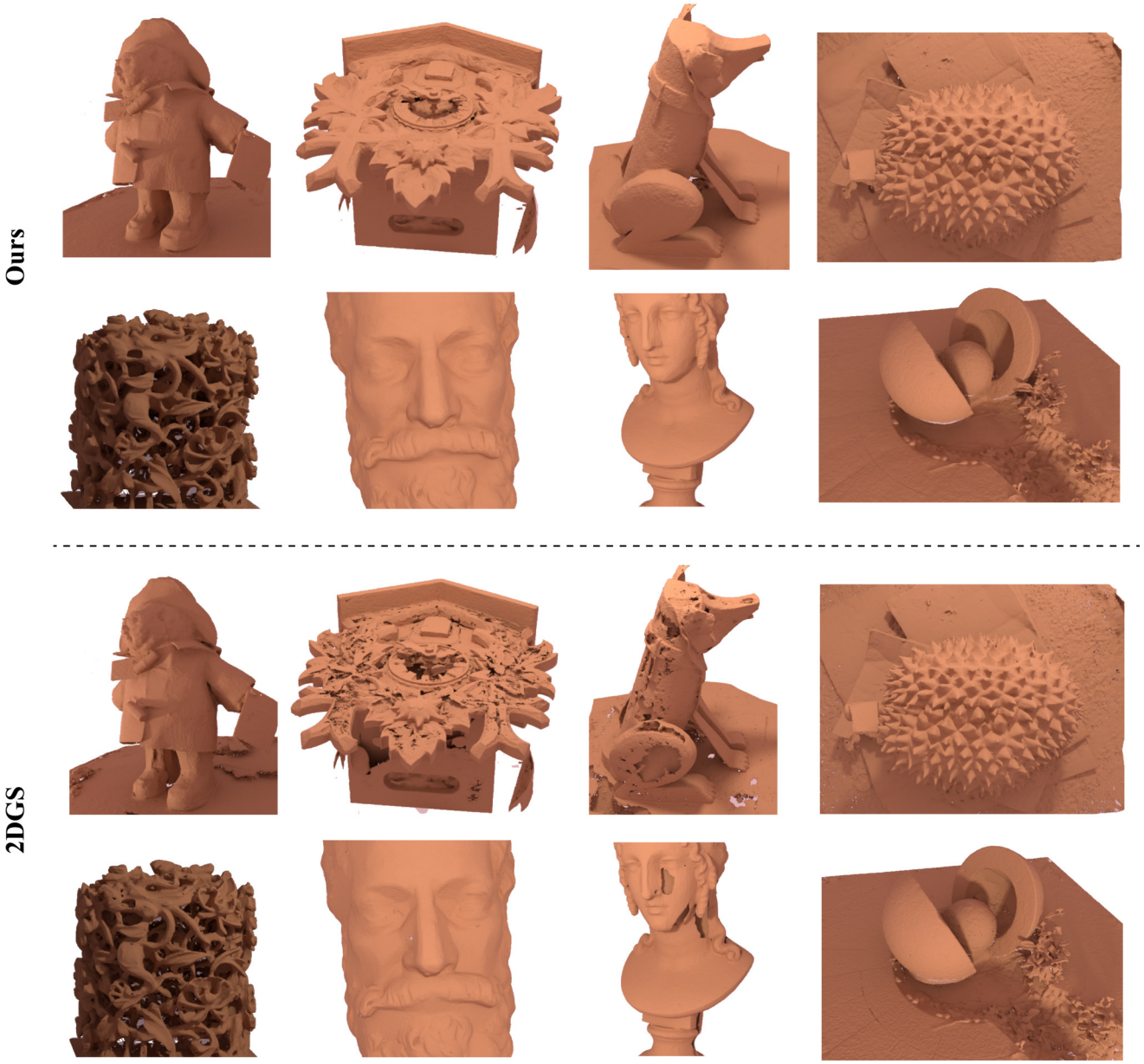

Figure 4. Comparison of rendering of meshes extracted by our method (at the top) and 2DGS (at the bottom) on the object cases of the BMVS dataset.

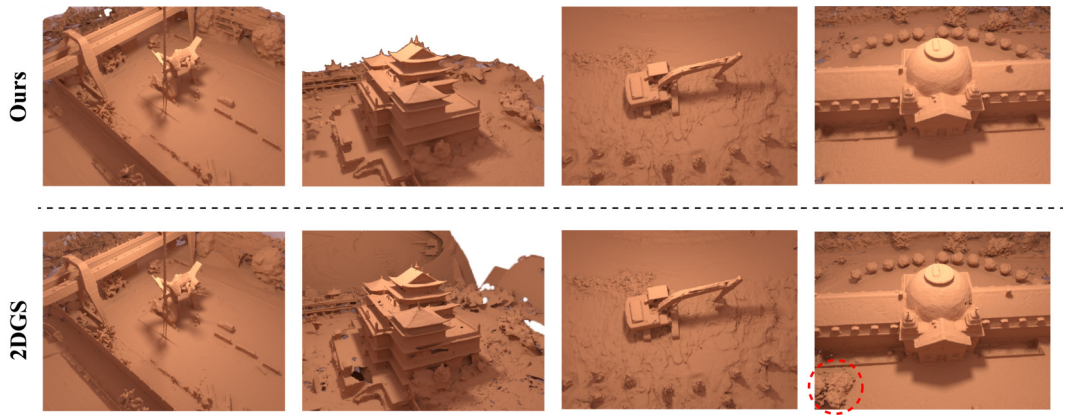

Figure 5. Comparison of rendering of meshes extracted by our method (at the top) and 2DGS (at the bottom) on the scene cases of BMVS dataset. Non-obvious differences are highlighted in red circles.

Figure 6. Rendering of meshes extracted by our method on the Tanks&Temples dataset.

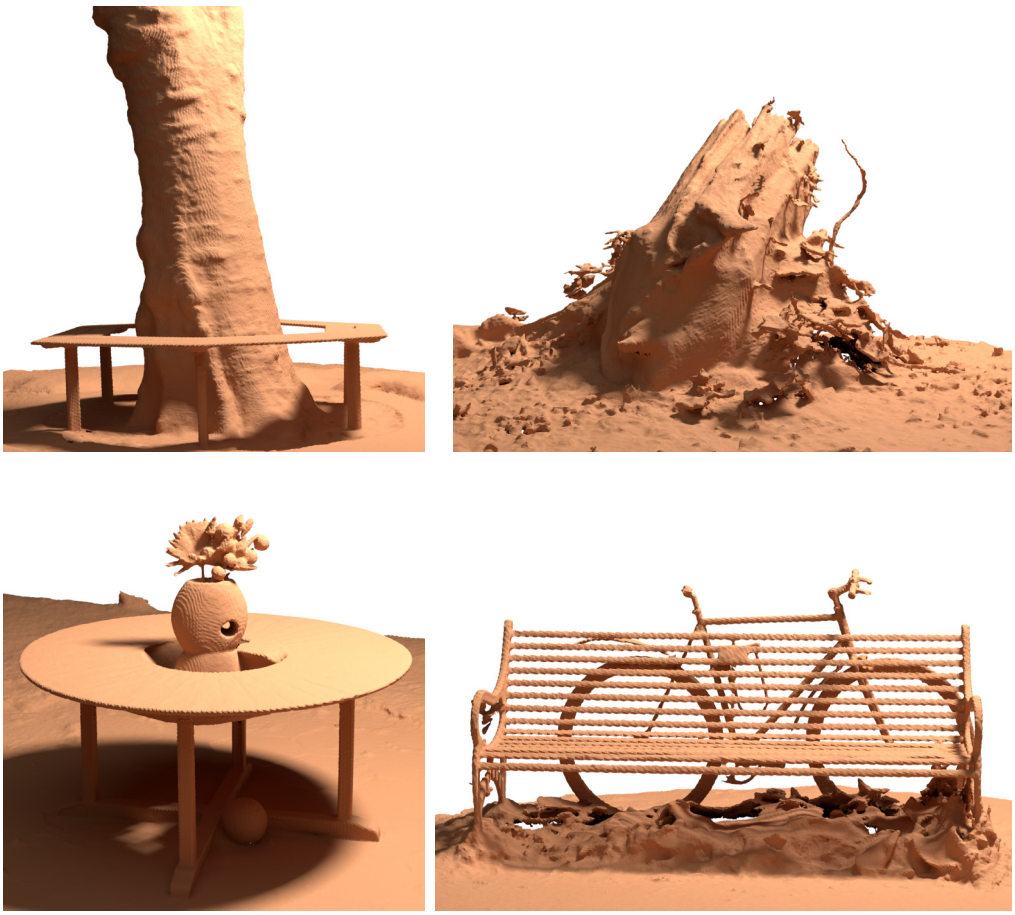

Figure 7. Rendering of meshes extracted by our method on the Treehill, Garden, Bicycle and Stump scenes of MipNeRF 360 dataset.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 4

[2] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. High-quality surface splatting on today's gpus. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics*, page 17–24, Goslar, DEU, 2005. Eurographics Association. 5

[3] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *arXiv preprint arXiv:2406.06521*, 2024. 5

[4] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 2, 5

[5] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR*, 2024. 4

[6] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 2, 3, 4, 5

[7] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413, 2014. 3, 4

[8] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5322–5332, 2024. 5

[9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 2, 3, 4

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

[11] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and

Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *European Conference on Computer Vision*, pages 210–227. Springer, 2022. 4

[12] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 4

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

[14] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: surface elements as rendering primitives. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, page 335–342, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 2

[15] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering. *ACM Transactions on Graphics (TOG)*, 43(4): 1–17, 2024. 1, 5

[16] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 3

[17] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023. 4

[18] Christian Reiser, Stephan Garbin, Pratul P. Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan T. Barron, Peter Hedman, and Andreas Geiger. Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis. *SIGGRAPH*, 2024. 4

[19] Jiaxiang Tang. Torch-ngp: a pytorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp. 3

[20] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022. 4

[21] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. *Computer Vision and Pattern Recognition (CVPR)*, 2020. 4

[22] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*, New York, NY, USA, 2023. Association for Computing Machinery. 4

[23] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details in 3d gaussian splatting. In *ACM Multimedia 2024*, 2024. 5

[24] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19447–19456, 2024. 4, 5

[25] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient high-quality compact surface reconstruction in unbounded scenes. *arXiv:2404.10772*, 2024. 4, 5

[26] Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. Rgb↔x: Image decomposition and synthesis using material-and lighting-aware diffusion models. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 5

[27] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *arXiv preprint arXiv:2406.01467*, 2024. 4, 5

[28] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538, 2001. 1, 2, 5